Optimizing Sensing
Theory and Applications

Andreas Krause

CMU-CS-08-171
December 2008

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Thesis Committee:
Carlos Guestrin, Chair
Anupam Gupta
Tom Mitchell
Tommi Jaakkola, MIT
William Kaiser, UCLA

Submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

This research was sponsored by the Office of Naval Research under grant number N000140810752, the National Science Foundation under grant numbers CNS-0509383 and CNS-0625518, and a Microsoft Research Graduate Fellowship.

Keywords: Machine learning, sensor networks, probabilistic models, optimization, approximation algorithms, observation selection, value of information, active learning, submodularity

Abstract

Many practical applications, such as environmental monitoring or placing sensors for event detection, require to select among a set of informative but possibly expensive observations. When monitoring spatial phenomena with sensor networks or mobile robots, for example, we need to decide which locations to observe in order to most effectively decrease the uncertainty, at minimum cost. These problems are usually **NP**-hard. Previous approaches for tackling these sensing problems have mainly relied on myopic heuristics, i.e., approaches which only consider the next best observation to add, without planning ahead for future sensing opportunities. For those approaches, typically no performance guarantees are known. Existing nonmyopic approaches have involved computationally intensive techniques which are very difficult to scale to larger problems.

In this Thesis, we present a new class of approaches for observation selection, using techniques from combinatorial optimization. We show that many observation selection objectives satisfy *submodularity*, an intuitive diminishing returns property – adding a sensor to a small deployment helps more than adding it to a large deployment. Examples include mutual information for spatial prediction and placing sensors for outbreak detection.

We also develop a suite of non-greedy approaches that systematically exploit this submodularity property in order to *efficiently* obtain provably *near-optimal* solutions to complex sensing problems. For example, the chosen observations often need to be robust against sensor failure or uncertainty about model parameters. Examples include minimizing the maximum posterior variance in spatial prediction and robust experimental design. We show, that many such problems require the optimization of an adversarially chosen submodular objective function. We will demonstrate that for this problem, existing greedy algorithms perform arbitrarily badly. We develop an algorithm, SATURATE, which performs near-optimally in this setting.

In addition to the problem of finding the best k observations (sensor locations), we consider problems with complex combinatorial constraints. For example, when placing wireless sensor networks, the chosen locations should not only be very informative, but also allow efficient communication. When using robots for making observations, the chosen locations need to lie on a collection of paths. We present PSPIEL, an efficient algorithm which finds solutions which near-optimally trade off sensing quality and cost.

When deploying wireless sensor networks, another fundamental constraint is battery lifetime. Since every measurement draws power, sensors can typically be activated only a fraction of the time. Hence, the problem of scheduling the sensors becomes of crucial importance. Traditionally, sensor placement and scheduling have been considered separately from each other. In this Thesis, we present ESPASS, an efficient algorithm for simultaneously optimizing the placement and scheduling. We show that this simultaneous approach leads to drastic improvements in network lifetime when compared to the traditional, stage-wise approach.

A key question in many observation selection problems is how much better a sequential algorithm, which decides on the next observation based on previous observations, can perform when compared to the best fixed set of observations chosen a priori. We present a partial answer to the question for spatial prediction in Gaussian Processes. We develop a theoretical bound quantifying the gap between the best sequential and a priori selections, and use it to develop an exploration–exploitation approach for active learning in Gaussian Processes.

Lastly, we look beyond submodular observation selection problems, and consider the problem of selecting *optimal* observations in graphical models. We show that in chain-graphical (Markovian) models, it is possible to efficiently find the optimal sequential policy. However, even for slightly larger classes of graphical models, we prove strong hardness results.

In addition to providing algorithms and theoretical analyses, we present extensive empirical evaluation of our approaches on several real-world case studies. These include building a sensing chair for posture recognition, monitoring environmental phenomena with mobile robots, securing municipal water distribution networks, and selecting informative weblogs to read on the Internet.

Acknowledgments

I would like to thank everybody who helped to make this Thesis possible. First and foremost, I am deeply grateful to my advisor, Carlos Guestrin, who continuously encouraged, challenged and supported me. His guidance extended well beyond my thesis research, most recently to my successful job search. Through his invaluable advice I learned so much about doing research and being part of the scientific community.

I would also like to thank Anupam Gupta for many interesting discussions and fruitful collaborations. I am also thankful to the other members of my thesis committee, Tom Mitchell, Tommi Jaakkola and William Kaiser, for their helpful and careful feedback.

To Dan Siewiorek, my undergraduate thesis mentor, I am also very thankful, as he taught me so much about research and encouraged me to pursue an academic career.

It was very enjoyable and rewarding to collaborate with all my coauthors – Carlos Guestrin, Anupam Gupta, Jure Leskovec, Jon Kleinberg, Ajit Singh, Amarjeet Singh, Vipul Singhvi, William Kaiser, Jeanne VanBriesen, Christos Faloutsos, Brendan McMahan, Natalie Glance, Bilge Mutlu, Eric Horvitz, Jodi Forlizzi, Jessica Hodgins, Feng Zhao, Aman Kansal and Ram Rajagopal – on the research that eventually became part of this Thesis, and beyond. I am also thankful to Jeff Bilmes and Mukund Narasimhan for many interesting discussions about submodularity in machine learning.

I would also like to thank all the members of the SELECT lab for great discussions and many fun meetings: Carlos Guestrin, Geoff Gordon, Stanislav Funiak, Khalid El-Arini, Joseph Gonzalez, Sue Ann Hong, Jonathan Huang, Anton Chechetka, Joseph Bradley, Byron Boots, Austin McDonald, Alexandra Meliou, Ram Ravichandran, Sajid Siddiqi, Dafna Shahaf, Ajit Singh, Gaurav Veda, Felipe Trevizan and Miro Dudik.

Many thanks to all my friends at CMU. In particular, Jure Leskovec, Joseph Gonzalez, Khalid El-Arini and Stano Funiak for sharing thoughts and advice, for conquering (or being conquered by) the winds with me, playing squash, hiking and many other fun activities. Also thanks to Jim Ferla and the members of the Guitar Ensemble at CMU – it was a great pleasure and a lot of fun to rehearse and perform music with them. There are many more wonderful people I met at CMU, in Munich and elsewhere, who I am thankful to for many great discussions, for lending a helpful hand and getting me away from my laptop.

Last but not least, I cannot thank enough my family for always being there for me, for giving everything and letting me choose my way, and Kathrin for her endless patience, support and inspiration. Without them, I would never have been able to accomplish what I have.

Contents

Part I

Background and Survey

Chapter 1

Introduction

Consider the problem, where we want to place a set of sensors in order to monitor a complex spatiotemporal phenomenon, such as temperature and light in a building, pollution in a lake, precipitation in an extended geographic region, traffic conditions in a road network or water quality in a municipal water distribution network. Or let us think about an online service which crawls the web, deciding which weblogs and news websites to suggest to its users. Or about a physician deciding between which tests to administer to a patient in order to decide on the treatment.

In such problems, we can make observations by placing a set of sensors or detector stations, display a set of links to blogs to a user, or apply a medical testing procedure. In practice, such observations are typically expensive, measured, e.g., in terms of sensor and deployment cost, power consumption, time and effort required for performing an experiment, patience and attention of a user, or cost for performing medical tests. A key question common to all these problems is thus:

> Which observations should we make, in order to acquire the most useful information as cost-effectively as possible?

We call this class of problems *sensing problems*, where we take a wide definition of the notion of a "sensor". Such sensing problems have gained much attention, in areas such as statistical experimental design, decision theory, operations research, probabilistic planning, sensor networks. Most of the optimization problems are **NP**-hard, thus in general not likely to be amenable for efficient exact solution.

There are typically two classes of approaches: Heuristic approaches, which do not strive for optimality, but quickly find solutions with often reasonable performance. However, typically, no performance guarantees are known for these approaches. These approaches include *myopic* algorithms such as greedy heuristics, or continuous relaxation approaches. On the contrary, *nonmyopic* approaches try to find the optimal solution, but often are very difficult to scale to larger problems. These approaches include techniques such as probabilistic planning using Partially Observable Markov Decision Processes, or Mixed Integer Programming.

1.1 Thesis Statement and main contributions

In this Thesis, we present a new class of nonmyopic approaches for solving complex sensing problems. We claim the following statement:

> In many practical applications, it is important to select among useful but expensive observations. By exploiting problem structure, one can efficiently and near-optimally solve many of the resulting sensing problems.

In order to substantiate this claim, we present the following contributions.

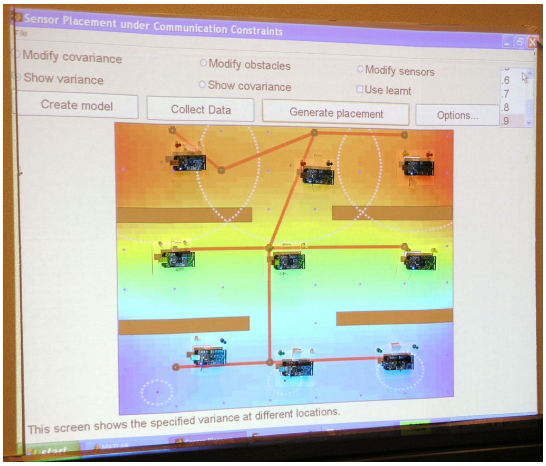
1.1.1 Submodular sensing

We first consider *subset selection* problems, where we choose all observations (locations to place sensors, links to blogs to present) before acquiring the information. In sensor placement problems for example, we are given a set $\mathcal{V}$ of possible sensing locations, and would like to pick a small subset $\mathcal{A} \subseteq \mathcal{V}$ of locations that maximizes a (problem-specific) sensing quality function $F(\mathcal{A})$. For example, when placing sensors to monitor temperature in a building, $F(\mathcal{A})$ can model the reduction in prediction error after obtaining measurements at locations $\mathcal{A}$. When placing sensors for outbreak detection, $F(\mathcal{A})$ could measure the expected improvement in the time required to detect some event after placing sensors at locations $\mathcal{A}$.

We show that a large class of sensing problems satisfy *submodularity*, an intuitive diminishing returns property: Adding a sensor to an existing deployment helps more if we have placed few sensors so far, and less if we have already placed many sensors. A fundamental result by Nemhauser et al. [1978] then guarantees that a simple greedy algorithm, which iteratively adds the sensor that increases the sensing quality of the network the most, achieves a near-optimal solution. Hence identifying submodularity in a problem is very useful, as it allows to use a simple, efficient greedy algorithm to near-optimally solve a complex, $\mathbf{NP}$-hard optimization problem.

Traditionally, many sensing problems have been modeled as a *set cover* problem [*c.f.*, Abrams et al., 2004, Bai et al., 2006, Deshpande et al., 2008, Hochbaum and Maas, 1985, Kershner, 1939]. In such problems, each sensor is associated with a fixed sensing region, and the sensing quality $F(\mathcal{A})$ of a set $\mathcal{A}$ of sensors is the total area covered by the union of all selected sensing regions. The set cover function $F(\mathcal{A})$ is an example of a submodular function. However, in many practical sensing problems, this traditional assumption is too strong. For example, when placing sensors for environmental monitoring, we are interested in directly optimizing the prediction performance of the sensor network. Hence, traditional algorithms developed for the set cover sensing quality function cannot be applied to such sensing problems.

In Part II of this Thesis, we show that much more complex sensing quality functions are submodular as well. Examples that we study in detail include spatial prediction in Gaussian Processes, sensor placement for outbreak detection and selecting informative variables in graphical models. By exploiting this structural property of submodularity, many sensing problems can be cast as submodular optimization problems. In this Thesis, we will demonstrate many benefits of this insight. Our research in this area received the *Best Paper Runner Up award* at UAI 2005 and ICML 2005, as well as the *Best Student Paper award* at KDD 2007.



(a) *Demo*

(b) *NIMS-AQ*

Figure 1.1: (a) Live demonstration of PSPIEL for trading off informativeness and communication cost at IPSN 2006 (also demonstrated at NIPS 2005). The demonstration setup uses light sensor motes and a video projector. (b) The NIMS-AQ system (image courtesy of CENS) for environmental monitoring of lakes.

1.1.2 Non-greedy algorithms for complex sensing problems

In Part III of this Thesis, we study complex sensing problems that go beyond selecting a set of observations maximizing a submodular sensing quality function. As we show, for these problems, the greedy algorithm fails arbitrarily badly. In order to tackle these sensing problems, we develop efficient non-greedy algorithms with strong theoretical guarantees.

Robust sensing problems. An important requirement, which frequently arises in practice, is robustness. For example, when deploying a network of sensors, hardware failures can occur, resulting in reduced coverage. Often, when using models for prediction, there is uncertainty in the model parameters. We show how this robustness requirement leads to the problem of simultaneously maximizing multiple submodular sensing quality functions $F_1, \ldots, F_m$, e.g., each quantifying informativeness under a different parameter setting. In order to obtain robust solutions to the resulting sensing problems we then have to optimize the minimum of the submodular functions, i.e., maximize $\min_i F_i$. We present an efficient non-greedy algorithm, SATURATE, which provides strong theoretical guarantees for this problem.

Complex combinatorial constraints. In practice, often complex constraints are associated with the sensing task. For example, when placing a network of wireless sensors, the sensors need to be able to reliably communicate over lossy links, constraining their locations not to be too far apart. When using robots to collect observations, the chosen sensing locations need to lie on a path traversed by the robot. Fuel and time constraints limit the length of this path. We present PSPIEL, an efficient, randomized, non-greedy approximation algorithm, which exploits problem structure to nonmyopically select observations even under such complex constraints. Figure 1.1(a) shows a demonstration of our PSPIEL approach that we presented at IPSN 2006 and NIPS 2005. Our research in this area was awarded the *Best Paper award* at IPSN 2006.

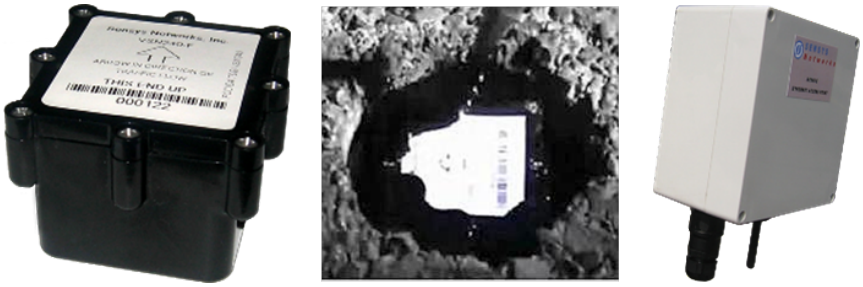


Figure 1.2: Sensys Networks wireless traffic sensor. (left) encased unit, (middle) sensor deployed in pavement, (right) GPRS/CDMA base station.

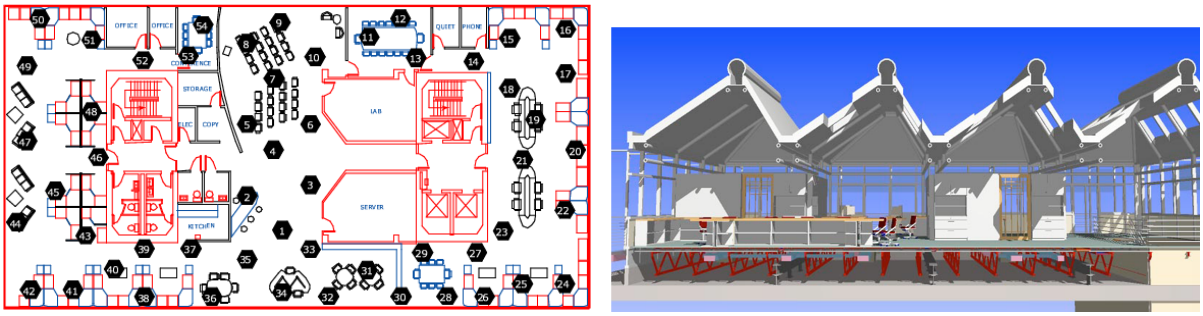
Simultaneous placement and scheduling. When deploying sensor networks, battery power is a fundamental constraint. For many practical applications, such as using sensors for traffic monitoring, where placing sensors requires closing roads from traffic, a minimum network lifetime is required to ensure feasibility of the deployment. Since every measurement that a sensor acquires drains its battery, often, the only possibility to meet such lifetime requirements is to *schedule* sensors, i.e., decide when to activate each sensor. Traditionally, sensor placement and scheduling have been considered separately from each other: One first places the sensors, and then optimizes the schedule to maximize lifetime. We develop a novel algorithm, ESPASS, that can simultaneously optimize the sensor placement and the schedule. We apply ESPASS to the problem of increasing the deployment lifetime of traffic sensors (as shown in Figure 1.2) under California highways. In this and other applications, our simultaneous approach drastically outperforms the traditional, stage-wise approach.

1.1.3 Algorithms for sequential sensing problems

In sensor placement and many other sensing problems it is natural to commit to all observations (sensor locations) before any measurements are made. In other sensing problems, however, we would like to take into account measurements made in the past before we adaptively decide on the next observation (sensing location).

In Part IV of this Thesis, we study such *sequential sensing problems*. Instead of optimizing for a set of locations, we try to find a *policy* (e.g., a decision tree, conditional plan), which decides on the next observation based on the previous measurements. Since submodularity is a property of sets of observations (and not policies), this concept cannot be used to analyze sequential sensing problems. However, we show that in some interesting cases it is nevertheless possible to exploit problem structure in order to develop nonmyopic algorithms.

The adaptivity gap for sequential sensing. A key question in sequential observation selection is the following: How much better can a sequential algorithm possibly perform compared to the best fixed a priori set of observations? We present a theoretical bound quantifying this *adaptivity gap* for the important case of spatial prediction in Gaussian Processes, and develop an efficient, nonmyopic, sequential algorithm with sample complexity guarantees. This bound exploits a structural property of Gaussian Processes.



(a) *Sensor network deployment* (b) *Intelligent workplace*

Figure 1.3: (a) A deployment of a sensor network with 54 nodes at the Intel Berkeley Lab. (b) Intelligent Workplace at CMU Architecture Department.

Optimal sequential sensing in chain models. Lastly, we show how *conditional independence*, another kind of problem structure, can be exploited to develop algorithms which find the *optimal* sequential strategy for an interesting class of graphical models, containing Hidden Markov models. Even though our algorithms can only find the optimal solutions for chain graphical models, they nevertheless perform well on more general graphical models. We apply our algorithms to several tasks, including active learning for named entity recognition and sensor scheduling in a building management testbed deployment, as shown in Figure 1.4(a).

1.1.4 Hardness results

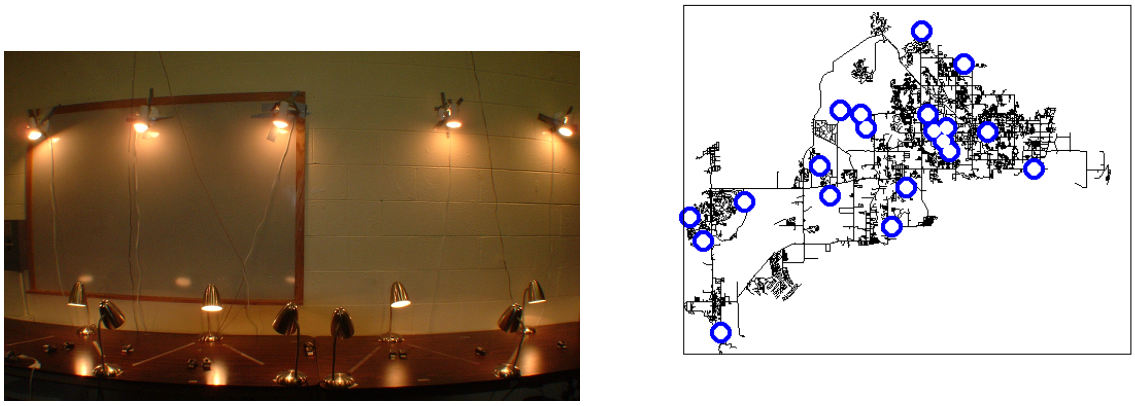
In addition to developing new algorithms for sensing problems, we present several hardness results, which in some cases even match the performance of our algorithms.

For example, we show that for the problem of selecting most informative variables in graphical models, the constant factor $(1 - 1/e)$ approximation guarantee obtained by the greedy algorithm is in fact best possible, unless $\mathbf{P} = \mathbf{NP}$. We also show strong hardness results for solving robust sensing problems. These results imply that, under reasonable complexity theoretic assumptions, our algorithm SATURATE provides best possible guarantees for this problem.

Most notably, we present a surprising result about optimizing sensing in graphical models. Most problems in graphical models, such as probabilistic inference, which can be efficiently solved for chain (Markovian) models, generalize to polytrees, i.e., models with tree-like conditional independence structure. While optimal sensing is tractable for chains, we show, that even for the slight generalization to polytrees, the problem becomes wildly intractable ($\mathbf{NP}^{\mathbf{PP}}$-complete).

1.1.5 Applications and empirical studies

One important goal of this work is to provide empirical evidence of the effectiveness of the proposed algorithms on several real-world sensing problems.



(a) *Building management testbed* (b) *Metropolitan area Water Network*

Figure 1.4: (a) Our testbed for the building automation setting. We schedule sensors to only turn on in situations which most improve our sensing quality. Our techniques lead to a factor of four improvement in deployment lifetime. (b) Example sensor placement in a realistic water network.

Sensor placement for building monitoring. A major focus of this Thesis are spatial prediction problems in the context of Gaussian Processes. We provide empirical studies on data from real sensor network deployments in the area of building and environmental monitoring. In addition to extensive comparison of our algorithms with existing techniques on standard data sets (such as data from a sensor network deployment at Intel Research, Berkeley, Figure 1.3(a)), we performed a full proof-of-concept study.

In this study, our goal was to compare the performance of our placement methods to *manual placements* and existing algorithms. We deployed a network of 46 wireless sensor nodes in the Intelligent Workplace at CMU (*c.f.*, Figure 1.3(b)). As a baseline deployment, we manually selected 20 locations that seemed to capture the overall variation in light intensity. We also used our algorithms to propose optimized placements. In this case study, which is described in detail in Chapter 11, our algorithms *drastically reduced* the prediction error by about 50% compared to the manual deployment.

Securing municipal water distribution networks. Drinking water distribution networks are complex dynamical systems, which are of fundamental importance to our everyday lives. The intentional introduction of a contaminant disrupts the system and could be detected by a network of sensors placed in the system. Optimal placement is crucial and especially difficult in this application domain, because the system has complex temporal dynamics and the number of possible intrusion scenarios is vast. On the other hand, sensors are very expensive (roughly $14,000 per unit), and hence only a small number of sensors could potentially be deployed.

In August 2006 we participated in the *Battle of the Water Sensor Networks*, an international challenge of designing sensor network deployments for realistic, metropolitan-area-sized water distribution networks. The goal was a multi-criterion optimization problem, trading off several natural objective functions: minimizing the time to detection, minimizing the population affected and contaminated water consumed prior to detection and maximizing the detection likelihood. We showed that all these objectives are submodular, which allowed us to apply the algorithms developed in this Thesis. Empirically, our algorithms often found the optimal solution.

Figure 1.5: Expensive high fidelity sensor (4032 point sensors). We achieve comparable classification accuracy using only 20 flat and point sensors.

We applied our approach to a large realistic distribution system, comprising more than 12,000 nodes respectively (*c.f.*, Figure 1.4(b)), simulated over several days at high resolution, and determined optimal locations for a budget of 20 sensors. We were the only group who managed to accurately compute the above-mentioned objective criteria by exhaustive simulation of all possible intrusion scenarios, therefore being able to prove guarantees about our solutions. We solved this massive computational challenge, involving Terabytes of simulation data, using distributed computation. This massive amount of simulation data additionally allowed us to get empirical insights on the relationship between the different objective criteria. Our results showed significant improvement over randomly placed sensors and over sensors placed using more simplistic models. We presented our results at the Water Distribution System Analysis Symposium 2006 [Krause et al., 2006b], and our solution obtained the *top score* in the competition.

Sensor placement for seating activity recognition. Motivated by the fact that a large part of the population spends most of their day sitting on a chair (in the office, in a car, etc.) we consider the problem of seating activity recognition. Using a high fidelity sensor (one measurement per cm^2, *c.f.*, Figure 1.5), we achieve a high classification accuracy for distinguishing between ten different postures of 84% when generalizing between different people. The best performance achieved by previous work on this benchmark data set was 79%. Unfortunately, the sensor is very expensive (roughly $16,000). In addition, handling the high resolution pressure data in real-time is difficult for, e.g., application in a car. These reasons motivated us to apply our sensor placement algorithms. We used our algorithms to optimize for placements of increasing size. Surprisingly, using only 20 optimally placed sensors we achieve 82% accuracy compared to 84% when using all 4032 sensors, at a small fraction of the cost and computational load. We study this application in collaboration with the Human Computer Interaction Department at CMU and the Ford Motor Company.

Monitoring environmental phenomena using mobile sensors. In collaboration with the Center of Networked Embedded Sensing at the University of California, Los Angeles, we considered the problem of monitoring the ecological state of San Joaquin River and Lake Fulmor in James Reserve, CA. High levels

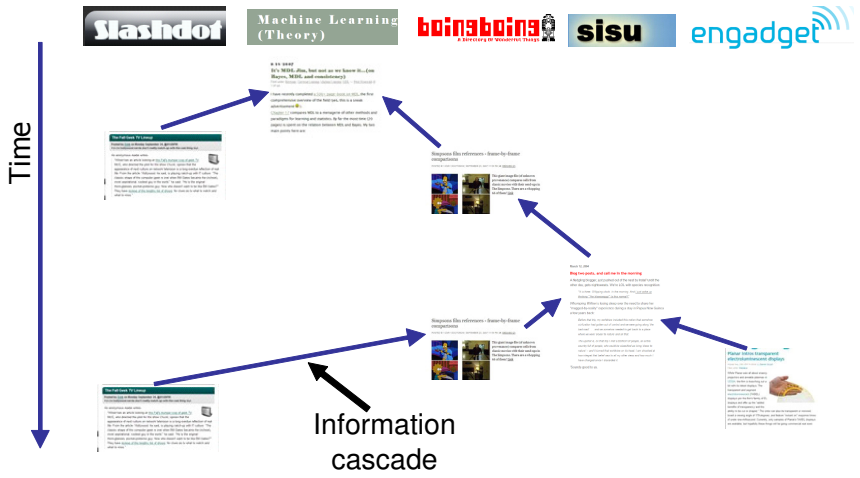


Figure 1.6: Information cascade. We would like to select blogs that participate in the large cascades as early as possible.

of pollutants, such as nitrates, can lead to the development of algal blooms. These nuisance algal blooms impair the beneficial use of aquatic systems, by blocking sunlight to underwater vegetation, consuming oxygen in the water, and producing surface scum and odors. The precise sensing of pollutants, nutrients, and oxygen levels can provide biologists with a fundamental characterization of the state of such a lake. Unfortunately, such sensors are expensive, and we are not able to cover the lake in sufficient detail with such static sensors. An alternative to deploying a static sensor network is to coordinate a set of robotic sensors to move such sensors to various locations in the lake (*c.f.*, Figure 1.1(b)). In order to enable this coordination, we apply the algorithms developed in this Thesis. We also study monitoring problems such as placing sensors for monitoring precipitation in the Pacific Northwest of the United States.

Selecting informative weblogs. All applications introduced above include some form of physical sensing. However, the algorithms developed in this Thesis apply to more general information gathering problems as well. To support this claim, we apply our algorithms to an online information network rather than a physically connected sensor network: We study the problem of selecting informative weblogs to read on the Internet. Our approach is based on the intuitive notion of an *information cascade*: A blogger writes a posting, and, after some time, other blogs link to it (*c.f.*, Figure 1.6). An information cascade is a directed acyclic graph of vertices (each vertex corresponds to a posting at some blog), where edges are annotated by the time difference between the postings. Based on this notion of an information cascade, we would like to select blogs, that detect big cascades (containing many nodes) as early as possible (i.e., we want to learn about an important event before most other readers). We show how one can formalize this intuition using a nondecreasing submodular function that measures the informativeness of a subset of all blogs. We perform extensive experimentation on a large data set containing 45,000 blogs, 10.5 million postings and 16.2 million links. After we published our results, our project website `http://www.blogcascades.org` experienced a "Slashdot effect" with over 30,000 visits. Our research was also reported on in ACM TechNews (November 08) and on MSNBC (January 08).

1.2 Summary of key contributions

The following table summarizes the key contributions of this Thesis, and their organization in the three main parts of this Thesis.

Part	Algorithms	Hardness results	Applications
Submodular sensing	CELF for approximate budgeted optimization	$(1 - 1/e)$ inapproximability of optimizing information gain	Environmental monitoring, Sensing chair, Water networks, Informative blogs
Non-greedy algorithms for complex sensing problems	SATURATE for robust sensing, PSPIEL for complex constraints, ESPASS for placement and scheduling	Inapproximability of robust optimization	Environmental monitoring, Water networks, Traffic monitoring
Sequential sensing	EEGP for active learning in Gaussian processes, VOIDP for value of information in chain models	$\mathbf{NP}^{\mathbf{PP}}$-completeness of optimizing value of information in polytree models	Environmental monitoring, Sensing for building management

1.3 Organization of this Thesis

In Part I of this Thesis, we will provide background on observation selection, and introduce relevant terminology and concepts (Chapter 2). We will also survey existing work that is relevant to most chapters of this thesis (Chapter 3). More specific discussions of work related is presented in the subsequent chapters.

In Part II, we will review the concept of submodularity (Chapter 5), and review existing results on optimizing submodular set functions. We will then establish submodularity for several important classes of observation selection problems, including problems of spatial prediction in Gaussian Processes (Chapter 6), outbreak detection (Chapter 8) and diagnosis in graphical models (Chapter 7). We also report on results obtained by applying existing algorithms for submodular function maximization.

Part III presents novel approximation algorithms for observation selection. In Chapter 10, we develop an algorithm for selecting observations under adversarial notions of robustness. In Chapter 11, we consider the problem of placing sensors subject to communication constraints. In Chapter 12, we propose ESPASS, an efficient algorithm for simultaneously optimizing sensor placements and schedules.

Part IV presents results on sequential observation selection, in the case of Gaussian Processes (Chapter 14) and chain graphical models (Chapter 15).

Lastly, in Part V we consider interesting open problems for future work (Chapter 16) and present our conclusions (Chapter 17).

In Appendix A, we review learning of nonstationary Gaussian Process models, as used in several chapters of this Thesis. Appendix B presents the proofs of all theoretical results described in this Thesis.

Chapter 2

Background on Observation Selection

In this chapter, we will provide background on observation selection problems, as well as the terminology used in this Thesis.

2.1 Model-based observation selection

In observation selection, our goal is to acquire the most useful information as cost-effectively as possible. Key questions which have to be answered in observation selection hence are the following:

> How useful would it be to me if I had a piece x of information? How expensive would it be to acquire this information?

These questions lead to further questions, such as "What does *useful* mean?", "How do we quantify *cost*?" or, "What am I likely to observe"? Such questions cannot be answered without a model of the world, and without the definition of the task we are trying to solve by acquiring the information. In this Thesis, we will adopt a Bayesian framework, and use *probabilistic models* which describe the uncertainty about the state of the world. By making observations, this uncertainty can be reduced. We can then quantify usefulness, e.g., by measuring how much this decrease in uncertainty can help us make better decisions.

2.2 Probabilistic models

Let us consider a finite set $\mathcal{V}$ indexing the possible observations we can make. Each $s \in \mathcal{V}$ corresponds, e.g., to a possible location where we can place a sensor, information we can present to a user of an online service, or a possible medical test which we can perform. With each element $s \in \mathcal{V}$, we associate a random variable $\mathcal{X}_s$. The random variables can be real-valued or discrete. For a subset $\mathcal{A} \subseteq \mathcal{V}$, with $\mathcal{A} = \{s_1, \ldots, s_k\}$, we use $\mathcal{X}_\mathcal{A}$ to refer to the random vector $(\mathcal{X}_{s_1}, \ldots, \mathcal{X}_{s_k})$. A *realization* $\mathbf{x}_\mathcal{A}$ is a vector of values $\mathbf{x}_\mathcal{A} = (x_{s_1}, \ldots, x_{s_k})$. A *probabilistic model* is a joint probability distribution (a prior distribution) over all the random variables $\mathcal{X}_\mathcal{V}$. Hence, it assigns, to each realization $\mathbf{x}_\mathcal{V}$, a probability (or density, which we assume to exist in this Thesis, in the case of continuous distributions) $P(\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V})$. Later in this Thesis, we will consider specific instances of the probability distributions P.

Upon observing a realization of a subset $\mathcal{A}$ of the variables, $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, we can use probabilistic inference [*c.f.*, Koller and Friedman, 2008] to estimate a posterior distribution over the unobserved variables, $P(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$. This posterior distribution encodes our belief about the state of the world, which we can use to make decisions.

For sets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ of observations, we use

$$\mathrm{E}[\mathcal{X}_\mathcal{A}] = \int \mathbf{x}_\mathcal{A} P(\mathbf{x}_\mathcal{A}) d\mathbf{x}_\mathcal{A}$$

to refer to the *expectation* of $\mathcal{X}_\mathcal{A}$, and

$$\mathrm{E}[\mathcal{X}_\mathcal{A} \mid \mathbf{x}_\mathcal{B}] = \int \mathbf{x}_\mathcal{A} P(\mathbf{x}_\mathcal{A} \mid \mathbf{x}_\mathcal{B}) d\mathbf{x}_\mathcal{A}$$

to refer to the *conditional expectation* of $\mathcal{X}_\mathcal{A}$ given $\mathcal{X}_\mathcal{B} = \mathbf{x}_\mathcal{A}$.

2.2.1 Examples

In *spatial prediction*, our goal is to estimate some hidden spatial process[1], such as temperature in a building, precipitation over extended geographic regions, etc. A common approach in the geostatistics community has been to think about the hidden process as a sample from a probability distribution. Here, every random variable $\mathcal{X}_s$, for example, models the temperature at some location s. One possible distribution that has been considered in the literature [*c.f.*, Deshpande et al., 2004] is to use the multivariate Gaussian distribution,

$$P(\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}) = \frac{1}{(2\pi)^{n/2}|\Sigma_{\mathcal{V}\mathcal{V}}|} e^{-\frac{1}{2}(\mathbf{x}_\mathcal{V} - \mu_\mathcal{V})^T \Sigma_{\mathcal{V}\mathcal{V}}^{-1} (\mathbf{x}_\mathcal{V} - \mu_\mathcal{V})},$$

where $\mu_\mathcal{V}$ is the mean vector and $\Sigma_{\mathcal{V}\mathcal{V}}$ is the covariance matrix. The advantage of the multivariate Gaussian distribution is that it can be compactly represented (only the mean vector $\mu_\mathcal{V} \in \mathbb{R}^n$ and the covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}} \in \mathbb{R}^{n \times n}$ have to be specified or estimated from data). It also allows efficient inference, i.e., computation of conditional distributions $P(\mathcal{X}_\mathcal{A} \mid \mathcal{X}_\mathcal{B})$.

While useful for many spatial modeling problems [*c.f.*, Deshpande et al., 2004], in some applications, the Gaussian distribution does not model the phenomena of interest well. When modeling *medical diagnosis* (as done, e.g., by Bayer-Zubek [2004], Heckerman et al. [1993]) for example, the random variables describe the absence or presence of a particular disease, as well as the outcomes of a collection of possible tests which can be performed. In such settings the variables of interest are often discrete, or take only positive or bounded continuous values. Completely specifying a joint distribution over n binary random variables requires $2^n - 1$ free parameters, which is intractable for most practical problems. *Probabilistic graphical models* have been developed as a powerful tool to overcome such challenges. In these models, the joint distribution is modeled as

$$P(\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}) = \frac{1}{Z} \prod_{j=1}^{m} \psi_j(\mathcal{X}_{\mathcal{B}_j}),$$

where Z is a normalization constant, and $\psi_1, \ldots, \psi_m$ are nonnegative *potential functions* defined over subsets of variables $\mathcal{B}_1, \ldots, \mathcal{B}_j \subseteq \mathcal{V}$ respectively. The advantage of such models is that they often allow to compactly represent complex joint distributions. Moreover, for certain classes of graphical models, efficient inference procedures are available. Instead of giving a general introduction to the topic of

[1] Many of the results presented in this Thesis apply to spatio-temporal processes as well.

probabilistic graphical models, we will refer to Koller and Friedman [2008], and only introduce specific concepts as they become necessary in latter chapters of this Thesis.

In *outbreak detection*, we think about a phenomenon spreading smoothly over some space. One example, which we consider in this Thesis, are contamination events spreading over a municipal water distribution network. In this case, a set of differential equations can be used in order to model the spreading of a contaminant through the network. If we consider random (e.g., accidental) contaminant introductions into the network, then the contaminant concentration $\mathcal{X}_{s,t}$ at each node s and point in time t is again modeled probabilistically. Here, we can quantify usefulness by measuring, e.g., the time a hypothetical sensor network deployment takes until the outbreak is detected.

2.3 Quantifying sensing quality

In order to decide which observations to select, we need to define a notion of *sensing quality* that allows us to prefer one set of observations over another. We consider the following formulation, which goes back to Howard [1966] and Lindley and Smith [1972]. Hereby, we consider a utility function $u(\mathbf{x}_\mathcal{V}, \mathcal{A})$ which depends on the state of the world, $\mathbf{x}_\mathcal{V}$, and the chosen observations, $\mathcal{A} \subseteq \mathcal{V}$. The sensing quality of a set of observations $\mathcal{A}$ is then the expected utility

$$F(\mathcal{A}) = \int p(\mathbf{x}_\mathcal{V}) u(\mathbf{x}_\mathcal{V}, \mathcal{A}) d\mathbf{x}_\mathcal{V}, \tag{2.1}$$

which is a *set function* $F : 2^\mathcal{V} \rightarrow \mathbb{R}$, mapping subsets $\mathcal{A}$ of observations to the real numbers. Our goal is to choose a set $\mathcal{A}$ such that this set function is maximized. Without loss of generality, we will require that $u(\mathbf{x}_\mathcal{V}, \emptyset) = 0$, and hence $F(\emptyset) = 0$, i.e., no observations provide no utility. Lindley [1956] proposed this approach for the special case of

$$u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = H(\Theta) - H(\Theta \mid \mathbf{x}_\mathcal{A}) = -\int p(\theta) \log_2 p(\theta) d\theta + \int p(\theta \mid \mathbf{x}_\mathcal{A}) \log_2 p(\theta \mid \mathbf{x}_\mathcal{A}) d\theta,$$

i.e., the most informative set of observations $\mathcal{A}$ is the one which maximizes the reduction of Shannon entropy [Shannon, 1948] about some unknown set of parameters. Here, and in the following, we will use the notation Θ to refer to a designated subset of the random variables $\mathcal{X}_\mathcal{V}$, with instantiations $\Theta = \theta$. With this notation, Lindley [1956] considered the case of $F(\mathcal{A}) = H(\Theta) - H(\Theta \mid \mathcal{X}_\mathcal{A})$.

The formulation (2.1) is very general; it makes the only assumption, that there is some known prior distribution $P(\mathcal{X}_\mathcal{V})$ about the state of the world. This distribution can be estimated from data or specified as a belief in the subjective Bayesian sense. However, in practice it is sometimes difficult to determine this distribution. In Chapter 10, we will hence consider problems where even this assumption is relaxed.

By choosing different forms for $u(\mathbf{x}_\mathcal{V}, \mathcal{A})$, we can model very different classes of observation selection problems. In the following, we give a few examples. In Part II of this Thesis, we will examine particular utility functions u and the associated sensing problems in more detail.

2.3.1 Shannon entropy and mutual information

One sensing quality function that is commonly used in practice [*c.f.*, O'Hagan, 1978, Shewry and Wynn, 1987], is the *entropy criterion*, where

$$u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = -\log_2 p(\mathbf{x}_\mathcal{A}).$$

In this case, the sensing quality $F(\mathcal{A})$ is the Shannon entropy, $F(\mathcal{A}) = H(\mathcal{X}_\mathcal{A})$ of the random vector $\mathcal{X}_\mathcal{A}$. Hence, when maximizing this criterion, the goal is to select observations that are as *uncertain* as possible.

Often, in practice, the goal of selecting observations is to reduce the uncertainty in unobserved variables. For example, in a sensor placement example, our goal would be to select sensor locations which are maximally informative about the unobserved locations. Hereby, we can, similarly to the case considered by Lindley [1956], be interested in reducing the uncertainty about a specified set $\mathcal{B} \subseteq \mathcal{V}$ of variables of interest, corresponding to, e.g., the temperature at a set of points of interest. In this case, the choice

$$u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = -\log_2 p(\mathbf{x}_\mathcal{B}) + \log_2 p(\mathbf{x}_\mathcal{B} \mid \mathbf{x}_\mathcal{A})$$

leads to the objective function

$$F(\mathcal{A}) = H(\mathcal{X}_\mathcal{B}) - H(\mathcal{X}_\mathcal{B} \mid \mathcal{X}_\mathcal{A}) = I(\mathcal{X}_\mathcal{B}; \mathcal{X}_\mathcal{A}). \tag{2.2}$$

Hereby, we use $I(\mathcal{X}_\mathcal{B}; \mathcal{X}_\mathcal{A}) = I(\mathcal{X}_\mathcal{A}; \mathcal{X}_\mathcal{B})$ to refer to the *mutual information* between the observation sets $\mathcal{A}$ and $\mathcal{B}$ (*c.f.*,Cover and Thomas 1991). This criterion is known under the name of *Bayesian D-optimality* [*c.f.*, Chaloner and Verdinelli, 1995]. We consider this criterion in the context of graphical models in detail in Chapter 7.

If we are interested in predicting, e.g., the temperature at all unobserved locations, we can set $\mathcal{B} = \mathcal{V} \setminus \mathcal{A}$, i.e., the locations of interest depend on the observed locations $\mathcal{A}$. In this setting, our goal is to maximize

$$F(\mathcal{A}) = H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A}).$$

This criterion has been first considered in the context of spatial prediction by Caselton and Zidek [1984], and we will study this criterion in detail in Chapter 6.

2.3.2 Predictive variance

Another important sensing quality function for spatial prediction is the reduction of predictive variance. Hereby, for some set $\mathcal{B}$ of reference points, we choose

$$u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = \sum_{s \in \mathcal{B}} \mathrm{Var}(\mathcal{X}_s) - \mathrm{Var}(\mathcal{X}_s \mid \mathbf{x}_\mathcal{A}),$$

where

$$\mathrm{Var}(\mathcal{X}_s \mid \mathbf{x}_\mathcal{A}) = \mathrm{E}\big[(\mathcal{X}_s - \mathrm{E}[\mathcal{X}_s \mid \mathbf{x}_\mathcal{A}])^2 \mid \mathbf{x}_\mathcal{A}\big]$$

denotes the predictive variance of $\mathcal{X}_s$ after observing $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$. This criterion is also known under the name of *Bayesian A-optimality* (*c.f.*, Chaloner and Verdinelli 1995). We will consider this criterion in Chapters 10 and 12.

2.3.3 Decision theoretic value of information

Often, the goal of observation selection is not to minimize the uncertainty about a set of variables of interest. Instead, observations are acquired in order to facilitate a decision making process.

A general, decision-theoretic formulation was proposed by Howard [1966]. In this setting, there is some set of parameters Θ which encode the information relevant for making a decision. For example, in a medical domain, Θ would encode the presence or absence of a particular disease. Observations $\mathcal{A} \subseteq \mathcal{V}$ can be selected from a set of possible tests $\mathcal{V}$. Upon observation of $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, a decision d is made from a set of possible decisions $\mathcal{D}$. If the true state of the world is $\Theta = \theta$, then this decision would result in a utility $g(d, \theta)$. For example, if θ encodes the presence of a disease, then typically the utility of choosing the "treat" decision would be higher than that for not treating. Since θ is unobserved, one would choose the decision d maximizing the expected utility, i.e., select

$$d^* = \operatorname*{argmax}_{d \in \mathcal{D}} \int P(\theta \mid \mathbf{x}_\mathcal{A}) g(d, \theta) d\theta.$$

Typically, g also encodes some kind of cost associated with obtaining the information. Hence, e.g., if the information obtained does not help the diagnosis much, but the cost of the applied medical procedure is high, the value of information can be negative.

We can encode this decision making process by encoding u in (2.1) as

$$u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = \max_{d \in \mathcal{D}} \int P(\theta \mid \mathbf{x}_\mathcal{A}) g(d, \theta) d\theta.$$

The goal would then be to select observations $\mathcal{A} \subseteq \mathcal{V}$ such that the *decision theoretic value of information*

$$F(\mathcal{A}) = \int P(\mathbf{x}_\mathcal{V}) u(\mathbf{x}_\mathcal{V}, \mathcal{A}) d\mathbf{x}_\mathcal{V} = \int \max_{d \in \mathcal{D}} \int g(d, \theta) P(\theta \mid \mathbf{x}_\mathcal{A}) P(\mathbf{x}_\mathcal{A}) d\theta d\mathbf{x}_\mathcal{A} \tag{2.3}$$

is maximized. We study this objective in detail in Chapter 15.

2.3.4 Penalty reduction for outbreak detection

The last class of objective functions we will consider is related to the following application. We consider dynamic phenomena, where an event (e.g., contaminant introduction in a water network) originates from a node $s' \in \mathcal{V}$ of a network $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and spreads through the network, affecting other nodes (e.g., by flow through pipes). Eventually, it reaches a monitored node $s \in \mathcal{A} \subseteq \mathcal{V}$ (i.e., pipe junction we instrument with sensors), and gets detected. During the time the outbreak is not detected, some negative impact is incurred, which we want to minimize. We can model this setting in the following way.

Let $\mathcal{X}_{s,t}$ describe the state (i.e., contaminant concentration; 0 if not contaminated) of node $s \in \mathcal{V}$ at time t. Hereby, $t \in \mathcal{T}$ ranges over a discrete sequence of time points $\mathcal{T}$. For each possible state of the world, we evaluate a penalty function $\pi(\mathbf{x}_\mathcal{V}, \mathcal{A})$ which depends on the state of the network $\mathbf{x}_{\mathcal{V},\mathcal{T}}$ at all nodes and all times, as well as the placement $\mathcal{A}$. This class of penalty functions is very general. For example, in the water monitoring example, it can estimate the expected population affected by consuming contaminated water until the placed sensor network $\mathcal{A}$ detected the contamination at time t (i.e., $\mathbf{x}_{s,t} > 0$ for some monitored node $s \in \mathcal{A}$ and time t). We discuss concrete examples of penalty functions in Chapter 8. Based on this notion of a penalty function π, we can then define the utility by the *penalty reduction*:

$$u(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \mathcal{A}) = \pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \emptyset) - \pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \mathcal{A}).$$

The penalty reduction satisfies the property that selecting no sensors gives zero sensing quality. If we assume a probability distribution $P(\mathbf{x}_{\mathcal{V},\mathcal{T}})$ over the state of contamination, the problem of minimizing the

expected negative impact is hence equivalent to the problem of maximizing

$$F(\mathcal{A}) = \int P(\mathbf{x}_{\mathcal{V},\mathcal{T}}) u(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \mathcal{A}) d\mathbf{x}_{\mathcal{V},\mathcal{T}} = const - \int P(\mathbf{x}_{\mathcal{V},\mathcal{T}}) \pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \mathcal{A}) d\mathbf{x}_{\mathcal{V},\mathcal{T}}.$$

2.4 Quantifying cost

In practice, the observations $\mathcal{A}$ are typically expensive (e.g., in terms of sensor hardware or deployment cost), and we only have limited resources we can spend on making observations. While we could, in principle, incorporate the cost directly into the utility $u(\mathbf{x}_{\mathcal{V}}, \mathcal{A})$, we choose to model the cost $C(\mathcal{A})$ as a separate set function

$$C : 2^{\mathcal{V}} \rightarrow \mathbb{R},$$

mapping sets of observations $\mathcal{A} \subseteq \mathcal{V}$ to the real numbers. Without loss of generality, we will require that $C(\emptyset) = 0$, i.e., selecting no observations does not cost anything.

In the following, we consider several classes of possible cost functions.

2.4.1 Additive cost functions

In the case of *additive cost* functions, which we will mainly consider in this Thesis (with the exception of Chapter 11), we assign a cost $c(s) \in \mathbb{R}$ to each possible observation $s \in \mathcal{V}$, and define the cost of a set $C(\mathcal{A})$ to be

$$C(\mathcal{A}) = \sum_{s \in \mathcal{A}} c(s).$$

For example, if we only care about the number of observations made (e.g., sensors placed), then we would set $c(s) = 1$, and hence $C(\mathcal{A}) = |\mathcal{A}|$.

However, in practice, different observations often have different costs. For example, when placing a sensor network, some locations might be more easily accessible than others, hence their deployment costs would vary. Similarly, we might have different types of sensors – some expensive high accuracy sensors, as well as cheaper, but lower-fidelity sensing devices. In these cases, the per-sensor cost function $c(s)$ would not be constant.

2.4.2 Subadditive cost functions

Often in practice, observations get cheaper, the more observations we have already selected. For example, if we want to send out a robot to make measurements for us, a natural cost function is the length of the path (or time) the robot takes to collect all observations. Consider the case presented in Figure 2.1, where the robot, starting at location s, can decide to visit and make observations at locations v_1, v_2 or v_1 and v_2. Since the robot does not have to go back to its starting point between each observation, the cost of observing $C(\{v_1, v_2\}) = 5.5$ is less than the sum $C(\{v_1\}) + C(\{v_2\}) = 6$ of observing v_1 and v_2 separately.

Similarly, in a medical scenario, performing a set of tests might be cheaper than the sum of the costs of the tests taken separately. For example, whether we want to study only one parameter (like the hemoglobin

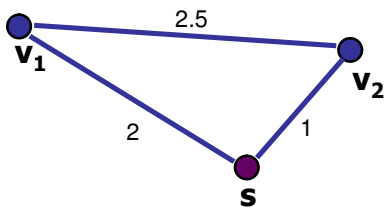


Figure 2.1: Cost of visiting v_1 and v_2 together is cheaper than the sum of the costs of visiting them separately.

value) or multiple parameters of a patient's blood, we nevertheless incur the cost of physically acquiring the blood sample. In this case, the cost of multiple observations can be far less than the sum of the individual costs.

Another example of such a "buy in a bulk" property arises when we consider computational cost for computing features as a preprocessing step to data analysis. Whether we we are interested in only a small part of a frequency spectrum of a signal, or the entire spectrum, we nevertheless typically have to perform the FFT of the entire signal.

In all these examples, the cost function is *subadditive*: For any two sets of observations, $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ it holds that

$$C(\mathcal{A} \cup \mathcal{B}) \leq C(\mathcal{A}) + C(\mathcal{B}).$$

In Chapter 11, we present an algorithm, PSPIEL, that can solve sensing problems even under some complex cost functions C.

2.4.3 Superadditive cost functions

In some application, the opposite property of subadditivity holds: A cost function C is called *superadditive* if for all sets $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ it holds that

$$C(\mathcal{A} \cup \mathcal{B}) \geq C(\mathcal{A}) + C(\mathcal{B}).$$

A typical example is the computational cost of running an algorithm. Here, the set $\mathcal{V}$ of possible observations corresponds to some features/dimensions of the data, and $C(\mathcal{A})$ models the computational cost required to process the data. Many algorithms scale superlinearly with the dimensionality of the data, i.e., $C(\mathcal{A}) = h(|\mathcal{A}|)$ where h is a monotonically increasing convex function, e.g., $h(n) = n^3$ for a cubic-time algorithm.

2.5 Formulation of sensing problems

Using the concepts of sensing quality and cost functions, in this section we will consider various kinds of sensing problems.

2.5.1 Constrained maximization vs. coverage

Our general goal is to select observations $\mathcal{A} \subseteq \mathcal{V}$ such that the sensing quality $F(\mathcal{A})$ is maximized, and simultaneously the cost $C(\mathcal{A})$ is minimized. Technically, this is a *multicriterion optimization problem* (*c.f.*, Boyd and Vandenberghe 2004), and two sets $\mathcal{A}$ and $\mathcal{B}$ can be *incomparable*, i.e., $F(\mathcal{A}) > F(\mathcal{B})$ but $C(\mathcal{A}) > C(\mathcal{B})$.

One possibility is to consider a *budget* $B \in \mathbb{R}$, and then try to find the set $\mathcal{A}^*$ such that

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A} \subseteq \mathcal{V}} F(\mathcal{A}) \text{ subject to } C(\mathcal{A}) \leq B, \tag{2.4}$$

i.e., to find the most informative set of observations subject to a constraint on the cost. We call Problem (2.4) a *maximization problem*.

Similarly, we can specify a *quota* $Q \in \mathbb{R}$, and then try to find the set $\mathcal{A}^*$ such that

$$\mathcal{A}^* = \operatorname*{argmin}_{\mathcal{A} \subseteq \mathcal{V}} C(\mathcal{A}) \text{ subject to } F(\mathcal{A}) \geq Q, \tag{2.5}$$

i.e., try to find the cheapest solution, which achieves a specified level Q of information. Problem (2.5) is called a *covering problem*.

An alternative approach is to use *scalarization* [*c.f.*, Boyd and Vandenberghe, 2004]. Here, we would specify a cost-to-sensing quality conversion factor λ, and then try to find the set $\mathcal{A}^*$ such that

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A} \subseteq \mathcal{V}} F(\mathcal{A}) - \lambda C(\mathcal{A}), \tag{2.6}$$

i.e., try to find the solution with maximum overall value. Problem (2.6) is called a *scalarized (maximization) problem*. In fact, it can be shown that if we are able to solve problems (2.4) and (2.5) for arbitrary choices of B and Q, we can obtain all optimal solutions of (2.6), but not necessarily vice versa [Boyd and Vandenberghe, 2004, Papadimitriou and Yannakakis, 2000]. Hence, in this Thesis, we concentrate on Problems (2.4) and (2.5).

Instead of specifying feasibility of observation sets $\mathcal{A}$ by requiring that $C(\mathcal{A}) \leq B$ (2.4) or $F(\mathcal{A}) \geq Q$ (2.5), we can consider more general classes of constraints, by defining a collection of *feasible sets* $\mathfrak{A} \subseteq 2^{\mathcal{V}}$, and require that the chosen sets $\mathcal{A}$ satisfy $\mathcal{A} \in \mathfrak{A}$. In this case, we would like to solve

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A}} F(\mathcal{A}) \text{ such that } \mathcal{A} \in \mathfrak{A}. \tag{2.7}$$

For example, while we might be interested in predicting a spatial phenomenon at all locations $\mathcal{V}$, not all of these locations might be accessible for sensor placement. In this case, we could have constraints restricting the possible sensor placements $\mathcal{A}$ to only consist of accessible locations. Incorporating a particular class of constraints (called *matroid constraints*, *c.f.*, Section 5.6) will allow us to develop the ESPASS algorithm for simultaneous placement and scheduling of sensors (Chapter 12).

2.5.2 A priori vs. sequential design

So far, we considered problems, where our goal is to select a *set* of observations $\mathcal{A} \subseteq \mathcal{V}$. In these problems, we commit to the chosen observations a priori, i.e., before getting to see the observed values.

These *a priori* selection problems are appropriate in cases where we, e.g., place a network of sensors, or design an experiment.

In some applications however it makes sense to *sequentially* select observations, i.e., decide on the next observation based on values which have been observed in the past. We can extend the concepts described in this section to this more general setting as follows. Instead of reasoning about *sets*, we reason about *policies* (sometimes called *conditional plans*). A policy

$$\pi : \mathbb{R}^{|\mathcal{V}|} \to 2^{\mathcal{V}}$$

is a set-valued random function, mapping the state of the world $\mathbf{x}_\mathcal{V}$ to a subset $\mathcal{A} \subseteq \mathcal{V}$. Hereby, $\pi(\mathbf{x}_\mathcal{V})$ is the set of observations chosen by the policy if the world is in state $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$.

For two policies π and π', we say $\pi \subseteq \pi'$ if for all $\mathbf{x}_\mathcal{V}$ it holds that $\pi(\mathbf{x}_\mathcal{V}) \subseteq \pi'(\mathbf{x}_\mathcal{V})$, i.e., in any state of the world, π selects a subset of the observations of policy π'. We say $|\pi| = m$ if for all $\mathbf{x}_\mathcal{V} \in \mathcal{V}$ it holds that $|\pi(\mathbf{x}_\mathcal{V})| = m$, and $|\pi| \leq m$ if for all $\mathbf{x}_\mathcal{V} \in \mathcal{V}$ it holds that $|\pi(\mathbf{x}_\mathcal{V})| \leq m$.

We will require the policy π to be *sequential*[2], i.e., be the output of a deterministic algorithm, which, after having observed a set of values $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, decides which observation to make next solely based on the observations made so far. We can formalize this notion of sequential selection in the following way:

Definition 2.1. *A policy π, $|\pi| \leq m$ is called* sequential*, if the following two conditions hold:*

- *There exist a sequence of policies, $\emptyset \subseteq \pi_1 \subseteq \cdots \subseteq \pi_m = \pi$ such that for all $1 \leq i \leq m$ it holds that $|\pi_i| \leq i$. In this case, we write $\pi_{1:i}$ to refer to the i-step policy π_i. We use the convention $\pi_{1:0} = \emptyset$.*
- *Let $0 \leq i < m$, $\mathcal{A} \subseteq \mathcal{V}$ and let $\mathbf{x}_\mathcal{V}$ and $\mathbf{x}'_\mathcal{V}$ be two states such that $\mathbf{x}_\mathcal{A} = \mathbf{x}'_\mathcal{A}$ (i.e., $\mathbf{x}_\mathcal{V}$ and $\mathbf{x}'_\mathcal{V}$ agree on $\mathcal{A}$). Furthermore suppose that $\pi_{1:i}(\mathbf{x}_\mathcal{V}) = \mathcal{A}$. Then it must hold that $\pi_{1:i+1}(\mathbf{x}_\mathcal{V}) = \pi_{1:i+1}(\mathbf{x}'_\mathcal{V})$.*

The second condition in above definition implies that if two states $\mathbf{x}_\mathcal{V}$ and $\mathbf{x}'_\mathcal{V}$ of the world agree on a set $\mathcal{A}$, and $\pi_{1:i}$ selects this set $\mathcal{A}$ in state $\mathbf{x}_\mathcal{V}$, then the next element $s = \pi_{1:i+1}(\mathbf{x}_\mathcal{V}) \setminus \pi_{1:i}(\mathbf{x}_\mathcal{V})$ is uniquely determined, and its choice cannot depend on any unobserved part of the state, i.e., any x_s for $s \in \mathcal{V} \setminus \mathcal{A}$.

Using this notation, we can define a sequential notion of sensing quality as

$$F(\pi) = \int p(\mathbf{x}_\mathcal{V}) u(\mathbf{x}_\mathcal{V}, \pi(\mathbf{x}_\mathcal{V})) d\mathbf{x}_\mathcal{V}. \tag{2.8}$$

Note, that in contrary to (2.1), the sensing quality u now depends on the selected observations $\pi(\mathbf{x}_\mathcal{V})$. Setting $\pi(\mathbf{x}_\mathcal{V}) = \mathcal{A}$ for all $\mathbf{x}_\mathcal{V}$ recovers (2.1) as a special case.

Similarly, we can define an expected cost as

$$C(\pi) = \int p(\mathbf{x}_\mathcal{V}) C(\pi(\mathbf{x}_\mathcal{V})) d\mathbf{x}_\mathcal{V}. \tag{2.9}$$

Based on these notions of expected sensing quality and cost, we can consider the same optimization problems as described in Section 2.5.1. For example, we can consider a sequential maximization problem of the form

$$\pi^* = \operatorname*{argmax}_\pi F(\pi) \text{ subject to } C(\pi) \leq B.$$

[2]Technically, we also require the mapping to be *measurable*, i.e., for any subset $\mathcal{A} \subseteq \mathcal{V}$, $\pi^{-1}(\mathcal{A})$ has to be a subset of $\mathbb{R}^{|\mathcal{V}|}$ which is measurable [*c.f.*, Dudley, 2003] with respect to the probability distribution $P(\mathcal{X}_\mathcal{V})$.

We will consider sequential design in Part IV of this Thesis. There we will consider particular instances of policies π, and provide more intuition about the concept.

2.6 Some background on approximation algorithms

A main focus of this Thesis will be to develop algorithms for solving sensing problems such as those discussed in Section 2.5.1. Most of these optimization problems are computationally complex, so that we cannot expect to find the optimal solutions for general problems. Instead, our goal will be to develop efficient algorithms, for which we can prove that the obtained solutions will be close to the optimal solution.

Any instance of Problem (2.4) (the following definitions for Problem (2.5) are analogous) is defined by a ground set $\mathcal{V}$, sensing quality function F, cost function C and budget B. A solution $\mathcal{A}$ which satisfies the constraint $C(\mathcal{A}) \leq B$ is called feasible. Since $\mathcal{V}$ is finite, there is at least one optimal solution $\mathcal{A}^*$ for which $F(\mathcal{A}^*) \geq F(\mathcal{A})$ for all feasible solutions $\mathcal{A}$. This solution $\mathcal{A}^*$ obtains the optimal value $OPT = F(\mathcal{A}^*)$. In this Thesis, we will consider restricted sets $\mathcal{I}$ of problem instances. For example, $\mathcal{I}$ could consider the set of problem instances, where $F(\mathcal{A})$ is the mutual information, as defined in Section 2.3.1, for a restricted class of probabilistic models (such as Gaussian Processes), $C(\mathcal{A}) = |\mathcal{A}|$ is the cardinality function, and B is a nonnegative integer. Each instance $I \in \mathcal{I}$ has an optimal value OPT_I. Similarly, a (deterministic) algorithm a will compute a solution $a(I) \subseteq \mathcal{V}$ for each instance $I \in \mathcal{V}$. Let $\alpha : \mathcal{I} \to (0, 1]$ and $\epsilon : \mathcal{I} \to [0, \infty)$ be functions that map problem instances $I \in \mathcal{I}$ to real numbers.

Definition 2.2. *We call a an* approximation algorithm *with multiplicative error α and additive error ϵ for problem instances $\mathcal{I}$ if for all $I \in \mathcal{I}$ it holds that*

$$F(a(I)) \geq \alpha(I) OPT_I - \epsilon(I).$$

Often, we require α to depend the "size" of the problem instance, for example, on $|\mathcal{V}|$. If α is constant, a is called a constant factor approximation algorithm. A more formal definition of approximation algorithms can be found in [Vazirani, 2003].

Chapter 3

Survey of Related Work

There is a large body of work related to the selection of observations for the purpose of coverage, prediction and for optimizing value of information. Variations of this problem appear in spatial statistics, active learning, experimental design, decision analysis and operations research. Generally, the methods define an objective function (Section 3.1), such as area coverage, predictive accuracy or detection time, and then apply a computational procedure (Section 3.2) to optimize this objective function. In this chapter, we review related work that is relevant to most chapters of this thesis. In each subsequent chapter, we will review additional related work that is specific to the respective chapter.

3.1 Objective functions

In the context of sensing for spatial prediction (which is an important part of this Thesis), we distinguish *geometric* and *model-based* approaches, which differ according to their assumptions made about the phenomenon to be monitored.

3.1.1 Geometric approaches

Geometric approaches do not build a probabilistic model of the underlying process, but instead use geometric properties of the space in which the process occurs. The most common approaches for optimizing sensing using geometric criteria assume that sensors have a fixed region [*c.f.*, Bai et al., 2006, Gonzalez-Banos and Latombe, 2001, Hochbaum and Maas, 1985]. These regions are usually convex or even circular. Furthermore, it is assumed that everything within this region can be perfectly observed, and everything outside cannot be measured by the sensors.

In the case where the sensing area is a disk (the *disk model*), Kershner [1939] has shown that an arrangement of the sensors in the centers of regular hexagons is asymptotically optimal, in the sense that a given set is fully covered by uniform disks. In Chapter 6, we experimentally show that when we apply the disk model to nonstationary phenomena, which frequently arise on spatial prediction problems, the geometric disk model approach leads to worse sensing configurations in terms of prediction accuracy, when compared to probabilistic model-based approaches.

If many sensing locations can be chosen, then one can optimize the deployment *density* instead of selecting a discrete set of individual sensing locations [Toumpis and Gupta, 2005]. The sensing locations are then assumed to be randomly sampled from this distribution. In the applications we consider, sensors and measurements are quite expensive, and optimal placement or selection of a small set of them is desired.

3.1.2 Probabilistic model-based approaches

In this Thesis, we consider model-based approaches, in which one takes a probabilistic model of the world (*c.f.*, Section 2.1) and selects observations which facilitate inferences in and decisions based on this model.

Many different objective functions have been proposed for model-based observation selection. In the statistics community, classical and Bayesian experimental design focused on the question of selecting observations to maximize the quality of parameter estimates or predictions, with a special focus on linear models [*c.f.*, Atkinson, 1988, Lindley, 1956]. In spatial statistics, information-theoretic measures, notably entropy, have been frequently used [Caselton and Hussain, 1980, Caselton and Zidek, 1984, Caselton et al., 1992, Federov and Mueller, 1989, Guttorp et al., 1992, Shewry and Wynn, 1987, Wu and Zidek, 1992]. These objectives minimize the uncertainty in the spatial prediction, after the observations are made.

Classical experimental design criteria

In the statistics literature, the problem of optimal experimental design has been extensively studied [*c.f.*, Atkinson, 1988, 1996, Boyd and Vandenberghe, 2004, Pukelsheim, 1987]. The problem commonly addressed there is to estimate the parameters θ of a function,

$$y = f_\theta(\mathbf{x}) + w,$$

where w is normally distributed measurement noise with zero mean and variance σ^2, y a scalar output and $\mathbf{x}$ a vectorial input. The assumption is, that the input $\mathbf{x}$ can be selected from a menu of design options, $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$. Each input corresponds to a possible experiment which can be performed. When using experimental design for sensor placement, one $\mathbf{x}$ would be associated with each sensing location, y would be the measurement at the location, and θ would correspond to the values of the phenomenon at the unobserved locations. Usually, the assumption is that f_θ is linear, i.e. $y = \theta^T\mathbf{x} + w$.

For the linear model $y = \theta^T\mathbf{x} + w$, if all n observations were available, then

$$\text{Var}(\hat{\theta}) = \sigma^2(X^TX)^{-1} \tag{3.1}$$

$$\text{Var}(\hat{y}_i) = \sigma^2\mathbf{x}_i^T(X^TX)^{-1}\mathbf{x}_i, \tag{3.2}$$

where X is the design matrix, which consists of the inputs $\mathbf{x}_1, \ldots, \mathbf{x}_n$ as its rows. We can see that the variance of both the parameter estimate $\hat{\theta}$ and the predictions $\hat{y}_i$ depends on the matrix $M = (X^TX)^{-1}$, which is called the inverse moment matrix. If this matrix is "small", then the parameter estimates and predictions will be accurate. A design consists of a selection $\mathcal{A}$ of the inputs (with repetitions allowed). We write $\mathcal{X}_\mathcal{A}$ to denote the selected experiments, and $M_\mathcal{A}$ for the corresponding inverse moment matrix. Classical experimental design considers different notions of "smallness" for this inverse moment matrix $M_\mathcal{A}$; D-optimality refers to the determinant, A-optimality to the trace and E-optimality to the spectral

radius (the maximum eigenvalue). There are several more scalarizations of the inverse moment matrix, and they are commonly referred to as "alphabetical" optimality criteria. We will empirically compare our approaches to these classical experimental design criteria in Chapter 6.

Equation (3.2) shows that the distribution of the test data is not taken into account, when attempting to minimize the inverse moment matrix $M_{\mathcal{A}}$. Yu et al. [2006] extend classical experimental design to the transductive setting, which takes the distribution of test data into account. The information-theoretic approaches, described below, also directly take into account the unobserved locations, as they minimize the uncertainty in the posterior $P(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}})$.

Bayesian design criteria

Classical experimental design is a Frequentist approach, which attempts to minimize the estimation error of the maximum likelihood parameter estimate. If one places a prior on the model parameters, one can formalize a Bayesian notion of experimental design. We introduced this Bayesian formalism (which was originally proposed by Lindley [1956]) in Section 2.3. By choosing different sensing quality functions $u(\mathbf{x}_{\mathcal{V}}, \mathcal{A})$, Bayesian versions of A-, D-, and E- optimality can be developed [Chaloner and Verdinelli, 1995].

Usually, Bayesian experimental design considers the task of parameter estimation [Paninski, 2005, Sebastiani and Wynn, 2000, Ylvisaker, 1975]. In this setting, a subset of the random variables $\Theta \subseteq \mathcal{V}$, called the parameters, is of special interest. Lindley [1956] suggested using Shannon information as sensing quality function, $u(\mathbf{x}_{\mathcal{V}}, \mathcal{A}) = H(\Theta) - H(\Theta \mid \mathbf{x}_{\mathcal{A}})$, which is equivalent to maximizing the expected Kullback-Leibler divergence between the posterior and prior over the parameters.

If we consider distributions $P(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}})$ over all unobserved variables $\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}$ instead of a subset of parameters $P(\Theta)$, this formulation exactly recovers the mutual information of Caselton and Zidek [1984] (as introduced in Section 2.3 and, in more detail, in Chapter 6) between the observed and unobserved variables.

Information-theoretic criteria

The special case of Bayesian experimental design, where an information-theoretic functional (such as entropy or mutual information) is used as a sensing quality function, and where the predictive uncertainty in the unobserved variables is concerned (as in Equation (2.2)) is of special importance for spatial prediction.

Such information-theoretic criteria have been used as design criteria in a variety of fields and applications. Maximizing the posterior entropy $H(\mathcal{X}_{\mathcal{A}})$ of a set of observations, as discussed in Section 2.3, has been used in the design of computer experiments [Currin et al., 1991, Sacks et al., 1989], function interpolation [O'Hagan, 1978] and spatial statistics [Shewry and Wynn, 1987]. This criteria is sometimes also referred to as D-optimality, since the scalarization of the posterior variance in the spatial literature and the scalarization of the parameter variance in classical experimental design both involve a determinant.

Maximizing mutual information between sets of random variables has a long history of use in statistics [Bernardo, 1979, Lindley, 1956], machine learning [Luttrell, 1985, MacKay, 1992]. The mutual information between a set of chosen variables and its complement, $I(\mathcal{X}_{\mathcal{A}}; \mathcal{X}_{\mathcal{V}\setminus\mathcal{A}})$ (as introduced in Section 2.3), has been used in spatial statistics [Caselton and Zidek, 1984, Caselton et al., 1992]. Mutual information

requires an accurate estimate of the joint model $P(\mathcal{X}_\mathcal{V})$, while the entropy criterion only requires an accurate estimate at the selected locations, $P(\mathcal{X}_\mathcal{A})$. Caselton et al. [1992] argue that latter is easier to estimate from a small amount of data, thus arguing against mutual information. Nowadays, however, effective techniques for learning complex nonstationary spatial models are available (*c.f.*, Chapter 6 for more details), thus mitigating these concerns and enabling the optimization of mutual information.

Decision theoretic value of information

Decision-theoretic value of information has been frequently used for principled information gathering [*c.f.*, Heckerman et al., 1993, Howard, 1966, Lindley, 1956], and popularized in decision analysis in the context of influence diagrams [Howard and Matheson, 1984]. In a sense, value of information problems are special cases of Bayesian experimental design problems, where the prior distribution has a particular structure, typically given by a graphical model as considered in Chapters 7 and 15. While value of information problems have been studied in a variety of areas, the predominant approach is to use myopic, i.e., greedy approaches for selecting observations [Dittmer and Jensen, 1997, Kapoor et al., 2007, Scheffer et al., 2001, van der Gaag and Wessels, 1993]. In Chapter 15, we will present an efficient, optimal, nonmyopic algorithm for optimizing value of information in an interesting class of graphical models.

Bandit problems and exploration / exploitation

An important class of sequential value of information problems is the class of *Bandit problems*. In the classical k-armed bandit problem, as formalized by Robbins [1952], a slot machine is given with k arms. A draw from arm i results in a reward with success probability p_i that is fixed for each arm, but different (and independent) across each arm. When selecting arms to pull, an important problem is to trade off exploration (i.e., estimation of the success probabilities of the arms) and exploitation (i.e., repeatedly pulling the best arm known so far). A celebrated result by Gittins and Jones [1979] shows that for a fixed number of draws, an optimal strategy can be computed in polynomial time, using a dynamic programming based algorithm. In general, bandit problems are optimization problems – rather than accurately estimating the success probability of all arms, the focus is on quickly identifying the best arm. In this Thesis, we will focus mainly on prediction and detection rather than such optimization problems. In Chapter 15 however, we will present a dynamic programming algorithm, different from the Gittins index algorithm, which can be used to solve a particular instance of bandit problems, where the arms are not required to be independent.

3.2 Optimization techniques

All of the criteria discussed thus far yield challenging combinatorial optimization problems. Several approaches are used to solve them in the literature, which can be roughly categorized into those that respect the integrality constraint and those which use a continuous relaxation.

3.2.1 Combinatorial search

For both geometric and model-based approaches, one must search for the best design or set of observations (e.g., sensor locations) among a very (usually exponentially) large number of candidate solutions. In a classical design, e.g., the inverse moment matrix on a set of selected experiments $\mathcal{X}_\mathcal{A}$ can be written as

$$M_\mathcal{A} = \left(\sum_{i=1}^{n} k_i \mathbf{x}_i \mathbf{x}_i^T \right)^{-1},$$

where k_i is the number of times experiment $\mathbf{x}_i$ is performed in design $\mathcal{A}$. Since k_i must be an integer, a combinatorial number of potential experimental designs has to be searched. Similarly, when placing a set $\mathcal{A}$ of k sensors out of a set $\mathcal{V}$ of possible locations, as we will do in Chapter 6, all sets of size k have to be searched. For both entropy (Ko et al. 1995), mutual information (Chapter 6) and value of information (Chapter 15), this search can be shown to be $\mathbf{NP}$-hard, hence efficient exact solutions are likely not possible.

Since exhaustive search is usually infeasible, local, heuristic searches without theoretical guarantees have commonly been applied. Approaches to the difficult combinatorial optimization include simulated annealing [Meyer and Nachtsheim, 1988], pairwise exchange [Cook and Nachtsheim, 1980, Fedorov, 1972, Mitchell, 1974a,b, Nguyen and Miller, 1992], forward and backward greedy heuristics [Caselton and Zidek, 1984, MacKay, 1992] and shortest path search [Zhao et al., 2002]. All these approaches provide no guarantees about the quality of the solution. Since optimal solutions are highly desirable, mixed integer program formulations [Koushanfary et al., 2006] and branch-and-bound approaches to speed up the exhaustive search have been developed [Ko et al., 1995, Welch, 1982]. Although they enable exhaustive search for slightly larger problem instances, the computational complexity of the problems puts strong limits on their effectiveness.

For geometric approaches, where sensors are associated with a fixed sensing region, and a spatial domain needs to be covered by the regions associated with the selected sensors, special properties of the sensing quality function can be used to develop approximation algorithms with theoretical guarantees. Examples include the art gallery approach to sensor placement [Hochbaum and Maas, 1985] and algorithms for sensor scheduling Abrams et al. [2004]. Deshpande et al. [2008] presents an approach for this problem based on semidefinite programming (SDP), handling more general constraints and providing tighter approximations. However, none of these approaches can handle objective functions based on complex probabilistic models.

By exploiting submodularity, in this Thesis, we provide the first approach to optimizing sensing which applies to such complex objectives, and has guarantees both on the runtime and on the quality of the achieved solutions.

3.2.2 Continuous relaxation

In some formulations, the integrality constraint is relaxed. For example, in classical experimental design, the number of experiments to be selected is often large compared to the number of design choices. In these cases, one can find a fractional design (i.e., a non-integral solution defining the proportions by which experiments should be performed), and round the fractional solutions. In the fractional formulation, A-, D- and E-optimality criteria can be solved exactly using a semi-definite program [Boyd and Vandenberghe,

2004]. There are however no known bounds on the integrality gap, i.e., the loss incurred by this rounding process.

In other approaches [Seo et al., 2000, Snelson and Ghahramani, 2005], a set of locations is chosen not from a discrete, but a continuous space. If the objective function is differentiable with respect to these locations, gradient-based optimization can be used instead of requiring combinatorial search techniques. Nevertheless, optimality of the solution is not guaranteed since there is no known bound on the discrepancy between local and global optima.

Another method that yields a continuous optimization, in the case of geometric objective functions, is the potential field approach [Heo and Varshney, 2005, Howard et al., 2002]. An energy criterion similar to a spring model is used. This optimization results in uniformly distributed (in terms of inter-sensor distances), homogeneous placements. The advantage of these approaches is that they can adapt to irregular spaces (such as hallways or corridors), where a simple grid-based deployment is not possible. Since the approach uses coordinate ascent, it can be performed using a distributed computation, making it useful for robotics applications where sensors can move.

Another well-known example of a continuous relaxation technique is the LASSO for subset selection in linear regression [Tibshirani, 1996]. Instead of solving the combinatorial problem of searching through all possible subsets of explanatory variables, a ℓ_1 penalty on the coefficients is introduced which encourages sparse solutions. Continuous relaxation techniques using ℓ_1 penalties have also been particularly successful in sparse reconstruction / Compressive Sensing [Candès et al., 2006, Donoho, 2006]. These results state that a high-dimensional signal can be reconstructed from a small number of measurements if it is known a priori that the true signal has a sparse representation in some (unknown) basis. For example, Candès et al. [2006] prove that a discrete time signal $f \in \mathbb{C}^N$ which is sparse (i.e., has at most T nonzero entries) whose Fourier coefficients $\hat{f}(\omega)$ are known for a set of frequencies $\omega \in \Omega$, where

$$|T| \leq \mathcal{O}(|\Omega|/\log N)$$

can be reconstructed exactly as the solution of the optimization problem

$$\min_{g \in \mathbb{C}^N} ||g||_{\ell_0} \text{ such that } \hat{g}(\omega) = \hat{f}(\omega) \text{ for all } \omega \in \Omega.$$

Hereby, $||g||_{\ell_0}$ is the number of non-zero elements of the complex vector g. However, solving this optimization problem requires to solve a combinatorial optimization problem, searching for the smallest subset of nonzero coefficients of g. Surprisingly, as Candès et al. [2006] show, if the set Ω of frequencies is chosen at random, then with overwhelmingly high probability the optimal solution of the above optimization problem is obtained using the following convex relaxation, which can be solved efficiently:

$$\min_{g \in \mathbb{C}^N} ||g||_{\ell_1} \text{ such that } \hat{g}(\omega) = \hat{f}(\omega) \text{ for all } \omega \in \Omega,$$

i.e., replacing $||\cdot||_{\ell_0}$ by the $||\cdot||_{\ell_1}$ norm. This celebrated result led to a number of extensions and follow-up work. For example, instead of requiring to observe Fourier coefficients, it suffices to use random measurements (more generally, linear measurements that satisfy a particular incoherence requirement). For a detailed introduction to the topic see [Candès and Wakin, 2008]. Instead of using random measurements, Tropp [2004] analyzes greedy selection using Orthogonal Matching Pursuit (originally proposed by Chen et al. [1989]). He shows that if the measurements that the algorithm can choose from are sufficiently incoherent, the greedy algorithm provides an $\mathcal{O}(\sqrt{k})$ approximation to the best reconstruction

error achievable when expanding the signal as a linear combination of at most k terms from the dictionary. While a number of sensing devices that implement the acquisition of incoherent measurements (as required for compressed sensing) have been developed [*c.f.*, Duarte et al., 2008], for the monitoring problems in this Thesis, compressive sensing theory does not apply, as the local, point measurements do not satisfy the required incoherence condition. Instead, the approaches described in this Thesis exploit prior knowledge about the signal that should be reconstructed. Our approaches also apply to a large variety of objective functions, beyond the squared reconstruction error. In addition, our algorithms apply to settings where there is a complex cost function associated with the set of chosen measurements (such as the one arising when using robots to obtain measurements, where the cost corresponds to the length of a shortest path connecting the measurements, considered as nodes in a graph, *c.f.*, Chapter 11).

3.2.3 Probabilistic planning

Optimized information gathering has been also extensively studied in the planning community. Bayer-Zubek [2004] for example proposed a heuristic method for optimizing value of information based on the Markov Decision Process framework. However, her approach makes approximations without theoretical guarantees.

The problem of optimizing decision theoretic value of information can be naturally formalized as a (finite-horizon) Partially Observable Markov Decision Process (POMDP) [Smallwood and Sondik, 1973]. Hence, in principle, algorithms for planning in POMDPs, such as the anytime algorithm by Pineau et al. [2006], can be employed for optimizing value of information. Unfortunately, the state space grows exponentially with the number of variables that are considered in the selection problem. In addition, the complexity of planning in POMDPs grows exponentially in the cardinality of the state space, hence doubly-exponentially in the number of variables for selection. This steep increase in complexity makes application of black-box POMDP solvers infeasible. Recently, Ji et al. [2007] demonstrated the use of POMDP planning on a multi-sensor scheduling problem. While presenting promising empirical results, their approach however uses approximate POMDP planning techniques without theoretical guarantees.

In the robotics literature, Stachniss et al. [2005], Sim and Roy [2005] and Kollar and Roy [2008] have presented approaches to information gathering in the context of Simultaneous Localization and Mapping (SLAM). None of these approaches however provide guarantees about the quality of the obtained solutions.

3.3 Observation selection applications in Machine Learning

Several important observation selection problems arise in Machine Learning, including feature selection and active learning. Further, more specific connections will be reviewed in subsequent chapters.

3.3.1 Feature selection and dimension reduction

Observation selection is closely related to dimension reduction approaches. However, while dimension reduction techniques such as Principal Component Analysis (PCA) or Canonical Correlation Analysis (CCA) can be used to find a lower dimensional representation of high dimensional data, these techniques usually find projections which are non-sparse, i.e., which are linear combinations of (almost) all input

variables. However, for interpretation purposes (and considering data acquisition cost), one often desires *sparse* projections, which are linear combinations of only a small subset of input variables. Moghaddam et al. [2005] and Moghaddam et al. [2006] consider the problem of selecting such sparse linear projections (subject to a constraint on the number of nonzero entries) of minimum reconstruction error (for PCA) and class separation (for LDA). In order to find these sparse projections, they propose two approaches: A mixed integer program, which can solve the problem optimally – albeit generally not in polynomial time, and a heuristic approach, using a greedy forward search followed by a greedy backward elimination. They provide several theoretical bounds, including a guarantee that this backward greedy algorithm achieves a solution of at least $\frac{k}{n}\lambda_{\max}$ where $n = |\mathcal{V}|$. This guarantee however depends on the problem size n.

3.3.2 Active learning

In the machine learning community, information-theoretic criteria have been used for active learning, techniques which allow the learning algorithm to influence the choice of training samples. For example, information-theoretic criteria have been used in the analysis of query-by-committee to select samples [Axelrod et al., 2001, Freund et al., 1997, Sollich, 1996]. Following Lindley [1956], MacKay [1992] proposes selecting observations that maximize expected information gain, either in terms of entropy or cross entropy, using Federov exchange. As opposed to the results presented in this Thesis, which address the optimization problem of selecting near-optimal observations, MacKay [1992] focuses on comparing the different objective criteria. Cohn [1994] proposes scoring each potential observation by measuring the average reduction in predicted variance at a set of reference points (a sequential implementation of Bayesian A-optimal design). A similar approach has been applied for observation selection in Gaussian process regression [Seo et al., 2000]. Common to all these active learning approaches, as well as to problems presented in this Thesis, is the problem of selecting a set (or sequence) of most informative observations. Unlike the results presented in this Thesis however, we are not aware of any prior work in this area which provides rigorous approximation guarantees for active learning using (sequential) Bayesian experimental design.

There is also a body of work on identifying the classes of learning problems for which active learning can achieve asymptotically faster error reduction than standard passive learning for classification [*c.f.*, Dasgupta, 2005] and regression [*c.f.*, Castro et al., 2005]. Often, the resulting algorithms [Balcan et al., 2006, Dasgupta et al., 2007, Dasgupta, 2004] are efficient in terms of label complexity, but intractable in general in terms of computational complexity. In contrast, the emphasis of the results presented in this Thesis is to develop observation selection algorithms that are both efficient and perform provably near-optimally, albeit under somewhat stronger (Bayesian) assumptions.

Part II

Submodular Sensing

Chapter 4

Overview of Part II

In Part II of this Thesis, we consider three examples of observation selection problems: selecting observations for *spatial prediction in Gaussian Processes*, selection of *informative variables in a graphical model*, and selecting observations for *detecting outbreaks* spreading over a graph.

Submodularity of sensing problems. We will show that in all three applications, the sensing quality function (*c.f.*, Section 2.3) satisfies an intuitive diminishing returns property: Adding a sensor helps more if we have placed few sensors so far, and less if we have already placed many sensors. This intuition can be formalized using the notion of *submodular set functions*, which we will review in Chapter 5.

Identifying submodularity in a problem is extremely useful. For example, suppose we want to find the best k locations to place sensors. In this setting, a fundamental result by Nemhauser et al. [1978] proves that a simple greedy algorithm, which iteratively adds a sensor at the location which increases the utility the most results in a near-optimal solution, obtaining a constant fraction of 63% of the optimal value of this $\mathbf{NP}$-hard optimization problem (*c.f.*, Theorem 5.2).

Each of the three settings which we will study in Part II of this Thesis does not allow to directly apply the existing results about maximizing submodular functions however, and we will discuss approaches for extending the existing results to these settings.

Sensor placement for spatial monitoring in Gaussian Processes. In Chapter 6, we consider the problem of selecting measurement locations that maximize the *mutual information* criterion for spatial prediction (Chapter 6). While this sensing quality function is submodular, it is not nondecreasing, as required by most existing results for maximizing submodular functions, such as Theorem 5.2. The key contribution provided in this chapter is the realization, that mutual information is *approximately* nondecreasing, if the number of placed sensors is small in comparison to the number of locations we can select from. We generalize the existing results about maximizing nondecreasing submodular functions to the setting of approximately nondecreasing functions, thus deriving the first approximation algorithm for placing sensors in Gaussian Processes. We also provide an extensive set of empirical results on several real-world sensor placement problems.

Selecting informative variables in graphical models. In Chapter 7, we consider the problem of selecting most informative variables in a graphical model. More formally, we address the problem of selecting a set of k variables which maximize the information gained about a set of target variables of interest. While the information gain objective about target variables in general is *not submodular*, we show that under some conditional independence assumptions, it does satisfy submodularity. Unfortunately, even in these cases, computing the information gain is difficult even for graphical models, where efficient inference is possible. We nevertheless show that the information gain can be approximated efficiently using sampling. We extend the methodology for maximizing submodular functions described in Chapter 5 to this case, and provide a randomized approximation algorithm for selecting the most informative set of variables in graphical models. We empirically analyze our algorithm on a real-world traffic prediction problem.

Sensor placement for outbreak detection. In the case of outbreak detection (Chapter 8), we deal with the problem that observations have non-constant cost, and the existing algorithms for optimizing such problems do not scale to the size of the outbreak detection problems considered in this chapter. In addition, we consider cases where multiple criteria (e.g., time to detection and detection likelihood) need to be simultaneously optimized. We present extensive experiments, on two real-world problems: Placing a set of sensors for contamination detection in a real water distribution network, and deciding which weblogs to read, in order to effectively capture most important information cascades.

Summary of contributions. The key contributions of this part of the Thesis are:

1. We prove that several important sensing quality functions satisfy *submodulariy*. These include
   - mutual information for spatial prediction in Gaussian Processes,
   - information gain with respect to a set of target variables in graphical models and
   - outbreak detection in networks.
2. We show how this submodularity can be exploited to
   - obtain provably near-optimal solutions for selecting the k most informative sensing locations, by invoking powerful existing results about optimizing submodular functions,
   - speed up algorithms using lazy evaluations and
   - compute novel tight, data dependent-bounds.
3. We prove hardness of sensing problems. Under reasonable complexity theoretic assumptions, optimizing information gain is hard to approximate within a factor of $(1 - 1/e)$ even for Naive Bayes models. This bound matches the performance of the greedy algorithm for this problem.
4. We demonstrate our approach on several real-world case studies, including
   - sensor placement for spatial prediction in Gaussian Processes,
   - sensor placement for posture recognition on a sensing chair, where our algorithms obtain 30-fold reduction in hardware cost without significant reduction in prediction accuracy,
   - sensor placement for outbreak detection in a metropolitan-area drinking water distribution network, where our algorithms obtained a top score in an international competition and
   - selecting informative weblogs to read on the Internet, showing that our algorithms scale to a blog data set with 45,000 blogs and millions of postings.

Chapter 5

Submodularity

As we will show in the subsequent Chapters 6, 7 and 8, many observation selection problems satisfy an intuitive diminishing returns property: Adding an observation helps more, if we have observed little so far, and helps less, if we already have made a lot of observations. This concept is formalized by the combinatorial property of *submodularity* (*c.f.*, Nemhauser et al. 1978). In this chapter, we review the concept of and classical results about submodularity, and give examples of its usefulness. In a strong sense, submodularity is the discrete analogue to convexity [Lovasz, 1983]. Just as convexity makes continuous functions more amenable to optimization, so does submodularity in the case of combinatorial problems. We also provide several contributions, including the CELF algorithm for maximizing submodular functions subject to arbitrary non-constant cost functions, as well as analyses using approximate oracles and online bounds.

5.1 Submodular set functions

Consider a set function

$$F : 2^{\mathcal{V}} \to \mathbb{R},$$

which maps subsets $\mathcal{A} \subseteq \mathcal{V}$ of a finite ground set $\mathcal{V}$ to the real numbers. F is called *submodular* [*c.f.*, Fujishige, 2005, Nemhauser et al., 1978] if, for all $\mathcal{A}, \mathcal{B} \subseteq \mathcal{V}$ it holds that

$$F(\mathcal{A}) + F(\mathcal{B}) \geq F(\mathcal{A} \cup \mathcal{B}) - F(\mathcal{A} \cap \mathcal{B}).$$

Note, that if F is nonnegative (i.e., $F(\mathcal{A}) \geq 0$ for all $\mathcal{A}$), submodularity implies subadditivity, as considered in Section 2.4.2. Hence, from a sensing optimization perspective, combining two sets of information has less sensing quality than the sum of the sensing qualities of the individual sets.

An equivalent characterization of submodular set functions is the following:
Proposition 5.1 (Nemhauser et al. [1978])**.** *A set function is submodular if and only if, for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $s \in \mathcal{V} \setminus \mathcal{B}$, it holds that*

$$F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{s\}) - F(\mathcal{B}). \tag{5.1}$$

This second characterization is the formalization of diminishing returns alluded to in the introduction: Adding an element s helps at least as much if we add it to a small set $\mathcal{A}$ than if we add it to a superset

$\mathcal{B}$. In addition to this intuitive interpretation, the characterization (5.1) is typically easier to verify when proving submodularity.

A function F such that $-F$ is submodular is called *supermodular*. A set function is additive (or *modular*) if and only if it is both submodular and supermodular [Lovasz, 1983].

Another important property of some set functions is *nondecreasingness*: A set function F is called *nondecreasing*, if for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ it holds that $F(\mathcal{A}) \leq F(\mathcal{B})$. This is an intuitive requirement for sensing quality functions: Gathering more information can never reduce the sensing quality.

5.1.1 An example: Set coverage

An important example of a nondecreasing submodular set function is the following. Consider finite sets $\mathcal{V}$ and $\mathcal{W}$. With each element $s \in \mathcal{V}$, we associate a subset $\mathcal{B}_s \subseteq \mathcal{W}$. Define a set function F on $\mathcal{V}$ by

$$F(\mathcal{A}) = \left| \bigcup_{s \in \mathcal{A}} \mathcal{B}_s \right|. \tag{5.2}$$

It is straightforward to verify that F is nondecreasing and submodular. Using the unit cost function, $C(\mathcal{A}) = |\mathcal{A}|$, Problem (2.5) with $Q = |\mathcal{W}|$, i.e., the problem

$$\mathcal{A}^* = \operatorname*{argmin}_{\mathcal{A} \subseteq \mathcal{V}} |\mathcal{A}| \text{ such that } F(\mathcal{A}) = F(\mathcal{V}),$$

is called the *set cover* problem. Similarly, for $B = k$, the problem (2.4), i.e.,

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A}),$$

is called the *max-k-cover* problem. F is often called the *set cover function* or the *cardinality-of-union function*.

Note, that this objective function has an intuitive interpretation in the observation selection context. Consider the case, where we have a spatial region $\mathcal{W}$ to cover, and with each location $s \in \mathcal{V}$, we associate a sensing region $\mathcal{B}_s$ (e.g., a disk). Then, our goal would be to, e.g., place k sensors such that the covered area $\left|\bigcup_{s \in \mathcal{A}} \mathcal{B}_s\right|$ is maximized. Indeed, this objective function has been extensively used in the sensor placement literature [*c.f.*, Abrams et al., 2004, Bai et al., 2006, Deshpande et al., 2008, Hochbaum and Maas, 1985, Kershner, 1939].

We can also fit this example into our utility-based formulation (2.1): Define a random variable Z which is uniformly distributed over the index set $\mathcal{W}$. With each possible observation $s \in \mathcal{V}$ associate an indicator variable $\mathcal{X}_s = [Z \in \mathcal{B}_s]$, which is 1 iff Z takes a value in $\mathcal{B}_s$, 0 otherwise. Define a sensing quality function $u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = \max_{s \in \mathcal{A}} x_s$. Hence, $u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = 1$ if any of the chosen observations "detects" Z, i.e., $Z \in \left|\bigcup_{s \in \mathcal{A}} \mathcal{B}_s\right|$, and 0 otherwise. Then

$$F(\mathcal{A}) = \int P(\mathbf{x}_\mathcal{V}) u(\mathbf{x}_\mathcal{V}, \mathcal{A}) d\mathbf{x}_\mathcal{V} = \frac{1}{|\mathcal{W}|} \left| \bigcup_{s \in \mathcal{A}} \mathcal{B}_s \right|.$$

Note, that it is straightforward to associate a nonnegative "importance" weight $w(x)$ with each element $x \in \mathcal{W}$. The modified objective

$$F(\mathcal{A}) = \sum_x w(x) \left[x \in \bigcup_{s \in \mathcal{A}} \right] \tag{5.3}$$

Algorithm 5.1: GREEDY: Greedy algorithm for maximizing submodular function F.

Input: Submodular function $F : 2^{\mathcal{V}} \to \mathbb{R}$, k
Output: Subset $\mathcal{A} \subseteq \mathcal{V}$
begin
  $\mathcal{A} \leftarrow \emptyset$;
  for *j = 1* **to** k **do**
    for $y \in \mathcal{V} \setminus \mathcal{A}$ **do**
1     $\delta_s \leftarrow F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})$;
    end
2   $s^* \leftarrow \operatorname{argmax}_{s \in \mathcal{V} \setminus \mathcal{A}} \delta_s$;
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{s^*\}$;
  end
end

remains submodular and nondecreasing.

We can allow even more extensions: For example, we can allow each element $x \in \mathcal{W}$ to be covered multiple times. This is useful in sensor placement problems, where we want to achieve coverage which is robust against sensor failures. Examples of such *set multi-cover* or even *multi-set multi-cover* problems are given by Rajagopalan and Vazirani [1998]. All these problems remain nondecreasing and submodular, and have useful observation selection interpretations.

This example already suggests that submodularity is an interesting structural property of observation selection problems. In the next sections, we will see how submodularity is very useful in practice.

5.2 The greedy algorithm for nondecreasing submodular set functions

For many nondecreasing submodular set functions, the maximization and covering problems (2.4) and (2.5) are $\mathbf{NP}$-hard. One example is the *max-k-cover* problem, where the best set $\mathcal{A}$ of size at most k maximizing the *set cover* objective F is desired [Feige, 1998]. Hence we cannot expect to solve such problems efficiently in general, unless $\mathbf{P} = \mathbf{NP}$. As discussed in the following, the simple greedy algorithm provides the best possible performance achievable in polynomial time in general under reasonable complexity theoretic assumptions.

5.2.1 Greedy maximization algorithm in the unit cost case

A simple heuristic approach to solving the maximization problem (2.4) is the *greedy algorithm* (sometimes also called *forward selection*), which starts with the empty set $\mathcal{A} = \emptyset$, and iteratively adds the element $s^* \in \mathcal{V} \setminus \mathcal{A}$ which maximally increases the objective value, i.e.,

$$s^* = \operatorname*{argmax}_{s \in \mathcal{V} \setminus \mathcal{A}} F(\mathcal{A} \cup \{s\}), \tag{5.4}$$

until k elements have been selected. Algorithm 5.1 provides pseudo-code for this procedure.

Perhaps surprisingly, the greedy algorithm provides strong theoretical guarantees:

Theorem 5.2 (Nemhauser et al. [1978])**.** *Let F be a nondecreasing submodular function defined on a ground set $\mathcal{V}$ such that $F(\emptyset) = 0$, and k be an integer. Then the solution $\mathcal{A}_{greedy}$ returned by the greedy algorithm satisfies*

$$F(\mathcal{A}_{greedy}) \geq \left(1 - \left(\frac{k-1}{k}\right)^k\right) \mathrm{OPT} \geq (1 - 1/e)\,\mathrm{OPT},$$

where $\mathrm{OPT} = \max_{|\mathcal{A}| \leq k} F(\mathcal{A})$ is the value obtained by an optimal solution.

Hence, the greedy algorithm is guaranteed to obtain a solution which achieves at least a constant fraction of $(1 - 1/e) \approx 63\%$ of the optimal value. Perhaps even more surprisingly, if $\mathbf{P} = \mathbf{NP}$, the greedy algorithm is in a strong sense optimal for maximizing submodular functions:

Theorem 5.3 (Feige [1998])**.** *If there were a constant $\alpha > (1 - 1/e)$, and a polynomial-time algorithm which, for any nondecreasing submodular function, integer k would be guaranteed to find a solution such that*

$$F(\mathcal{A}_{greedy}) \geq \alpha\,\mathrm{OPT},$$

then $\mathbf{P} = \mathbf{NP}$.

Hence, for any nondecreasing submodular function F, and for $C(\mathcal{A}) = |\mathcal{A}|$, the greedy algorithm solves the problem (2.4) near-optimally.

What if the greedy rule (5.4) can not be evaluated exactly? This problem can arise, e.g., if F can only be evaluated approximately (e.g., through sampling). In the following, we consider the case where, instead of F, an approximate version $\widehat{F}$ of F is evaluated. We say $\widehat{F}$ evaluates F with *multiplicative error* $\alpha \geq 1$, if for all $\mathcal{A} \in \mathcal{V}$ it holds that

$$\frac{1}{\alpha} F(\mathcal{A}) \leq \widehat{F}(\mathcal{A}) \leq \alpha F(\mathcal{A}).$$

We say $\widehat{F}$ evaluates F with *additive error* $\varepsilon \geq 0$, if for all $\mathcal{A} \in \mathcal{V}$ it holds that

$$|F(\mathcal{A}) - \widehat{F}(\mathcal{A})| \leq \varepsilon.$$

We can generalize Theorem 5.2 to the case, where F is evaluated only approximately:

Theorem 5.4. *If $\widehat{F}$ evaluates F with multiplicative error α or absolute error ε, then the solution $\mathcal{A}_{greedy}$ returned by the greedy algorithm applied to $\widehat{F}$ satisfies*

$$F(\mathcal{A}_{greedy}) \geq \left(1 - 1/e^{\frac{1}{\alpha}}\right) \mathrm{OPT} - k\varepsilon,$$

where $\mathrm{OPT} = \max_{|\mathcal{A}| \leq k} F(\mathcal{A})$ is the value obtained by an optimal solution.

The proof of Theorem 5.4 and all other results in this Thesis is presented in the Appendix. This result was shown previously by Kempe et al. [2003] for the special case of multiplicative error.

5.2.2 Greedy maximization algorithm in the non-constant cost case

If the cost function is additive, but the cost $c(s)$ of each element varies, then the simple greedy algorithm, which adds elements until the specified budget B is exhausted, can perform arbitrarily badly, since it is indifferent to the costs (*i.e.*, a very expensive element providing reward r is preferred over a cheaper

element providing reward $r - \varepsilon$. To avoid this issue, the greedy algorithm can be modified to take costs into account, by selecting the next element s^* after $j-1$ elements have already been chosen as

$$s^* = \operatorname*{argmax}_{s \in \mathcal{V} \setminus \mathcal{A}} \frac{F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})}{c(s)}. \tag{5.5}$$

i.e., the greedy algorithm picks the element maximizing the benefit/cost ratio. The algorithm stops once no element can be added to the current set $\mathcal{A}$ without exceeding the budget B. Unfortunately, this intuitive generalization of the greedy algorithm can perform arbitrarily worse than the optimal solution. Consider the case where we have two locations, s_1 and s_2, $c(s_1) = \varepsilon$ and $c(s_2) = B$. Let our objective function F satisfy $F(\emptyset) = 0$, $F(\{s_1\}) = 2\varepsilon$, and $F(\{s_2\}) = B$. Now, $(F(\{s_1\}) - F(\emptyset))/c(s_1) = 2$, and $(F(\{s_2\}) - F(\emptyset))/c(s_2) = 1$. Hence the greedy algorithm would pick s_1. After selecting s_1, it cannot afford s_2 anymore, and our total objective value would be ε. However, the optimal solution would pick s_2, achieving a total value of B. As ε goes to 0, the performance of the greedy algorithm becomes arbitrarily bad.

However, the greedy algorithm can be improved to achieve a constant factor approximation. This new algorithm, CEF (for cost-effective forward-selection) computes the solution $\mathcal{A}_{GCB}$ using the benefit-cost greedy algorithm, using rule (5.5), and also computes the solution $\mathcal{A}_{GUC}$ using the unit-cost greedy algorithm (ignoring the costs), using rule (5.4). For both rules, CEF only considers elements which do not exceed the budget B. CEF then returns the solution with higher score. Even though both solutions can be arbitrarily bad, the following result shows that there is at least one of them which is not too far away from optimum, and hence CEF provides a constant factor approximation.

Theorem 5.5. *Let F be a nondecreasing submodular function with $F(\emptyset) = 0$. Then*

$$\max\{F(\mathcal{A}_{GCB}), F(\mathcal{A}_{GUC})\} \geq \frac{1}{2}(1 - 1/e) \max_{\mathcal{A}, C(\mathcal{A}) \leq B} F(\mathcal{A}).$$

Theorem 5.5 was proved by Khuller et al. [1999] for the special case of the set-covering objective function. We extend their result and analysis to *arbitrary* nondecreasing submodular functions. Theorem 5.5 states that the better solution of $\mathcal{A}_{GBC}$ and $\mathcal{A}_{GUC}$ (which is returned by CEF) is at most a constant factor $\frac{1}{2}(1 - 1/e) \approx 0.31$ of the optimal solution.

We can generalize Theorem 5.5 to the case where F is evaluated only approximately:

Theorem 5.6. *If $\widehat{F}$ evaluates F with multiplicative error at most α, or additive error at most ε, then*

$$\max\{F(\mathcal{A}_{GCB}), F(\mathcal{A}_{GUC})\} \geq \frac{1}{2\alpha}(1 - 1/e^{1/\alpha}) \max_{\mathcal{A}, C(\mathcal{A}) \leq B} F(\mathcal{A}) - (1 + \alpha)\left(\frac{B}{2c_{min}}\right)\varepsilon.$$

For an algorithm similar to CEF, Wolsey [1982b] proved a slightly tighter bound of $1 - e^{-\beta}$ with β chosen as the solution of $e^{\beta} = 2 - \beta$. This bound evaluates to $1 - e^{-\beta} \approx 0.35$, whereas $\frac{1}{2}(1 - 1/e) \approx 0.31$. However, Wolsey [1982b] did not consider the case of approximate function evaluations, which will be required for our analysis in the subsequent chapters.

Note that the running time of CEF is $\mathcal{O}(B|\mathcal{V}|)$ in the number of possible locations $|\mathcal{V}|$ (if we consider a function evaluation $F(\mathcal{A})$ as atomic operation, and the lowest cost of a node is constant). Sviridenko [2004] showed that even in the non-constant cost case, the approximation guarantee of $(1 - 1/e)$ can be achieved. However, his algorithm is $\Omega(B|\mathcal{V}|^4)$ in the size of possible locations $|\mathcal{V}|$ we need to select from, which is prohibitive in the applications we consider.

5.2.3 Greedy covering algorithm

We now consider the covering Problem (2.5). An intuitive algorithm for covering is an alternative version of the greedy algorithm, which does not stop until $F(\mathcal{A}) = F(\mathcal{V})$, i.e., the maximum attainable score has been achieved. We again consider the setting of arbitrary nonnegative additive cost functions, and use the greedy rule (5.5). The following result is known about this algorithm: (*c.f.*,

Theorem 5.7 (Wolsey [1982a])**.** *Let F be a nondecreasing,* integer-valued*, submodular function defined on a ground set $\mathcal{V}$ such that $F(\emptyset) = 0$, and let $C(\mathcal{A})$ be a nonnegative, additive cost function. Then the solution $\mathcal{A}_{greedy}$ returned by the greedy covering algorithm satisfies*

$$C(\mathcal{A}_{greedy}) \leq \left(1 + \ln\left(\max_{s \in \mathcal{V}} F(\{s\})\right)\right) \mathrm{OPT}$$

where $\mathrm{OPT} = \min_{\mathcal{A}: F(\mathcal{A}) = F(\mathcal{V})} C(\mathcal{A})$ is the cost incurred by an optimal solution.

Hence, the solution returned by the greedy algorithm is at most a logarithmic factor more expensive than the optimal solution. Similarly to the maximization problem, it is possible to show that this guarantee is best possible for the covering case as well:

Theorem 5.8 (Feige [1998])**.** *If there were a constant $\alpha > 0$, and a polynomial-time algorithm which, for any nondecreasing submodular function would be guaranteed to find a solution such that*

$$C(\mathcal{A}_{greedy}) \leq (1 - \alpha)\left(\ln \max_{s \in \mathcal{V}} F(\{s\})\right) \mathrm{OPT},$$

then $\mathbf{P} \subseteq \mathrm{DTIME}(n^{\log \log n})$.

Hereby, $\mathrm{DTIME}(n^{\log \log n})$ is a class of slightly superpolynomial algorithms, and the inclusion $\mathbf{P} \subseteq \mathrm{DTIME}(n^{\log \log n})$ is considered unlikely [Feige, 1998].

Note that if $F(\mathcal{A})$ is nondecreasing and submodular, the function $F'(\mathcal{A}) = \min(F(\mathcal{A}), Q)$ is submodular as well [Fujito, 2000]. Furthermore, if $Q \leq F(\mathcal{V})$, then $F'(\mathcal{A}) = F'(\mathcal{V}) = Q$ if and only iff $F(\mathcal{A}) \geq Q$. Hence, the greedy covering algorithm can be used to approximately solve Problem (2.5) for arbitrary submodular functions and arbitrary quotas Q.

5.3 Online bounds for submodular maximization problems

For nondecreasing submodular functions, Theorems 5.2 and 5.5 prove an *offline* approximation guarantee of $(1 - 1/e)$, which we can state *a priori*, i.e., before running the algorithm. As we will show empirically Chapters 6 and 8, for many practical problems, this bound is very loose.

In the case of additive cost functions with unit cost, the following observation allows to compute online bounds on the optimal value of the maximization problem with sets of size k:

Proposition 5.9 (Minoux [1978])**.** *Let F be a nondecreasing submodular function on $\mathcal{V}$, and let $\mathcal{A} \subseteq \mathcal{V}$, $|\mathcal{A}| = k$. For all $s \in \mathcal{V} \setminus \mathcal{A}$, let $\delta_s = F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})$. Sort the δ_s in decreasing order, and consider the sequence $\delta^{(1)}, \ldots, \delta^{(k)}$ of the first k elements. Then $\max_{|\mathcal{A}'| \subseteq k} F(\mathcal{A}') \leq F(\mathcal{A}) + \sum_{i=1}^{k} \delta^{(i)}$.*

The proof of this proposition follows directly from submodularity. In many applications, especially for large placements, this bound can be much tighter than the bound guaranteed by Theorem 5.2.

Algorithm 5.2: GETBOUND: Getting bound $\hat{F}$ on optimal solution.

Input: Reward function F, cost function C, budget B
Output: bound on optimal value
begin
  $\mathcal{A} \leftarrow \emptyset$; $\mathcal{B} \leftarrow \emptyset$; $\hat{F} = F(\mathcal{A})$;
  foreach $s \in \mathcal{V}$ **do** $\delta_s \leftarrow F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})$; $r_s = \frac{\delta_s}{C(\{s\})}$;
  while $\exists s \in \mathcal{V} \setminus (\mathcal{A} \cup \mathcal{B}) : C(\mathcal{A} \cup \mathcal{B} \cup \{s\}) \leq B$ **do**
    $s^* \leftarrow \underset{s \in \mathcal{V} \setminus \{\mathcal{A} \cup \mathcal{B}\}, C(\mathcal{A} \cup \mathcal{B} \cup \{s\}) \leq B}{\operatorname{argmax}} r_s$;
    $\hat{F} \leftarrow \hat{F} + \delta_{s^*}$; $\mathcal{B} \leftarrow \mathcal{B} \cup \{s^*\}$;
  end
  $s^* \leftarrow \underset{s \in \mathcal{V} \setminus \{\mathcal{A} \cup \mathcal{B}\}, C(\mathcal{A} \cup \mathcal{B} \cup \{s\}) \leq B}{\operatorname{argmax}} r_s$; $\lambda \leftarrow \frac{B - C(\mathcal{A} \cup \mathcal{B})}{C(\{s^*\})}$;
  return $\hat{F} + \lambda \delta_{s^*}$;
end

The following result allows to compute online bounds on the optimal value of the maximization problem for arbitrary additive cost functions.

Theorem 5.10. *For a set $\mathcal{A} \subseteq \mathcal{V}$, and each $s \in \mathcal{V} \setminus \mathcal{A}$, let $\delta_s = F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})$. Let $r_s = \delta_s / c(s)$, and let $s_1, \ldots, s_m$ be the sequence of locations with r_s in decreasing order. Let k be such that $C = \sum_{i=1}^{k-1} c(s_i) \leq B$ and $\sum_{i=1}^{k} c(s_i) > B$. Let $\lambda = (B - C)/c(s_k)$. Then*

$$\max_{\mathcal{A}', C(\mathcal{A}') \leq B} F(\mathcal{A}') \leq F(\mathcal{A}) + \sum_{i=1}^{k-1} \delta_{s_i} + \lambda \delta_{s_k}. \tag{5.6}$$

Theorem 5.10 presents a way of computing how far an *arbitrary* given solution $\mathcal{A}$ (obtained using CEF or *any* other algorithm) is from the optimal solution. In Chapters 6 and 8, we show, how this online bound can be much tighter than the offline guarantee of $(1 - 1/e)$ for many applications.

This theorem can be readily turned into an algorithm, as formalized in Algorithm 5.2.

We start with set $\widehat{\mathcal{A}}$, compute the benefit-cost ratios r_i, arrange them in decreasing order, and keep on adding their marginal benefits δ_i to $F(\widehat{\mathcal{A}})$ according to this order, until our budget is exhausted. We then pick the next best element, and add it fractionally according to the weight λ. The resulting score is our bound. Algorithm 5.2 presents pseudo-code for this procedure. In our experiments, we empirically show that this bound is much tighter than the bound $\frac{1}{2}(1 - 1/e)$, which is roughly 31%, or even the $(1 - 1/e)$ bound obtained using the GREEDY algorithm. Furthermore, we can essentially compute it for free when running CEF.

For additional practical usefulness of the offline and online bounds (Theorem 5.2, Theorem 5.5 and Theorem 5.10), note that they can be used to automatically derive an online performance guarantee for any other algorithm maximizing a submodular function F.

Remark 5.11. *Let $F(\mathcal{A})$ be the value of the solution which is guaranteed (e.g., by Theorem 5.10) to achieve at least a factor $0 < \alpha \leq 1$ of the optimal value. Let $F(\mathcal{A}')$ be the value of a feasible solution computed by another algorithm. Then $F(\mathcal{A}')$ is guaranteed to be a $\frac{\alpha F(\mathcal{A}')}{F(\mathcal{A})}$ approximation, and we can compute the guarantee at runtime.*

While fairly straightforward, this observation is very useful for the analysis of stochastic approximation algorithms such as simulated annealing [Kirkpatrick et al., 1983], which are in certain cases guaranteed to converge to the optimal solution with high probability, but for which one usually does not know whether they have already achieved the optimum or not. Remark 5.11 provides a lower bound on the quality of approximation achieved and a stopping criterion for these algorithm. The CEF solution is also a viable starting point for such local search heuristics.

5.4 Lazy evaluations and the CELF algorithm

It is possible to improve the performance of Algorithm 5.1 directly under certain conditions by lazy evaluation of the incremental improvements in Line 1. The idea of lazy evaluations is due to Minoux [1978], and has been used by Robertazzi and Schwartz [1989] in the context of D-optimal design. The key insight is that the improvements

$$\delta_s(\mathcal{A}) = F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}),$$

which are used in greedy selection, are monotonically nonincreasing in $\mathcal{A}$. Furthermore, the greedy algorithm produces such a monotonically increasing sequence of sets $\mathcal{A}$. This insight can be exploited in a lazy variant of the greedy algorithm in the following way.

At the start of the algorithm, all δ_s will be initialized to $+\infty$. The algorithm will maintain information about which δ_s are current, i.e., have been computed for the current set $\mathcal{A}$. Now, the greedy rule in Line 1 will find the element s with largest δ_s. If this δ_s has not been updated for the current $\mathcal{A}$, the value is updated and reintroduced into the queue. This process is iterated until the location with maximum δ_s has an updated value. As we show in Chapters 6 and 8, this lazy approach can save significant computation time.

To understand the efficacy of this procedure, consider the following intuition: If a location s^* is selected, nearby locations will become significantly less desirable and their marginal increases δ_y will decrease significantly. When this happens, these location will not be considered as possible maxima for the greedy step for several iterations.

Note, that this lazy evaluation technique can be applied to the CEF algorithm as well. We call the resulting algorithm CELF (for Cost-effective Lazy Forward-Selection), which is formalized in Algorithm 5.4.

This approach can be efficiently implemented by using a priority queue to maintain the advantages δ_s. Line 2 (or Line 3 respectively) calls **deletemax** with complexity $\mathcal{O}(\log n)$ and Line 4 uses the **insert** operation with complexity $\mathcal{O}(1)$. Also, as stated Line 1 has an $\mathcal{O}(n)$ complexity, and was introduced for simplicity of exposition. In reality, we annotate the δ_s's with the last iteration that they were updated, completely eliminating this step.

5.5 Exact algorithms for optimizing submodular functions

Even though maximizing submodular functions is **NP**-hard in general, for certain special cases, the optimal solution can be found efficiently.

Algorithm 5.3: LAZYFORWARD: Lazy unit cost/cost-benefit greedy algorithm.

Input: Reward function F, cost function C, budget B, type$\in \{UC, CB\}$
Output: Node selection $\mathcal{A} \subseteq \mathcal{V}$
begin
  $\mathcal{A} \leftarrow \emptyset$; **foreach** $s \in \mathcal{V}$ **do** $\delta_s \leftarrow +\infty$;
  while $\exists s \in \mathcal{V} \setminus \mathcal{A} : C(\mathcal{A} \cup \{s\}) \leq B$ **do**
1 **foreach** $s \in \mathcal{V} \setminus \mathcal{A}$ **do** $cur_s \leftarrow$ **false**;
    while true do
      if *type=UC* **then** $s^* \leftarrow \underset{s \in \mathcal{V} \setminus \mathcal{A}, C(\mathcal{A} \cup \{s\}) \leq B}{\operatorname{argmax}} \delta_s$;
2
      if *type=CB* **then** $s^* \leftarrow \underset{s \in \mathcal{V} \setminus \mathcal{A}, C(\mathcal{A} \cup \{s\}) \leq B}{\operatorname{argmax}} \frac{\delta_s}{C(\{s\})}$;
3
      if cur_s **then** $\mathcal{A} \leftarrow \mathcal{A} \cup \{s^*\}$; **break** ;
4
      else $\delta_s \leftarrow F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})$; $cur_s \leftarrow$ **true**;
    end
  end
  return $\mathcal{A}$;
end

Algorithm 5.4: CELF: The CELF algorithm.

Input: Reward function F, cost function C, budget B
Output: Node selection $\mathcal{A} \subseteq \mathcal{V}$
begin
  $\mathcal{A}_{UC} \leftarrow$`LazyForward(`$F, C, B$,***UC***`)`;
  $\mathcal{A}_{CB} \leftarrow$`LazyForward(`$F, C, B$,***CB***`)`;
  return $\operatorname{argmax}\{F(\mathcal{A}_{UC}), F(\mathcal{A}_{CB})\}$
end

5.5.1 Mixed integer programming

One approach, which can be used to get even tighter online bounds than those of Theorem 5.10, or even compute the optimal solution in certain cases, is based on mixed integer programming.

The mixed integer program for this approach, which is due to Nemhauser and Wolsey [1981], is given by:

$$\begin{aligned} &\max \eta; \\ &\quad \eta \;\leq\; F(\mathcal{B}) + \sum_{s_i \in \mathcal{V} \setminus \mathcal{B}} \alpha_i [F(\mathcal{B} \cup s_i) - F(\mathcal{B})], \quad \forall \mathcal{B} \subseteq \mathcal{V}; \qquad (5.7) \\ &\sum_i \alpha_i \;\leq\; k, \quad \forall i; \qquad (5.8) \\ &\quad \alpha_i \;\in\; \{0, 1\}, \quad \forall i; \end{aligned}$$

where $\alpha_i = 1$ means that location $s_i \in \mathcal{V}$ should be selected. Note that this MIP can be easily extended to handle the case in which each location can have a different cost, by replacing the constraint (5.8) by $\sum_i \alpha_i c_i \leq B$, where B is the budget and $c_i = c(s_i)$.

Unfortunately, this MIP has exponentially many constraints. Nemhauser and Wolsey [1981] proposed the following constraint generation algorithm: Let $\alpha^{\mathcal{A}}$ denote an assignment to $\alpha_1, \ldots, \alpha_n$ such that $\alpha_i = 1$ iff $s_i \in \mathcal{A}$. Starting with no constraints of type (5.7), the MIP is solved, and one checks whether the current solution $(\eta, \alpha^{\mathcal{B}})$ satisfies $\eta \leq F_{MI}(\mathcal{B})$). If it does not, a violated constraint has been found. Since solving individual instances (even with only polynomially many constraints) is **NP**-hard, we need to resort to search heuristics such as Branch and Bound and Cut during the constraint generation process.

The analysis of this MIP, as presented by Nemhauser and Wolsey [1981], assumes that F is nondecreasing. There is another MIP formulation for maximizing general submodular functions without requiring nondecreasingness. The details can be found in Nemhauser and Wolsey [1981].

5.5.2 Branch and bound search

By exploiting specific properties of submodular functions, Goldengorin et al. [1999] developed a branch and bound search approach for maximizing arbitrary (not necessarily nondecreasing) submodular functions. Similarly to the mixed-integer programming approach, this approach can potentially be used to avoid exhaustive search, but it is not guaranteed to find the optimal solution in polynomial time.

5.6 Existing work on submodular function maximization

In their seminal work, Nemhauser et al. [1978] and Wolsey [1982a] analyze the greedy algorithm for optimizing nondecreasing submodular functions. Sviridenko [2004] presents an algorithm for the budgeted maximization of submodular functions, as discussed in Section 5.2.2.

Another interesting submodular optimization problem is the problem of maximizing a nondecreasing submodular function subject to a *matroid constraint*, i.e., where the set $\mathfrak{A}$ in Problem (2.7) consists of independent sets of a matroid. Hereby, a matroid is a pair $(\mathcal{V}, \mathfrak{A})$ where $\mathfrak{A} \subseteq 2^{\mathcal{V}}$ is a collection of subsets of $\mathcal{V}$ called *independent* sets, satisfying

1. $\emptyset \in \mathfrak{A}$, i.e., the empty set is independent.
2. If $\mathcal{A} \subseteq \mathcal{B}$, and $\mathcal{B} \in \mathfrak{A}$, then $\mathcal{A} \in \mathfrak{A}$, i.e., subsets of independent sets are independent.
3. If $\mathcal{A}, \mathcal{B} \in \mathfrak{A}$, $|\mathcal{A}| < |\mathcal{B}|$ then there exists a $s \in \mathcal{B} \setminus \mathcal{A}$ such that $\mathcal{A} \cup \{s\} \in \mathfrak{A}$. I.e., if $\mathcal{A}$ and $\mathcal{B}$ are independent, and $\mathcal{A}$ is smaller than $\mathcal{B}$, then there is an element s in $\mathcal{B}$ which can be added to $\mathcal{A}$ such that the resulting set is still independent.

Matroid theory originated from generalizing the notion of linear independence in vector spaces to other combinatorial problems. One well-known example is the following: Take a graph with edges $\mathcal{E}$, and call a subset $\mathcal{A} \subseteq \mathcal{E}$ independent (i.e., $\mathcal{A} \in \mathfrak{A}$) if $\mathcal{A}$ has no cycle. Then $(\mathcal{E}, \mathfrak{A})$ is a matroid. Also note that for any set $\mathcal{V}$ and the collection of sets $\mathfrak{A} = \{\mathcal{A} \subseteq \mathcal{V} : |\mathcal{A}| \leq k\}$, $(\mathcal{V}, \mathfrak{A})$ is a matroid as well, called the *uniform matroid*. Hence, the maximization problem

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F(\mathcal{A})$$

is a special case of the problem of maximizing a submodular function subject to a matroid constraint. As Nemhauser et al. [1978] proved, the greedy algorithm gives a $(1 - 1/e)$ approximation to this problem.

Until very recently, the problem of whether a $(1 - 1/e)$ approximation is possible for *arbitrary* matroid constraints was open. Fisher et al. [1978] proved the powerful result:

Theorem 5.12 (Fisher et al. [1978])**.** *Let $\mathcal{V}$ be a finite set, and $(\mathcal{V}, \mathfrak{A}_1), \ldots, (\mathcal{V}, \mathfrak{A}_p)$ be matroids, and F a nondecreasing submodular function. Then the greedy algorithm results in a solution $\mathcal{A}_G$ such that*

$$F(\mathcal{A}_G) \geq \frac{1}{p+1} \max_{\mathcal{A}} F(\mathcal{A}) \text{ such that } \mathcal{A} \in \mathfrak{A}_1 \cap \cdots \cap \mathfrak{A}_p.$$

In the case of a single matroid constraint ($p = 1$), this theorem guarantees a factor 1/2 approximation to the problem of maximizing a submodular function to a matroid constraint, which is slightly weaker than the $(1 - 1/e)$ guarantee for the uniform matroid. Vondrak [2008] proved that a $(1 - 1/e)$ guarantee can be obtained for *arbitrary* matroid constraints, albeit using a more complex algorithm than the greedy algorithm. In Chapter 12 we will see how matroids are extremely useful for solving complex sensor scheduling problems.

5.7 Minimization of submodular functions

A large part of the theory of optimizing submodular functions is concerned with *minimizing* instead of maximizing a single submodular function. Queyranne [1995] present the first algorithm for minimizing symmetric submodular functions; Iwata et al. [2001] and Schrijver [2000] present combinatorial algorithms for minimizing *arbitrary* (not necessarily symmetric) submodular functions. Narasimhan and Bilmes [2006, 2004], Narasimhan et al. [2005] present several results exploiting these minimization algorithms to solve difficult machine learning problems such as clustering and structure learning of graphical models. While these approaches show promise for solving complex machine learning problems, we have not yet identified applications in the context of optimized information gathering.

5.8 Matlab Toolbox for optimizing submodular functions

All the algorithms for optimizing submodular functions presented in this Thesis are available in a Matlab toolbox, which is available at our website `http://www.submodularity.org`. The toolbox implements CELF for budgeted maximization, SATURATE for robust optimization, PSPIEL for optimization under complex cost functions, as well as ESPASS for simultaneous placement and scheduling. The Branch and Bound algorithm by Goldengorin et al. [1999] for exact maximization of submodular functions is included as well.

In addition, the toolbox also implements various algorithms for submodular function optimization beyond this thesis, including the Minimum Norm Point algorithm by Fujishige [2005], the algorithm for minimizing symmetric submodular functions by Queyranne [1995] as well as the submodular-supermodular procedure by Narasimhan and Bilmes [2006]. All algorithms are illustrated by optimization tasks arising from machine learning problems such as active learning and experimental design, clustering and inference in Markov Random Fields.

Chapter 6

Sensor Placements for Spatial Prediction

When monitoring spatial phenomena, such as temperatures in an indoor environment as shown in Figure 6.1, using a limited number of sensing devices, deciding where to place the sensors is a fundamental task. One approach is to assume that sensors have a fixed sensing radius and to solve the task as an instance of the art-gallery problem [*c.f.*, Gonzalez-Banos and Latombe, 2001, Hochbaum and Maas, 1985], similar to the *set covering* approach discussed in Section 5.1.1. In practice, however, this geometric assumption is too strong; sensors make noisy measurements about the nearby environment, and this "sensing area" is not usually characterized by a regular disk, as illustrated by the temperature correlations in Figure 6.2(a). In addition, note that correlations can be both positive and negative, as shown in Figure 6.2(b), which again is not well-characterized by a disk model. Fundamentally, the notion that a single sensor needs to predict values in a nearby region is too strong. Often, correlations may be too weak to enable prediction from a single sensor. In other settings, a location may be "too far" from existing sensors to enable good prediction if we only consider one of them, but combining data from multiple sensors we can obtain accurate predictions. This notion of combination of data from multiple sensors in complex spaces is not easily characterized by existing geometric models.

An alternative approach from spatial statistics [*c.f.*, Caselton and Zidek, 1984, Cressie, 1991], making weaker assumptions than the geometric approach, is to use a pilot deployment or expert knowledge to learn a probabilistic *Gaussian process* (GP) model for the phenomena, a non-parametric generalization of linear regression that allows for the representation of uncertainty about predictions made over the sensed field. We can use data from a pilot study or expert knowledge to learn the (hyper-)parameters of this GP. The learned GP model can then be used to predict the effect of placing sensors at particular locations, and thus optimize their positions.[1]

Given a GP model, many criteria have been proposed for characterizing the quality of placements, including placing sensors at the points of highest entropy (variance) in the GP model, and A-, D-, or E-optimal design, and mutual information (*c.f.*, Shewry and Wynn 1987, Caselton and Zidek 1984, Cressie 1991,

[1]This initial GP is, of course, a rough model, and a sensor placement strategy can be viewed as an inner-loop step for a sequential, or *active learning* algorithm [MacKay, 2003]. We discuss such approaches in more depth in the last part of this Thesis. Alternatively, if we can characterize the uncertainty about the parameters of the model, we can explicitly optimize the placements over possible models [Zhu and Stein, 2006, Zidek et al., 2000, Zimmerman, 2006]. More details on this approach are given in Chapter 10

Zhu and Stein 2006, Zimmerman 2006). A typical sensor placement technique is to greedily add sensors where uncertainty about the phenomena is highest, i.e., the highest entropy location of the GP [Cressie, 1991, Shewry and Wynn, 1987]. Unfortunately, this criterion suffers from a significant flaw: entropy is an *indirect* criterion, not considering the prediction quality of the selected placements. The highest entropy set, i.e., the sensors that are most uncertain about each other's measurements, is usually characterized by sensor locations that are as far as possible from each other. Thus, the entropy criterion tends to place sensors along the borders of the area of interest [Ramakrishnan et al., 2005], e.g., Figure 6.5. Since a sensor usually provides information about the area around it, a sensor on the boundary "wastes" sensed information.

An alternative criterion, proposed by Caselton and Zidek [1984], *mutual information*, seeks to find sensor placements that are most informative about unsensed locations. This optimization criterion *directly* measures the effect of sensor placements on the posterior uncertainty of the GP. In this chapter, we consider the observation selection problem of selecting placements which maximize this criterion. We first prove that maximizing mutual information is an $\mathbf{NP}$-complete problem. We then show that mutual information is a *submodular* function (*c.f.*, Chapter 5). Unfortunately, mutual information does *not* satisfy *nondecreasingness*, a requirement which is fundamental, e.g., in the analysis of the greedy algorithm as discussed in Chapter 5. A key result which we establish in this chapter is that if the number of placed sensors is small in comparison to the total number of possible locations, then mutual information is *approximately* nondecreasing. This result allows us the apply the techniques described in Chapter 5 to optimize sensor placements. For example, the CELF algorithm is guaranteed to provide a constant-factor approximation to the problem of finding the best set of sensor locations in polynomial time. To the best of our knowledge, no such guarantee exists for any other GP-based sensor placement approach.

Though polynomial, the complexity of our basic algorithm is relatively high due to the cost of inference in the GP – $\mathcal{O}(kn^4)$ to select k out of n possible sensor locations. We address this problem in two ways: First, we show that the lazy evaluation technique discussed in Section 5.4 typically significantly reduces the number of sensor locations that need to be checked, thus greatly speeding up computation. Second, we show that if we exploit locality in sensing areas by trimming low covariance entries, we reduce the complexity to $\mathcal{O}(kn)$.

We provide a very extensive experimental evaluation, showing that data-driven placements outperform placements based on geometric considerations only. We also show that the *mutual information* criterion leads to improved prediction accuracies with a reduced number of sensors compared to several more commonly considered experimental design criteria, such as an entropy-based criterion, and A-optimal, D-optimal and E-optimal design criteria.

In summary, in this chapter,

- We approach the problem of maximizing the information-theoretic *mutual information* criterion of Caselton and Zidek [1984] for optimizing sensor placements, empirically demonstrating its advantages over more commonly used criteria.
- Even though we prove $\mathbf{NP}$-hardness of the optimization problem, we present a polynomial time approximation algorithm with constant factor approximation guarantee, by exploiting *submodularity*. To the best of our knowledge, no such guarantee exists for any other GP-based sensor placement approach, and for any other criterion.
- We also show that submodularity provides online bounds for the quality of our solution, which can be used in the development of efficient branch-and-bound search techniques, or to bound the quality

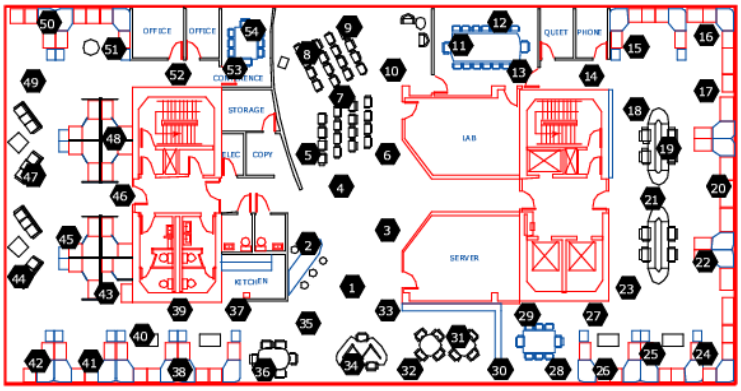


Figure 6.1: A deployment of a sensor network with 54 nodes at the Intel Berkeley Lab.

of the solutions obtained by other algorithms.

- We provide two practical techniques that significantly speed up the algorithm, and prove that they have no or minimal effect on the quality of the answer.
- Extensive empirical evaluation of our methods on several real-world sensor placement problems and comparisons with several classical design criteria.

This chapter is organized as follows. In Section 6.1, we introduce Gaussian Processes. We review the mutual information criterion in Section 6.2, and describe our approximation algorithm to optimize mutual information in Section 6.3. Section 6.4 presents several approaches towards making the optimization more computationally efficient. Section 6.5 describes related work, and Section 6.6 presents our experiments.

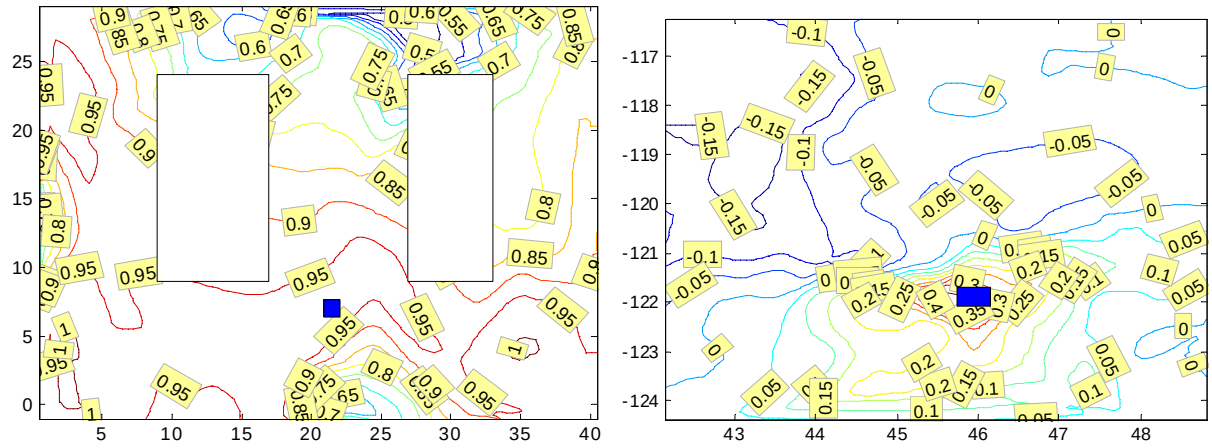
6.1 Gaussian Processes

In this section, we review *Gaussian Processes*, the probabilistic model for spatial phenomena that forms the basis of our sensor placement algorithms.

6.1.1 Modeling sensor data using the multivariate normal distribution

Consider, for example, the sensor network we deployed as shown in Figure 6.1 that measures a temperature field at 54 discrete locations. In order to predict the temperature at one of these locations from the other sensor readings, we need the joint distribution over temperatures at the 54 locations. A simple, yet often effective [*c.f.*, Deshpande et al., 2004], approach is to assume that the temperatures have a (multivariate) Gaussian joint distribution. Denoting the set of locations as $\mathcal{V}$, in our sensor network example $|\mathcal{V}| = 54$, we have a set of $n = |\mathcal{V}|$ corresponding random variables $\mathcal{X}_\mathcal{V}$ with joint distribution:

$$P(\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}) = \frac{1}{(2\pi)^{n/2}|\Sigma_{\mathcal{V}\mathcal{V}}|} e^{-\frac{1}{2}(\mathbf{x}_\mathcal{V}-\mu_\mathcal{V})^T \Sigma_{\mathcal{V}\mathcal{V}}^{-1}(\mathbf{x}_\mathcal{V}-\mu_\mathcal{V})},$$



(a) *Temperature correlations* (b) *Precipitation correlations*

Figure 6.2: Correlations are often nonstationary as illustrated by (b) temperature data from the sensor network deployment in Figure 6.1, showing the correlation between a sensor placed on the blue square and other possible locations; (c) precipitation data from measurements made across the Pacific Northwest, Figure 6.12(b).

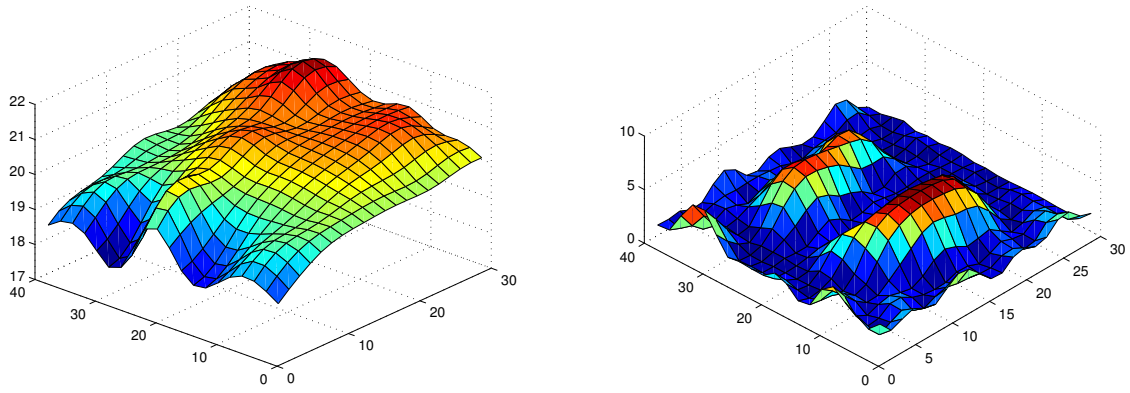
where $\mu_{\mathcal{V}}$ is the mean vector and $\Sigma_{\mathcal{V}\mathcal{V}}$ is the covariance matrix. Interestingly, if we consider a subset, $\mathcal{A} \subseteq \mathcal{V}$, of our random variables, denoted by $\mathcal{X}_{\mathcal{A}}$, then their joint distribution is also Gaussian.

6.1.2 Modeling sensor data using Gaussian Processes

In our sensor network example, we are not just interested in temperatures at sensed locations, but also at locations where no sensors were placed. In such cases, we can use regression techniques to perform prediction [Golub and Van Loan, 1989, Hastie et al., 2003]. Although linear regression often gives excellent predictions, there is usually no notion of uncertainty about these predictions, e.g., for Figure 6.1, we are likely to have better temperature estimates at points near existing sensors, than in the two central areas that were not instrumented. A *Gaussian process* (GP) is a natural generalization of linear regression that allows us to consider uncertainty about predictions.

Intuitively, a GP generalizes multivariate Gaussians to an infinite number of random variables. In analogy to the multivariate Gaussian above where the the index set $\mathcal{V}$ was finite, we now have a (possibly uncountably) infinite index set $\mathcal{V}$. In our temperature example, $\mathcal{V}$ would be a subset of $\mathbb{R}^2$, and each index would correspond to a position in the lab. GPs have been widely studied [*c.f.*, Lindley and Smith, 1972, MacKay, 2003, O'Hagan, 1978, Paciorek, 2003, Seeger, 2004a, Shewry and Wynn, 1987], and generalize Kriging estimators commonly used in geostatistics [Cressie, 1991].

An important property of GPs is that for every finite subset $\mathcal{A}$ of the indices $\mathcal{V}$, which we can think about as locations in the plane, the joint distribution over the corresponding random variables $\mathcal{X}_{\mathcal{A}}$ is Gaussian, e.g., the joint distribution over temperatures at a finite number of sensor locations is Gaussian. In order to specify this distribution, a GP is associated with a *mean function* $\mathcal{M}(\cdot)$, and a symmetric positive-definite *kernel function* $\mathcal{K}(\cdot,\cdot)$, often called the covariance function. For each random variable with index $u \in \mathcal{V}$, its mean μ_u is given by $\mathcal{M}(u)$. Analogously, for each pair of indices $u, v \in \mathcal{V}$, their covariance σ_{uv} is given by $\mathcal{K}(u, v)$. For simplicity of notation, we denote the mean vector of some set of variables $\mathcal{X}_{\mathcal{A}}$ by



(a) *Temperature prediction using GP*

(b) *Variance of temperature prediction*

Figure 6.3: Posterior mean and variance of the temperature GP estimated using all sensors: (a) Predicted temperature; (b) predicted variance.

$\mu_{\mathcal{A}}$, where the entry for element u of $\mu_{\mathcal{A}}$ is $\mathcal{M}(u)$. Similarly, we denote their covariance matrix by $\Sigma_{\mathcal{A}\mathcal{A}}$, where the entry for u, v is $\mathcal{K}(u, v)$.

The GP representation is extremely powerful. For example, if we observe a set of sensor measurements $\mathcal{X}_{\mathcal{A}} = \mathbf{x}_{\mathcal{A}}$ corresponding to the finite subset $\mathcal{A} \subseteq \mathcal{V}$, we can predict the value at any point $y \in \mathcal{V}$ conditioned on these measurements, $P(\mathcal{X}_y \mid x_{\mathcal{A}})$. The distribution of $\mathcal{X}_y$ given these observations is a Gaussian whose conditional mean $\mu_{y|\mathcal{A}}$ and variance $\sigma^2_{y|\mathcal{A}}$ are given by:

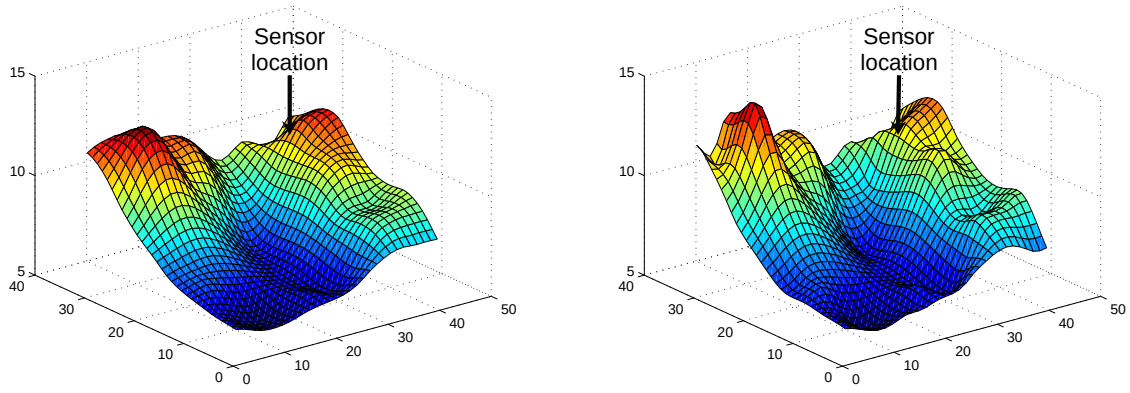
$$\mu_{y|\mathcal{A}} = \mu_y + \Sigma_{y\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}(x_{\mathcal{A}} - \mu_{\mathcal{A}}), \tag{6.1}$$

$$\sigma^2_{y|\mathcal{A}} = \mathcal{K}(y, y) - \Sigma_{y\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}y}, \tag{6.2}$$

where $\Sigma_{y\mathcal{A}}$ is a covariance vector with one entry for each $u \in \mathcal{A}$ with value $\mathcal{K}(y, u)$, and $\Sigma_{\mathcal{A}y} = \Sigma^T_{y\mathcal{A}}$. Figure 6.3(a) and Figure 6.3(b) show the posterior mean and variance derived using these equations on 54 sensors at Intel Labs Berkeley. Note that two areas in the center of the lab were not instrumented. These areas have higher posterior variance, as expected. An important property of GPs is that the posterior variance (6.2) does not depend on the actual observed values $x_{\mathcal{A}}$. Thus, for a given kernel function, the variances in Figure 6.3(b) will not depend on the observed temperatures. This property will be important when analyzing sequential design in GPs (*c.f.*, Chapter 14).

6.1.3 Nonstationarity

In order to compute predictive distributions using (6.1) and (6.2), the mean and kernel functions have to be known. The mean function can usually be estimated using regression techniques. Estimating kernel functions is difficult, and usually, strongly limiting assumptions are made. For example, it is commonly assumed that the kernel $\mathcal{K}(u, v)$ is *stationary*, which means that the kernel depends only on the difference between the locations, considered as vectors v, u, i.e., $\mathcal{K}(u, v) = \mathcal{K}_\theta(u - v)$. Hereby, θ is a set of parameters. Very often, the kernel is even assumed to be *isotropic*, which means that the covariance only depends



(a) *Example kernel function.* (b) *Data from the empirical covariance matrix.*

Figure 6.4: Example kernel function learned from the Berkeley Lab temperature data: (a) learned covariance function $\mathcal{K}(x, \cdot)$, where x is the location of sensor 41; (b) "ground truth", interpolated empirical covariance values for the same sensors. Observe the close match between predicted and measured covariances.

on the distance between locations, i.e., $\mathcal{K}(u, v) = \mathcal{K}_\theta(||u - v||_2)$. Common choices for isotropic kernels are the exponential kernel,

$$\mathcal{K}_\theta(\delta) = \exp\left(-\frac{|\delta|}{\theta}\right), \tag{6.3}$$

and the Gaussian kernel,

$$\mathcal{K}_\theta(\delta) = \exp\left(-\frac{\delta^2}{\theta^2}\right). \tag{6.4}$$

These assumptions are frequently strongly violated in practice, as illustrated in the real sensor data shown in Figures 6.2(a) and 6.2(b). In Section 6.5.2, we discuss how placements optimized from models with isotropic kernels reduce to geometric covering and packing problems.

In this chapter, we *do not* assume that $\mathcal{K}(\cdot, \cdot)$ is stationary or isotropic. Our approach is general, and can use *any* kernel function. In our experiments, we use the approach of Nott and Dunsmuir [2002] to estimate nonstationary kernels from data collected by an initial deployment. More specifically, their assumption is that an estimate of the empirical covariance $\Sigma_{\mathcal{A}\mathcal{A}}$ at a set of observed locations is available, and that the process can be locally described by a collection of isotropic processes, associated with a set of reference points. Details of this method are reviewed in Appendix A.3. An example of a kernel function estimated using this method is presented in Figure 6.4(a). In Section 6.6.2, we show that placements based on such nonstationary GPs lead to far better prediction accuracies than those obtained from isotropic kernels.

6.2 Optimizing sensor placements

Usually, we can only deploy a small number of sensors, and thus must carefully choose where to place them. In spatial statistics this optimization is called *sampling* or *experimental design*: finding the k best sensor locations out of a finite subset $\mathcal{V}$ of possible locations, e.g., out of a grid discretization of $\mathbb{R}^2$.

6.2.1 The entropy criterion

We first have to define what a good design is. Intuitively, we want to place sensors which are most informative with respect to the entire design space. A natural notion of uncertainty is the conditional entropy of the unobserved locations $\mathcal{V} \setminus \mathcal{A}$ after placing sensors at locations $\mathcal{A}$,

$$H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}) = -\int p(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}}, \mathbf{x}_{\mathcal{A}}) \log p(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathbf{x}_{\mathcal{A}}) d\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}} d\mathbf{x}_{\mathcal{A}}, \tag{6.5}$$

where we use $\mathcal{X}_{\mathcal{A}}$ and $\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}$ to refer to sets of random variables at the locations $\mathcal{A}$ and $\mathcal{V} \setminus \mathcal{A}$. Intuitively, minimizing this quantity aims at finding the placement which results in the lowest uncertainty about all uninstrumented locations $\mathcal{V} \setminus \mathcal{A}$ after observing the placed sensors $\mathcal{A}$. A good placement would therefore minimize this conditional entropy, i.e., we want to find

$$\mathcal{A}^* = \operatorname*{argmin}_{\mathcal{A}\subseteq\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}).$$

Using the identity $H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}) = H(\mathcal{X}_{\mathcal{V}}) - H(\mathcal{X}_{\mathcal{A}})$, we can see that

$$\mathcal{A}^* = \operatorname*{argmin}_{\mathcal{A}\subseteq\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}) = \operatorname*{argmax}_{\mathcal{A}\subseteq\mathcal{V}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{A}}).$$

In the terminology of Section 2.3, this criterion corresponds to choosing the sensing quality function

$$u(\mathbf{x}_{\mathcal{V}}, \mathcal{A}) = -\log_2 P(\mathbf{x}_{\mathcal{A}}),$$

and then solving the maximization problem (2.4) with the cost $C(\mathcal{A})$ defined as $C(\mathcal{A}) = |\mathcal{A}|$, and budget $B = k$. So we can see that we need to find a set of k sensors $\mathcal{A}$ which is most uncertain about each other. Unfortunately, this optimization problem, often also referred to as D-optimal design in the experiment design literature [*c.f.*, Currin et al., 1991], has been shown to be **NP**-hard [Ko et al., 1995]:

Theorem 6.1 (Ko et al., 1995)**.** *Given rational M and rational covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}}$ over Gaussian random variables $\mathcal{V}$, deciding whether there exists a subset $\mathcal{A} \subseteq \mathcal{V}$ of cardinality k such that $H(\mathcal{X}_{\mathcal{A}}) \geq M$ is* **NP***-complete.*

Therefore, the following greedy heuristic has found common use [Cressie, 1991, McKay et al., 1979]: One starts from an empty set of locations, $\mathcal{A}_0 = \emptyset$, and greedily adds sensors until $|\mathcal{A}| = k$. At each iteration, starting with set $\mathcal{A}_i$, the greedy rule used is to add the location $y^*_H \in \mathcal{V} \setminus \mathcal{A}$ that has highest conditional entropy,

$$y^*_H = \operatorname*{argmax}_y H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{A}_i}), \tag{6.6}$$

i.e., the location we are most uncertain about given the sensors placed thus far. If the set of selected locations at iteration i is $\mathcal{A}_i = \{y_1, \ldots, y_i\}$, using the chain-rule of entropies, we have that:

$$H(\mathcal{X}_{\mathcal{A}_i}) = H(\mathcal{X}_{y_i} \mid \mathcal{X}_{\mathcal{A}_{i-1}}) + \ldots + H(\mathcal{X}_{y_2} \mid \mathcal{X}_{\mathcal{A}_1}) + H(\mathcal{X}_{y_1} \mid \mathcal{X}_{\mathcal{A}_0}).$$

Note that the (differential) entropy of a Gaussian random variable $\mathcal{X}_y$ conditioned on some set of variables $\mathcal{X}_{\mathcal{A}}$ is a nondecreasing function of its variance:

$$H(\mathcal{X}_y|\mathcal{X}_{\mathcal{A}}) = \frac{1}{2}\log(2\pi e\sigma^2_{\mathcal{X}_y|\mathcal{X}_{\mathcal{A}}}) = \frac{1}{2}\log\sigma^2_{\mathcal{X}_y|\mathcal{X}_{\mathcal{A}}} + const, \tag{6.7}$$

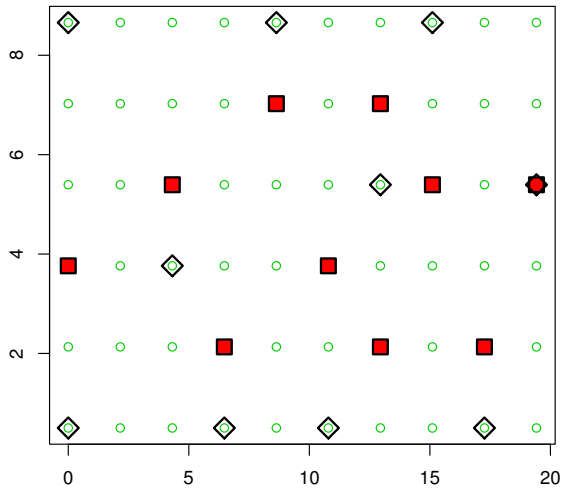


Figure 6.5: An example of placements chosen using entropy and mutual information criteria on a subset of the temperature data from the Intel deployment. Diamonds indicate the positions chosen using entropy; squares the positions chosen using MI.

which can be computed in closed form using Equation (6.2). Since for a fixed kernel function, the variance does not depend on the observed values, this optimization can be done before deploying the sensors, i.e., a sequential, closed-loop design taking into account previous measurements bears no advantages over an open-loop design, performed before any measurements are made. If the parameters are not known, this is not necessarily true. We will study this sequential design problem in Chapter 14.

6.2.2 An improved design criterion: mutual information

The entropy criterion described above is intuitive for finding sensor placements, since the sensors that are most uncertain about each other should cover the space well. Unfortunately, this entropy criterion suffers from the problem shown in Figure 6.5, where sensors are placed far apart along the boundary of the space. Since we expect predictions made from a sensor measurement to be most precise in a region around it, such placements on the boundary are likely to "waste" information. This phenomenon has been noticed previously by Ramakrishnan et al. [2005], who proposed a weighting heuristic. Intuitively, this problem arises because the entropy criterion is *indirect*: the criterion only considers the entropy of the selected sensor locations, rather than considering prediction quality over the space of interest. This indirect quality of the entropy criterion is surprising, since the criterion was derived from the "predictive" formulation $H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}})$ in Equation (6.5), which is equivalent to maximizing $H(\mathcal{X}_{\mathcal{A}})$.

Caselton and Zidek [1984] proposed a different optimization criterion, which searches for the subset of sensor locations that most significantly reduces the uncertainty about the estimates in the rest of the space. More formally, we consider our space as a discrete set of locations $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$ composed of two parts: a set $\mathcal{S}$ of possible positions where we can place sensors, and another set $\mathcal{U}$ of positions of interest, where no sensor placements are possible. The goal is to place a set of k sensors that will give us good predictions at all uninstrumented locations $\mathcal{V} \setminus \mathcal{A}$. Specifically, we want to find

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A}\subseteq\mathcal{S}:|\mathcal{A}|=k} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}), \tag{6.8}$$

that is, the set $\mathcal{A}^*$ that maximally reduces the entropy over the rest of the space $\mathcal{V} \setminus \mathcal{A}^*$. Note that this criterion $F_{MI}(\mathcal{A}) \equiv H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}})$ is equivalent to finding the set that maximizes the *mutual information* $I(\mathcal{X}_{\mathcal{A}}; \mathcal{X}_{\mathcal{V}\setminus\mathcal{A}})$ between the locations $\mathcal{A}$ and the rest of the space $\mathcal{V} \setminus \mathcal{A}$. In the notiation from Section 2.3, this criterion corresponds to choosing the utility function

$$u(\mathbf{x}_{\mathcal{V}}, \mathcal{A}) = \log_2 P(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathbf{x}_{\mathcal{A}}) - \log_2 P(\mathbf{x}_{\mathcal{V}\setminus\mathcal{A}}).$$

In their follow-up work, Caselton et al. [1992] and Zidek et al. [2000], argue against the use of mutual information in a setting where the entropy $H(\mathcal{X}_{\mathcal{A}})$ in the observed locations constitutes a significant part of the total uncertainty $H(\mathcal{X}_{\mathcal{V}})$. Caselton et al. [1992] also argue that, in order to compute $F_{MI}(\mathcal{A})$, one needs an accurate model of $P(\mathcal{X}_{\mathcal{V}})$. Since then, the entropy criterion has been dominantly used as a placement criterion. Nowadays however, the estimation of complex nonstationary models for $P(\mathcal{X}_{\mathcal{V}})$, as well as computational aspects, are quite well understood and handled. Furthermore, we show empirically, that even in the sensor selection case, mutual information outperforms entropy on several practical placement problems.

On the same simple example in Figure 6.5, this mutual information criterion leads to intuitively appropriate central sensor placements that do not have the "wasted information" property of the entropy criterion. Our experimental results in Section 6.6 further demonstrate the advantages in performance of the mutual information criterion. Notice that in our notation $F_{MI}(\mathcal{A}) = I(\mathcal{X}_{\mathcal{A}}; \mathcal{X}_{\mathcal{V}\setminus\mathcal{A}})$ the process $\mathcal{X}$ and the set of locations $\mathcal{V}$ is implicit.

Similarly to the entropy criterion, mutual information is also hard to optimize:
Theorem 6.2. *Given rational M and a rational covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}}$ over Gaussian random variables $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$, deciding whether there exists a subset $\mathcal{A} \subseteq \mathcal{S}$ of cardinality k such that $F_{MI}(\mathcal{A}) \geq M$ is* $\mathbf{NP}$*-complete.*

Due to the problem complexity, we cannot expect to find optimal solutions in polynomial time in general. However, if we implement the simple greedy algorithm for the mutual information criterion (details given below), and optimize designs on real-world placement problems, we see that the greedy algorithm gives almost optimal solutions, as presented in Figure 6.6. In this small example, where we could compute the optimal solution, the performance of the greedy algorithm was at most five percent worse than the optimal solution. In the following sections, we will give theoretical bounds and empirical evidence justifying this near-optimal behavior.

6.3 Approximation algorithm

Optimizing the mutual information criterion is an $\mathbf{NP}$-complete problem. In the following, we show that the greedy algorithm provides a constant-factor approximation guarantee, by exploiting the concept of submodularity (*c.f.*, Chapter 5).

6.3.1 The greedy algorithm

The greedy algorithm simply adds sensors in sequence, choosing the next sensor which provides the maximum increase in mutual information. More formally, using $F_{MI}(\mathcal{A}) = I(\mathcal{X}_{\mathcal{A}}; \mathcal{X}_{\mathcal{V}\setminus\mathcal{A}})$, our goal is to

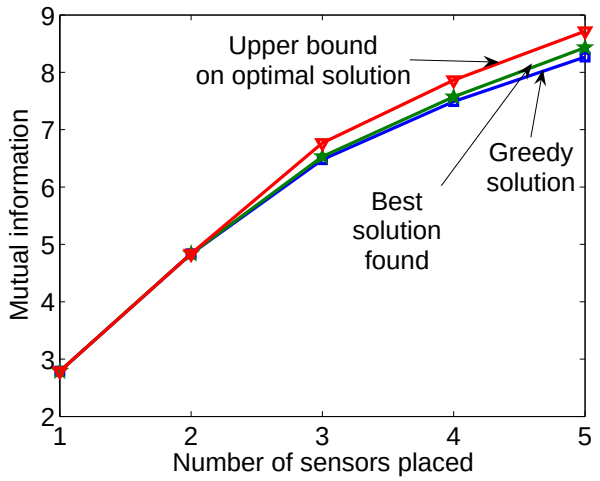


Figure 6.6: Comparison of the greedy algorithm with the optimal solutions on a small problem. We select from 1 to 5 sensor locations out of 16, on the Intel Berkeley temperature data set as discussed in Section 6.6. The greedy algorithm is always within 95 percent of the optimal solution.

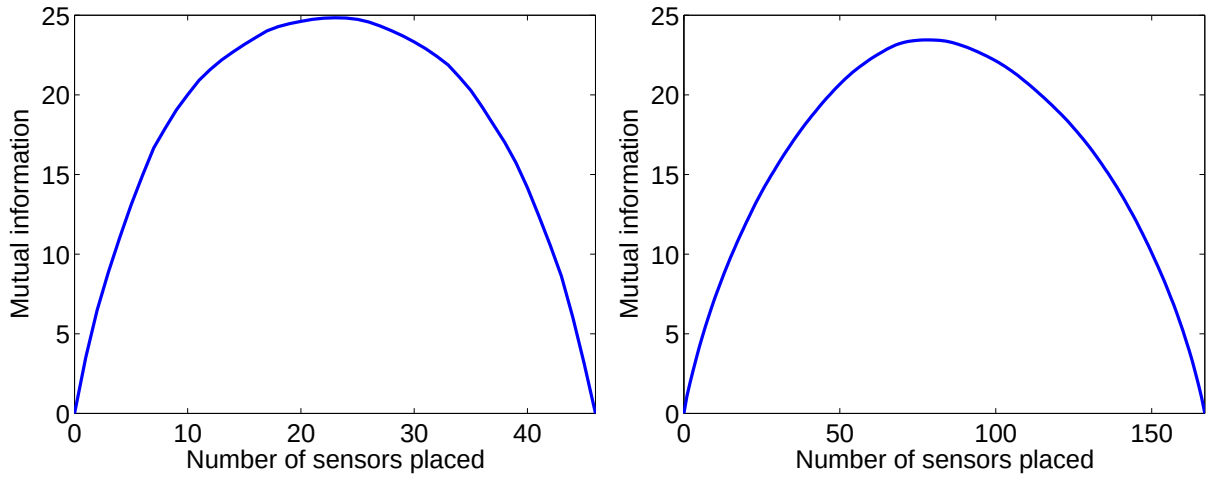
Algorithm 6.1: Approximation algorithm for maximizing mutual information.

Input: Covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}}$, k, $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$
Output: Sensor selection $\mathcal{A} \subseteq \mathcal{S}$
begin
  $\mathcal{A} \leftarrow \emptyset$;
  for $j = 1$ **to** k **do**
    for $y \in \mathcal{S} \setminus \mathcal{A}$ **do** $\delta_y \leftarrow \dfrac{\sigma_y^2 - \Sigma_{y\mathcal{A}}\Sigma_{\mathcal{A}\mathcal{A}}^{-1}\Sigma_{\mathcal{A}y}}{\sigma_y^2 - \Sigma_{y\bar{\mathcal{A}}}\Sigma_{\bar{\mathcal{A}}\bar{\mathcal{A}}}^{-1}\Sigma_{\bar{\mathcal{A}}y}}$;
1
    $y^* \leftarrow \operatorname{argmax}_{y \in \mathcal{S}\setminus\mathcal{A}} \delta_y$;
2
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{y^*\}$;
  end
end

greedily select the next sensor y that maximizes:

$$\begin{aligned} F_{MI}(\mathcal{A} \cup \{y\}) - F_{MI}(\mathcal{A}) &= H(\mathcal{X}_{\mathcal{A}} \cup \{\mathcal{X}_y\}) - H(\mathcal{X}_{\mathcal{A}} \cup \{\mathcal{X}_y\} \mid \mathcal{X}_{\bar{\mathcal{A}}}) - \left[H(\mathcal{X}_{\mathcal{A}}) - H(\mathcal{X}_{\mathcal{A}} \mid \mathcal{X}_{\bar{\mathcal{A}}\cup\{y\}})\right], \\ &= H(\mathcal{X}_{\mathcal{A}} \cup \{\mathcal{X}_y\}) - H(\mathcal{X}_{\mathcal{V}}) + H(\mathcal{X}_{\bar{\mathcal{A}}}) - [H(\mathcal{X}_{\mathcal{A}}) - H(\mathcal{X}_{\mathcal{V}}) + H(\mathcal{X}_{\bar{\mathcal{A}}} \cup \{\mathcal{X}_y\})] \\ &= H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{A}}) - H(\mathcal{X}_y \mid \mathcal{X}_{\bar{\mathcal{A}}}). \end{aligned} \tag{6.9}$$

Note that the greedy rule for entropy in Equation (6.6) only considers the $H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{A}})$ part of Equation (6.9), measuring the uncertainty of location y with respect to the placements $\mathcal{A}$. In contrast, the greedy rule for mutual information trades off this uncertainty with $-H(\mathcal{X}_y \mid \mathcal{X}_{\bar{\mathcal{A}}})$, which forces us to pick a y that is "central" with respect to the unselected locations $\bar{\mathcal{A}}$, since those "central" locations will result in the least conditional entropy $H(\mathcal{X}_y \mid \mathcal{X}_{\bar{\mathcal{A}}})$. Expanding the definition of conditional entropy in Equation (6.7), Algorithm 6.1 summarizes the greedy sensor placement algorithm.



(a) *Temperature data* (b) *Precipitation data*

Figure 6.7: Mutual information of greedy sets of increasing size. It can be seen that clearly mutual information is not nondecreasing. MI is nondecreasing, however, in the initial part of the curve corresponding to small placements. This allows us to prove approximate nondecreasingness.

6.3.2 An approximation bound

We now prove that, if the discretization $\mathcal{V}$ of locations of interest in the Gaussian process is fine enough, our greedy algorithm gives a $(1 - 1/e)$ approximation, approximately 63% of the optimal sensor placement: If the algorithm returns set $\widehat{A}$, then

$$F_{MI}(\widehat{A}) \geq (1 - 1/e) \max_{\mathcal{A} \subset \mathcal{S}, |\mathcal{A}| = k} F_{MI}(\mathcal{A}) - k\varepsilon,$$

for some small $\varepsilon > 0$.

To prove this result, we use the concept of submodularity as reviewed in Chapter 5. In order to verify this property, we use the diminishing returns characterization of submodularity: a set function F is submodular if for all $\mathcal{A} \subseteq \mathcal{A}' \subseteq \mathcal{V}$ and $y \in \mathcal{V} \setminus \mathcal{A}'$ it holds that $F(\mathcal{A} \cup \{y\}) - F(\mathcal{A}) \geq F(\mathcal{A}' \cup \{y\}) - F(\mathcal{A}')$. In the case of $F(\mathcal{A}) = F_{MI}(\mathcal{A})$, we need to consider the increments, $F_{MI}(\mathcal{A} \cup \{s\}) - F_{MI}(\mathcal{A})$ which are equal to $H(y \mid \mathcal{A}) - H(y \mid \bar{\mathcal{A}})$ as shown above. Now, the "information never hurts" bound proves that $H(y \mid \mathcal{A}) \geq H(y \mid \mathcal{A}')$ whenever $\mathcal{A} \subseteq \mathcal{A}'$ [Cover and Thomas, 1991]. Hence, this bound immediately proves that

$$F_{MI}(\mathcal{A}' \cup \{y\}) - F_{MI}(\mathcal{A}') \leq F_{MI}(\mathcal{A} \cup \{y\}) - F_{MI}(\mathcal{A}),$$

whenever $\mathcal{A} \subseteq \mathcal{A}'$, and for an arbitrary choice $y \in \mathcal{V} \setminus \mathcal{A}'$, i.e., adding y to $\mathcal{A}$ helps more than adding y to $\mathcal{A}'$. Hence we have shown:

Lemma 6.3. *The set function $\mathcal{A} \mapsto F_{MI}(\mathcal{A})$ is submodular.*

As explained in Chapter 5, the analysis of the greedy algorithm by Nemhauser et al. [1978] does not only require submodularity, but also *nondecreasingness*, i.e., that $F_{MI}(\mathcal{A}) \leq F_{MI}(\mathcal{A}')$ whenever $\mathcal{A} \leq \mathcal{A}'$. The third requirement, $F_{MI}(\emptyset) = I(\emptyset; \mathcal{V}) = 0$ is clearly satisfied.

However, the nondecreasingness of mutual information is not apparent. Since

$$F_{MI}(\mathcal{V}) = I(\mathcal{V}, \emptyset) = I(\emptyset, \mathcal{V}) = F_{MI}(\emptyset) = 0,$$

the objective function will increase and then decrease, and, thus, is *not* nondecreasing, as shown in Figures 6.7(a) and 6.7(b). Fortunately, the proof of Nemhauser et al. [1978] does not use nondecreasingness for all possible sets, it is sufficient to prove that F_{MI} is nondecreasing for all sets of size up to $2k$. Intuitively, mutual information is not nondecreasing when the number of chosen sensor locations approaches $|\mathcal{V}|/2$. If the discretization level $|\mathcal{V}|$ is significantly larger than $2k$ points, then mutual information should meet the conditions of the proof of Theorem 5.2.

Thus the heart of our analysis of Algorithm 6.1 will be to prove that if the discretization of the Gaussian process is fine enough, then mutual information is *approximately nondecreasing* for sets of size up to $2k$. More precisely we prove the following result:

Lemma 6.4. *Let $\mathcal{X}$ be a Gaussian process on a compact subset $\mathcal{C}$ of $\mathbb{R}^m$ with a positive-definite, continuous covariance kernel $\mathcal{K} : \mathcal{C} \times \mathcal{C} \to \mathbb{R}_0^+$. Assume the sensors have a measurement error with variance at least σ^2. Then, for any $\varepsilon > 0$, and any finite maximum number k of sensors to place there exists a discretization $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$, $\mathcal{S}$ and $\mathcal{U}$ having mesh width δ such that*

$$F_{MI}(\mathcal{A} \cup \{y\}) \geq F_{MI}(\mathcal{A}) - \varepsilon$$

for all $\mathcal{A} \subseteq \mathcal{S}$, $|\mathcal{A}| \leq 2k$ and $y \in \mathcal{V} \setminus \mathcal{A}$.

If the covariance function is Lipschitz-continuous, such as the Gaussian Radial Basis Function (RBF) kernel, the following corollary gives a bound on the required discretization level with respect to the Lipschitz constant:

Corollary 6.5. *If $\mathcal{K}$ is Lipschitz-continuous with constant L, then the required discretization is*

$$\delta \leq \frac{\varepsilon\sigma^6}{4kLM\left(\sigma^2 + 2k^2M + 6k^2\sigma^2\right)},$$

where $M = \max_{x \in C} \mathcal{K}(x, x)$, for $\varepsilon < \min(M, 1)$.

Corollary 6.5 guarantees that for any $\varepsilon > 0$, a polynomial discretization level is sufficient to guarantee that mutual information is $\varepsilon-$approximately nondecreasing. These bounds on the discretization are, of course, worst case bounds. The worst-case setting occurs when the sensor placements $\mathcal{A}$ are arbitrarily close to each other, since the entropy part $H(\mathcal{X}_y \mid \mathcal{X}_\mathcal{A})$ in Equation (6.9) can become negative. Since most GPs are used for modeling physical phenomena, both the optimal sensor placement and the sensor placement produced by the greedy algorithm can be expected to be spread out, and not condensed to a small region of the sensing area. Hence we expect the bounds to be very loose in the situations that arise during normal operation of the greedy algorithm.

Combining our Lemmas 6.3 and 6.4 with Theorem 5.2, we obtain our constant-factor approximation bound on the quality of the sensor placements obtained by our algorithm:

Theorem 6.6. *Under the assumptions of Lemma 6.4, Algorithm 6.1 is guaranteed to select a set $\mathcal{A}$ of k sensors for which*

$$F_{MI}(\mathcal{A}) \geq (1 - 1/e)(\mathrm{OPT} - k\varepsilon),$$

where OPT *is the value of the mutual information for the optimal placement.*

Note that our bound has two implications: First, it shows that the greedy algorithm has a guaranteed minimum performance level of $1 - 1/e$ when compared to the optimal solution. Second, our approach also provides an upper-bound on the value of the optimal placement, which can be used to bound the quality of the placements by other heuristic approaches, such as local search, that may perform better than our greedy algorithm on specific problems.

If the locations $s \in \mathcal{V}$ can have different costs, we can use Theorem 5.6 to still get approximation guarantees. Analogously we can also extend the online bounds, lazy evaluation and mixed integer programming techniques described in Chapter 5 to the case of ε-nondecreasing submodular functions.

6.3.3 Sensor placement with non-constant cost functions

In many real-world settings, the cost of placing a sensor depends on the specific location. As described in Section 2.4, such cases can often be formalized by specifying an additive cost function $C(\mathcal{A})$ and a total budget B, and the task is to select placements $\mathcal{A}$ whose total cost $C(\mathcal{A})$ is within our budget $C(\mathcal{A}) \leq B$. Recently, the submodular function maximization approach of Nemhauser et al. [1978] has been extended to address this budgeted case [Sviridenko, 2004] in the case of additive cost functions. The combination of the analysis in this chapter with these new results also yields a constant-factor $(1 - 1/e)$ approximation guarantee for the sensor placement problem with non-uniform costs.

6.3.4 Online bounds

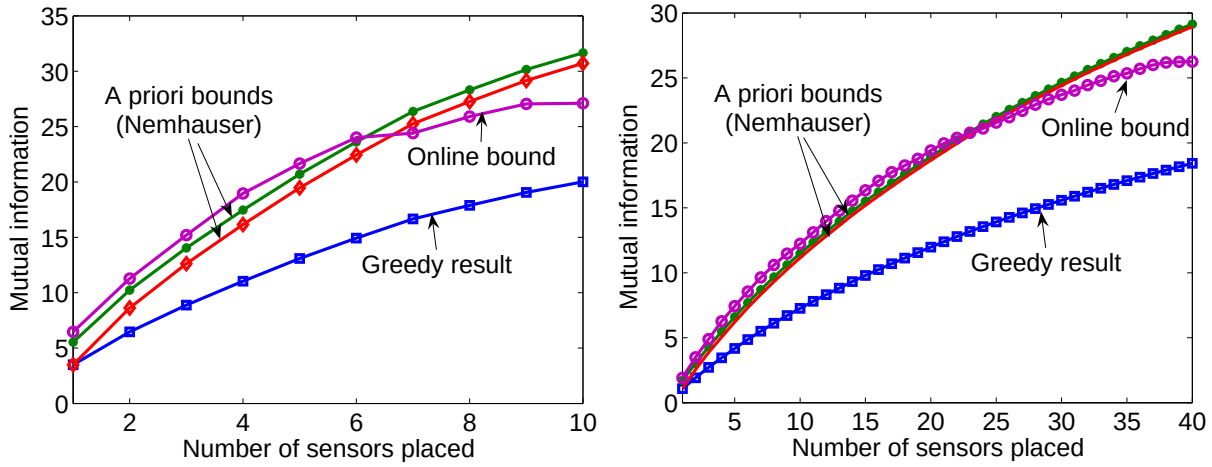
Since mutual information is approximately nondecreasing and submodular, Theorem 6.6 proves an *a priori* approximation guarantee of $(1 - 1/e)$. For most practical problems however, this bound is very loose. The bounds provided in Section 5.3 can be extended to the case of *approximately nondecreasing* submodular functions as formalized in the following observation:

Proposition 6.7. *Assume that the discretization is fine enough to guarantee ε-nondecreasingness for mutual information and sets of size at most k. Let $\mathcal{A} \subseteq \mathcal{S}$ be a placement of k sensors, and for all $y \in \mathcal{S}$, let $\delta_y = F_{MI}(\mathcal{A} \cup \{y\}) - F_{MI}(\mathcal{A})$. Sort the δ_y in decreasing order, and consider the sequence $\delta^{(1)}, \ldots, \delta^{(k)}$ of the first k elements. Then* $\mathrm{OPT} \leq F_{MI}(\mathcal{A}) + \sum_{i=1}^{k} \delta^{(i)} + k\varepsilon$.

The proof of this proposition follows directly from submodularity and ε-nondecreasingness. In many applications, especially for large placements, this bound can be much tighter than the bound guaranteed by Theorem 6.6. Figures 6.8(a) and 6.8(b) compare the a priori and online bounds for the data sets discussed in Section 6.6.1.

6.3.5 Exact optimization and using mixed integer programming

There is another way to get even tighter bounds, or even compute the optimal solution. This approach is based on branch & bound algorithm for solving a mixed integer program for nondecreasing submodular functions by Nemhauser and Wolsey [1981], as discussed in Section 5.5.1. We used this algorithm to bound the value of the optimal solution in Figure 6.6.



(a) *Temperature data* (b) *Precipitation data*

Figure 6.8: Online bounds: mutual information achieved by the greedy algorithm, the $(1 - 1/e)$ and $1 - (1 - 1/k)^k$ a priori bounds and the online bound described in Section 5.3.

The mixed integer program is given by:

$$\begin{aligned} &\max \eta; \\ &\quad \eta \;\leq\; F_{MI}(\mathcal{B}) + \sum_{y_i \in \mathcal{S}\setminus\mathcal{B}} \alpha_i \big[F_{MI}(\mathcal{B}\cup\{y_i\}) - F_{MI}(\mathcal{B})\big], \quad \forall \mathcal{B} \subseteq \mathcal{S}; \qquad (6.10) \\ &\sum_i \alpha_i \;\leq\; k, \quad \forall i; \qquad (6.11) \\ &\quad \alpha_i \;\in\; \{0,1\}, \quad \forall i; \end{aligned}$$

where $\alpha_i = 1$ means that location y_i should be selected. The MIP is solved using constraint generation, as detailed in Section 5.5.1.

The analysis of this MIP, as presented by Nemhauser and Wolsey [1981], assumes nondecreasingness. In the case of mutual information, the objective is only approximately nondecreasing. In particular, consider a a placement defined by $\alpha^{\mathcal{A}}$. Then, by submodularity, for all $\mathcal{B}$, we have that:

$$\begin{aligned} F_{MI}(\mathcal{B}) + \sum_{y_i \in \mathcal{S}\setminus\mathcal{B}} \alpha_i^{\mathcal{A}}[F_{MI}(\mathcal{B}\cup\{y_i\}) - F_{MI}(\mathcal{B})] &= F_{MI}(\mathcal{B}) + \sum_{y_i \in \mathcal{A}\setminus\mathcal{B}} [F_{MI}(\mathcal{B}\cup\{y_i\}) - F_{MI}(\mathcal{B})], \\ &\geq F_{MI}(\mathcal{A}\cup\mathcal{B}). \end{aligned}$$

By approximate nondecreasingness:

$$F_{MI}(\mathcal{A}\cup\mathcal{B}) \geq F_{MI}(\mathcal{A}) - k\varepsilon.$$

Thus, $(\hat{\eta}, \alpha^{\mathcal{A}})$, for $\hat{\eta} \leq F_{MI}(\mathcal{A}) - k\varepsilon$ is a feasible solution for the mixed integer program. Since we are maximizing η, for the optimal solution $(\eta^*, \alpha^{\mathcal{A}^*})$ of the MIP it holds that

$$F_{MI}(\mathcal{A}^*) \geq \mathrm{OPT} - k\varepsilon.$$

There is another MIP formulation for maximizing general submodular functions without the ε-nondecreasingness requirement. The details can be found in Nemhauser and Wolsey [1981]. We however found this formulation to produce much looser bounds, and to take much longer to converge.

6.4 Scaling up

Greedy updates for both entropy and mutual information require the computation of conditional entropies using Equation (6.7), which involves solving a system of $k = |\mathcal{A}|$ linear equations. For entropy maximization, where we consider $H(\mathcal{X}_y|\mathcal{X}_\mathcal{A})$ alone, the complexity of this operation is $\mathcal{O}(k^3)$. To maximize the mutual information, we also need $H(\mathcal{X}_y|\mathcal{X}_{\bar{\mathcal{A}}})$ requiring $\mathcal{O}(n^3)$, for $n = |\mathcal{V}|$. Since we need to recompute the score of all possible locations at every iteration of Algorithm 6.1, the complexity of our greedy approach for selecting k sensors is $\mathcal{O}(kn^4)$, which is not computationally feasible for very fine discretizations (large n). In Section 6.4.1 we first observe, that a lazy strategy (as discussed in Section 5.4) can be used even though mutual information is only approximately nondecreasing. This strategy frequently reduces the number of evaluations of the greedy rule, thereby often reducing the complexity to $\mathcal{O}(kn^3)$. In Section 6.4.2 we present a way of exploiting the problem structure by using local kernels, which often reduces the complexity to $\mathcal{O}(kn)$. Both approaches can be combined for even more efficient computation.

6.4.1 Lazy evaluation

In Chapter 5, we discussed how the greedy algorithm can be made more efficient by using lazy evaluations. These lazy evaluations rely on the fact, that, for a submodular function F, the greedy increments,

$$\delta_y \equiv \delta_y(A) \equiv F(\mathcal{A} \cup \{y\}) - F(\mathcal{A}),$$

are monotonically nonincreasing in $\mathcal{A}$, i.e., $\delta_y(\mathcal{A}) \geq \delta_y(\mathcal{A}')$ whenever $\mathcal{A} \subseteq \mathcal{A}'$. Note that this property does not require nondecreasingness, hence it applies to the case of mutual information $F(\mathcal{A}) = F_{MI}(\mathcal{A})$.

6.4.2 Local kernels

In this section, we exploit locality in the kernel function to speed up the algorithm significantly: First, we note that, for many GPs, correlation decreases exponentially with the distance between points. Often, variables which are far apart are actually independent. These weak dependencies can be modeled using a covariance function $\mathcal{K}$ for which $\mathcal{K}(x, \cdot)$ has compact support, i.e., that has non-zero value only for a small portion of the space. For example, consider the following isotropic covariance function proposed by Storkey [1999]:

$$\mathcal{K}(x, y) = \begin{cases} \frac{(2\pi - \Delta)(1 + (\cos\Delta)/2) + \frac{3}{2}\sin\Delta}{3\pi}, & \text{for } \Delta < 2\pi, \\ 0, & \text{otherwise,} \end{cases} \tag{6.12}$$

where $\Delta = \beta\|x - y\|_2$, for $\beta > 0$. This covariance function resembles the Gaussian kernel $\mathcal{K}(x, y) = \exp(-\beta\|x - y\|_2^2/(2\pi))$ as shown in Figure 6.9, but is zero for distances larger than $2\pi/\beta$.

Even if the covariance function does not have compact support, it can be appropriate to compute $H(y|\tilde{\mathcal{B}}) \approx H(y|\mathcal{B})$ where $\tilde{\mathcal{B}}$ results from removing all elements x from $\mathcal{B}$ for which $|\,\mathcal{K}(x, y)| \leq \varepsilon$ for some small value of ε. This truncation is motivated by noting that:

$$\sigma^2_{y|\mathcal{B}\setminus x} - \sigma^2_{y|\mathcal{B}} \leq \frac{\mathcal{K}(y, x)^2}{\sigma^2_x} \leq \frac{\varepsilon^2}{\sigma^2_x}.$$

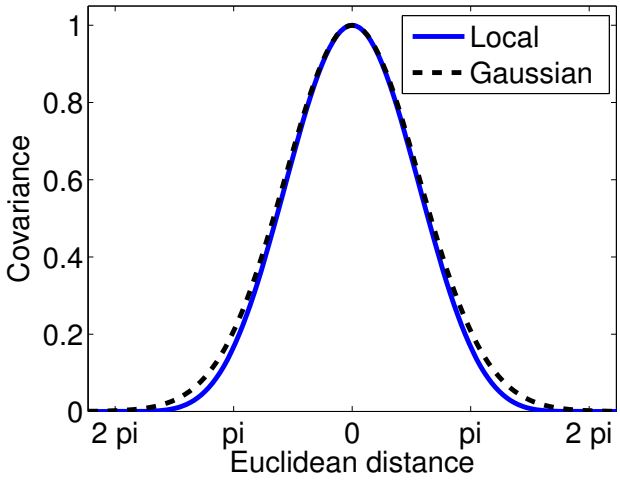


Figure 6.9: Comparison of local and Gaussian kernels.

This implies that the decrease in entropy $H(\mathcal{X}_y|\mathcal{X}_{\mathcal{B}\setminus\{x\}}) - H(\mathcal{X}_y|\mathcal{X}_{\mathcal{B}})$ is at most $\varepsilon^2/(\sigma^2\sigma_x^2)$ (using a similar argument as the one in the proof of Lemma 6.4), assuming that each sensor has independent Gaussian measurement error of at least σ^2. The total decrease of entropy $H(\mathcal{X}_y \mid \mathcal{X}_{\tilde{\mathcal{B}}}) - H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{B}})$ is bounded by $n\varepsilon^2/\sigma^4$. This truncation allows to compute $H(\mathcal{X}_y \mid \mathcal{X}_{\bar{\mathcal{A}}})$ much more efficiently, at the expense of this small absolute error. In the special case of isotropic kernels, the number d of variables x with $\mathcal{K}(x, y) > \varepsilon$ can be computed as a function of the discretization and the covariance kernel. This reduces the complexity of computing $H(\mathcal{X}_y|\mathcal{X}_{\bar{\mathcal{A}}})$ from $\mathcal{O}(n^3)$ to $\mathcal{O}(d^3)$, which is a constant.

Our truncation approach leads to the more efficient optimization algorithm shown in Algorithm 6.2. Here, $\tilde{H}_\varepsilon$ refers to the truncated computation of entropy as described above, and $N(y^*; \varepsilon) \leq d$ refers to the set of elements $x \in \mathcal{S}$ for which $|\mathcal{K}(y^*, x)| > \varepsilon$. Using this approximation, our algorithm is significantly faster: Initialization (Line 6) requires $\mathcal{O}(nd^3)$ operations. For each one of the k iterations, finding the next sensor (Line 6) requires $\mathcal{O}(n)$ comparisons, and adding the new sensor y^* can only change the score of its neighbors ($N(y^*; \varepsilon) \leq d$), thus Line 6 requires $\mathcal{O}(d \cdot d^3)$ operations. The total running time of Algorithm 6.2 is $\mathcal{O}(nd^3 + kn + kd^4)$, which can be significantly lower than the $\mathcal{O}(kn^4)$ operations required by Algorithm 6.1. Theorem 6.8 summarizes our analysis:

Theorem 6.8. *Under the assumptions of Lemma 6.4, guaranteeing ε_1-approximate nondecreasingness and truncation parameter ε_2, Algorithm 6.2 selects $\mathcal{A} \subseteq \mathcal{S}$ such that*

$$F_{MI}(\mathcal{A}) \geq (1 - 1/e)(\mathrm{OPT} - k\varepsilon_1 - 2kn\varepsilon_2/\sigma^4),$$

in time $\mathcal{O}(nd^3 + nk + kd^4)$.

This approach can be efficiently implemented by using a priority queue to maintain the advantages δ_y. Using for example a Relaxed Heaps data structure, the running time can be decreased to $\mathcal{O}(nd^3 + kd \log n + kd^4)$: Line 6 uses the **insert** operation with complexity $\mathcal{O}(1)$, Line 6 calls **deletemax** with complexity $\mathcal{O}(\log n)$, and Line 6 uses **delete** and **insert**, again with complexity $\mathcal{O}(\log n)$. This complexity improves on Algorithm 6.2 if $d \log n \ll n$. This assumption is frequently met in practice, since d can be considered a constant as the size n of the sensing area grows. Of course, this procedure can also be combined with the lazy evaluations described in the previous section for further improvement in running time.

Algorithm 6.2: Approximation algorithm for maximizing mutual information using local kernels.

Input: Covariance $\Sigma_{\mathcal{V}\mathcal{V}}$, k, $\mathcal{V} = \mathcal{S} \cup \mathcal{U}, \varepsilon > 0$
Output: Sensor selection $\mathcal{A} \subseteq \mathcal{S}$
begin
  $\mathcal{A} \leftarrow \emptyset$;
  foreach $y \in \mathcal{S}$ **do**
1 $\delta_y \leftarrow H(\mathcal{X}_y) - \tilde{H}_\varepsilon(\mathcal{X}_y|\mathcal{X}_{\mathcal{V}\setminus y})$;
  end
  for *j = 1* **to** k **do**
2 $y^* \leftarrow \arg\max_y \delta_y$;
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{y^*\}$;
    foreach $y \in N(y^*; \varepsilon)$ **do**
3 $\delta_y \leftarrow \tilde{H}_\varepsilon(\mathcal{X}_y|\mathcal{X}_\mathcal{A}) - \tilde{H}_\varepsilon(\mathcal{X}_y|\mathcal{X}_{\bar{\mathcal{A}}})$;
    end
  end
end

6.5 Related work

In this section, we will review the related work that specific to this chapter, and that has not been reviewed in Chapter 3 yet.

6.5.1 Relationship to canonical correlation analysis

Given that the joint distribution of $\mathcal{X}_\mathcal{A}$ and $\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}$ is Gaussian, their mutual information is also

$$F_{MI}(\mathcal{A}) = -\frac{1}{2}\sum_i \log\left(1 - \rho_i^2\right) \tag{6.13}$$

where $\rho_1^2 \geq \cdots \geq \rho_{|\mathcal{V}|}^2$ are the canonical correlation coefficients between $\mathcal{X}_\mathcal{A}$ and $\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}$ [Caselton and Zidek, 1984]. McCabe [1984] show that maximizing the canonical correlations between observed and unobserved variables can be interpreted as a form of principal components analysis, where one realizes that selecting subsets of variables is a special kind of linear projection. A similar analysis is presented for entropy and other common design criteria. Using Equation (6.13), a similar relationship can be made to canonical correlation analysis (CCA; Hotelling 1936), which finds linear projections for $\mathcal{V} \setminus \mathcal{A}$ and $\mathcal{A}$ that maximize the correlations in the lower dimensional space. By considering these lower-dimensional projections, one can determine how much variance is shared (jointly explained) by $\mathcal{V} \setminus \mathcal{A}$ and $\mathcal{A}$.

6.5.2 Relationship with the disk model

The disk model for sensor placement [*c.f.* Bai et al., 2006, Hochbaum and Maas, 1985] assumes that each sensor can perfectly observe everything within a radius of r and nothing else. Hence we can associate with every location $y \in \mathcal{V}$ a sensing region D_y, which, for a discretization of the space, corresponds to the locations contained within radius r of location y. For a set of locations $\mathcal{A}$, we can define the coverage

$F_D(\mathcal{A}) = \bigcup_{y \in \mathcal{A}} D_y$. It can be easily seen that this criterion is nondecreasing submodular and $F(\emptyset) = 0$. Hence optimizing placements for the disk model criterion is a submodular maximization problem, and the greedy algorithm can guarantee a constant factor $(1 - 1/e)\,\mathrm{OPT}$ approximation guarantee for finding the placement of k sensors with maximum coverage.

There is a sense, in which the approach of sensor placements in GPs can be considered a generalization of the disk model. If we assume an isotropic GP with local kernel function as the one presented in Figure 6.9, then a sensor measurement is correlated exactly with the locations within a disk around its location. If the process has constant variance, then the greedy algorithm will, for the first few sensors placed, only try to achieve a disjoint placement of the disks, and as such behave just like the greedy algorithm for disk covering.

However, once enough sensors have been placed so that these "disks" start to overlap, the behavior of the two approaches begins to differ: in a disk model there is no advantage in placing sensors that lead to overlapping disks. In a GP model, even an isotropic one, "overlapping disks" lead to better predictions in the overlapping area, a very natural consequence of the representation of the uncertainty in the process.

6.5.3 Fast Gaussian Process methods

Information-theoretic criteria are also used in sparse GP modeling, which attempts to reduce the cost of inference by selecting a representative subset of the training data. Sample selection criteria have included KL-divergence [Seeger et al., 2003] and entropy [Lawrence et al., 2003]. In contrast to sensor placement, where locations are chosen to minimize predictive uncertainty, in sparse GP methods, the samples are chosen such that the approximate posterior matches the true posterior (which uses the entire training set) as accurately as possible. Instead of choosing a subset of the training data, Snelson and Ghahramani [2005] propose to optimize the location of a set of "hallucinated" inputs. This approach results in a continuous optimization problem, which appears to be easier to solve (albeit with no performance guarantees) than the discrete subset selection problem.

6.5.4 Sensor placement with model uncertainty

The discussion thus far has focused on the case where the joint model $P(\mathcal{X}_\mathcal{V})$ is completely specified, i.e., the mean and covariance of the GP are known[2]. With model uncertainty, one has to distinguish between observation selection for predictive accuracy in a fixed model and observation selection for learning parameters. Model uncertainty also introduces computational issues. If the mean and covariance are fixed in a Gaussian process then the posterior is Gaussian. This makes it easy to compute quantities such as entropy and mutual information. If the mean and covariance are unknown, and we have to learn hyperparameters (*e.g.*, kernel bandwidth of an isotropic process), then the predictive distributions and information-theoretic quantities often lack a closed form.

Caselton et al. [1992] extend their earlier work on maximum entropy sampling to the case where the mean and covariance are unknown by using a conjugate Bayesian analysis. The limitations of this approach are that the conjugate Bayesian analysis makes spatial independence assumptions in the prior and that

[2]Or one assumes the uncertainty on these parameters is small enough that their contribution to the predictive uncertainty is negligible.

complete data with repeated observations are required at every potential sensing site. This leads to a determinant maximization problem, much like D-optimality, that precludes the use of submodularity.

Another approach is the development of hybrid criteria, which balance parameter estimation and prediction. For example, Zimmerman [2006] proposes local EK-optimality, a linear combination of the maximum predictive variance and a scalarization of the covariance of the maximum likelihood parameter estimate. While this criterion selects observations which reduce parameter uncertainty and predictive uncertainty given the *current parameter*, it does not take into account the effect of parameter uncertainty on prediction error. To address this issue, Zhu and Stein [2006] derive an iterative algorithm which alternates between optimizing the design for covariance estimation and spatial prediction. This procedure does not provide guarantees on the quality of designs.

An alternative approach to addressing model uncertainty, in the context of classical experimental design, is presented by Flaherty et al. [2006]. There, instead of committing to a single value, the parameters of a linear model are constrained to lie in a bounded interval. Their robust design objective, which is based on E-optimality, is then defined with respect to the worst-case parameter value. Flaherty et al. [2006] demonstrate how a continuous relaxation of this problem can be formulated as a SDP, which can be solved exactly. No guarantees are given however on the integrality gap on this relaxation.

By exploiting submodularity, we can extend our approach described in this chapter to handle parameter uncertainty as well. Details will be discussed in Chapter 10.

6.5.5 Non-constant cost functions

In Section 6.3.3, we discuss the case where every sensor can have a different cost, and one has a budget which one can spend. An alternate approach to sensor costs is presented by Zidek et al. [2000]. They propose a criteria that makes a trade off between achieved reduction in entropy using an entropy-to-cost conversion factor, i.e., they optimize the sum of the entropy with a factor times the cost of the placements. This criterion yields an unconstrained optimization problem. Our approach to sensor costs (Section 6.3.3) yields a constrained optimization, maximizing our criteria given a fixed budget that can be spent when placing sensors. Such a budget-based approach seems more natural in real problems (where one often has a fixed number of sensors or amount of money to spend). Moreover, our approach provides strong a priori theoretical guarantees and tighter online bounds, which are not available for the approach of Zidek et al. [2000].

6.6 Experiments

We performed experiments on two real-world data sets, as described in Section 6.6.1. In Section 6.6.2 we compare placements on stationary and nonstationary GP models. In Sections 6.6.3, 6.6.4 and 6.6.5 we compare mutual information with the disk model, with entropy and with other classical experimental design criteria, in terms of prediction accuracy. In 6.6.6 we compare the performance of the greedy algorithm with other heuristics, and in Section 6.6.7 we analyze the effect of exploiting local kernels.

6.6.1 Data sets used in our experiments

We first introduce the data sets we consider in our experiments. In our first data set, we analyze temperature measurements from the network of 46 sensors shown in Figure 6.1. Our training data consisted of samples collected at 30 sec. intervals on 3 consecutive days (starting Feb. 28th 2004), the testing data consisted of the corresponding samples on the two following days.

Our second data set consists of precipitation data collected during the years 1949 - 1994 in the states of Washington and Oregon [Widmann and Bretherton, 1999]. Overall 167 regions of equal area, approximately 50 km apart, reported the daily precipitation. To ensure the data could be reasonably modeled using a Gaussian process we applied a log-transformation, removed the daily mean, and only considered days during which rain was reported. After this preprocessing, we selected the initial two thirds of the data as training instances, and the remaining samples for testing purposes. From the training data, we estimated the mean and empirical covariance, and regularized it by adding independent measurement noise[3] of $\sigma^2 = 0.1$.

We computed error bars for the prediction accuracies in all of our experiments, but due to the violated independence of the collected samples (which are temporally correlated), these error bars are overconfident and hence not reported here. The estimated standard errors under the independence assumption are too small to be visible on the plots.

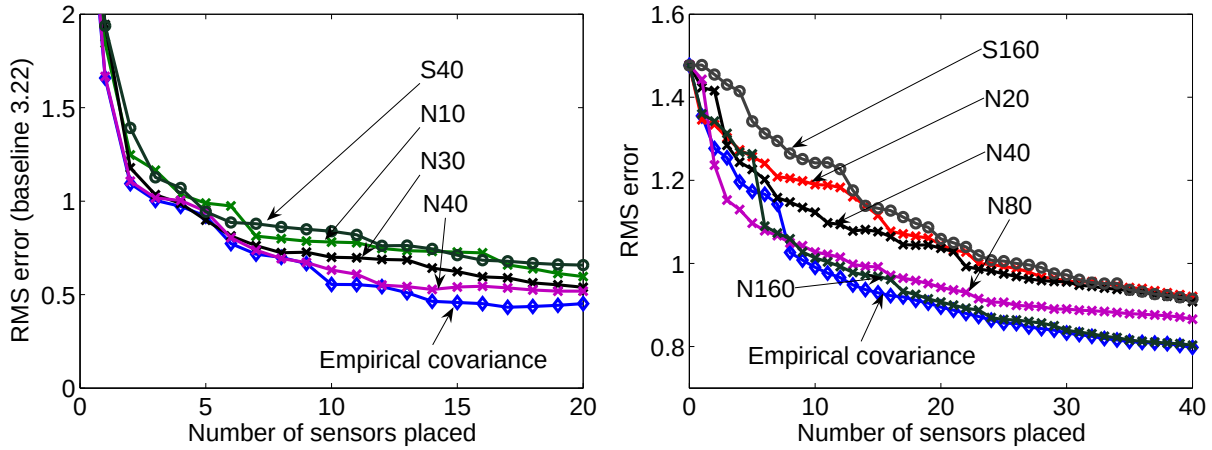
6.6.2 Comparison of stationary and nonstationary models

To see how well the stationary and nonstationary models capture the phenomenon, we performed the following experiment: We learned both stationary and non-stationary GP models from an increasing number of sensors. The model with stationary correlation function used an isotropic Exponential kernel with bandwidth fitted using least-squares fit of the empirical variogram [Cressie, 1991]. We also learned a nonstationary GP using the technique from Nott and Dunsmuir [2002]. Both GP models were estimated from an initial deployment of an increasing number of sensors. We used non-stationary the same variance process for both stationary and nonstationary models (i.e., giving more information to the stationary model than commonly done). Since with increasing amount of data, the empirical covariance matrix will exactly capture the underlying process, we consider the empirical covariance as the ground truth both for placements and prediction. Hence we also selected placements using the entire estimated covariance matrix.

We optimized the designs using mutual information on all the models, and evaluated the prediction accuracy for an increasing number of near-optimally placed sensors, using the estimated model and the measured values for the selected sensors. Figure 6.10(a) presents the results for the temperature data set. We can see that the nonstationary model learned from 10 sensors performs comparably to the stationary model with 40 sensors, even with non-stationary variance process. As we increase the number of sensors in the initial deployment, the Root Mean Squared error (RMS) prediction accuracies we get for placements of increasing size converge to those obtained for optimizing the placements based on the empirical covariance matrix.

Figure 6.10(b) presents the results of the same experiment for the precipitation data. Here we can see that the nonstationary model estimated using 20 sensors leads to better RMS accuracies than the stationary model, even if latter is estimated using 160 sensors.

[3] The measurement noise σ^2 was chosen by cross-validation.



(a) *Comparison of fits for temperature data.*

(b) *Comparison of fits for precipitation data.*

Figure 6.10: RMS curves for placements of increasing size, optimized using stationary and nonstationary GPs. Prediction is done using estimated models. (a) Stationary GP estimated for the temperature data from 40 sensors (S40), nonstationary GPs estimated from 10, 30 and 40 sensors (N10, N30, N40). (b) Stationary GP estimated for the precipitation data from 160 sensors (S160), nonstationary GPs estimated from 20, 40, 80 and 160 sensors (N20, N40, N80, N160).

6.6.3 Comparison of data-driven placements with geometric design criteria

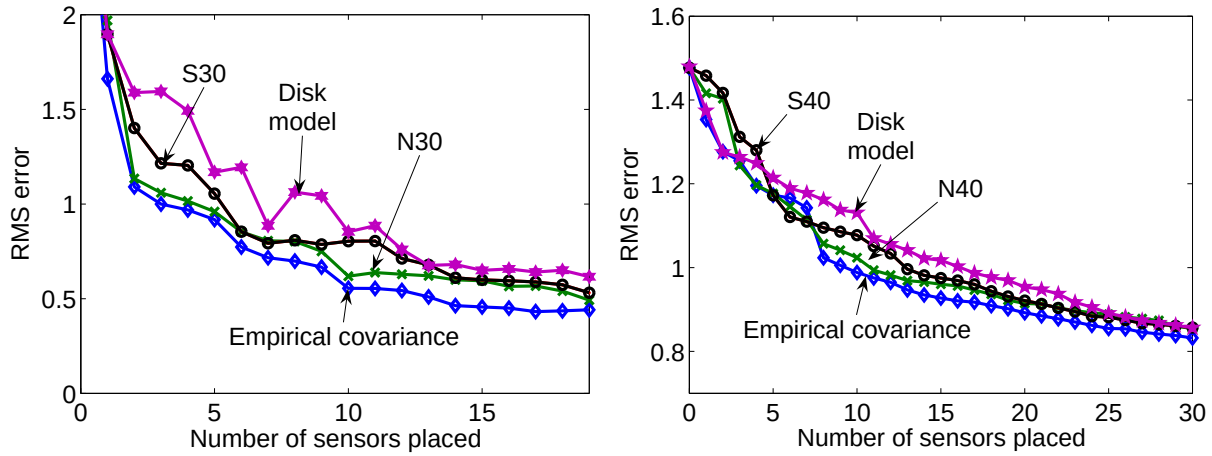
We now compare placements based on our data-driven GP models with those based on the traditional disk model. This model assumes that every sensor can perfectly measure the phenomenon within a radius of r, and have no information outside this radius. Since choosing an appropriate radius for the disk model is very difficult in practice, we decided to choose $r = 5m$ since for this radius 20 sensors could just spatially cover the entire space. We also learned stationary and non-stationary GP models as discussed in Section 6.6.2.

For the disk model, we used the greedy set covering algorithm. The design on both GP models was done using our greedy algorithm to maximize mutual information. For an increasing number of sensors, we compute the Root Mean Squares (RMS) prediction error on the test data. In order to separate the placement from the prediction task, we used the empirical covariance matrix estimated from the training data on all 46 locations for prediction, for all three placements.

Figure 6.11(a) presents the results of this experiment. The geometrical criterion performs poorly compared to the model based approaches. We can see that the placements based on the empirical covariance matrix perform best, quite closely followed by the accuracies obtained by the designs based on the nonstationary process. Figure 6.11(b) shows the results for the same experiment on the precipitation data set.

6.6.4 Comparison of the mutual information and entropy criteria

We also compared the mutual information criterion to other design criteria. We first compare it against the entropy (variance) criterion. Using the empirical covariance matrix as our process, we use the greedy algorithm to select placements of increasing size, both for mutual information and for entropy. Figure 6.13(a)



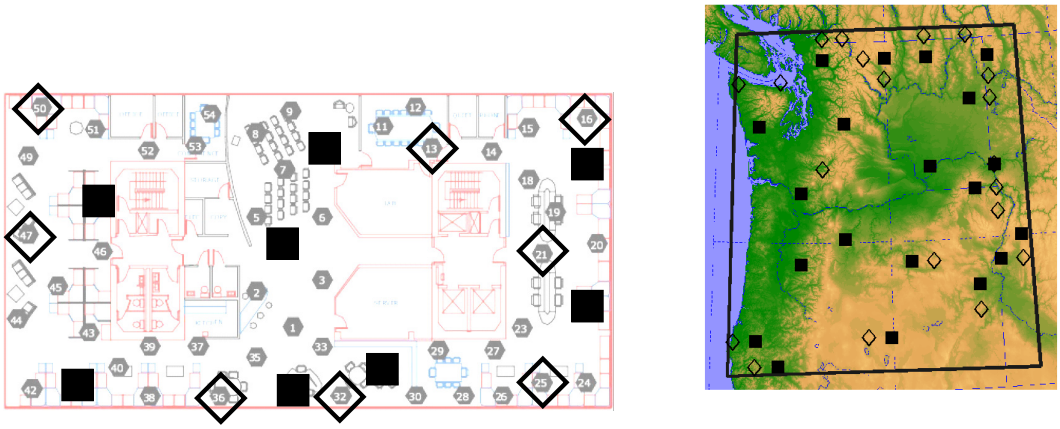
(a) *Comparison with disk model (Temperature)* (b) *Comparison with disk model (Precipitation)*

Figure 6.11: RMS curves for placements of increasing size, optimized using the disk model, stationary and nonstationary GPs. Prediction for all placements is done using the empirical covariance. Stationary GPs and nonstationary GPs estimated from 30 sensors (N30, S30, for temperature data) or 40 sensors (N40, S40, for precipitation data).

and Figure 6.13(b) show the results of this comparison on models estimated for the morning (between 8 am and 9 am) and noon (between 12 pm and 1 pm) in the Intel lab data. Figure 6.13(a) and Figure 6.13(b) plot the log-likelihood of the test set observations with increasing number of sensors for both models. Figure 6.13(e) presents the RMS error for a model estimated from the Intel lab data for the entire day. We can see that mutual information outperforms entropy, achieving better prediction accuracies with a smaller number of sensors.

Figure 6.13(f) presents the same results for the precipitation data set. Mutual information significantly outperforms entropy as a selection criteria – often several sensors would have to be additionally placed for entropy to reach the same level of prediction accuracy as mutual information. Figure 6.12(b) shows where both objective values would place sensors to measure precipitation. It can be seen that entropy is again much more likely to place sensors around the border of the sensing area than mutual information.

To gain further insight into the qualitative behavior of the selection criteria we learned a GP model using all sensors over one hour starting at noon. The model was fit with a isotropic Gaussian kernel and quadratic trend for the mean, using the *geoR* Toolkit [Ribeiro Jr. and Diggle, 2001]. Figures 6.14(a) and 6.14(b) show the posterior mean and variance for the model. Using our algorithms, 22 sensors were chosen using the entropy and mutual information criteria. For each set of selected sensors, additional models were trained using only the measurements of the selected sensors. Predicted temperature surfaces for the entropy and mutual information configurations are presented in Figures 6.14(c) and 6.14(d). Entropy tends to favor placing sensors near the boundary as observed in Section 6.2, while mutual information tends to place the sensors on the top and bottom sides, which exhibited the most complexity and should have a higher sensor density. The predicted variances for each model are shown in figures 6.14(e) and 6.14(f). The mutual information version has significantly lower variance than the entropy version almost everywhere, displaying, as expected, higher variance in the unsensed areas in the center of the lab.



(a) *Placements of temperature sensors*

(b) *Placements of rain sensors*

Figure 6.12: Example sensor placements for temperature and precipitation data. Squares indicate locations selected by mutual information, diamonds indicate those selected by entropy. Notice how entropy places sensors closer to the border of the sensing field.

6.6.5 Comparison of mutual information with classical experimental design criteria

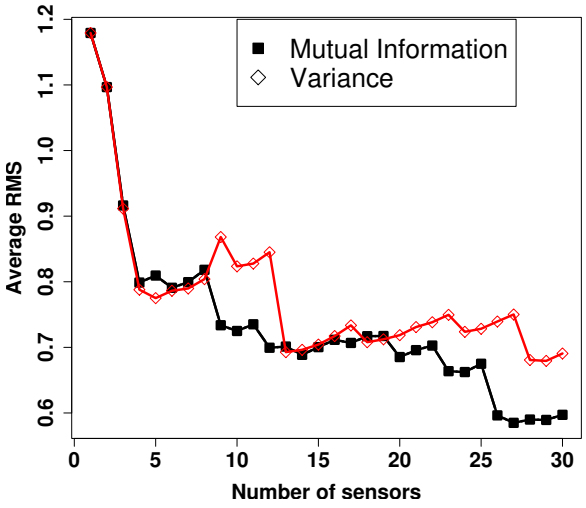
In order to compare the mutual information placements with the classical optimality criteria, we performed the following experiment. We uniformly selected 12 target locations $\mathcal{U}$ in the lab as locations of interest. We then set up the linear model

$$\mathbf{x}_{\mathcal{S}} = \Sigma_{\mathcal{SU}}\Sigma_{\mathcal{UU}}^{-1}\,\mathbf{x}_{\mathcal{U}} + \mathbf{w}.$$

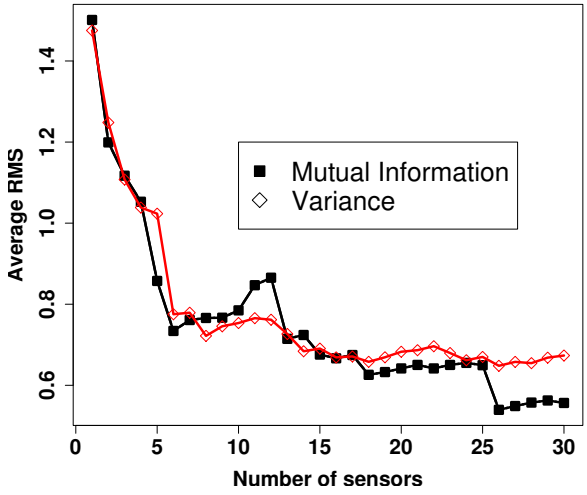
Hereby, $\mathbf{x}_{\mathcal{S}}$ denotes measurements at the locations $\mathcal{S}$, among which we choose our placement, $\mathbf{x}_{\mathcal{U}}$ are the values at the locations of interest (no sensors can be placed there), and $\mathbf{w}$ models independent normal measurement noise with constant variance. After subtraction of the training set mean, this model uses the Best Linear Unbiased (Kriging) estimator for predicting $\mathbf{x}_{\mathcal{S}}$ from $\mathbf{x}_{\mathcal{U}}$.

The problem becomes to select the sensor locations $\mathcal{A} \subseteq \mathcal{S}$ which allow most precise prediction of the variables of interest, in the sense of minimizing the error covariance $\frac{1}{\sigma^2}(A^TA)^{-1}$, where $A = \Sigma_{\mathcal{SU}}\Sigma_{\mathcal{UU}}^{-1}$. The different classical design criteria vary in how the scalarization of the error covariance is done. D-optimal design minimizes the log-determinant, A-optimal design minimizes the trace, and E-optimal design minimizes the spectral radius (the magnitude of the largest eigenvalue) of the error covariance. Note that this problem formulation favors the classical design criteria, which are tailored to minimize the error of predicting the values at the target locations $\mathcal{U}$, whereas mutual information and entropy just try to decrease the uncertainty in the entire space.

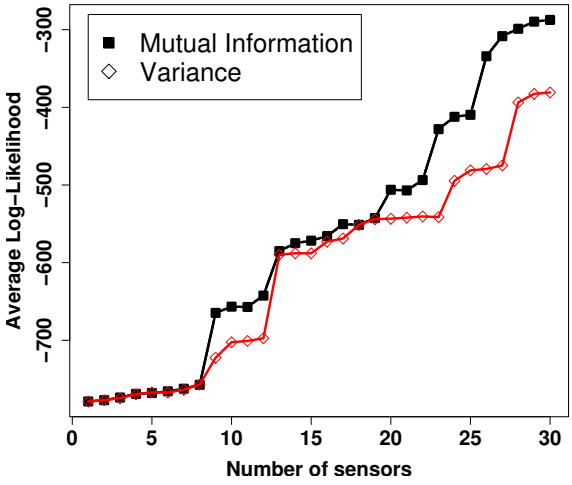
In order to solve the classical experimental design problems, we use the formulations as a semidefinite program (SDP) as discussed by Boyd and Vandenberghe [2004]. We use SeDuMi [Sturm, 1999] for solving these SDPs. Since the integral optimization is hard, we solve the SDP relaxation to achieve a fractional design. This fractional solution defines the best way to distribute an infinite (or very large) budget of experiments to the different choices on the design menu (the variables in $\mathcal{S}$). In the sensor selection problem however, we have to solve the integral problem, since we face the binary decision of whether a sensor should be placed at a particular location or not. This is a hard combinatorial optimization problem. Since



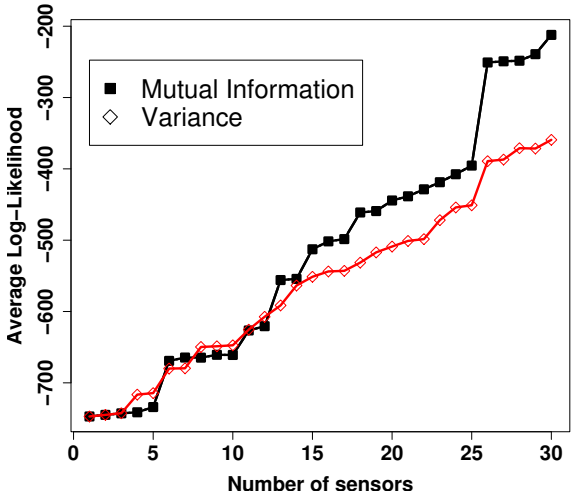
(a) *Morning RMS*



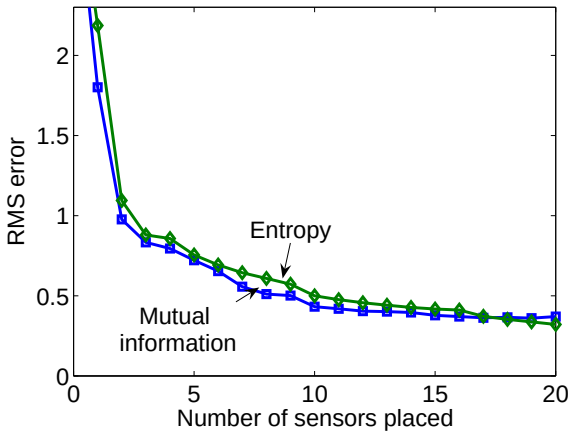
(b) *Noon RMS*



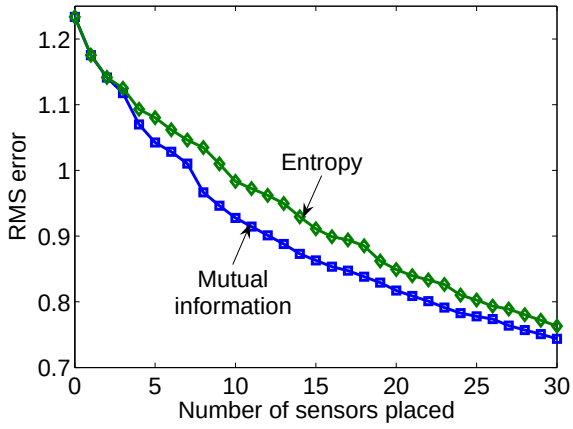
(c) *Morning log-likelihood*



(d) *Noon log-likelihood*

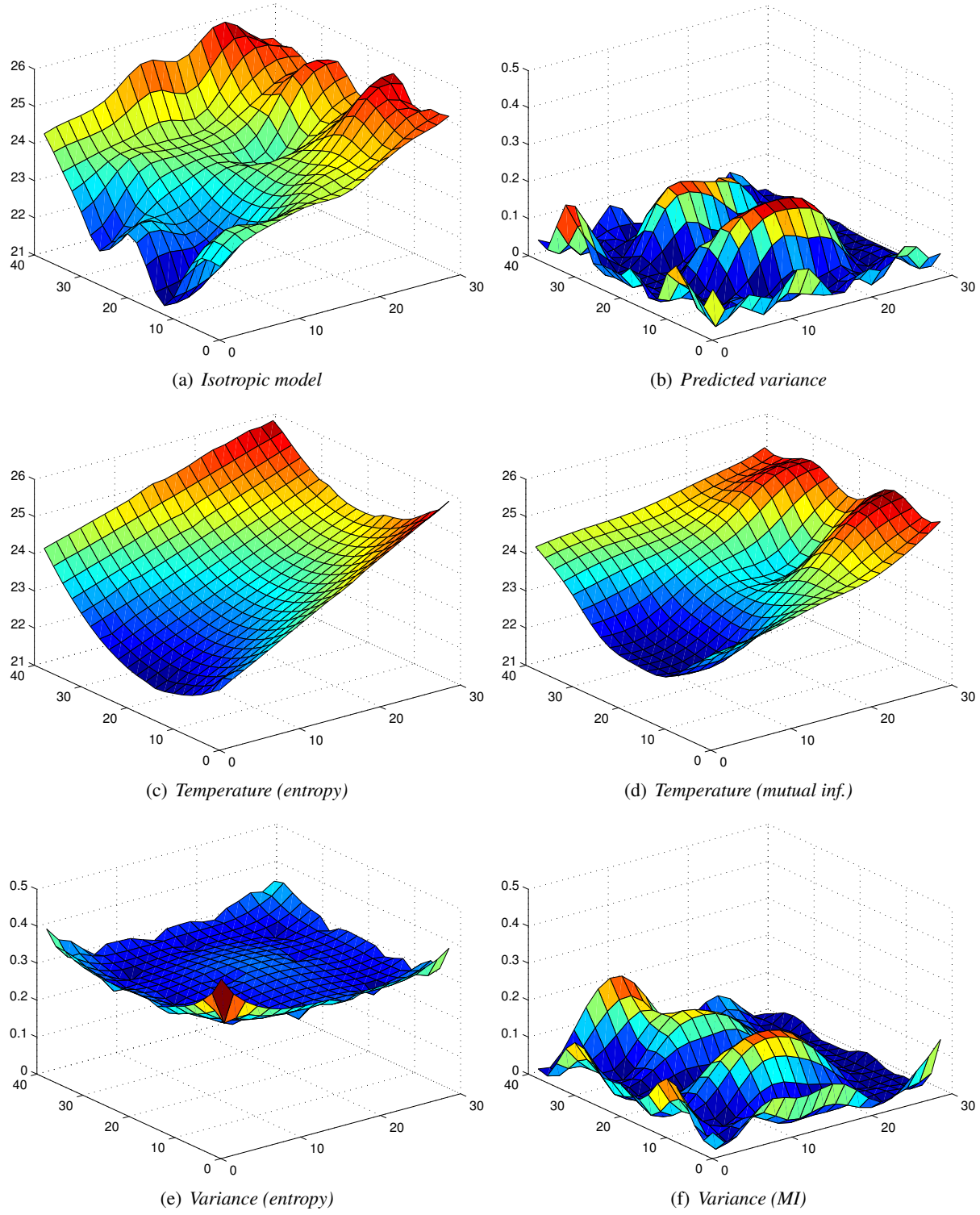


(e) *Temperature data*



(f) *Precipitation RMS*

Figure 6.13: Prediction error for temperature (a-e) and precipitation (f) data in sensor network deployments, for an increasing number of sensors.



(a) *Isotropic model*

(b) *Predicted variance*

(c) *Temperature (entropy)*

(d) *Temperature (mutual inf.)*

(e) *Variance (entropy)*

(f) *Variance (MI)*

Figure 6.14: Comparison of predictive quality of subsets selected using MI and entropy.

no near-optimal solution is known, we select the locations corresponding to the top k coefficients of the design menu, as is common practice. We compare the placements using the classical design criteria to those using the mutual information and entropy criteria, and evaluate each of them on the RMS prediction accuracy on the hold-out locations $\mathcal{U}$.

Figure 6.15(a) presents the results of this experiment on the temperature data. We can see that even though mutual information optimizes for prediction accuracy in the entire space and not specifically for the target locations $\mathcal{U}$, it incurs the least RMS prediction error, apart from the placements consisting only of a single sensor. E-optimal design performs comparably with the entropy criterion, and D- and A-optimality perform worse.

When we performed the same experiment with the precipitation data, SeDuMi ran out of memory (1 GB) for the SDP required to solve the A-optimality criterion. The largest subsample we could solve for all A-, D- and E-optimality on this data set was limited to 111 locations. Figure 6.15(b) presents the results. For the entire data set of 167 locations, we could still solve the D- and E-optimality SDPs. The results are presented in Figure 6.15(c). We can observe that for the 111 locations, D-optimality slightly outperforms mutual information. We have to consider, however, that the classical criteria are optimized to minimize the error covariance with respect to the locations $\mathcal{U}$ of interest, whereas mutual information merely tries to achieve uniformly low uncertainty over the entire space. For the full set of 167 locations, mutual information outperforms the other design criteria.

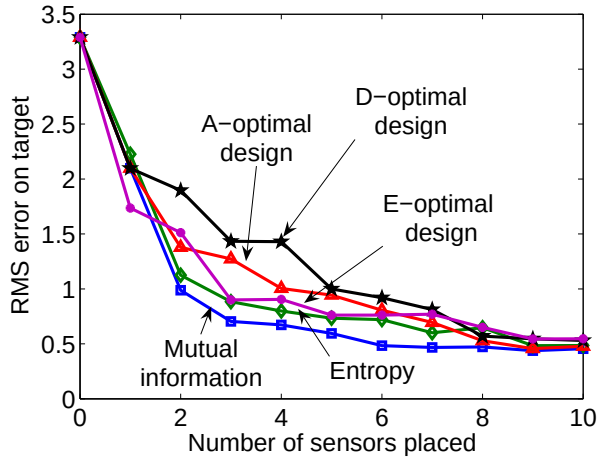
Figure 6.15(d) presents the running time for optimizing A-, D-, E-optimality, and mutual information, mutual information with truncation parameter $\varepsilon = 1$ and entropy on the 111 node subsample of the precipitation data on a Pentium M 1.7 GHz processor. We can see that optimizing entropy is fastest, closely followed by the truncated mutual information criterion described in Section 6.4.2 that is further evaluated in Section 6.6.7. Even without truncation, optimizing mutual information is three times faster than (fractionally) optimizing D-optimality and 24 times faster than A-optimality.

6.6.6 Empirical analysis of the greedy algorithm

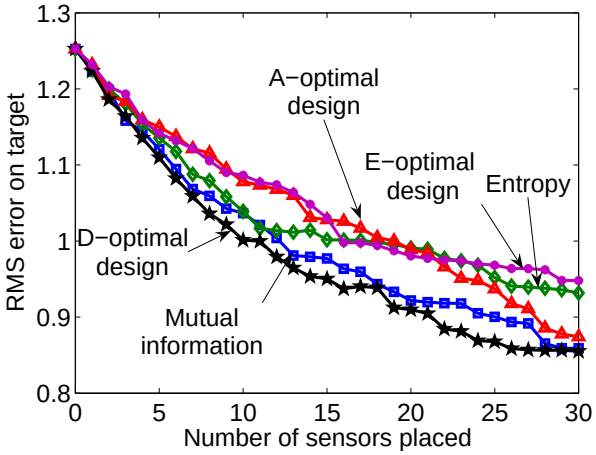
To study the effectiveness of the greedy algorithm, we compared the mutual information of the sets selected by our greedy algorithm to random selections, to a hill climbing method that uses a pairwise exchange heuristic, and – for small subsamples – to the bounds proved by the MIP from Section 6.3.5.

In this experiment, we used the empirical covariance matrix as the input to the algorithms. Figure 6.16(b) shows that the greedy algorithm provided significantly better results than the random selections, and even the maximum of a hundred random placements did not reach the quality of the greedy placements. Furthermore, we enhanced the random and greedy selections with the pairwise exchange (PE) heuristic, which iteratively finds exchanges of elements $y \in \mathcal{A}$ and $y' \in \mathcal{S} \setminus \mathcal{A}$ such that exchanging y and y' improves the mutual information score. Figure 6.16(a) presents objective values of these enhanced selection methods for a subset size of 12, for which the maximum over 100 random selections enhanced with PE actually exceeded the greedy score (unlike with most other subset sizes, where random + PE did about as well as the greedy algorithm). Typically, the objective values of random + PE, greedy + PE and greedy did not differ much. Note that as mentioned in Section 6.3, the performance guarantee for the greedy algorithm always provides an online approximation guarantee for the other heuristics.

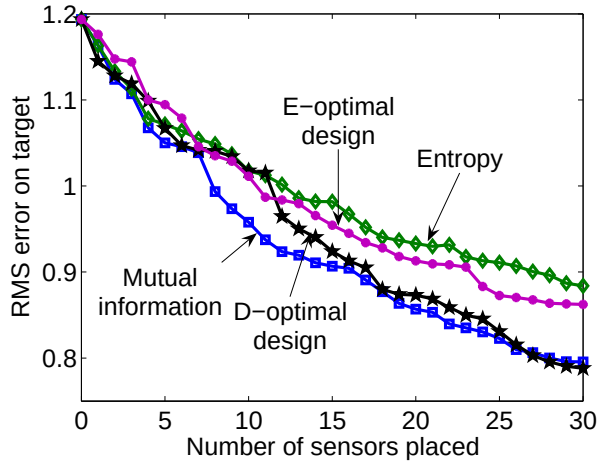
For a 16 node subsample of the temperature data set, we used the MIP from Section 6.3.5 to compute



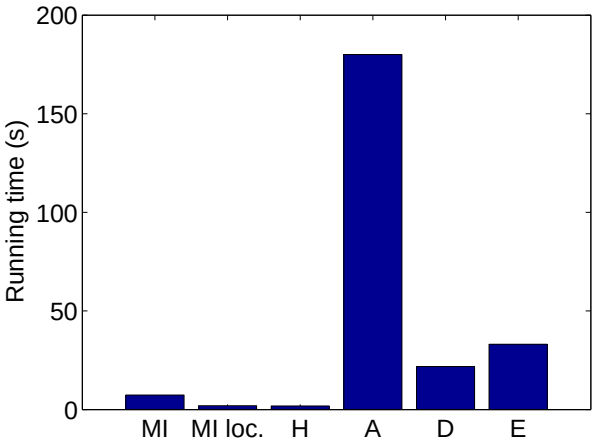
(a) *Comparison with A-, D- and E-optimality on temperature data*



(b) *Comparison with A-, D- and E-optimality on precipitation data, 111 node subsample*



(c) *Comparison with D- and E-optimality on precipitation data*

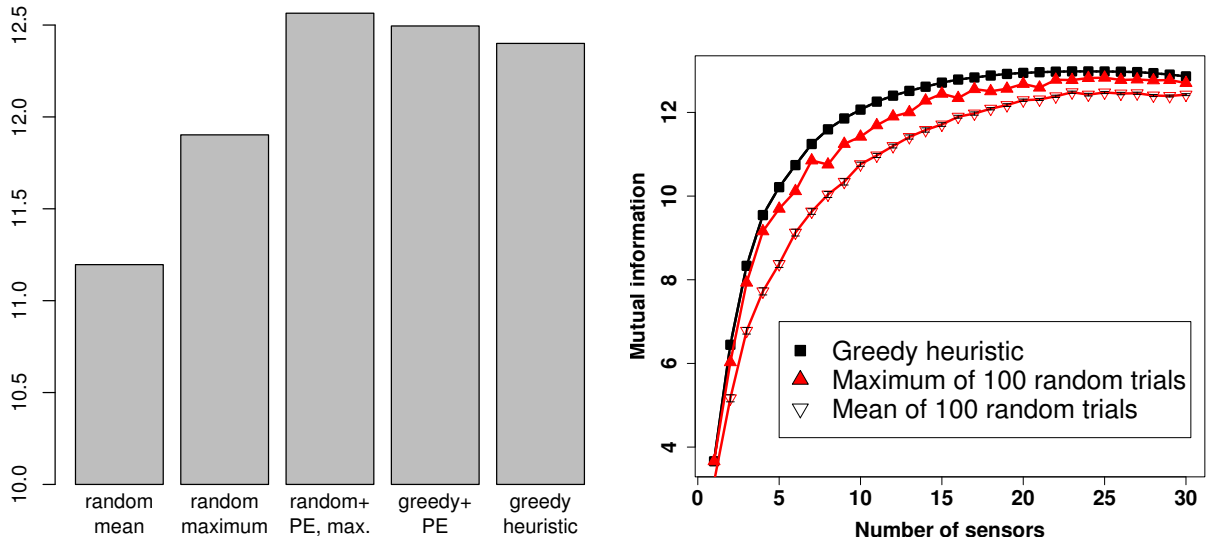


(d) *Running time for 111 node precipitation subsample*

Figure 6.15: Comparison with classical experimental design. We plot RMS prediction error on 25% hold out target locations $\mathcal{U}$.

bounds on the optimal mutual information. Figure 6.6 presents the results. It can be seen, that for this small subsample, the greedy solution is never more than 5 percent away from the optimal solution, which is a much tighter bound than the a priori approximation factor of $(1 - 1/e)$.

We also experimented with the lazy evaluation strategy discussed in Section 6.4.1. For example when picking placements of size 50 for the precipitation data set, the number of mutual information computations decreased from 7125 to 1172, and the computation time on a Pentium M 1.7 GHz processor decreased from 41.3 seconds to 8.7 seconds. The results for both temperature and precipitation data sets are presented in Figures 6.17(a) and 6.17(b).



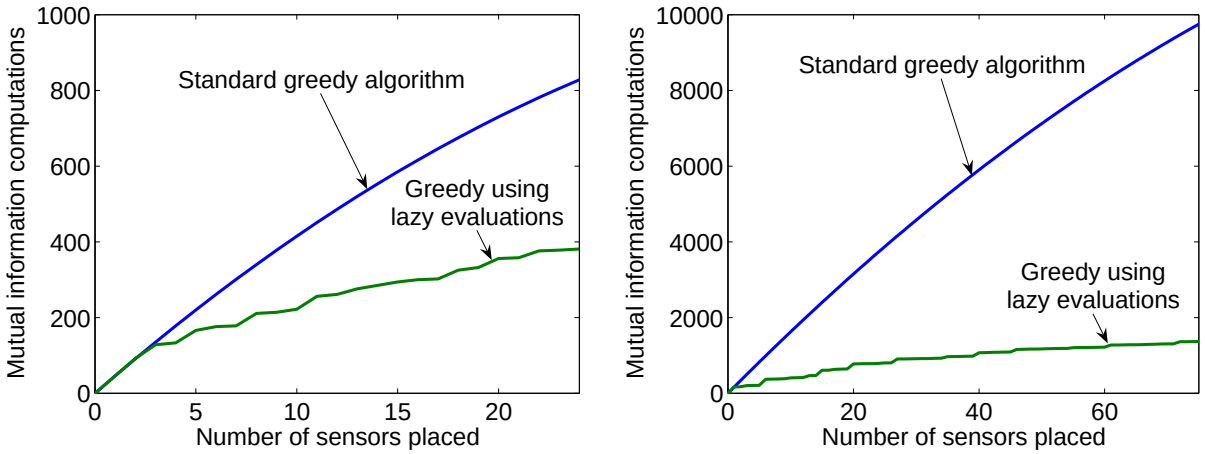
(a) *Objective values* (b) *Comparison of heuristics*

Figure 6.16: Comparison of the greedy algorithm with several heuristics.

6.6.7 Results on local kernels

We also performed experiments to assess the running time versus quality trade-off incurred by using approximate local kernels. To provide intuition about the covariance structure, note that the 25, 50 and 75 percentiles of the absolute covariance entries were 0.122, 0.263 and 0.442, the maximum was 3.51, the minimum was $8.78E-6$. For the variance (the diagonal entries), the median was 1.70, and the minimum was 0.990. Figure 6.18(a) shows that the computation time can be drastically decreased as we increase the truncation parameter ε from 0 to the maximum variance. Figure 6.18(b) shows the RMS prediction accuracy for the 20 element subsets selected by Algorithm 6.2. According to the graphs, the range $\varepsilon \in [0.5, 1]$ seems to provide the appropriate trade-off between computation time and prediction accuracy.

In order to study the effect of local kernels on the placements, we performed the following experiment. We created a regular 7 by 7 grid with unit distance between neighboring grid points, and generated covariance matrices using two different GPs, one using the Gaussian (squared exponential) kernel, and the other using the the local kernel (Equation (6.12)). We exponentially increased the bandwidth in eight steps from 0.1 to 12.8. Figures 6.21 and 6.22 show the corresponding placements using mutual information to select the locations. From this experiment, we can see that the placements obtained using the non-local Gaussian kernel tend to be spread out slightly more, as one might expect. Overall, however, the placements appear to be very similar. In light of the computational advantages provided by local kernels, these results provide further evidence in the spirit of Section 6.6.7, namely that local kernels can be a valuable tool for developing efficient model-based sensor placement algorithms.



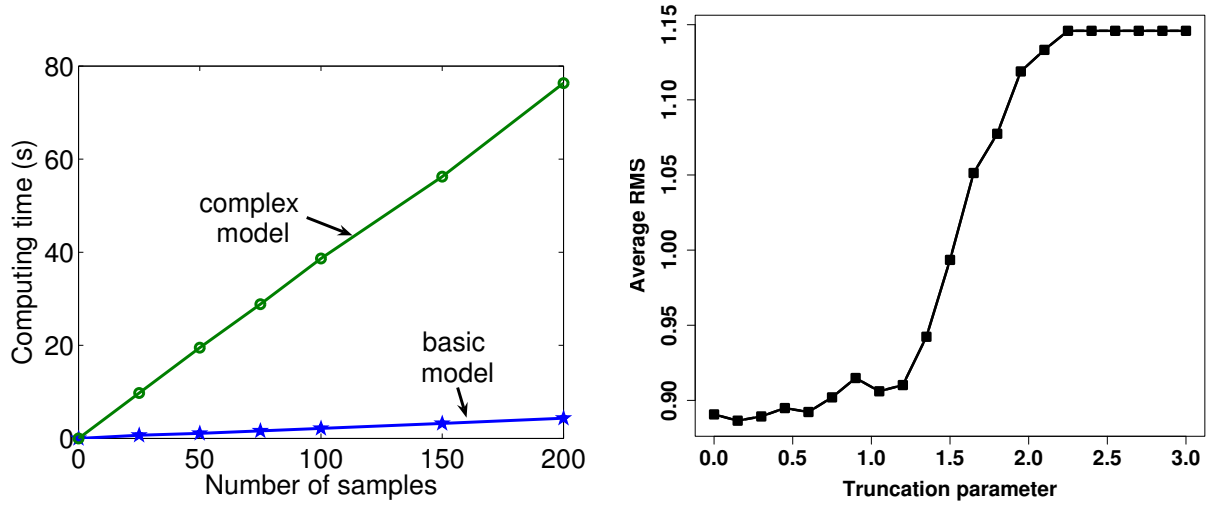
(a) *Temperature data* (b) *Precipitation data*

Figure 6.17: Performance improvements by using lazy evaluations of mutual information.

6.7 Summary

In this chapter, we tackled the problem of maximizing mutual information in order to optimize sensor placements, i.e., select observations for the purpose of spatial prediction. We proved that the exact optimization of mutual information is **NP**-complete, and provided an approximation algorithm that is within $(1 - 1/e)$ of the maximum mutual information configuration by exploiting the submodularity in the criterion. We also illustrate that submodularity can be used to obtain online bounds, which are useful for bounding the quality of the solutions obtained by any optimization method, and for designing branch and bound algorithms for the mutual information criterion. In order to scale up the application of our approach, show how to exploit lazy evaluations and local structure in GPs to provide significant speedups. We also extend our submodularity-based analysis of mutual information to incorporate robustness to sensor failures and model uncertainty.

Our very extensive empirical results indicate that data-driven placements can significantly improve the prediction accuracy over geometric models. We find, in contrast to previous work [Caselton et al., 1992, Zidek et al., 2000], that the mutual information criterion is often better than entropy and other classical experimental design criteria, both qualitatively and in prediction accuracy. In addition, the results show that a simple greedy algorithm for optimizing mutual information provides performance that is very close to the optimal solution in problems that are small enough to be solved exactly, and comparable to more complex heuristics in large problems.



(a) *Running time* (b) *Approximation error*

Figure 6.18: Analysis of the experiments with local kernels. (a) running times for increasing level of truncation. (b) increase of average RMS error with increasing level of truncation. Note that for a truncation between 0.5 and 1.2, a good tradeoff between running time and error is achieved.

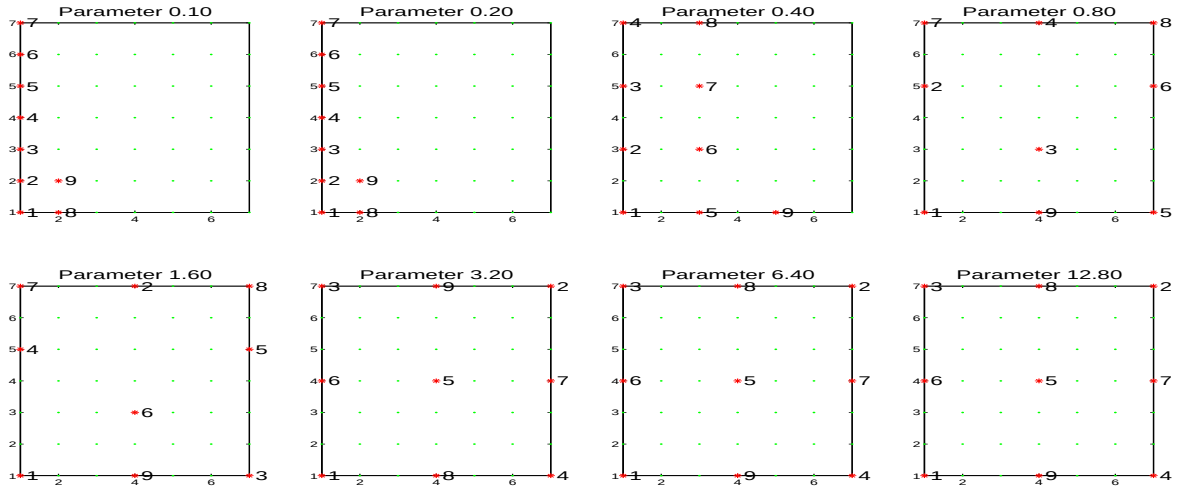


Figure 6.19: Placements under Gaussian kernel, entropy criterion, increasing bandwidth

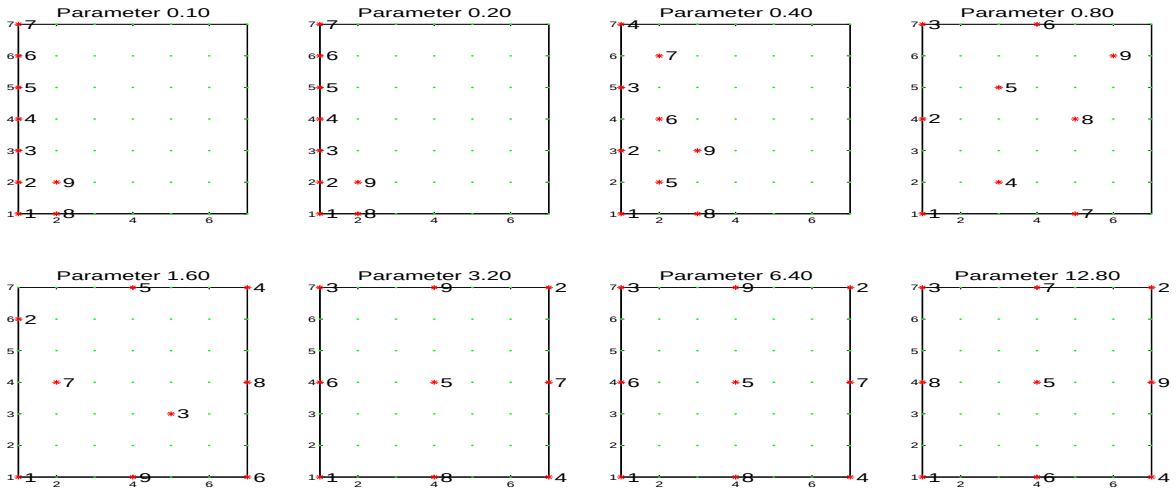


Figure 6.20: Placements under local kernel, entropy criterion, increasing bandwidth

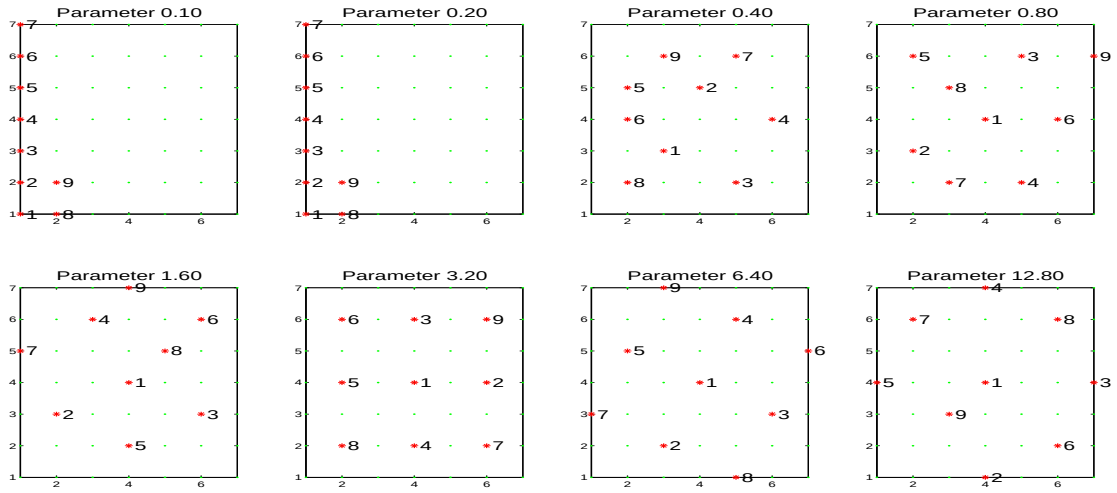


Figure 6.21: Placements under Gaussian kernel, mutual information criterion, increasing bandwidth

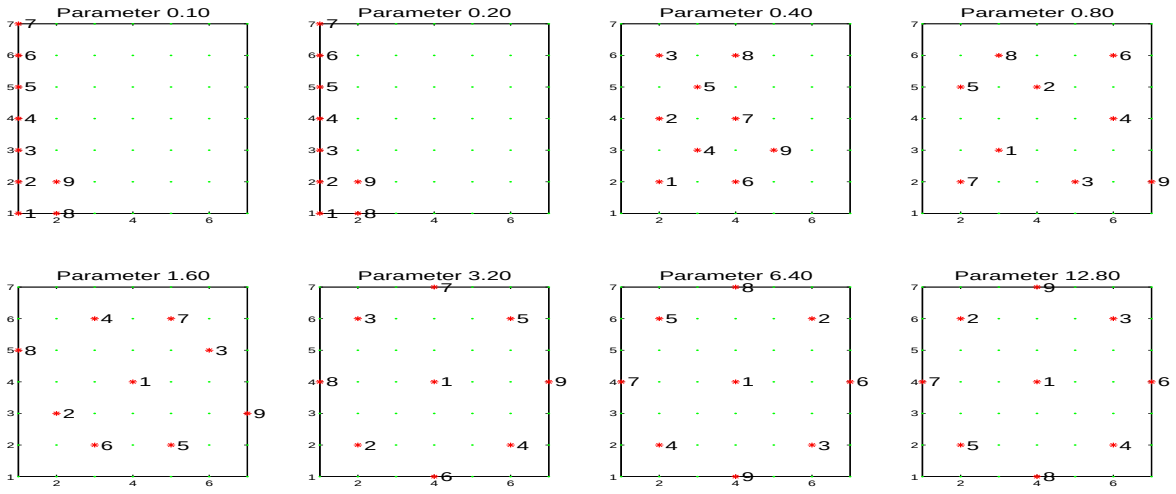


Figure 6.22: Placements under local kernel, mutual information criterion, increasing bandwidth

Chapter 7

Selecting Informative Variables in Graphical Models

In Chapter 6, we have seen that in Gaussian Processes, the problem of selecting the locations that are most informative about the non-selected locations, i.e.,

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}})$$

requires maximization of a (approximately) nondecreasing submodular function. In this chapter, we will consider a generalization of this problem, where we want to reduce the uncertainty about a specified set of target variables $\mathcal{U} \subseteq \mathcal{V}$,

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} H(\mathcal{X}_{\mathcal{U}}) - H(\mathcal{X}_{\mathcal{U}} \mid \mathcal{X}_{\mathcal{A}}).$$

We furthermore consider the general setting where the probability distribution P is specified by a graphical model over discrete random variables. This problem arises, for example, in feature selection for classification, where $\mathcal{U}$ is a single class variable we would like to predict, and $\mathcal{X}_{\mathcal{V}}$ is a collection of noisy features.

Unfortunately, in contrast to Gaussian models, where the information gain can be computed efficiently, this is not true for general graphical models, even if efficient inference can be performed, as we will discuss below. For this reason, similarly as in the setting of spatial prediction (as discussed in Chapter 6), it has been common practice [*c.f.*, Bayer-Zubek, 2004, Dittmer and Jensen, 1997, Ramakrishnan et al., 2005, van der Gaag and Wessels, 1993] to myopically (greedily) select the most *uncertain* (instead of the most *informative*) variable as the next observation, or, equivalently, the set of observations with maximum joint entropy. Unfortunately, selecting observations with maximal joint entropy is an *indirect* measure of progress towards the minimization of the remaining uncertainty about the target variables $\mathcal{U}$.

Additionally, in contrast to the mutual information criterion $H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}})$ considered in Chapter 6, the "targeted" information gain $H(\mathcal{X}_{\mathcal{U}}) - H(\mathcal{X}_{\mathcal{U}} \mid \mathcal{X}_{\mathcal{A}})$ is *not submodular* in general, as we will show below.

While the information gain is not submodular in general, in this chapter, we prove that submodularity does in fact hold under certain conditional independence conditions. However, due to the difficulty in evaluating conditional entropies $H(\mathcal{X}_{\mathcal{U}} \mid \mathcal{X}_{\mathcal{A}})$ for general graphical models, we cannot implement the

greedy algorithm directly. Fortunately, we are able to *approximately* compute $H(\mathcal{X}_\mathcal{U} \mid \mathcal{X}_\mathcal{A})$ by a sampling based algorithm with polynomial sample complexity guarantees.

Combined, these results allow us to leverage the theory of submodular functions (as discussed in Chapter 5), to present the first efficient algorithm for finding a near-optimal set of informative variables, providing a constant factor $(1 - 1/e - \varepsilon)$ approximation guarantee for any $\varepsilon > 0$ with high confidence, for any graphical model where inference can be performed efficiently, and the conditional independence conditions for submodularity are satisfied.

We furthermore prove that no polynomial time algorithm can provide an approximation with a constant factor better than $(1 - 1/e)$, unless $\mathbf{P} = \mathbf{NP}$.

Finally, we empirically evaluate our approach on several real-world sensing tasks, including building and traffic monitoring.

7.1 Selecting most informative variables

In this section, we formalize the problem addressed in this chapter: nonmyopic selection of the most informative subset of variables for graphical models.

Consider, as a running example, a temperature monitoring task, where wireless temperature sensors are distributed across a building as shown in Figure 7.1(a). Our goal in this example is to become most certain about the temperature distribution, whilst minimizing energy expenditure, a critically constrained resource [Deshpande et al., 2004]. In this example, we want to select the subset of sensors from our deployment indicated in Figure 7.1(a) that most effectively decreases expected uncertainty about temperatures in different areas of the lab.

More generally, as in Chapter 2, let $\mathcal{V}$ be a finite set indexing a set of random variables $\mathcal{X}_\mathcal{V}$. the finite set of discrete random variables in our graphical model, and $F : 2^\mathcal{V} \to \mathbb{R}$ be a set function, where $F(\mathcal{A})$ measures the reduction in uncertainty after by observing $\mathcal{X}_\mathcal{A} \subseteq \mathcal{X}_\mathcal{V}$. In most applications, observations are associated with cost, measuring, for example, the energy required to measure temperatures at particular locations. Given, a cost function $C : 2^\mathcal{V} \to \mathbb{N}$ and a budget B, we are interested in computing

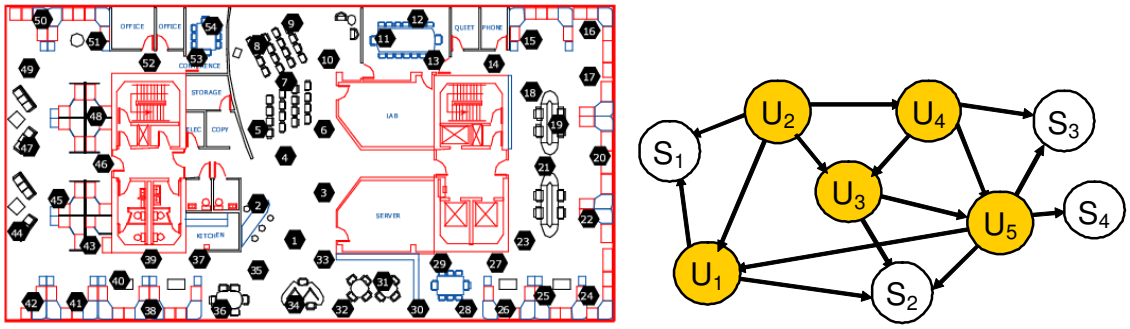
$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A} \subseteq \mathcal{V} : C(\mathcal{A}) \leq B} F(\mathcal{A}). \tag{7.1}$$

A basic version of this problem is the *unit cost* case, where every observation has unit cost, $C(\mathcal{A}) = |\mathcal{A}|$, and we are allowed to observe up to B sensors. Apart from the *unit cost* case, we will also present results for the *budgeted* case, where the cost function C is linear, i.e., each variable $s \in \mathcal{V}$ has a positive integer cost $c(s) \in \mathbb{N}$ associated with it, and $C(\mathcal{A}) = \sum_{s \in \mathcal{A}} c(s)$. Note that our approach easily extends to settings where certain variables cannot be observed at all, by setting their cost to a value greater than B.

A commonly used criterion for measuring uncertainty is the entropy of a distribution $P : \{x_1, \ldots, x_d\} \to [0, 1]$,

$$H(P) = -\sum_k P(x_k) \log P(x_k),$$

which, for discrete distributions as considered in this chapter, measures the number of bits required to encode $\{x_1, \ldots, x_d\}$ [Cover and Thomas, 1991]. If $\mathcal{X}_\mathcal{A}$ is a discrete random vector $\mathcal{X}_\mathcal{A} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$,



(a) *Sensor network deployment* (b) *Graphical model*

Figure 7.1: Sensor network deployment.

then its entropy $H(\mathcal{X}_\mathcal{A})$ is defined as the entropy of its joint distribution. The conditional entropy $H(\mathcal{X}_\mathcal{A} \mid \mathcal{X}_\mathcal{B})$ for two vectors $\mathcal{X}_\mathcal{A}, \mathcal{X}_\mathcal{B} \subseteq \mathcal{X}_\mathcal{V}$ is defined as

$$H(\mathcal{X}_\mathcal{A} \mid \mathcal{X}_\mathcal{B}) = -\sum_{\mathbf{x}_\mathcal{A}, \mathbf{x}_\mathcal{B}} P(\mathbf{x}_\mathcal{A}, \mathbf{x}_\mathcal{B}) \log P(\mathbf{x}_\mathcal{A} \mid \mathbf{x}_\mathcal{B}),$$

measuring the expected uncertainty about variables $\mathcal{X}_\mathcal{A}$ after variables $\mathcal{X}_\mathcal{B}$ are observed.

Similar to maximum entropy sampling Gaussian Processes (*c.f.*, Chapter 6), in practice, a commonly used algorithm for selecting observations is to greedily select the next variable to observe as the most uncertain variable given the ones observed thus far:

$$\mathcal{X}_k := \operatorname*{argmax}_{\mathcal{X}} H(\mathcal{X} \mid \{\mathcal{X}_1, \ldots \mathcal{X}_{k-1}\}). \tag{7.2}$$

Similarly as in Chapter 6, using the chain-rule of entropies [Cover and Thomas, 1991], $H(\mathcal{X}_\mathcal{A} \cup \mathcal{X}_\mathcal{B}) = H(\mathcal{X}_\mathcal{A} \mid \mathcal{X}_\mathcal{B}) + H(\mathcal{X}_\mathcal{B})$, we can decompose the entropy $H(\mathcal{X}_\mathcal{A})$ of a set of variables $\mathcal{X}_\mathcal{A} = \{\mathcal{X}_1, \ldots, \mathcal{X}_k\}$ as

$$H(\mathcal{X}_\mathcal{A}) = H(\mathcal{X}_k \mid \mathcal{X}_1, \ldots, \mathcal{X}_{k-1}) + \ldots + H(\mathcal{X}_2 \mid \mathcal{X}_1) + H(\mathcal{X}_1),$$

thus, suggesting that the greedy rule in Equation (7.2) is a heuristic that defines observation *subset selection* task in Equation (7.1) as the problem of selecting the set of variables that have the maximum joint entropy:

$$\operatorname*{argmax}_{\mathcal{A}: C(\mathcal{A}) \leq B} H(\mathcal{X}_\mathcal{A}). \tag{7.3}$$

A major limitation of this approach is that joint entropy is an indirect measure of information: It aims to maximize the uncertainty about the selected variables, but does not consider prediction quality for unobserved variables. In our sensor networks example, the entropy criterion often leads to the selection of sensors at the border of the sensing field, as sensors that are far apart are most uncertain about each other. Since sensors usually provide most information about the area surrounding them, these border placements "waste" part of their sensing capacity, as noticed in Chapter 6

A more *direct* measure of value of information is the information gain $I(\mathcal{X}_\mathcal{B}; \mathcal{X}_\mathcal{A})$, which is defined as

$$I(\mathcal{X}_\mathcal{B}; \mathcal{X}_\mathcal{A}) = H(\mathcal{X}_\mathcal{B}) - H(\mathcal{X}_\mathcal{B} \mid \mathcal{X}_\mathcal{A}),$$

i.e., the expected reduction in uncertainty over the variables in $\mathcal{X}_\mathcal{B}$ given the observations $\mathcal{X}_\mathcal{A}$. The analogous subset selection problem for the information gain is to compute

$$\operatorname*{argmax}_{\mathcal{A}\subseteq\mathcal{V}:C(\mathcal{A})\leq B} I(\mathcal{X}_\mathcal{V};\mathcal{X}_\mathcal{A}). \tag{7.4}$$

For example, we can model our temperature measurement task as indicated in by the graphical model in Figure 7.1(b). The temperature in the different rooms is modeled by a set $\mathcal{U} = \{\mathcal{U}_1,\ldots,\mathcal{U}_n\}$ of hidden nodes, whereas the sensors are modeled as a set of observable nodes $\mathcal{S} = \{\mathcal{S}_1,\ldots,\mathcal{S}_m\}$. For a set of sensors $\mathcal{A} \subseteq \mathcal{S}$, $H(\mathcal{X}_\mathcal{U} \mid \mathcal{X}_\mathcal{A})$ measures the expected uncertainty about $\mathcal{U}$ given the observations from the sensors in $\mathcal{A}$. Our goal is to define the set of sensor locations that most reduces uncertainty about the hidden target variables $\mathcal{U}$. In contrast to the mutual information criterion discussed in Chapter 6, this criterion allows specification of the target variables of interest $\mathcal{X}_\mathcal{U}$.

7.2 Submodularity of information theoretic objectives

In this section, we prove that under certain conditions, the information gain criterion is submodular (*c.f.*, Chapter 5). This property will be leveraged by our constant-factor approximation algorithms presented in Section 7.3.

7.2.1 Submodularity of the joint entropy

The first observation selection criterion we consider is joint entropy $H(\mathcal{X}_\mathcal{A})$ in Equation (7.3). To prove the submodularity, we use the "information never hurts" principle (*c.f.*, Chapter 6, Cover and Thomas 1991), $H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A}) \leq H(\mathcal{X}_s)$, i.e., in expectation, observing $\mathcal{A}$ cannot increase uncertainty about $\mathcal{X}_s$. Since the marginal increase can be written as $F(\mathcal{A}\cup\{s\}) - F(\mathcal{A}) = H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A})$, submodularity is simply a consequence of the information never hurts principle:

$$F(\mathcal{A}\cup\{s\}) - F(\mathcal{A}) = H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A}) \geq H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{A}'}) = F(\mathcal{A}'\cup\{s\}) - F(\mathcal{A}').$$

Submodularity of entropy has been established before [Fujishige, 1978]. Contrary to the differential entropy (*c.f.*, Chapter 6), which can be negative, in the discrete case, the entropy H is guaranteed to be nondecreasing, i.e., $F(\mathcal{A}\cup\{s\}) - F(\mathcal{A}) = H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A}) \geq 0$ for all sets $\mathcal{A} \subseteq \mathcal{V}$. Furthermore, $H(\emptyset) = 0$. Hence the approximate maximization result of Nemhauser et al. [1978], Theorem 5.2, can be used to provide approximation guarantees for observation selection based on the entropy criterion.

7.2.2 Submodularity of the information gain

As discussed in Section 7.1, information gain $F(\mathcal{A}) = I(\mathcal{X}_\mathcal{U};\mathcal{X}_\mathcal{A})$ is a more direct objective function for observation selection. Again using the information never hurts principle, we have that information gain is a nondecreasing function, and, by definition, that $F(\emptyset) = 0$. Unfortunately, the following counter-example shows that information gain is not submodular in general:

Example 7.1. *Let $\mathcal{X},\mathcal{Y}$ be independent boolean random variables with $\Pr[\mathcal{X}=1] = \Pr[\mathcal{Y}=1] = \frac{1}{2}$. Let $\mathcal{Z} = \mathcal{X}$ **XOR** $\mathcal{Y}$. Here, $H(\mathcal{Z}) = H(\mathcal{Z} \mid \mathcal{X}) = H(\mathcal{Z} \mid \mathcal{Y}) = 1$, but $H(\mathcal{Z} \mid \mathcal{X}\cup\mathcal{Y}) = 0$. Thus, $H(\mathcal{Z} \mid \mathcal{X}) - H(\mathcal{Z}) \geq H(\mathcal{Z} \mid \mathcal{X}\cup\mathcal{Y}) - H(\mathcal{Z} \mid \mathcal{Y})$, which implies that information gain, $I(\mathcal{Z};\cdot) = H(\mathcal{Z}) - H(\mathcal{Z} \mid \cdot)$, is not submodular.* □

Since submodularity is required to use the approximation result (Theorem 5.2) of Nemhauser et al. [1978], their result is not applicable to information gain, in general. Fortunately, under some weak conditional independence assumptions, we prove that information gain is guaranteed to be submodular:

Proposition 7.2. *Let $\mathcal{S},\mathcal{U}$ be disjoint subsets of $\mathcal{V}$, such that the variables $\mathcal{X}_\mathcal{S}$ are independent given $\mathcal{X}_\mathcal{U}$. Furthermore let $\mathcal{W} \subseteq \mathcal{S} \cup \mathcal{U}$. Then the set function*

$$F(\mathcal{A}) = H(\mathcal{X}_\mathcal{U}) - H(\mathcal{X}_\mathcal{U} \setminus \mathcal{X}_\mathcal{A} \mid \mathcal{X}_\mathcal{A})$$

is submodular and nondecreasing on $\mathcal{W}$, and $F(\emptyset) = 0$. □

The assumptions of Proposition 7.2 are satisfied, e.g., for graphical models with structure similar to Figure 7.1(b), where the variables $\mathcal{U}$ form a general graphical model, and the variables in $\mathcal{S}$ each depend on subsets of $\mathcal{U}$. In our sensor network example, we can interpret Proposition 7.2 in the following way: We want to minimize our uncertainty on the temperatures $\mathcal{U}$, which are measured by noisy sensors $\mathcal{S}$. We can select sensors in $\mathcal{S}$, and potentially make additional, more complicated measurements to estimate certain temperature variables in $\mathcal{U}$ directly (at some, potentially larger, cost).

The following observation identifies the joint entropy as a special case of our setting:

Corollary 7.3. *Let $F(\mathcal{A}) = H(\mathcal{X}_\mathcal{A}) = H(\mathcal{X}_\mathcal{V}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A})$. Then F is submodular and nondecreasing on any subset of $\mathcal{V}$, and $F(\emptyset) = 0$.* □

The information gain is another interesting special case:

Corollary 7.4. *Let $\mathcal{S},\mathcal{U}$ be subsets of $\mathcal{V}$ such that the variables in $\mathcal{S}$ are independent given $\mathcal{U}$. Let $F(\mathcal{A}) = I(\mathcal{X}_\mathcal{U}; \mathcal{X}_\mathcal{A}) = H(\mathcal{X}_\mathcal{U}) - H(\mathcal{X}_\mathcal{U} \mid \mathcal{X}_\mathcal{A})$. Then F is submodular and nondecreasing on $\mathcal{S}$, and $F(\emptyset) = 0$.* □

Note that Corollary 7.4 applies, for example, to the problem of attribute selection for Naive Bayes classifiers. Hence our algorithms also provide performance guarantees for this important feature selection problem. A convenient property of submodular functions is that they are closed under positive linear combinations. This allows us, for example, to have temperature models for different times of the day, and select sensors that are most informative *on average* for all models.

7.3 Approximation algorithms

In this section, we present approximation algorithms for the unit-cost and budgeted cases, leveraging the submodularity established in Section 7.2. These algorithms are based on the greedy algorithm (Algorithm 5.1) by Nemhauser et al. [1978].

We first consider the unit-cost case, where $c(s_i) = 1$ for all variables s_i, and we want to select B observations. This is exactly the setting addressed by Nemhauser et al. [1978], who prove that the greedy algorithm selects a set that is at most a factor of $1 - (1 - 1/B)^B > (1 - 1/e)$ worse than the optimal set. In our setting, the greedy rule is given by:

Proposition 7.5. *Under the same assumptions as for Proposition 7.2, when a set $\mathcal{A} \subseteq \mathcal{W}$ has already been chosen, the greedy heuristic for information gain must select the element $s^* \in \mathcal{W}$ which fulfils*

$$s^* \in \operatorname*{argmax}_{s \in \mathcal{W}\setminus\mathcal{A}} \begin{cases} H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A}) - H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{U}), & \text{for } s \in \mathcal{S}, \\ H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A}), & \text{otherwise.} \end{cases}$$ □

Algorithm 7.1: Approximation algorithm for the unit cost case.

Input: $B > 0$, graphical model G for $\mathcal{V} = \mathcal{S} \cup \mathcal{U}$, $\mathcal{W} \subseteq \mathcal{V}$
Output: Sensor selection $\mathcal{A} \subseteq \mathcal{W}$
begin
  $\mathcal{A} := \emptyset$;
  for *j := 1* **to** *B* **do**
    foreach $s \in \mathcal{W} \setminus \mathcal{A}$ **do**
      $\Delta_s := H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A})$;
      if $s \in \mathcal{S}$ **then** $\Delta_s := \Delta_s - H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{U})$;
    end
    $s^* := \operatorname{argmax}\{\Delta_s : s \in \mathcal{W} \setminus \mathcal{A}\}$;
    $\mathcal{A} := \mathcal{A} \cup \{s^*\}$;
  end
end

Proposition 7.5 has an interesting interpretation: our greedy rule for information gain is very similar to that of the greedy heuristic for maximizing joint entropy, but our information gain criterion is penalized for selecting sensors $s \in \mathcal{S}$ which are "unrelated" to the variables of interest, i.e., those where $H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{U})$ is large.

The greedy algorithm using the rule from Proposition 7.5 is presented in Algorithm 7.1, and we summarize our analysis in the following Theorem:

Theorem 7.6. *Algorithm 7.1 selects a subset $\mathcal{A} \subseteq \mathcal{W}$ of size B such that $F(\mathcal{A}) \geq (1 - 1/e)OPT$, where OPT is the information gain of the optimal set as defined in Equation (7.4), using $\mathcal{O}(B|\mathcal{W}|)$ computations of conditional entropies.* □

The result of Nemhauser et al. [1978] only provides guarantees for the maximization of $F(\mathcal{A})$ for the case where the cost of all observations is the same, and we are simply trying to find the best B observations. Often, different variables have different costs. In our sensor network example, if our sensors are equipped with solar power, sensing during the day time might be cheaper than sensing during the night. In general, the deployment cost for sensors might depend on the location. These considerations motivate the consideration of more general cost functions $C(\mathcal{A}) = \sum_{s \in \mathcal{A}} c(s)$, where each variable has a fixed cost $c(s)$. In this case, we can apply the CELF algorithm (Algorithm 5.4) to obtain a near-optimal solution.

7.4 Hardness of approximation for optimizing information gain

In this section, we prove that the problem of maximizing the information gain, even in our formulation using conditional independence, cannot be approximated by a constant factor better than $(1 - 1/e)$, unless $\mathbf{P} = \mathbf{NP}$.

We formalize the optimization problem MAXINFOGAIN in the following way: The instances of the problem consist of an integer B and a set $\mathcal{V} = \mathcal{U} \cup \mathcal{S}$ of random variables such that the variables in $\mathcal{S}$ are independent given all $\mathcal{U}$, and a polynomial time Turing machine M for computing $I(\mathcal{U}; \mathcal{A})$ for any subset $\mathcal{A} \subseteq \mathcal{S}$ of size up to B. The Turing machine is specified in binary, along with an integer N providing a runtime guarantee. This runtime guarantee is necessary in order to enable the verification of

Algorithm 7.2: Sampling algorithm for conditional entropy.

Input: $N > 0$, graphical model G for $\mathcal{V}$, $\mathcal{B} \subseteq \mathcal{V}$, $s \in \mathcal{V} \setminus \mathcal{B}$
Output: $\approx H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{B})$
begin
  $\hat{H} := 0$;
  for *i = 1* **to** N **do**
    generate sample $\mathbf{x}_\mathcal{B}$ of $\mathcal{X}_\mathcal{B}$ from G;
    use prob. inference in G to compute $P(\mathcal{X}_s \mid \mathbf{x}_\mathcal{B})$;
    $H(\mathcal{X}_s \mid \mathbf{x}_\mathcal{B}) \leftarrow -\sum_{x_s} P(x_s \mid \mathbf{x}_\mathcal{B}) \log P(x_s \mid \mathbf{x}_\mathcal{B})$;
    $\hat{H} \leftarrow \hat{H} + \frac{1}{N} H(\mathcal{X}_s \mid \mathbf{x}_\mathcal{B})$;
  end
  return $\hat{H}$;
end

the computations. The solutions comprise the maximum information gain achieved by a subset of size at most B.

Theorem 7.7. *The problem MAXINFOGAIN is not approximable within a constant factor better than* $(1 - 1/e)$*, unless* $\mathbf{P} = \mathbf{NP}$. □

Technically, we have to note that even computing conditional entropies can be an intractable problem on its own. Hence, we state our result for problem instances for which these computations can be carried out efficiently, as is the one used in our reduction.

7.5 Approximating conditional entropies

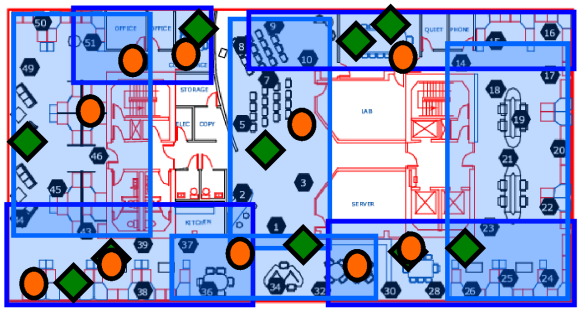
The algorithms presented in Section 7.3 require the evaluation of the marginal increases $F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})$. Proposition 7.5 states that this quantity can be computed using one-dimensional conditional entropies. Unfortunately, there is no known efficient algorithm for computing conditional entropies even for discrete polytree graphical models. In fact, given the hardness results which we will develop in Chapter 15, it seems unlikely that such an efficient algorithm can exist. This observation motivates the need for approximating these entropies. In this section, we describe a sampling approach that only requires a polynomial number of samples to approximate conditional entropies by an arbitrary ε difference, with high confidence. Our method in Algorithm 7.2, which applies to any problem where conditional entropies have to be computed, has sample complexity stated in the following Lemma:

Lemma 7.8. *Algorithm 7.2 approximates* $H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{B})$ *with absolute error* ε *and confidence* $1 - \delta$ *using*

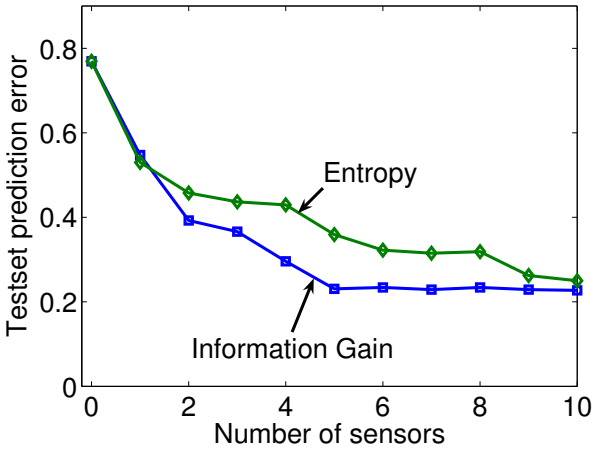
$$N = \left\lceil \frac{1}{2} \left(\frac{\log |\operatorname{dom}(\mathcal{X}_s)|}{\varepsilon} \right)^2 \log \frac{1}{\delta} \right\rceil$$

samples, for any $\varepsilon > 0$ *and* $\delta > 0$. □

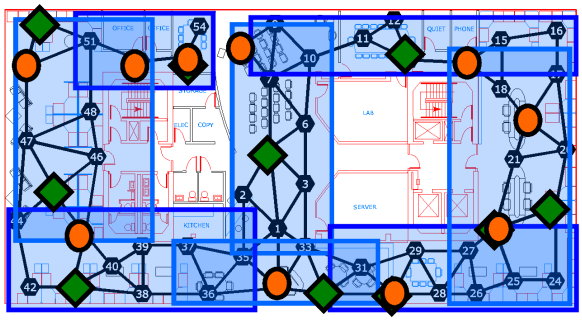
In order to use this approximation method for conditional entropies in our observation selection algorithm, we have to ensure that the approximate computation of the marginal increases does not significantly worsen our approximation guarantee. Fortunately, if we can compute the marginal increases



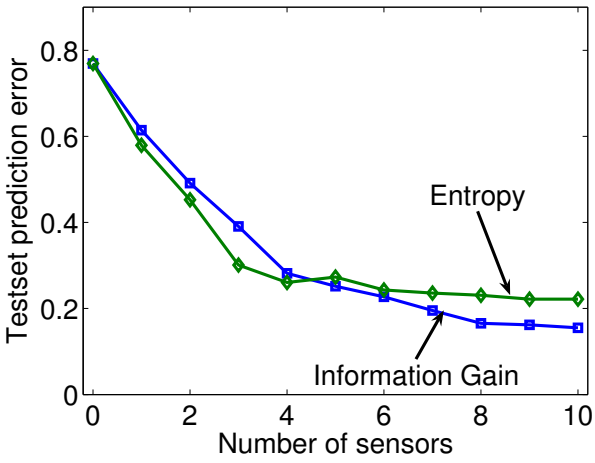
(a) *Placements for hierarc. model*



(b) *Prediction error*



(c) *Placements for violating model*



(d) *Prediction error*

Figure 7.2: Results for sensor network deployment.

$F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})$ with an absolute error of at most ε, an argument very similar to the one presented by Nemhauser et al. [1978] shows that the greedy algorithm will then provide a solution $\hat{\mathcal{A}}$ such that $F(\hat{\mathcal{A}}) \geq (1 - 1/e)OPT - 2B\varepsilon$ with high confidence. The following Theorem summarizes our analysis of Algorithm 7.1, using Algorithm 7.2 for approximate computation of conditional entropies:

Theorem 7.9. *Algorithm 7.1 computes a set $\hat{\mathcal{A}}$ such that*

$$F(\hat{\mathcal{A}}) \geq (1 - 1/e)OPT - \varepsilon,$$

with probability at least $1 - \delta$, in time

$$\mathcal{O}\left(c_{inf}\ B\ |\mathcal{W}|\ \left\lceil \left(\frac{B \log K}{\varepsilon}\right)^2 \log \frac{B|\mathcal{W}|}{\delta} \right\rceil\right)$$

for any $\varepsilon > 0$ and $\delta > 0$ where c_{inf} is the cost of probabilistic inference, and $K = \max_{s \in \mathcal{W}} |\operatorname{dom}(\mathcal{X}_s)|$. □

Similar guarantees can be obtained for the CELF algorithm (*c.f.*, Chapter 5) in the case of non-constant cost functions.

Note that, in order to guarantee polynomial running time for Algorithm 7.1, the cost c_{inf} of sampling from the graphical model and the probabilistic inference step in Algorithm 7.2 has to be polynomial. For the case of discrete polytrees however, efficient sampling and inference are possible, as is for all graphs of bounded treewidth.

7.6 Experimental results

In this section, we present empirical evidence of the effectiveness of method on two real-world data sets.

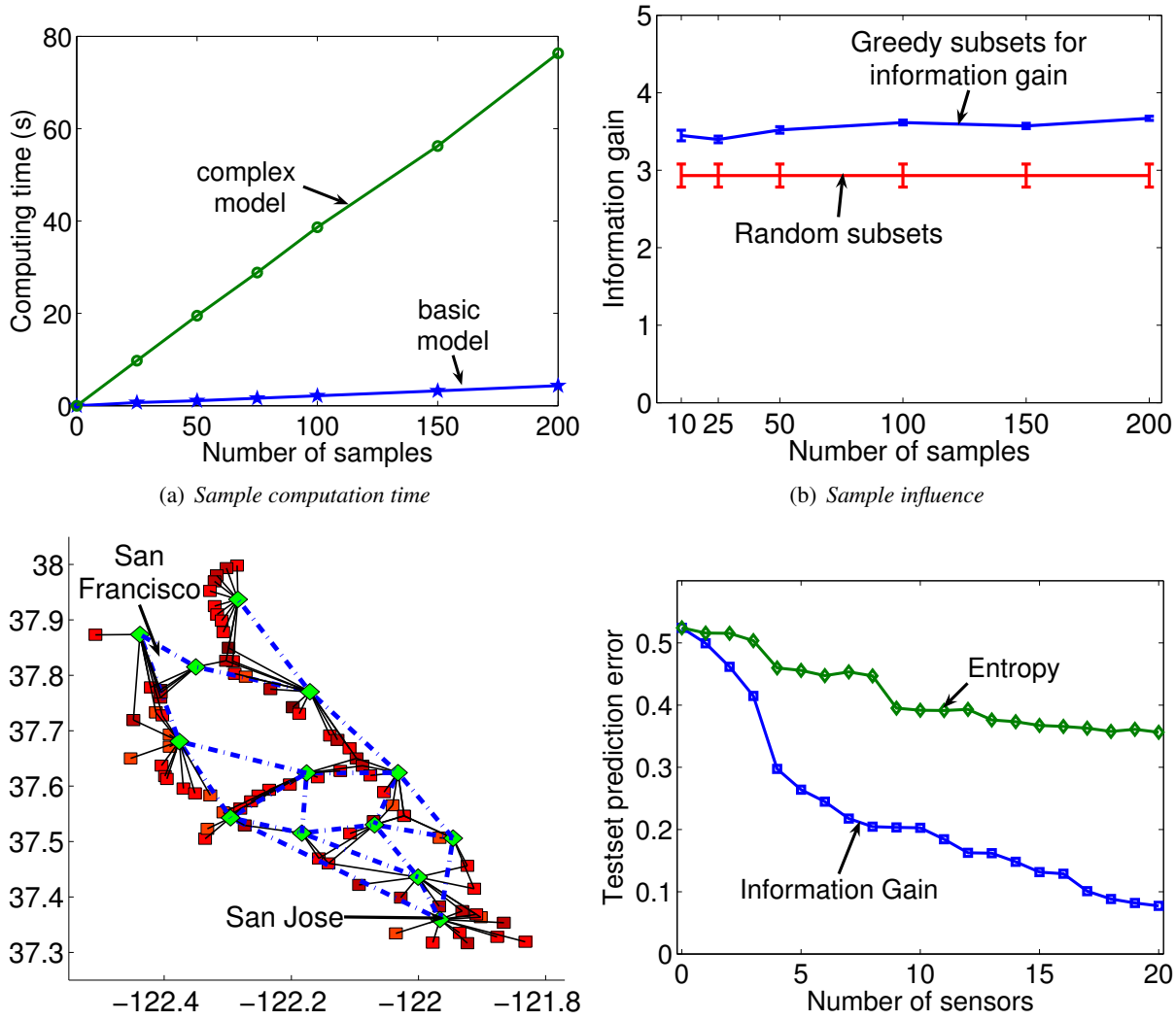
7.6.1 Temperature data

In our first set of experiments, we analyzed temperature measurements from the sensor network deployment at Intel Research Berkeley, as mentioned in our running example (and as considered in Chapter 6). We fit a hierarchical model to the data, as indicated in Figure 7.2(a). The overlapping rectangles correspond to larger areas of the research lab, where each region is associated with a random variable representing the mean of all sensors contained in this region. Since our regions overlap, sensors may be in more than one region. The sensors are assumed to be conditionally independent given the values of all aggregate nodes.

Our training data consisted of samples collected at 30 sec. intervals from 6 am till 8 pm on five consecutive days (starting Feb. 28th 2004). The test data consisted of the corresponding samples on the two following days. Data was discretized into five bins of three degrees Celsius each. We used our approximation algorithms to learn informative subsets of the original sensors. Figure 7.2(a) shows the sensors selected by the information gain criterion (circles) and by the entropy criterion (diamonds) for the unit-cost optimization with $B = 10$. Figure 7.2(b) compares the prediction quality (measured as misclassification errors) on the aggregate nodes for both subset selections, using varying subset sizes. The prediction quality of the subset selected by our information gain criterion is significantly better than that of the entropy criterion. Using only five sensors, information gain achieved a prediction accuracy which was not met by the entropy criterion even for subset sizes up to 20 sensors.

To verify whether the algorithm for maximizing the information gain achieves viable results even in the case where its assumptions of conditional independence are violated, we performed another experiment. In addition to the identical hierarchical structure, our new graphical model connects adjacent sensors as indicated in Figure 7.2(c), which also shows ten sensor subsets selected using the entropy and information gain criteria. Figure 7.2(d) presents the test set prediction errors for both criteria. It can be seen that, in comparison to Figure 7.2(b), the prediction accuracy is not much improved by this more complex model, though our information gain criterion manages to decrease the prediction error to below 20 percent, which posed a hard barrier for the basic model.

Figure 7.3(a) presents the time required for computing the greedy update steps for a varying number of samples, both for the basic and complex model. As predicted in Theorem 7.6, the run time is linear in the number of samples, the total number of sensors and the size of the selected subset. To assess how the number of samples affects the performance of Algorithm 7.1, Figure 7.3(b) displays the information gain computed for five element subsets for a varying number of samples. There is no significant increase of the



(a) *Sample computation time*

(b) *Sample influence*

(c) *Hierarchical model*

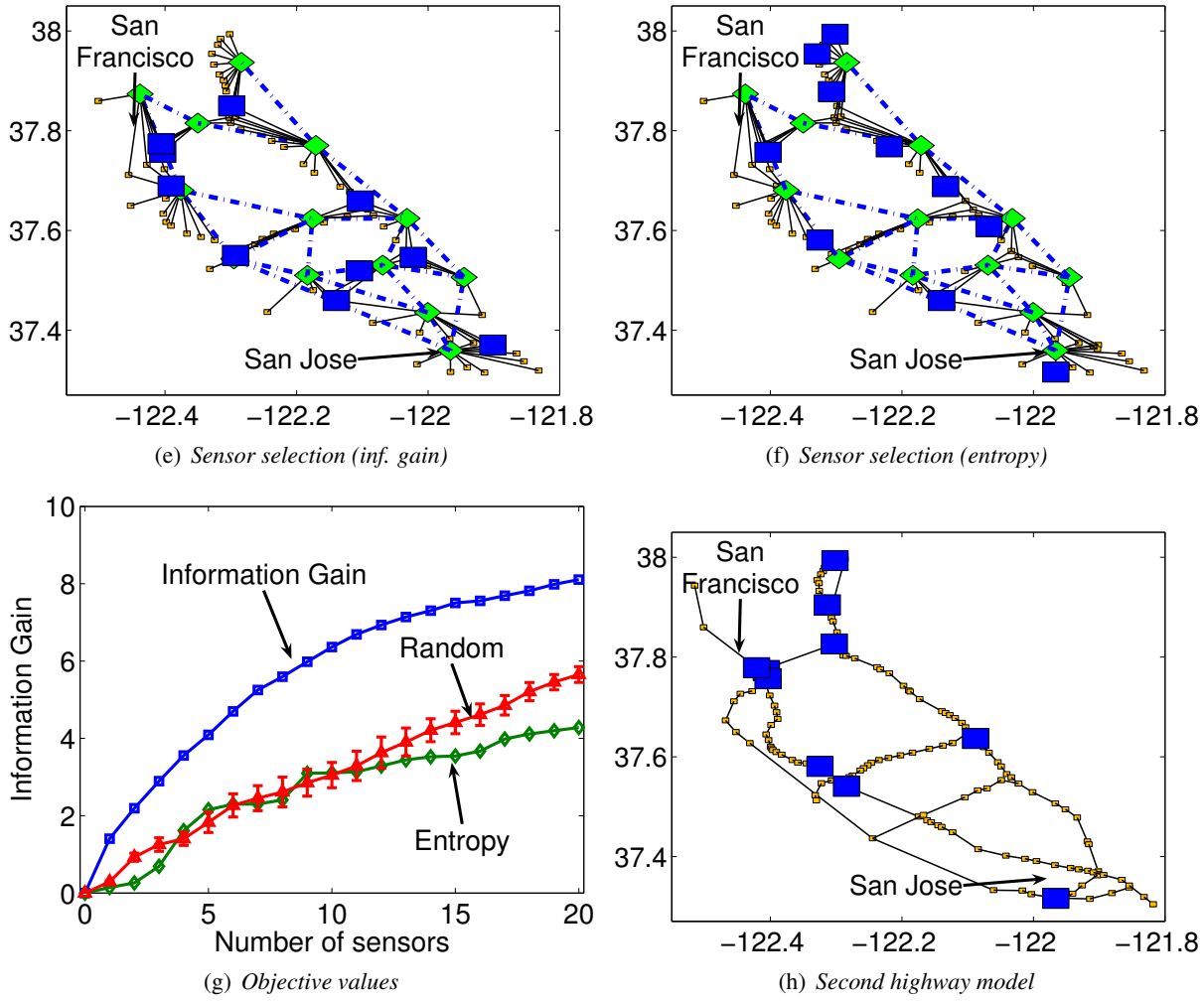
(d) *Prediction error*

mean information gain, suggesting that the bounds presented in Theorem 7.8 are very loose for practical applications.

7.6.2 Highway traffic data

In our second set of experiments, we analyzed highway traffic from the San Francisco Bay area [UC Berkeley, PATH and Caltrans, 2005]. Each detector station, 77 in total, computed aggregated measurements over 5 minutes, reporting the total number of vehicle miles traveled divided by the total vehicle hours for its region.

As for the the temperature data, we fit a hierarchical model, shown in Figure 7.3(c). The bay area was logically segmented into overlapping regions, covering bridges and major highway segments. The diamonds represent the aggregate nodes, and the detector stations (squares) are assumed to be conditionally



(e) *Sensor selection (inf. gain)*

(f) *Sensor selection (entropy)*

(g) *Objective values*

(h) *Second highway model*

Figure 7.3: Results for sensor network deployment and highway traffic data.

independent given all aggregate nodes. The data was collected over 18 days, of which the last two days were chosen for testing prediction accuracy. We chose a discretization into five bins: below 50 mph, between 50-60, 60-65, 65-70 and more than 70 mph, and the task was to predict the minimum speed over all detectors within each region. Figure 7.3(d) compares the prediction accuracy for subsets selected using the entropy and information gain criteria. Information gain significantly outperforms the maximum entropy selection. Figures 7.3(e) and 7.3(f) show the selected sensor locations for information gain and entropy.

In another experiment, we compared the information gain objective values computed for subsets selected using the greedy heuristic for both information gain and entropy criteria, and random selections; Figure 7.3(g) presents these results. Interestingly, the average information gain for randomly chosen subsets is even slightly higher than using the greedy entropy maximization. This indicates that maximizing entropy and information gain appear to be conflicting goals, resulting in qualitatively different behaviors.

Analogously to the temperature data, we performed another experiment with a different model. Instead of a hierarchical model satisfying the conditional independence assumptions for maximization of the in-

Figure 7.4: On the left is the commercially-available high-resolution Tekscan ConforMat sensor (with 2048 sensors) installed on a Herman Miller Aeron Chair. One the right is our deployed system (with 19 sensors) installed on the same chair.

formation gain, we fit a larger Bayesian network for which the 153 detector stations along the highways were connected, as indicated in Figure 7.3(h). This figure also indicates the placement of the most informative sensors, which are concentrated on very similar locations as in the hierarchical model (*c.f.*, Figure 7.3(f)).

7.7 Case study: Sensor placement for posture recognition

In this section, we present a detailed case study that uses the greedy algorithm for optimizing the information gain $H(\mathcal{X}_{\mathcal{U}}) - H(\mathcal{X}_{\mathcal{U}} \mid \mathcal{X}_{\mathcal{A}})$ in order to place sensors for seating posture recognition.

Many people spend a large part of their day sitting, be it in an office chair, couch or car seat. In these contexts, sustained poor sitting posture can cause significant adverse effects. For example, back pain affects 60 % to 80% of adults in the U.S. at some time in time in their lives, causing expensive absence from work [Deyo, 1994]. Elderly nursing home residents, who often spend long periods of time seated, are likely to develop ulcers from sitting for long times [AHCPR, 1992, Brandeis et al., 1994, Smith, 1995]. Improved seating comfort for these individuals augmented with technology interventions may have significant impact on health productivity and injury prevention.

Posture is a primary measure of sitting comfort (and discomfort) [de Looze et al., 2003]. Posture information could also help predict a sitter's state. For instance, subtle changes in seating posture (e.g., when talking to other people in the car) can indicate attentiveness; unattentive driving can lead to an increased chance of accidents. Hence, posture information can potentially be used to reduce severe adverse effects (health problems, decreased productivity or car accidents).

Previous work in posture recognition [Tan et al., 2001, Z. et al., 1997, Zhu et al., 2003] used a high-fidelity pressure sensor placed on the seat to acquire posture information. While these approaches demonstrated

significant potential for using posture recognition as input modality, their approach had several practical limitations:

- While they reported high classification accuracies on *familiar* subjects (subjects on which the posture recognition system was trained), the classification accuracy on *unfamiliar* subjects (not seen during the training process) is either not reported or significantly lower than results with *familiar* subjects. However, the training process is rather expensive and time-consuming, so for a widespread deployment, the recognition must generalize across subjects.
- They used an expensive sensor to acquire the necessary pressure information for classification. This high cost makes widespread deployment difficult.
- Due to the high resolution, signal processing is expensive, requiring sophisticated hardware satisfying high computational demand.

In our case study, we consider an alternative approach to the posture recognition problem in order to address these challenges. By using our variable selection algorithms, we replace the expensive, high resolution sensor by a small set of near-optimally placed sensors. Hence the amount of sensor data collected is reduced as well, reducing the computational load, and enabling real-time classification performance. Our system achieves high classification accuracy on *unfamiliar* subjects, while using less than 1% of the sensors (and less cost, even for our prototype deployment) when compared to previous work.

7.7.1 Methodology

Our goal is to recognize (classify) seating postures. As input, we are using sensor data acquired from a collection of pressure sensors placed on the seat and back of a chair, as displayed in Figure 7.4. At given intervals, sensor data is acquired, processed, and a label for the posture (such as *sitting straight*, *leaning left*, and *slouching*) is predicted, which can then be used as input to an adaptive system. Discussion on possible applications are provided at the end of the case study.

Learning in our problem context requires (1) an *algorithm* for learning and prediction of postural information, and (2) a set of *features* that explain the posture data. While there is a large literature on how classifier g can be learned from training data, both the training algorithm and the set of features should be selected with a deep understanding of the recognition problem. Therefore, as a first step in choosing an algorithm and creating meaningful features for our classification, we did extensive analyses of postural data.

Understanding Posture Data

In the posture classification problem, at every time step t our system is given the sensor values $\mathcal{X}_\mathcal{V}$, where m is the number of sensors. Our goal is to output a class label $\mathcal{Y} \in \{1, \ldots, C\}$, indicating the sitter's posture. The sensor values $\mathbf{x}_\mathcal{V}$ are in the following referred to as *pressure maps* – Figure 7.5 presents examples of such pressure maps for different postures, obtained by the Tekscan high resolution pressure sensor used in earlier work [Tan et al., 2001, Z. et al., 1997, Zhu et al., 2003]. Given a set of training data $\mathcal{D}_{train} = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, we would like to learn a classifier g – such that $\mathcal{Y} \approx g(\mathbf{x}_\mathcal{V})$, i.e., $g(\mathbf{x}_\mathcal{V})$ is approximately the true class label, not just on the training set, but also on unseen input data. This requirement is known as *generalization*. In posture recognition, we specifically want the classifier to *generalize* between different subjects, i.e., to work well on *unfamiliar* subjects.

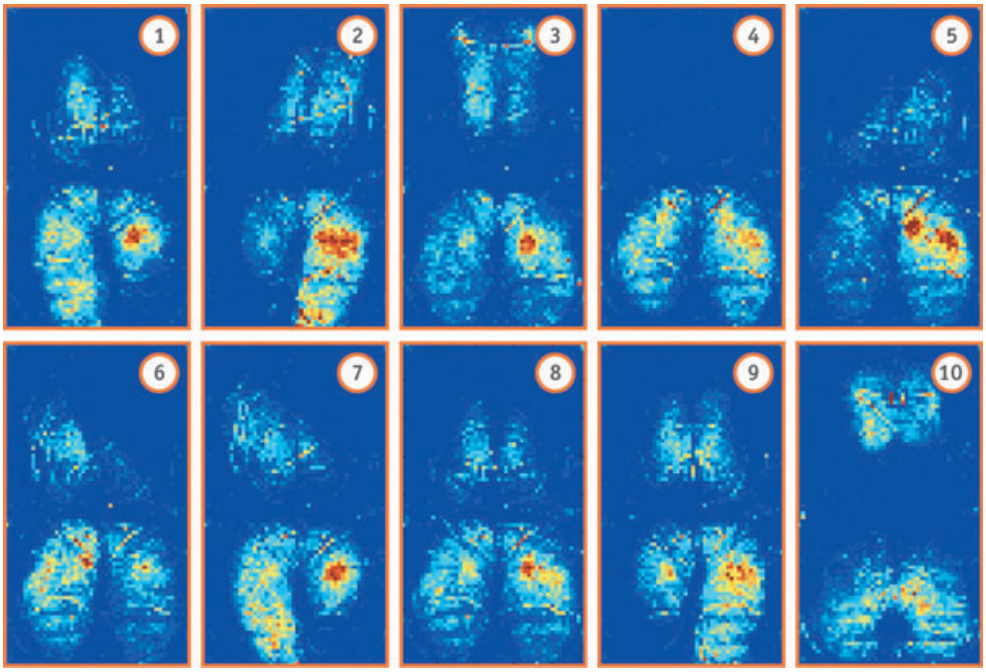


Figure 7.5: Sample data on pressure applied to the seat and back of an office chair. Samples display postures: *(1) Left leg crossed, (2) Right leg crossed, leaning left, (3) Leaning back, (4) Leaning forward, (5) Leaning left, (6) Leaning right, (7) Left leg crossed, leaning right, (8) Seated upright, (9) Right leg crossed, (10) Slouching*.

When we use the high-resolution Tekscan sensor, the number m of sensor values is very large, and we incur what is known as the *curse of dimensionality* problem [Bellman, 1957a] – generalization in a high dimensional data set is very difficult. Consequently, the two prior approaches in posture recognition [Tan et al., 2001, Zhu et al., 2003] deal with this problem by explicitly reducing the dimensionality of the input data using Principal Component Analysis (PCA) and Sliced Inverse Regression (SIR) respectively. These dimension reduction techniques aim at finding a low dimensional subspace which most effectively represents the training data, in the sense of minimizing the mean square reconstruction error. [1]

In our analyses of the data, we explored how domain knowledge could be used to engineer a small set of features $\mathbf{f}$, and learn a classifier g mapping the feature values to the class label, i.e., $\mathcal{Y} \approx g(\mathbf{f})$. Our experimental results show that this approach leads to improved generalization performance on unfamiliar subjects, compared to existing work. Furthermore, we demonstrate that, contrary to all work to date, our approach does not depend on the Tekscan high-resolution sensor – in the following, we show how we can reconstruct the chosen features from a small set of near-optimally placed sensors, thereby achieving similar classification accuracy at a fraction of the material and computational cost. Figure 7.6 provides a diagrammatic illustration of our approach.

[1]The difference in their approaches is that while PCA is independent of the classification labels, SIR (as a generalization of Linear Discriminant Analysis) aims at selecting features (dimensions) which are capture most of the class specific variance. Both approaches however do *not* explicitly select features which are tailored to improve the classification results.

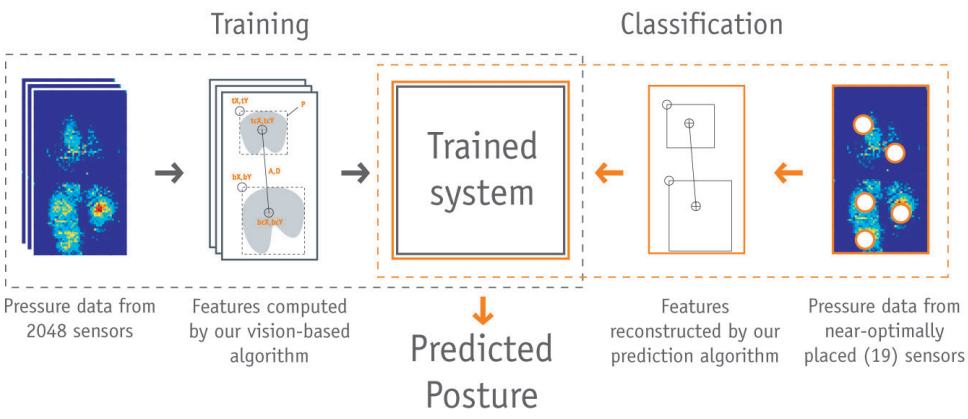


Figure 7.6: A diagrammatic illustration of our classifier.

Features

Zhu et al. [2003] proposed a normalization step for preprocessing, in which the pressure maps are *cropped* to contain only areas where the sensor reading is non-zero. We argue that this cropping step discards valuable information about user's posture. Furthermore, our experiments have shown that *where* pressure is applied is more informative about user's posture than the *amount* of pressure applied on the surface. However, prediction is improved by knowing how much pressure is applied, but maximized when sensor readings are aggregated into equal subdivisions of the pressure areas.

Our features represent these two kinds of information; (1) where the pressure was applied at the bottom of the seat and at the back of the seat, and how these two areas are interrelated (illustrated with items (1) to (4) in Figure 7.7), and (2) how much pressure was applied in these areas (item (5) in Figure 7.7). Following this framework, we designed a set of 51 features and used a feature selection algorithm to find the most informative features based on classification accuracy [Guyon et al., 2002] to identify a subset of 30 features that are the most informative of the class to be predicted (see Figure 7.10).

Description of Features.

1. The position and size of the bounding box of the pressure area on each surface (seat and back). In finding the bounding boxes, we first apply Hysteresis thresholding [Canny, 1986] to eliminate noisy edges for better detection of the pressure areas. The pressure areas are computed using a simple connected components algorithm [Rosenfeld and Pfaltz, 1966] and finding the bounding boxes of the clustered areas.

2. The distance of the bounding boxes of the pressure areas at the bottom and the back to the edges of the seat.

3. The distance and angle between the centers of the pressure areas at the back and the bottom surfaces.

4. The centers, radii, and orientations of two ellipses fit to the pressure areas created by the sitter's buttocks. These ellipses are calculated using a Least-Squares ellipse fitting algorithm described by Fitzgibbon et al. [1999].

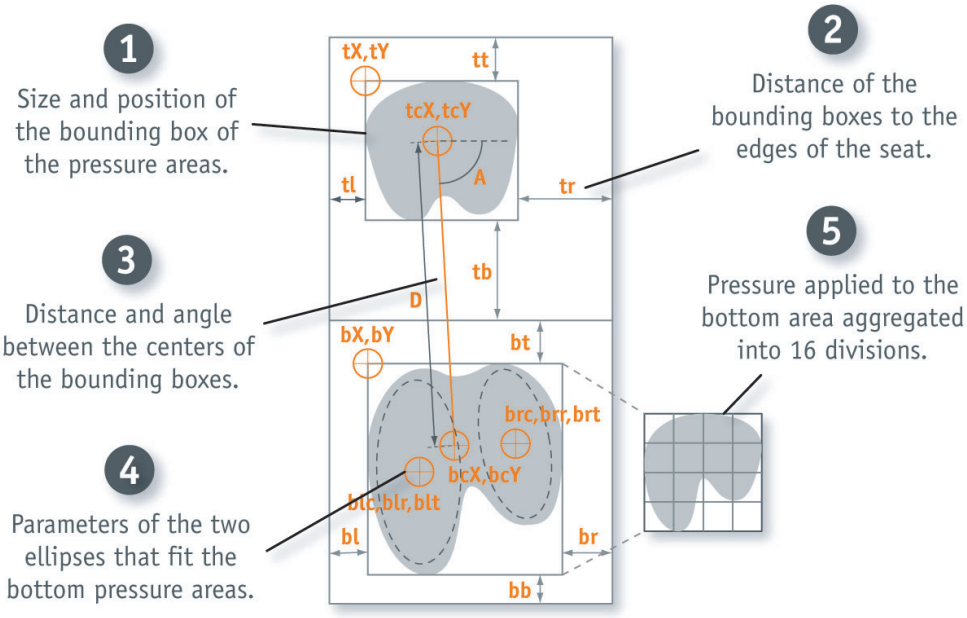


Figure 7.7: Our set of features derived from geometric and physical variability in seated postures as well as pressure information. (1) to (4) describe geometric features. (5) shows pressure applied to the bottom seating area divided into 16 equal aggregated pressure areas, which among all other divisions that we tested (pressure area divided into equal 1, 4, 9, and 25 regions) provided us with the highest classification accuracy.

5. Pressure applied to the bottom area divided into 16 aggregated sub-regions. We decided on dividing the pressure area into 16 aggregated sub-areas empirically. This division outperformed other candidate divisions (into 1, 4, 9, 16, or 25 regions) in a cross-validation experiment using a Logistic Regression classifier.

We tested the performance of our features against features used by existing work. For instance, the method described by Tan et al. [2001] uses Principal Component Analysis to reduce the dimensionality of the data and extract the features (Principal Components) that are most informative of the data. A cross-validation experiment using our features outperformed the accuracy provided by Tan et al. [2001]. We also tested linear separation performance achieved by our features $\mathcal{F}$ against that of PCA. Figure 7.8 displays the results of this experiment where our features achieved significantly less misclassified instances.

Learning Algorithm

After we extract the feature values from the training data, we train a classifier based on Logistic Regression, a classification technique which finds a linear decision boundary to separate the classes. For instance, to separate two postures, the classifier finds a vector $\mathbf{w} = (w_1, \ldots, w_m)$ such that $\mathbf{w}^T\mathbf{f} = \sum_{l=1}^{m} w_l f_l$ is positive for feature values $\mathbf{f} = (f_1, \ldots, f_m)$ from one posture, and negative for the other. For k postures,

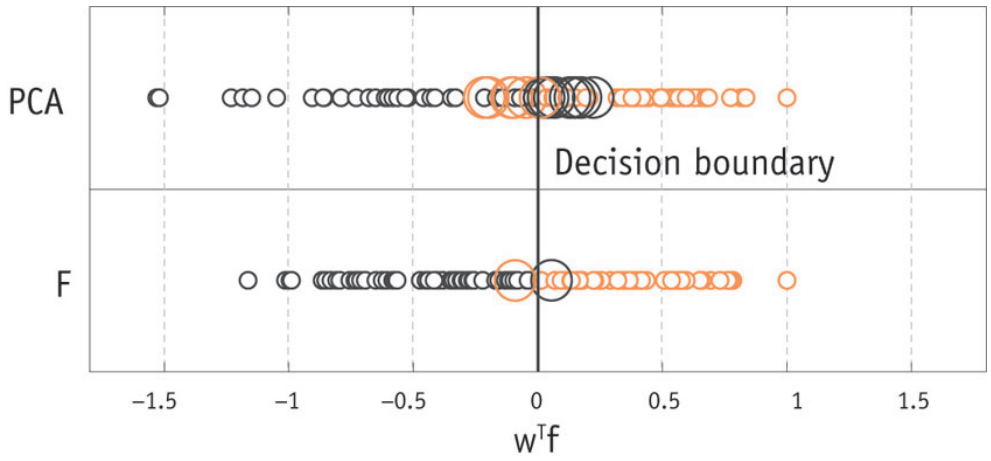


Figure 7.8: Best linear separation achieved using PCA (top) and our features (F at the bottom), for distinguishing postures *(2) Right leg crossed, leaning left* and *(9) Right leg crossed*. Big circles mark misclassified examples. For both PCA and F, we train a Logistic Regression classifier, and project the examples (i.e., compute $\mathbf{w}^T\mathbf{f}$) along the normal vector $\mathbf{w}$ of the respective decision boundary. F achieves significantly better separation.

it models the conditional likelihood of posture j, $P(y = j \mid \mathcal{F} = \mathbf{f})$ given features $\mathcal{F} = \mathbf{f}$ as

$$P(y = j \mid \mathcal{F} = \mathbf{f}) = \frac{\exp(-\mathbf{w}_j^T\mathbf{f})}{\sum_{l=1}^{k}\exp(-\mathbf{w}_l^T\mathbf{f})}.$$

Hereby, $\mathbf{w}_l$ are posture specific vectors found by the training algorithm. In order to classify an unseen example $\mathbf{f}$, the algorithm would select the most probable posture, i.e., select j such that $P(y = j \mid \mathcal{F} = \mathbf{f})$ is maximized.

One way to find the coefficients $\mathbf{w}_l$ is maximum conditional likelihood estimation, i.e., to find the coefficients which maximize the likelihood of observing all labels $y^{(i)}$ for all features $\mathbf{f}^{(i)}$, where i is the index of the training example. Another approach, which often is more robust, is to assume a *sparse* representation of $\mathbf{w}_j$. In this sparse representation, each weight vector $\mathbf{w}_j$ (for the j-th posture) depends only on a small set of features, i.e., has only few non-zero entries. Note that these non-zero entries can be different for each posture, i.e., every posture can be characterized by a different subset of features. This sparse approach favors *simpler* models, which leads to improved generalization to unseen data. The *SimpleLogistic* algorithm [Landwehr et al., 2003] finds such a sparse representation by using the LogitBoost algorithm of Friedman et al. [1998] to optimize an additive logistic regression model; each additive component is a simple linear function h depending on a single feature $h(\mathbf{f}) = w_l f_l$ for some scalar weight w_l and index l. The additive components are optimized on an appropriately weighted training set. Hereby, the weights for each new component are chosen as to minimize the residual classification error, i.e., a new weight is added to a new feature, such that the number of examples (misclassified using the existing weights) is minimized. This iterative procedure stops when the classification accuracy cannot substantially be improved by adding more features. Details of this approach can be found by Friedman et al. [1998], Landwehr et al. [2003]. We selected this approach after empirical comparison with several other classification techniques.

The advantage of using Logistic Regression in addition to achieving high classification accuracy, is that instead of simply outputting a class label, it outputs a probability distribution over class labels, which contains an estimate of the predictive uncertainty. This notion of confidence may be important for the application using the posture information.

Near-Optimal Sensor Placement

Although the approach described in the previous section achieves high classification accuracy, it requires input data from a high resolution pressure sensor, in order to accurately compute the features discussed above. The cost of this sensor provides a significant burden for its large scale deployment, e.g., in automobiles, wheelchairs, or office chairs. In addition, the features need to be computed from the high dimensional data in real-time, demanding high computational performance. In order to avoid these problems, we would like to replace this high resolution sensor with a small set of pressure sensors, placed in order to provide similar classification accuracy. If $\mathcal{V}$ is the set of all possible sensor locations (e.g., the grid cells of the Tekscan sensor), we want to find a subset $\mathcal{A} \subseteq \mathcal{V}$ of these sensors, such that, given the pressure $\mathbf{x}_\mathcal{A}$ at only these locations, our system can classify the posture as accurately as possible.

The high performance of our features motivated the following approach towards selecting near-optimal sensor locations: Our algorithm selects sensor locations, which allow us to *reconstruct* the features $\mathbf{f}$ (e.g. the bounding box of pressure, angle between mass centroids, etc.) as accurately as possible. After reconstruction, we use the classifier trained on the features $\mathbf{f}$ to classify the posture. In order to achieve the reconstruction, our classifier learns a probabilistic model that describes the correlation between the measured sensor values and the predicted feature values. This probabilistic model allows us to select sensor locations that minimize the *uncertainty* in the prediction of the feature variables.

More formally, we model the pressure maps using random variables $\mathcal{X}_\mathcal{V}$, where each $\mathcal{X}_s \in \mathcal{X}_\mathcal{V}$ for $s \in \mathcal{V}$ represents the sensor value obtained from the Tekscan sensor at location s. Furthermore, we define random variables $\mathcal{F}$ that model the values of the features (that depend on the sensor values). If we assume that we have a statistical model for the joint distribution, $P(\mathcal{X}_\mathcal{V}, \mathcal{F})$ (we will describe how we fit the model later), and if we know the values of all the sensors, $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$, we can compute the feature values $\mathcal{F} = \mathbf{f}$ exactly, without any uncertainty. If we can only observe a subset $\mathcal{A} \subseteq \mathcal{V}$ of the sensor values, i.e., $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, then the conditional probability $P(\mathcal{F} = \mathbf{f} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$ can be computed, and feature values $\mathbf{f}$ that maximize this conditional probability can be predicted.

Based on this model, we can then choose sensor locations $\mathcal{A}$ that are most informative about the feature variables $\mathcal{F}$. Hence, we need to find the subset of variables $\mathcal{A}$ which has the highest *information gain*

$$F(\mathcal{A}) = I(\mathcal{F}, \mathcal{X}_\mathcal{A}) = H(\mathcal{F}) - H(\mathcal{F} \mid \mathcal{X}_\mathcal{A})$$

with respect to the feature variables, as considered earlier in this chapter[2]. In our optimization, we constrain the set $\mathcal{A}$ to have a fixed number k of elements. The results derived in this chapter motivate the use of the greedy algorithm for selecting sensor locations, which starts with the empty set $\mathcal{A} = \emptyset$, and iteratively adds the sensor location s^* which increases the information gain the most, i.e., adds

$$s^* = \operatorname*{argmax}_s F(\mathcal{A} \cup \{s\})$$

to the set $\mathcal{A}$.

In order to use this algorithm for sensor placement, we need to specify a joint distribution $P(\mathcal{F}, \mathcal{X}_\mathcal{V})$. In our approach, we use the multivariate normal distribution; this distribution is fully parameterized by a

[2]We also considered the information gain $I(\mathcal{Y}; \mathcal{X}_\mathcal{A})$. However, using the greedy algorithm on this objective led to worse performance than for $I(\mathcal{F}; \mathcal{X}_\mathcal{A})$, as shown in our experiments. We hypothesize that this is due to the fact that $\mathcal{X}_\mathcal{V}$ is approximately independent given $\mathcal{F}$ and hence $I(\mathcal{F}; \mathcal{X}_\mathcal{A})$ is approximately submodular, whereas $\mathcal{X}_\mathcal{V}$ is not conditionally independent given $\mathcal{Y}$ alone, hence possibly violating submodularity of $I(\mathcal{Y}; \mathcal{X}_\mathcal{A})$

mean vector μ and a covariance matrix Σ, i.e.,

$$P(\mathbf{z}) = \frac{1}{\sqrt{(2\pi)^N |\Sigma|}} \exp\left(-\frac{1}{2}(\mathbf{z}-\mu)^T \Sigma^{-1}(\mathbf{z}-\mu)\right),$$

where $\mathbf{z} = (\mathbf{f}, \mathbf{x}_\mathcal{V})$, and $N = |\mathcal{V}| + |\mathcal{F}|$ is the total number of variables. These parameters can be estimated from the training data using maximum likelihood estimation.

Using this model, similarly as in Chapter 6, we can compute the conditional entropies

$$H(\mathcal{F}) = \frac{1}{2}\log_2(2\pi e)^N |\Sigma_\mathcal{F}|, \tag{7.5}$$

$$H(\mathcal{F} \mid \mathcal{X}_\mathcal{A}) = \frac{1}{2}\log_2(2\pi e)^N |\Sigma_{\mathcal{F}|\mathcal{A}}|. \tag{7.6}$$

Hence, these conditional entropies can be efficiently computed in closed form. Using Proposition 7.5, the greedy algorithm, given a set of sensors $\mathcal{A}$ already selected, should pick the sensor s which maximizes $H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A}) - H(\mathcal{X}_s \mid \mathcal{F}, \mathcal{X}_\mathcal{A})$, i.e., the algorithm prefers to add sensors which are as different as possible from the sensors already selected ($H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A})$ is large), but which are also relevant for predicting $\mathcal{F}$ (i.e., $H(\mathcal{X}_s \mid \mathcal{F}, \mathcal{X}_\mathcal{A})$ is small, e.g., if $\mathcal{X}_s$ is strongly correlated with $\mathcal{F}$).

Multi-resolution Sensor Placement

An alternative approach to selecting all pressure sensors from the same granularity is employing different types of sensors. We might, for instance, want small point sensors for detecting the boundary of the sitting region, but wider, flat sensors reporting the average pressure in a certain region. In order to allow this extension, we augment the set of variables $\mathcal{X}_\mathcal{V}$. Now, $\mathcal{V}$ is the union of two (or more) parts, $\mathcal{V}_1, \ldots, \mathcal{V}_r$, where each subset corresponds to sensors of a different resolution. Given the sensor values at the finest resolution (using, e.g., the Tekscan sensor), we can downsample the training data (i.e. Figure 7.9) to get coarser resolutions, i.e., by combining regions of 2×2, 4×4, etc. sensors, and averaging their values. Using this procedure, we can augment the training data set to simulate the values of coarser sensors. We then learn a joint distribution $P(\mathcal{F}, \mathcal{X}_\mathcal{V})$ on this new augmented set of variables, and the sensor placement algorithm as described in the previous section will automatically determine which resolution to use.

Furthermore, as discussed before, we can also allow to have different costs c_i for the different sensor types $\mathcal{V}_i$. Using the CELF algorithm described in Chapter 5, we can select near-optimal placements $\mathcal{A}$ which maximize the information gain $U(\mathcal{F}; \mathcal{X}_\mathcal{A})$ subject to a budget, a constraint on the total cost of the placement $\mathcal{A}$.

7.7.2 Experiments

Evaluation Using Tekscan Sensor Data

We performed our first set of experiments on the data set made available to us by Zhu et al. [2003] and Tan et al. [2001], thereby making our results comparable to these existing approaches. The data set contains pressure data for ten postures, collected from 26 male and 26 female participants. The postures

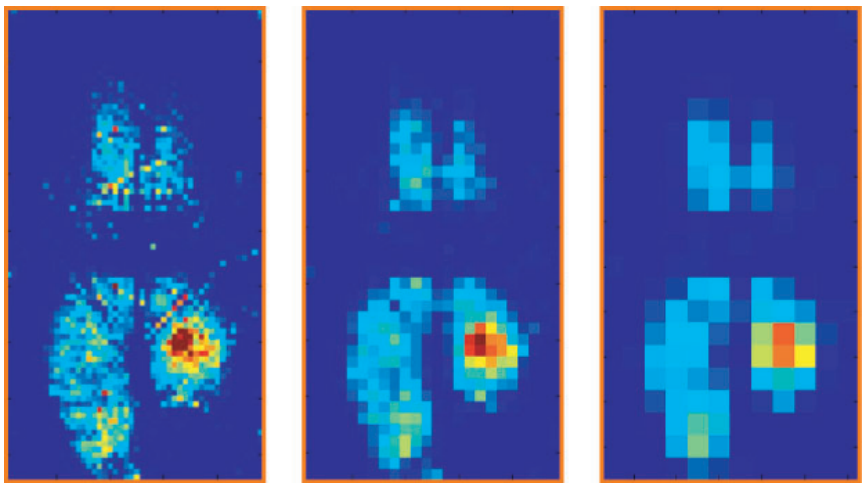


Figure 7.9: An example posture instance in multiple resolutions. The leftmost image is the raw pressure readings. In the center is data points aggregated into 2×2 regions. The image in the right shows data points combined into 4×4 regions.

included are *(1) Left leg crossed, (2) Right leg crossed, leaning left, (3) Leaning back, (4) Leaning forward, (5) Leaning left, (6) Leaning right, (7) Left leg crossed, leaning right, (8) Seated upright, (9) Right leg crossed, (10) Slouching*. From each participant and posture, five examples were selected. Data was collected using two sheets of the Tekscan sensor, each providing 42×48 sensing elements, at a 10mm distance. The elements provide a 8-bit resolution for sensing pressure.

We used ten-fold cross-validation in our experiments that repeatedly selected a gender-balanced random subset of 23 male and 23 female participants for training, and evaluated our classifier on the remaining six participants (three male, three female). The reported results are average scores with respect to ten such random splits. This split enables us to assess the generalization capabilities *between* participants, which we refer to as the *multi-user* setting, using the terminology of Tan et al. [2001]. The best performance reported by previous work in this setting was 79% [Tan et al., 2001].

We implemented the algorithms for computing the feature values in MATLAB, and used the *SimpleLogistic* classifier implementation from the Weka library [Witten and Frank, 2005]. We also implemented the greedy algorithm for sensor placement in MATLAB. To use this algorithm, we trained a multivariate normal model on the sensor values and the computed features on the training data using maximum likelihood estimation.

Feature Selection. Our first experiment aimed at prioritizing the features on our feature set using rankings created by an SVM-based feature selection algorithm [Guyon et al., 2002] and identifying a subset of these features that sufficiently produce our target classification performance. To identify this subset, we conducted a ten-fold cross-validation experiment, which provided us with information on how the classification accuracy produced by our algorithm changed as features were incrementally included in the learning model, without overfitting to a particular test set. Figure 7.10 presents results on this feature selection experiment. We can see that the classification accuracy quickly increases. As the number of features reaches ten, the classification performance levels off and a maximum is reached at 30 features. Based on this analysis, we decided to use the first 30 features to train our classifier. Using this number,

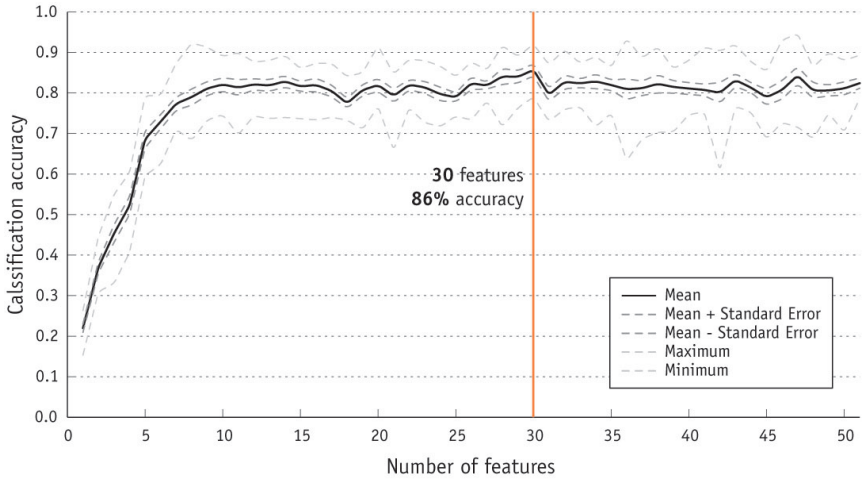


Figure 7.10: Ten-fold cross-validation results for change in classification accuracy as more features are added. The sequence of adding these features followed a ranking created by an SVM-based feature selection algorithm [Guyon et al., 2002]. Vertical line indicates the cut-off point we chose.

our classifier achieved an average (cross-validated) classification accuracy of 86%, compared to the trivial baseline of 10% if we guess the posture at random.

The top ten selected features, achieving an accuracy of 82%, included pressure readings from five aggregated pressure areas from the bottom of the seat, three of the variables that define the location of the pressure area at the back of the seat, and the angle and distance between the centers of the top and bottom pressure areas.

class	1	2	3	4	5	6	7	8	9	10
1	**235**	0	1	0	0	0	18	3	2	1
2	0	**240**	0	0	5	0	0	0	15	0
3	0	0	**187**	5	1	2	0	28	0	37
4	0	0	3	**244**	1	0	0	10	0	2
5	1	5	2	7	**241**	0	0	4	0	0
6	0	0	0	16	0	**222**	5	14	3	0
7	26	0	0	0	0	3	**231**	0	0	0
8	2	0	24	28	5	3	0	**196**	1	1
9	0	5	0	2	1	2	0	0	**250**	0
10	0	0	32	3	1	0	0	3	0	**221**

Figure 7.11: Confusion matrix. Rows indicate true classes, columns refer to assigned labels. Class labels represent *(1) Left leg crossed, (2) Right leg crossed, leaning left, (3) Leaning back, (4) Leaning forward, (5) Leaning left, (6) Leaning right, (7) Left leg crossed, leaning right, (8) Seated upright, (9) Right leg crossed, (10) Slouching*

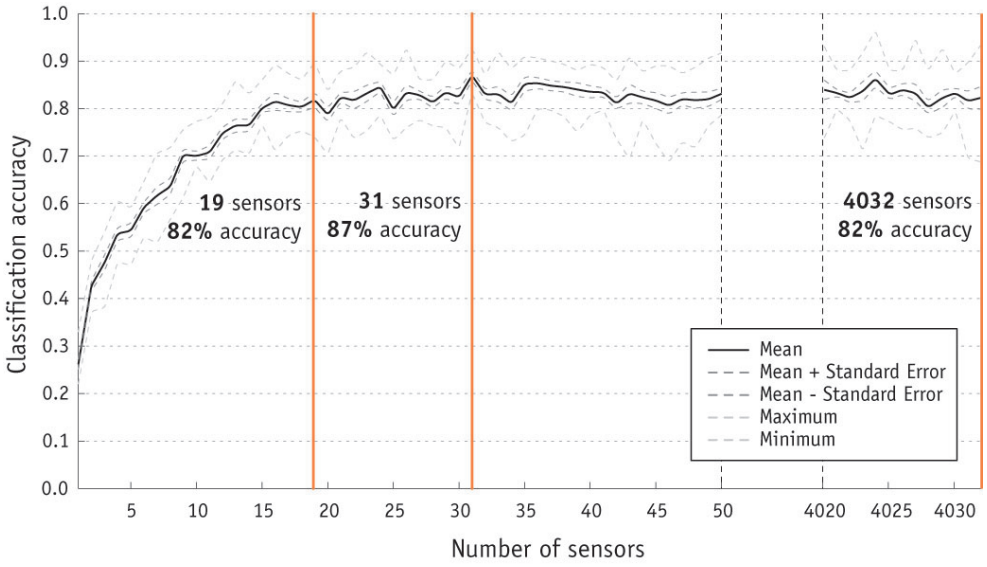


Figure 7.12: Ten-fold cross-validation results for the change in accuracy as the number of sensors increase. Best results were obtained with 31 sensors (87% accuracy), while we deployed 19 sensors, which produce the same accuracy (82%) as 4032 sensors do.

Sensor Placement. After feature selection, we conducted experiments on selecting the sufficient number of sensors and the near-optimal placement of these sensors to achieve the targeted classification performance. We conducted a second Ten-fold cross-validation experiment that looked at how classification accuracy changed as data from each sensor in our dataset were incrementally included in the learner. Our algorithm selected sensors at multiple resolutions of discretization, as described in the previous section. Figure 7.12 presents how the (ten-fold cross-validated) classification accuracy increases as more and more sensors are selected. However, the performance levels off quickly – even when placing 19 sensors, our classifier achieves an accuracy of 82% (equal to the accuracy obtained using all 4032 sensors). The best classification accuracy we obtained was **87%** using 31 sensors, which we believe is higher than existing work, while using only less than 1% of the sensors used in these work.

We limited our final system to use 19 sensors, to be near-optimally placed on the bottom and back of the chair. Interestingly, all of the 19 sensors were chosen at the same level of discretization (4x4), aggregating pressure values from 16 sensors. Based on this result, we limited our hardware design to only sensors of this granularity. Figure 7.13 shows the near-optimal placement of these 19 sensors overlaid on sample aggregated data as well as on our final deployed system.

We also compared the effectiveness of our near-optimal placement with other possible placements. Therefore, another ten-fold cross-validation experiment compared classification performance of a *near-optimally* placed subset of sensors against *random* and *uniform* placements of the same number of sensors. Additionally, we compared the performance of our algorithm with a potentially more direct sensor placement approach which directly optimizes the information gain with respect to the class variable (instead of the feature values, as our approach does). Figure 7.14 displays the results of this experiment. Our near-optimal placement significantly outperforms the uniform and random placements in classification accuracy[3]. As

[3]As the reader might have noticed, the cross-validation accuracies reported in different experiments are slightly different.

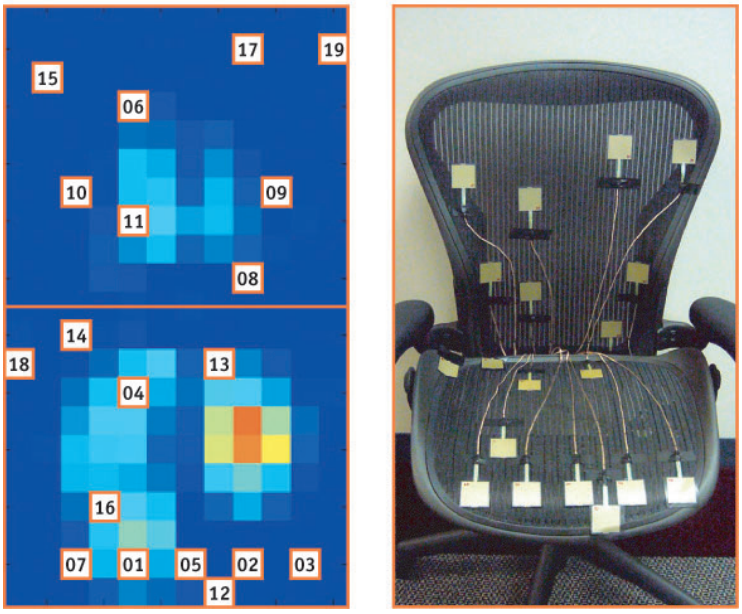


Figure 7.13: Final near-optimal placement of the sensors overlaid on an illustration of the downsampled data. The printed numbers on the sensor locations indicate ordering of sensors from the best predictor to the least. On the right shows our deployment of this near-optimal placement.

seen in Figure 7.14, it also performs better than the alternative, more direct placement strategy, again underlining the expressiveness of our chosen features.

Posture-Specific Performance. Most misclassified labels were (3), *Leaning back*, with true positive and negative rates of 75% and 72% respectively, and (8), *Seated upright*, with rates 76% and 75%. This result might not be surprising, since these postures are similar to several other postures in the data set, and there is high variance across subjects. Figure 7.11 shows the confusion matrix of the postures. We can see that postures (3), *Leaning back*, and (10), *Slouching*, were the ones which were most frequently confused, which is also very intuitive due to the similarity of these postures.

We also looked at which features were selected (i.e., given non-zero coefficients in the posture specific weight vectors $\mathbf{w}_l$) for the individual postures by the SimpleLogistic classifier. These weight vectors are very sparse – on average, every posture depends only on approximately three out of the 30 features. Inspection of these weights also gives valuable insight: For example, the two postures, (2) and (5), that included *Leaning left* are positively correlated with the horizontal bounding box location at the back, whereas the postures (6) and (7) that involved *Leaning right* are negatively correlated with this feature.

Reconstruction of $\mathcal{X}_\mathcal{V}$. We can also use the joint distribution $P(\mathcal{F}, \mathcal{X}_\mathcal{V})$ to reconstruct the unobserved sensor locations by estimating $P(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$. Hereby, $\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}$ are the values of all the locations $\mathcal{V} \setminus \mathcal{A}$ where no sensors were placed, and $\mathbf{x}_\mathcal{A}$ are the measurements made by our sensors at locations $\mathcal{A}$.

This is due to different random splits done for cross-validation in different experiments.

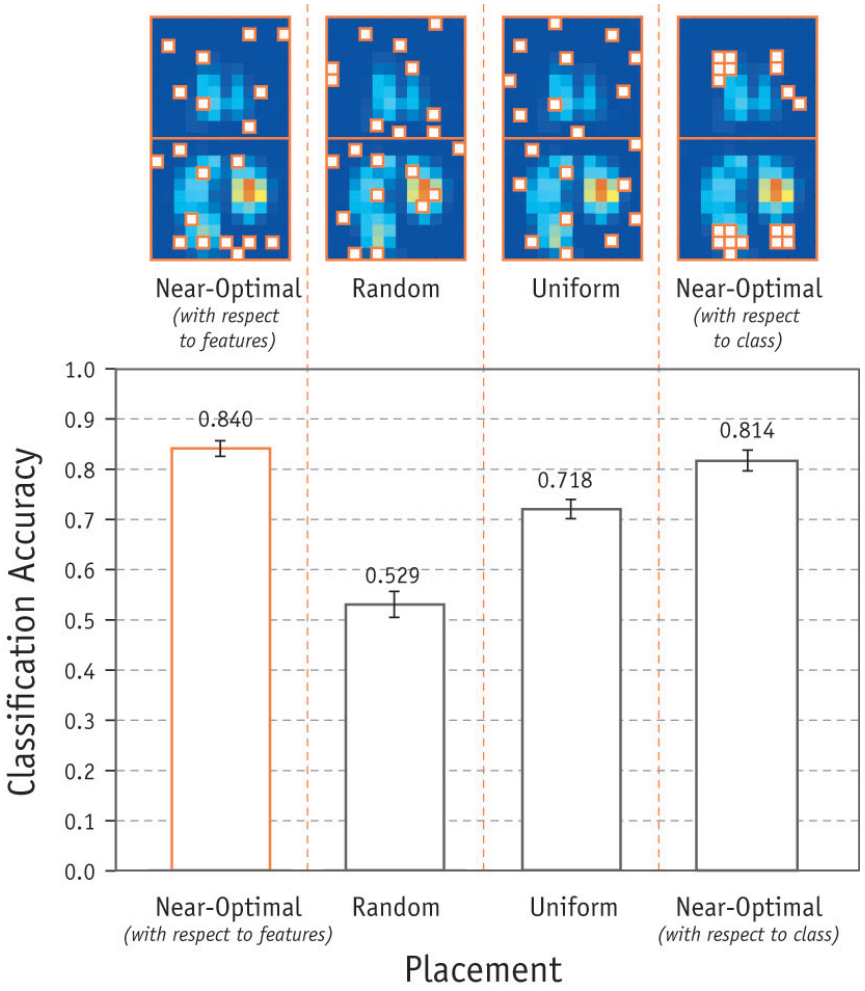


Figure 7.14: Cross-validation results that compare uniform, random, and near-optimal (with respect to class and features) placements of sensors on the chair surface. Images on the top row illustrate actual placements computed by these methods.

Figure 7.15 presents examples of these reconstructed pressure maps. Images in the top row display the original, high resolution pressure maps acquired from the Tekscan sensor, while the ones in the bottom row show the reconstructed data using only the measurements of 19 near-optimally chosen sensors. We can tell through visual inspection that the reconstructed pressure maps are rather accurate, which indicates that the selected sensor locations capture the most important characteristics of the different postures.

We also performed an experiment, where we reconstructed the pressure maps as described above, then *deterministically* computed the features on the full resolution maps, and used the computed features for classification. This approach however led to far worse classification accuracy of 54%, as opposed to 84% when probabilistically reconstructing the features directly.

Learning Algorithm Comparison. A final evaluation of our method aimed at comparing different learning algorithms. Therefore, we trained *SimpleLogistic*, *Naive Bayes*, *Artificial Neural Network* (Multi-

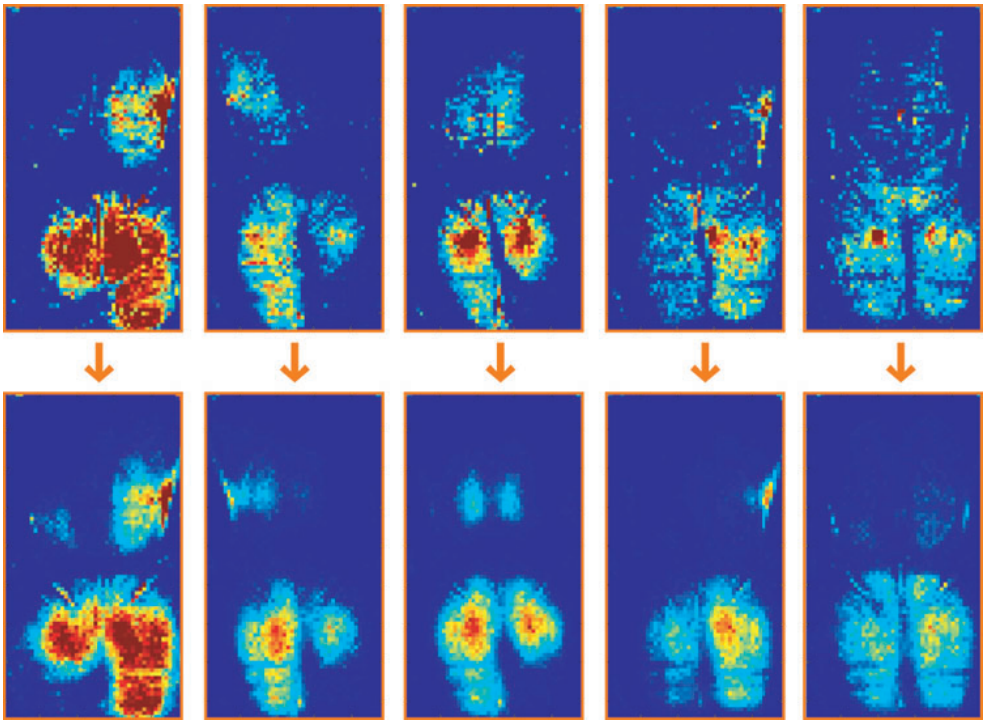


Figure 7.15: Reconstructed posture data examples on 6 postures. Top of each pair: original sensor values (4032 sensors); right image: reconstruction using only selected 19 sensors.

Layer Perceptron), and *Support Vector Machine* classifiers using our final feature set calculated by data from our near-optimally placed 19 sensors. Figure 7.16 presents the results of this experiment. *SimpleLogistic* outperformed all other algorithms in this experiment.

Evaluation of Our Prototype Sensor System

Based on the results described in the previous section, we deployed a proof-of-concept prototype sensor system, as illustrated in Figure 7.4, and installed it on an office chair. We chose the same chair (the Aeron chair produced by Herman Miller) as used in the original data collection by Zhu et al. [2003] and Tan et al. [2001]. The final system included 19 one-and-a-half-inch-square sensors near-optimally placed on the seat and back of the chair. We used FSR (Force Sensing Resistors) sensors produced by Interlink Electronics. Sensors were connected to a data acquisition board that read sensor values with the desired frequency and sent to a desktop computer via a USB connection. We implemented a Java application for real-time data acquisition, processing, and classification.

We used our deployed sensor system to evaluate the performance of our classifier in a real-time evaluation setting. We hired 20 naive subjects (ten male, ten female, college students aged between 19 and 34) to sit in the ten postures we used for training our classifier in ten trials (that produced 100 postures per participant). The order that the postured appeared in each trial was randomized to avoid learning or discomfort effects.

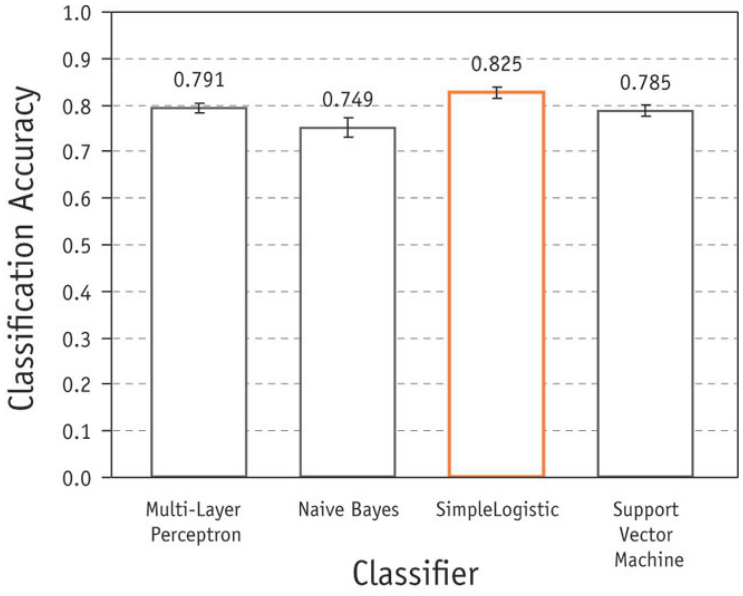


Figure 7.16: Ten-fold cross-validation results for classification accuracy with 4 classifiers (*SimpleLogistic*, *Naive Bayes*, *Support Vector Machine*, and *Multi-Layer Perceptron*) as well as Information Gain with respect to the target class variable.

Classification Accuracy. We analyzed the data collected from this deployment in several ways. In order to use the classifier trained on the high resolution data set, we need to calibrate the deployed sensors. We calibrated the sensors by performing linear regression, finding slope and bias for each deployed sensor to minimize the error in predicting the corresponding values from the high resolution sensor. Since we did not have data from the same subjects on both the deployed sensors and the Tekscan sensor, we performed this regression analysis on the posture specific means, as presented in Figure 7.17. Due to the low-fidelity nature of the deployed sensors, their different signal response compared to the Tekscan sensor, and the variance in the subjects' postures, this calibration was very difficult. In fact, when we used the calibrated data from the sensors, and classified it using the classifier trained on the high resolution data, our accuracy was only 63%.

Not satisfied with this result, we *re-trained* the classifier on the data from the deployed sensor only. We conducted a ten-fold, gender-balanced, cross-validation experiment. In each cross-validation split, we trained a *SimpleLogistic* classifier on randomly selected nine male and nine female participants and tested on the remaining two participants. We used the 19 measured values directly as features. Our classifier achieved a (ten-fold cross-validated) classification accuracy of **78%**. Considering the fact that the chosen sensors have quite different and lower-fidelity responses than the high-cost Tekscan sensor, as well as the fact that we trained the classifier on only 18 (instead of 46) subjects, this classification accuracy is very promising, and shows that the proposed sensor placement technique can lead to high quality sensor placements at a small fraction of the cost of existing approaches.

Realtime Performance. The second experiment was an explorative evaluation of the real-time performance of our deployed system. We identified that our deployed posture recognition system can predict postures with a 100 ms delay (an aggregation of delays in sensor reading, processing, and classifica-

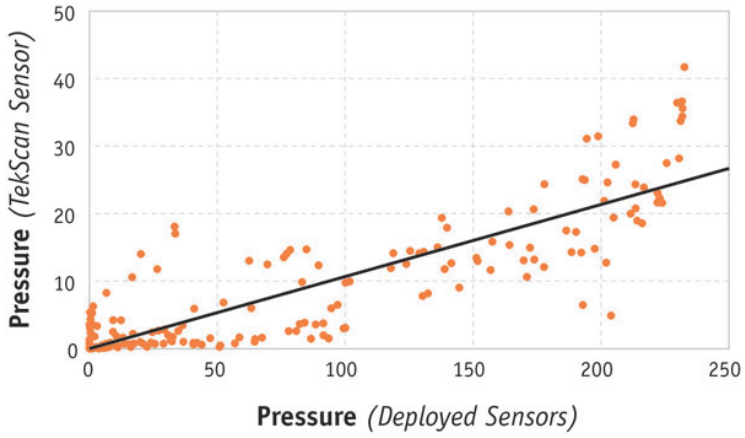


Figure 7.17: Regression analysis, comparing the of mean pressure values for each posture between the deployed sensors and the corresponding values of the Tekscan sensor.

tion), which we believe is a promising performance considering that no near-real-time systems on posture recognition is reported in the literature. We are also convinced, that if the classifier is implemented on a dedicated embedded device, far lower response times can be achieved.

Cost Analysis. Through deploying our sensor system, we were also able to get an estimate of the cost of the system. The total cost of our prototype deployment was approximately 300 USD in components, compared to the above 16,000 USD when using the high-resolution sensor. The cost of only the sensors in our system is 100 USD compared to the Tekscan sensors commercially available for 3,000 USD. We argue that having developed our low-cost sensor system will allow us to deploy the technology in different settings. Collecting data from a larger number of people and in a variety of seating contexts would provide us with a deeper and more precise understanding of posture data and seated activities. In the next section, we discuss possible applications of such understanding.

7.7.3 Discussion

We argue that our low-cost posture detection technology has a number of application domains. In the following few paragraphs, we provide related work on these application domains, which provide context for our work. We then identify four domains where posture recognition could play a vital role and describe rough usage scenarios for our technology. These domains include Home, Office, Automotive, and Mobile Assistive Technology. We provide four simple scenarios that illustrate how our system could be used to provide users with assistance and support in their daily activities.

Related Work

Seat Comfort. Seat comfort has gained particular attention for military, workplace, and assistive technology applications. Research has shown that uncomfortable aircrew seating can significantly affect attention, information processing, and task performance while improved comfort with more evenly distributed pressure patterns can improve aircrew task performance [Cohen, 1998]. Office ergonomics is another

context where seat comfort is extensively studied [Zhang, 1996]. Symptoms of seating discomfort in the office were found to be sore muscles, heavy legs, uneven pressure, stiffness, restlessness, fatigue, and pain [Halender and Zhang, 1997]. In assistive technology, seat discomfort is identified as a major problem among most individuals with disabilities [Bardsley, 1984, Monette et al., 1999, Shaw, 1992]. In a 1995 national survey in the United States, comfort is identified as one of the leading unmet needs of assistive technology users [Scherer, 1996].

A significant body of work, such as those provided above, acknowledges the importance of studying seating comfort or avoiding or treating seating-related injuries. However, technology development in providing seating comfort is still limited to a few context-specific examples. Recently, pressure sensing has been used as an important tool for studying seating comfort [Brienza et al., 1996]. Researchers have also proposed interventions to avoid negative effects of uncomfortable and long-term seating such as decubitus ulcers [Cooper et al., 1997] or the inability to use standard wheelchair seats by certain populations such as children with neuro-motor impairments [Hobson, 1972].

The literature on seating comfort indicate that technology development to understand and support seating is fairly nascent. We hope that our work will contribute to a systematic understanding of seating and the kinds of technologies that that could support seated activities. Low-cost sensing and recognition systems could be deployed in different seating contexts. Literature on ubiquitous computing and multi-modal user interfaces provide examples of these contexts.

Technology Development. The ubiquity of seated activities in the home, workplace, and on the go makes seating also interesting as a human-computer interface. An example is the Sensing Chair, a chair that uses high-fidelity pressure readings from the surface of the chair to detect sitter's postures, developed by Tan et al. [2001], Z. et al. [1997], Zhu et al. [2003]. Seat pressure data is also used in affective computing applications. Kapoor and others used readings of pressure applied on a seat along with other multi-modal inputs to recognize affective states of a student in order to facilitate learning [Kapoor et al., 2001].

This work cannot exist without at least rudimentary analysis of sitting activity and its long-term effects on people along with technologies for automatic processing and recognition of seated activities were also developed. Real-time processing and recognition systems, such as those described by Tan et al. [2001], can support other processes such as the automatic control of airbag deployment based on the size, weight, and sitting posture of a driver or passenger, which is also part of the main inspiration of our work. However, technologies such as those used by Tan et al. [2001], Z. et al. [1997], Zhu et al. [2003] are far from being inexpensive and robust enough to be deployed in real-world application. We argue that our proposed system overcomes this problem and provides a robust, inexpensive system that provides near-real-time processing and prediction of seated postures.

Application Areas

The work cited above provides examples of the kinds of contexts where seating-related technology could support people's daily activities. However, we would like to provide several possible contexts for the use of such technology that originally motivated our work.

Home. In the home, an intelligent chair could identify postures over a pattern and adjust itself to support the user's postural needs. For instance, if the user is falling asleep on the chair, the chair could identify the change in user's postures over a window of time, infer that the user wants to sleep, and automatically adjust the backrest and footrest to support user's actions. DiSalvo et al. [2005] discuss the design of an intelligent lounge chair for the home for a more detailed scenario.

Office. Fatigue due to sitting in unhealthy postures for extended periods of time is a significant and very common problem in the workplace. Identifying a sitter's postural patterns could help avoid this problem. Users could be made aware of how healthy their postural patterns are and be provided with recommendations as well as corrections to attain and maintain healthy posture.

Educational settings could also use posture recognition technology for similar purposes. Through identifying a pattern in a student's posture, an intelligent tutor could infer whether the student is interested in the topic.

Automotive. A car could use posture information over a window of time to infer its driver's level of drowsiness as well as attention. Such application might be more robust than existing vision-based technologies as, unlike camera data that is sensitive to lighting conditions and changes in driver's appearances, seating surface is predictable and consistent.

Assistive Technology. One of the application areas in the Assistive Technology domain is wheelchairs. Posture information could be used by an intelligent wheelchair to help avoiding seating ulcers that are caused by sitting in the same posture for extended periods of time. The chair could make minor adjustments on its surface or gently warn the user to change postures to avoid such problems.

7.8 Summary

We presented efficient randomized approximation algorithms for the long standing problem of nonmyopically selecting most informative subsets of variables in graphical models. Our algorithms provide a constant factor $(1 - 1/e - \varepsilon)$ performance guarantee with high confidence, both in the unit-cost and in the budgeted case. Our methods can also be used to derive online performance guarantees for other heuristics. The analysis of the algorithms leveraged the concept of submodular functions, and polynomial bounds on sample complexity. We also showed that $(1 - 1/e)$ is the best performance guarantee possible unless $\mathbf{P} = \mathbf{NP}$.

Our empirical results demonstrate the practical applicability of our method to real-world problems: relating the maximization of our submodular objective functions to improving prediction accuracy, and showing the superiority of the information gain criterion to entropy maximization.

In our detailed seating activity recognition case study, our approach achieves a classification accuracy of 87% in classifying ten postures on untrained subjects, a higher performance than that of previous work. Our system uses less than 1% of the deployed sensors compared to previous work, therefore drastically reducing hardware and computational cost. We used our methodology to build a prototype real-time sensing and recognition system using only 19 sensors. In a user study with 20 subjects, our low-cost prototype achieved 78% accuracy in the same classification task.

Chapter 8

Sensing for Outbreak Detection in Networks

In this chapter, we consider another important class of observation selection problems: In this class of problems, we are given a network and a dynamic process spreading over this network, and we want to select a set of nodes to detect the process as effectively as possible.

Many real-world problems can be modeled under this setting. Consider a city water distribution network, delivering water to households via pipes and junctions. Accidental or malicious intrusions can cause contaminants to spread over the network, and we want to select a few locations (pipe junctions) to install sensors, in order to detect these contaminations as quickly as possible. In August 2006, the *Battle of Water Sensor Networks (BWSN)* [Ostfeld et al., 2006] was organized by researchers from the civil engineering community as an international challenge to find the best sensor placements for a real (but anonymized) metropolitan area water distribution network. In this chapter, we discuss the approach we used in this competition. Typical epidemics scenarios also fit into this outbreak detection setting: We have a social network of interactions between people, and we want to select a small set of people to monitor, so that any disease outbreak can be detected early, when very few people are infected.

In the domain of weblogs (blogs), bloggers publish posts and use hyper-links to refer to other bloggers' posts and content on the web. Each post is time stamped, so we can observe the spread of information on the "blogosphere". In this setting, we want to select a set of blogs to read (or retrieve) which are most up to date, *i.e.*, catch (link to) most of the stories that propagate over the blogosphere. Figure 8.1 illustrates this setting. Each layer plots the propagation graph (also called *information cascade* [Bikhchandani et al., 1992]) of the information. Circles correspond to blog posts, and all posts at the same vertical column belong to the same blog. Edges indicate the temporal flow of information: the cascade starts at some post (*e.g.*, top-left circle of the top layer of Figure 8.1) and then the information propagates recursively by other posts linking to it. Our goal is to select a small set of blogs (two in case of Figure 8.1) which "catch" as many cascades (stories) as possible[1].A naive, intuitive solution would be to select the big, well-known blogs. However, these usually have a large number of posts, and are time-consuming to read. We show, that, perhaps counterintuitively, a more cost-effective solution can be obtained, by reading smaller, but higher quality, blogs, which our algorithm can find.

There are several possible criteria one may want to optimize in outbreak detection. For example, one

[1]In real-life multiple cascades can be on the same or similar story, but we still aim to detect as many as possible.

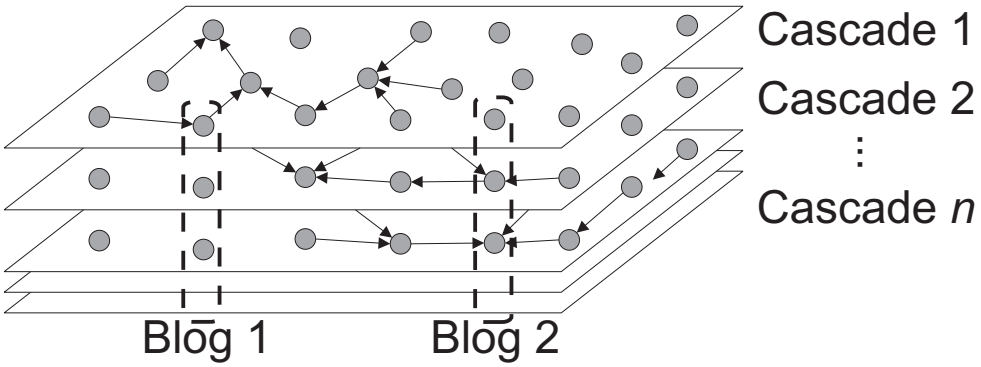


Figure 8.1: Spread of information between blogs. Each layer shows an information cascade, and all posts at the same vertical column belong to the same blog. Edges represent the flow of information. We want to pick a few blogs quickly capture most cascades.

criterion seeks to minimize *detection time* (*i.e.*, to know about a cascade as soon as possible, or avoid spreading of contaminated water). Similarly, another criterion seeks to minimize the *population affected* by an undetected outbreak (*i.e.*, the number of blogs referring to the story we just missed, or the population consuming the contamination we cannot detect). Optimizing these objective functions is $\mathbf{NP}$-hard, so in general, for large, real-world problems, we cannot expect to efficiently find the optimal solution.

In this chapter, we show, that these and many other realistic outbreak detection objectives are submodular (*c.f.*, Chapter 5). By exploiting this submodularity property, we can in principle use the machinery discussed in Chapter 5 to *efficiently obtain* solutions which are *provably close* to the optimal solution. These guarantees are important in practice, since selecting nodes is expensive (reading blogs is time-consuming, sensors have high cost), and we desire solutions which are not too far from the optimal solution.

However, the problems considered in this chapter are too large so that we cannot apply the algorithms discussed in Section 5 directly. Furthermore, in these outbreak detection problems, observations often have non-constant cost. For example, the time required to read a blog depends on the number of posts in the blog, etc. We provide an efficient algorithm, CELF, which scales to the problem sizes involved in the BWSN challenge, and the blog monitoring application.

The main contributions of this chapter are:

- We show that many objective functions for detecting outbreaks in networks are submodular, including detection time and population affected in the blogosphere and water distribution monitoring problems. We show that our approach also generalizes work by Kempe et al. [2003] on selecting nodes maximizing influence in a social network.
- We show that this submodularity observation helps to speed up the greedy algorithm by up to 700 times. We also derive novel online bounds on the quality of the placements obtained by *any* algorithm.
- We extensively evaluate our methodology on the applications introduced above – water quality and blogosphere monitoring. These are large real-world problems, involving a model of a water distribution network from the EPA with millions of contamination scenarios, and real blog data with millions of posts.

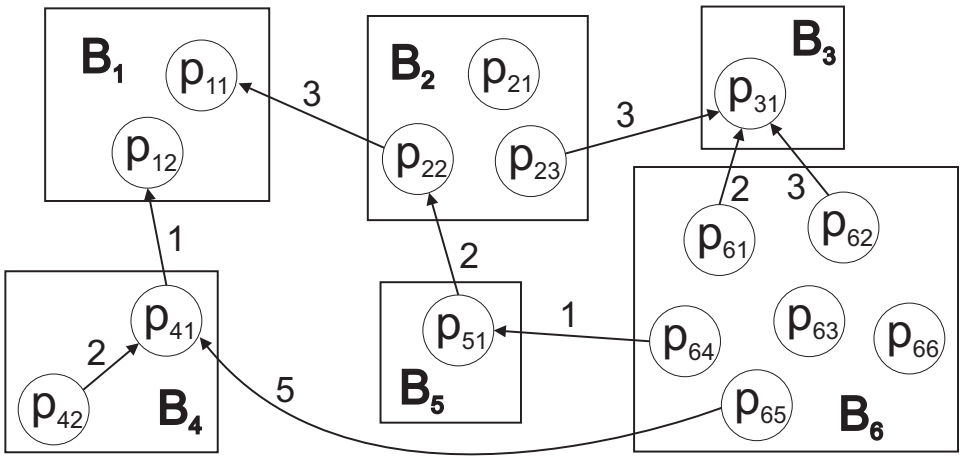


Figure 8.2: Blogs have posts, and there are time stamped links between the posts. The links point to the sources of information and the cascades grow (information spreads) in the reverse direction of the edges. Reading only blog B_6 captures all cascades, but late. B_6 also has many posts, so by reading B_1 and B_2 we detect cascades sooner.

- We show how the proposed methodology leads to deeper insights in both applications, including multicriterion, cost-sensitivity analyses and generalization questions.

8.1 The outbreak detection problem

8.1.1 Problem statement

The water distribution and blogosphere monitoring problems, even though in very different domains, share essential structure. In both problems, we want to select a subset $\mathcal{A}$ of nodes (sensor locations, blogs) in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which detect outbreaks (spreading of a virus/information) quickly.

Figure 8.2 presents an example of such a graph for blog network. Each of the six blogs consists of a set of posts. Connections between posts represent hyper-links, and labels show the time difference between the source and destination post, *e.g.*, post p_{41} linked p_{12} one day after p_{12} was published).

These outbreaks (*e.g.*, information cascades) initiate from a single node of the network (*e.g.*, p_{11}, p_{12} and p_{31}), and spread over the graph, such that the traversal of every edge $(s,t) \in \mathcal{E}$ takes a certain amount of time (indicated by the edge labels). As soon as the event reaches selected node, alarm is triggered. E.g., selecting blog B_6, would detect the cascades originating from post p_{11}, p_{12} and p_{31}, after 6, 6 and 2 timesteps after the start of the respective cascades.

We can formally model these outbreaks as a random process. With each node $s \in \mathcal{V}$ in the network and timestep $t \in \mathcal{T}$, we associate a nonnegative random variable $\mathcal{X}_{s,t}$, which is 0 if node s is not infected at time t, and positive otherwise. Hereby, $\mathcal{T} \subseteq \mathbb{R}$ is a discrete or continuous ordered set of time points. If $\mathcal{X}_{s,t} > 0$, then $\mathcal{X}_{s,t}$ denotes the degree of infection (e.g., the concentration of contaminated water). Hence, the joint realization $\mathbf{x}_{\mathcal{V},\mathcal{T}}$ describes the state of the network over the entire course of an outbreak. For each possible realization, we evaluate a penalty function $\pi(\mathbf{x}_{\mathcal{V}}, \mathcal{A})$ which depends on the state of the network

$\mathbf{x}_{\mathcal{V},\mathcal{T}}$ at all nodes and all times, as well as the placement $\mathcal{A}$, as follows. For each possible observation (sensor location) $s \in \mathcal{V}$, we use

$$T(\mathbf{x}_{\mathcal{V},\mathcal{T}}, s) = \min\{t \in \mathcal{T} : x_{s,t} > 0\}$$

to denote the *time of detection*. Hereby, $T(\mathbf{x}_{\mathcal{V},\mathcal{T}}, s) = \infty$ if s never detects the outbreak, i.e., $x_{s,t} = 0$ for all $t \in \mathcal{T}$. For a set of observations $\mathcal{A} \subseteq \mathcal{V}$, we define

$$T(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \mathcal{A}) = \min_{s \in \mathcal{A}} T(\mathbf{x}_{\mathcal{V},\mathcal{T}}, s),$$

i.e., the earliest time any sensor in $\mathcal{A}$ detects the outbreak. Furthermore, we define a *penalty function* $\pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, t)$[2], describing the penalty incurred if outbreak $\mathbf{x}_{\mathcal{V},\mathcal{T}}$ is detected at time t. As we describe below, this penalty models, e.g., the population affected by consuming contaminated water up to time t.

Based on this notation, we can define outbreak detection formally as an observation selection problem as discussed in Section 2.3. To this end, we can define a sensing quality function

$$u(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \mathcal{A}) = \pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \infty) - \pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, T(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \mathcal{A})).$$

If we assume a probability distribution $P(\mathbf{x}_{\mathcal{V},\mathcal{T}})$ over outbreaks, the corresponding expected sensing quality

$$F(\mathcal{A}) = \int P(\mathbf{x}_{\mathcal{V},\mathcal{T}}) u(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \mathcal{A}) d\mathbf{x}_{\mathcal{V},\mathcal{T}}$$

then models the expected penalty reduction obtained if observing subset $\mathcal{A} \subseteq \mathcal{V}$ of the nodes[3].

Depending on which nodes $\mathcal{A}$ we select, we achieve a certain placement score $F(\mathcal{A})$. Figure 8.2 illustrates several criteria one may want to optimize. If we only want to detect as many stories as possible, then reading just blog B_6 is best. However, reading B_1 would only miss one cascade (p_{31}), but would detect the other cascades immediately.

Since sensors are expensive, we also associate a *cost* $C(\mathcal{A})$ with every placement $\mathcal{A}$, and require, that this cost does not exceed a specified budget B which we can spend. For example, the cost of selecting a blog could be the number of posts in it (*i.e.*, B_1 has cost 2, while B_6 has cost 6). In the water distribution setting, accessing certain locations in the network might be more difficult (expensive) than other locations. Also, we could have several types of sensors to choose from, which vary in their quality (detection accuracy) and cost. In this chapter, we consider the case of *additive cost functions* (*c.f.*, Section 2.4), i.e., we associate a nonnegative cost $c(s)$ with every sensor s, and define the cost of placement $\mathcal{A}$: $C(\mathcal{A}) = \sum_{s \in \mathcal{A}} c(s)$.

Using this notion of sensing quality and cost, our goal is to solve the maximization problem (2.4):

$$\max_{\mathcal{A} \subseteq \mathcal{V}} F(\mathcal{A}) \text{ subject to } C(\mathcal{A}) \leq B, \tag{8.1}$$

where B is a budget we can spend for selecting the nodes.

[2]We require that π be nondecreasing in $\mathcal{T}$, i.e., we never prefer late over early detection. We also require π to be defined for $t = \infty$, describing the penalty incurred if the outbreak is never detected.

[3]Note, that we could equivalently allow to reason over the times of observing as well as over the observed nodes, i.e., select $\mathcal{A} \subseteq \mathcal{V} \times \mathcal{T}$

8.1.2 Placement objectives

Even though the water distribution and blogosphere monitoring problems are very different, similar placement objective scores make sense for both applications. The detection time $T(i, s)$ in the blog setting is the time difference in days, until blog s participates in the cascade i, which we extract from the data. In the water network, $T(i, s)$ is the time it takes for contaminated water to reach node s in scenario i (depending on outbreak location and time). In both applications we consider the following objective functions (penalty reductions):

(a) *Detection likelihood (DL)*: fraction of information cascades and contamination events detected by the selected nodes. Here, the penalty is $\pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, t) = 0$ for all $t < \infty$, and $\pi_i(\infty) = 1$, *i.e.*, we do not incur any penalty if we detect the outbreak in finite time, otherwise we incur penalty 1.

(b) *Detection time (DT)* measures the time passed from outbreak till detection by one of the selected nodes. Hence, $\pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, t) = \min\{t, T_{\max}\}$, where $T_{\max}$ is the time horizon we consider (end of simulation / data set).

(c) *Population affected (PA)* by scenario (cascade, outbreak). This criterion has different interpretations for both applications. In the blog setting, the affected population measures the number of blogs involved in a cascade before the detection. Here, $\pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, t)$ is the size of (number of blogs participating in) cascade $\mathbf{x}_{\mathcal{V},\mathcal{T}}$ at time $t < \infty$, and $\pi(\mathbf{x}_{\mathcal{V},\mathcal{T}}, \infty)$ is the size of the cascade at the end of the data set. In the water distribution application, the affected population is the expected number of people affected by not (or late) detecting a contamination event.

Note, that optimizing each of the objectives can lead to very different solutions, hence we may want to simultaneously optimize all objectives at once. We deal with this multicriterion optimization problem in Section 8.1.4.

8.1.3 Submodularity of the placement objectives

The penalty reduction function[4] $F(\mathcal{A})$ has several important and intuitive properties: Firstly, $F(\emptyset) = 0$, *i.e.*, we do not reduce the penalty if we do not place any sensors. Secondly, F is nondecreasing, *i.e.*, $F(\mathcal{A}) \leq F(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$. Hence, adding sensors can only decrease the incurred penalty. Thirdly, and most importantly, it is submodular (*c.f.*, Chapter 5):

Theorem 8.1. *For all placements $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and sensors $s \in \mathcal{V} \setminus \mathcal{B}$, it holds that*

$$F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{s\}) - F(\mathcal{B}).$$

Hence, if we add a sensor to a small placement $\mathcal{A}$, we improve our score at least as much, as if we add it to a larger placement $\mathcal{B} \supseteq \mathcal{A}$.

Hence, both the blogosphere and water distribution monitoring problems can be reduced to the problem of maximizing a nondecreasing submodular function, subject to a constraint on the budget we can spend for selecting nodes. More generally, any objective function that can be viewed as an expected penalty reduction is submodular. Thus, in principle, we can exploit the machinery for maximizing nondecreasing

[4]The objective F is similar to one of the examples of submodular functions described by [Nemhauser et al., 1978]. Our objective, however, preserves additional problem structure (sparsity) which we exploit in our implementation, and which we crucially depend on to solve large problem instances.

submodular functions as introduced in Chapter 5. For example, it follows from Theorem 5.2 that the greedy algorithm applied to the penalty reduction objective $F(\mathcal{A})$ is guaranteed to obtain a solution to the maximization problem (2.4) which achieves at least a constant fraction of $(1 - 1/e)$ of the value obtained by the optimal solution. We use the CELF algorithm for finding cost-effective solutions.

8.1.4 Multicriterion optimization

In practical applications, such as the blogosphere and water distribution monitoring, we may want to *simultaneously* optimize multiple objectives. Then, each placement has a vector of scores, $F(\mathcal{A}) = (F_1(\mathcal{A}), \ldots, F_m(\mathcal{A}))$. Here, the situation can arise that two placements $\mathcal{A}_1$ and $\mathcal{A}_2$ are *incomparable*, *e.g.*, $F_1(\mathcal{A}_1) > F_1(\mathcal{A}_2)$, but $F_2(\mathcal{A}_1) < F_2(\mathcal{A}_2)$. So all we can hope for are *Pareto-optimal solutions* [Boyd and Vandenberghe, 2004]. A placement $\mathcal{A}$ is called Pareto-optimal, if there does not exist another placement $\mathcal{A}'$ such that $F_i(\mathcal{A}') \geq F_i(\mathcal{A})$ for all i, and $F_j(\mathcal{A}') > F_j(\mathcal{A})$ for some j (*i.e.*, there is no placement $\mathcal{A}'$ which is at least as good as $\mathcal{A}$ in all objectives F_i, and strictly better in at least one objective F_j).

One common approach for finding such Pareto-optimal solutions is *scalarization* (*c.f.*, [Boyd and Vandenberghe, 2004]). Here, one picks positive weights $\lambda_1 > 0, \ldots, \lambda_m > 0$, and optimizes the objective $F(\mathcal{A}) = \sum_i \lambda_i F_i(\mathcal{A})$. Any solution maximizing $F(\mathcal{A})$ is *guaranteed* to be Pareto-optimal [Boyd and Vandenberghe, 2004], and by varying the weights λ_i, different Pareto-optimal solutions can be obtained. One might be concerned that, even if optimizing the individual objectives F_i is easy (*i.e.*, can be approximated well), optimizing the sum $F = \sum_i \lambda_i F_i$ might be hard. However, submodularity is closed under nonnegative linear combinations and thus the new scalarized objective is submodular as well.

8.1.5 Speeding up function evaluations

Evaluating the penalty reductions F can be very expensive. For example, in the water distribution application, we need to run physical simulations, in order to estimate the effect of a contamination at a certain node. In the blog networks, we need to consider several millions of posts, which make up the cascades. However, in both applications, most outbreaks are sparse, *i.e.*, affect only a small area of the network [*c.f.*, Krause et al., 2008a, Leskovec et al., 2007a], and hence are only detected by a small number of nodes. Hence, most nodes s do not reduce the penalty incurred by an outbreak (*i.e.*, $F_i(\{s\}) = 0$). Note, that this sparsity is *only* present if we consider penalty *reductions*. If for each sensor $s \in \mathcal{V}$ and scenario $i \in \mathcal{I}$ we store the actual penalty $\pi_i(s)$, the resulting representation is not sparse. Our implementation exploits this sparsity by representing the penalty function F as an *inverted index*[5], which allows fast lookup of the penalty reductions *by sensor index* s. By looking up all scenarios detected by all sensors in our placement $\mathcal{A}$, we can quickly compute the penalty reduction

$$F(\mathcal{A}) = \sum_{i: i \text{ detected by } \mathcal{A}} P(i) \max_{s \in \mathcal{A}} F_i(\{s\}),$$

without having to scan the entire data set.

The inverted index is the main data structure we use in our optimization algorithms. After the problem (water distribution network simulations, blog cascades) has been compressed into this structure, we use the same implementation for optimizing sensor placements and computing bounds.

[5]The index is inverted, since the data set facilitates the lookup *by scenario index* i (since we need to consider cascades, or contamination simulations for each scenario).

In the water distribution network application for example, exploiting this sparsity allows us to fit the set of all possible intrusions considered in the BWSN challenge – a footprint of 47 Terabytes – in main memory (16 GB), which leads to several orders of magnitude improvements in the running time, since we can avoid hard-drive accesses.

8.2 Case study: Water networks

8.2.1 Experimental setup

In the water distribution system application, we used the data and rules introduced by the Battle of Water Sensor Networks (BWSN) challenge [Ostfeld et al., 2006]. We considered both the small network on 129 nodes (BWSN1), and a large, realistic, 12,527 node distribution network (BWSN2) provided as part of the BWSN challenge. In addition we also consider a third water distribution network (NW3) of a large metropolitan area in the United States. The network (not including the household level) contains 21,000 nodes and 25,000 pipes (edges). To our knowledge, this is the largest water distribution network considered for sensor placement optimization so far. The networks consist of a static description (junctions and pipes) and dynamic parameters (time-varying water consumption demand patterns at different nodes, opening and closing of valves, pumps, tanks, etc.)

8.2.2 Objective functions

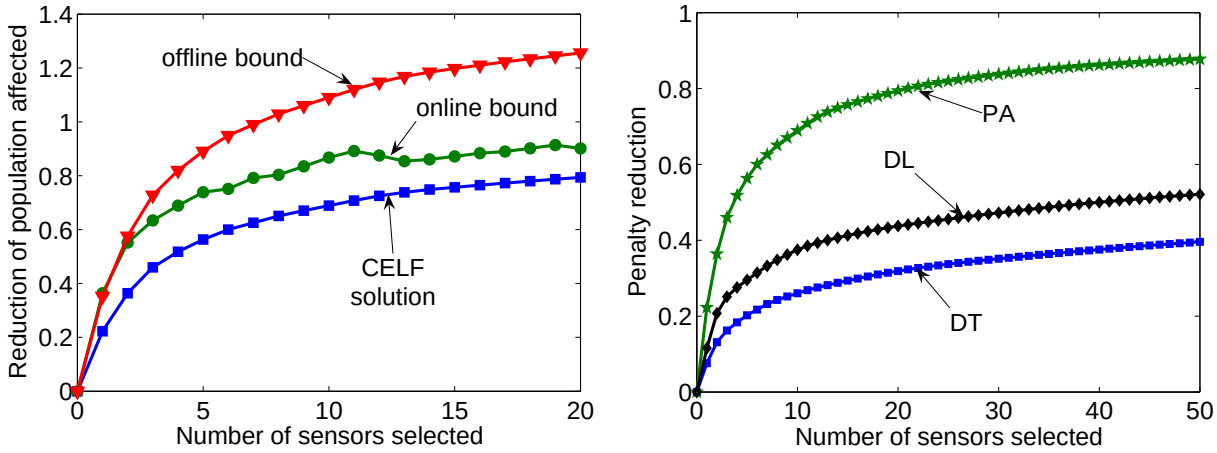
In the BWSN challenge, we want to select a set of 20 sensors, simultaneously optimizing the objective functions DT, PA and DL, as introduced in Section 8.1.2. To obtain cascades we use a realistic disease model defined by [Ostfeld et al., 2006], which depends on the demands and the contaminant concentration at each node. In order to evaluate these objectives, we use the EPANET simulator [Rossman, 1999], which is based on a physical model to provide realistic predictions on the detection time and concentration of contaminant for any possible contamination event. We consider simulations of 48 hours length, with 5 minute simulation timesteps. Contaminations can happen at any node and any time within the first 24 hours, and spread through the network according to the EPANET simulation. The time of the outbreak is important, since water consumption varies over the day and the contamination spreads at different rates depending on the time of the day. Altogether, we consider a set of 3.6 million possible contamination scenarios and each of these is associated with a "cascade" of contaminant spreading over the network.

8.2.3 Solution quality

We first used CELF to optimize placements of increasing size, according to the three criteria DL, DT, PA. We again normalized the scores to be between 0 and 1, where 1 is the best achievable score when placing sensors at every node.

Figure 8.3 (a) presents the CELF score, the off-line and on-line bounds for PA objective on the BWSN2 network. Consistently with the blog experiments, the on-line bound is much tighter than the off-line bound, and the solutions obtained by our CELF algorithm are very close to the optimum.

Figure 8.3 (b) shows CELF's performance on all 3 objective functions. Similarly to the blog data, the population affected (PA) score increases very quickly. The reason is that most contamination events only



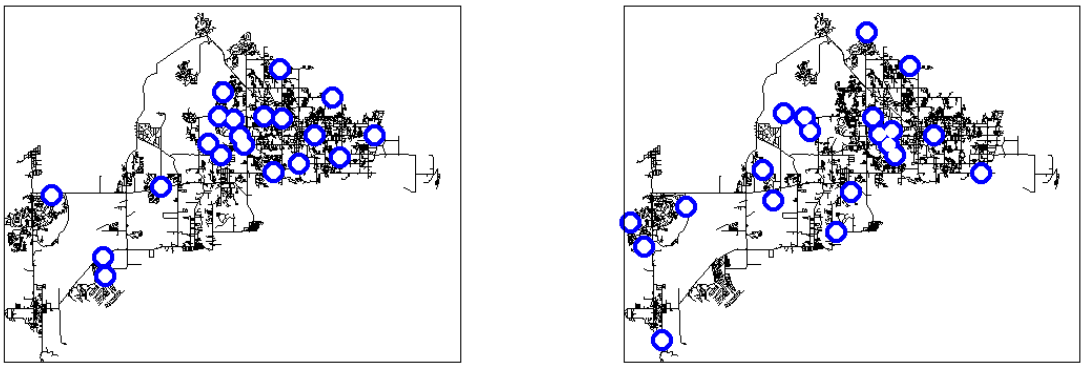
(a) Performance of CELF (b) Objective functions

Figure 8.3: (a) CELF with offline and online bounds for PA objective. (b) Different objective functions.

impact a small fraction of the network. Using few sensors, it is relatively easy to detect most of the high impact outbreaks. However, if we want to detect all scenarios, we need to place a large number of sensors (2,263 in our experiment). Hence, the DL (and correspondingly DT) increase more slowly than PA.

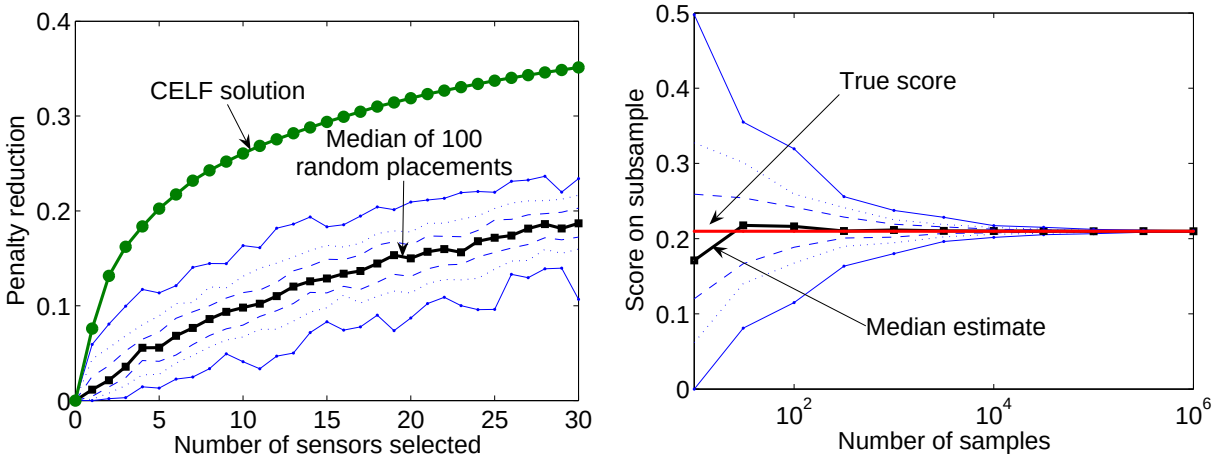
Figure 8.4 shows two 20 sensor placements after optimizing DL and PA respectively on BWSN2. When optimizing the population affected (PA), the placed sensors are concentrated in the dense high-population areas, since the goal is to detect outbreaks which affect the population the most. When optimizing the detection likelihood, the sensors are uniformly spread out over the network. Intuitively this makes sense, since according to BWSN challenge [Ostfeld et al., 2006], outbreaks happen with same probability at every node. So, for DL, the placed sensors should be as close to all nodes as possible.

We also compared the scores achieved by CELF with several heuristic sensor placement techniques, where we order nodes by some "goodness" criteria, and then pick top nodes. We consider the following criteria: population at the node, water flow through the node, and the diameter and the number of pipes at the node. Figure 8.7(a) shows the results for PA objective function. CELF outperforms best heuristic for 45%. Best heuristics are placing nodes at random, by degree or their population. We see heuristics perform poorly, since nodes which are close in the graph tend to have similar flow, diameter and population, and hence the sensors will be spread out too little. Even the maximum over one hundred random trials performs far worse than CELF. Figure 8.5(a) shows the statistics of choosing 100 random random placements on the water distribution network for the PA objective function. Notice that even best out of 100 random trials performs far worse than CELF. Figure 8.5(b) shows how many outbreaks one needs so that the score approaches the true score that one obtains if data on all outbreaks is available. Notice that estimates soon converge to true score and data on less than 100,000 outbreaks is needed.



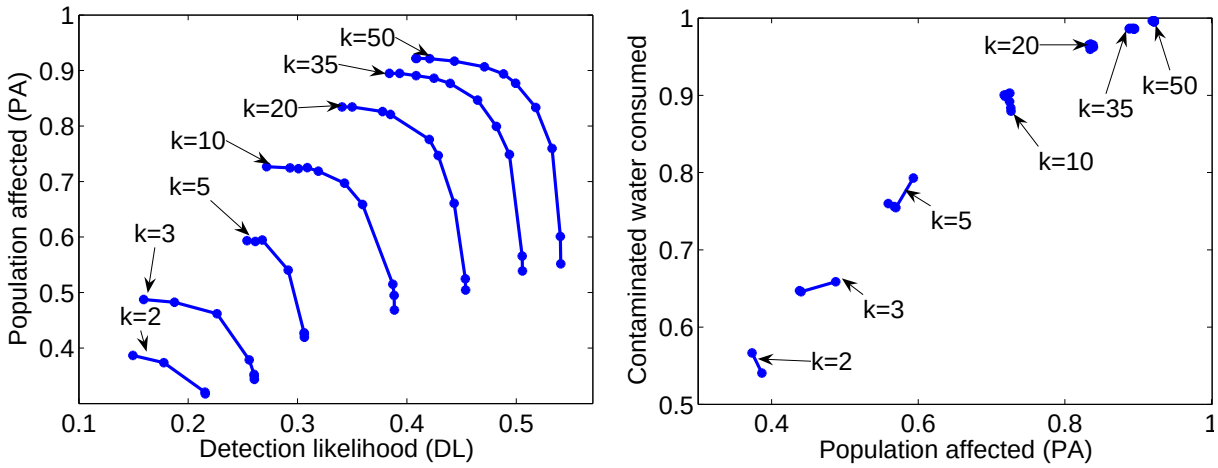
(a) PA (b) DL

Figure 8.4: Water network sensor placements: (a) when optimizing PA, sensors are concentrated in high population areas. (b) when optimizing DL, sensors are uniformly spread out.



(a) Random placements (b) Variance

Figure 8.5: (a) Performance of 100 random placements on the water distribution network for the PA objective function. (b) Performance with the number of simulated outbreaks. As more outbreaks are available we get better performance.



(a) (b)

Figure 8.6: (a) Trading off PA and DL. (b) Trading off PA and consumed contaminated water.

8.2.4 Multicriterion optimization and results from BWSN

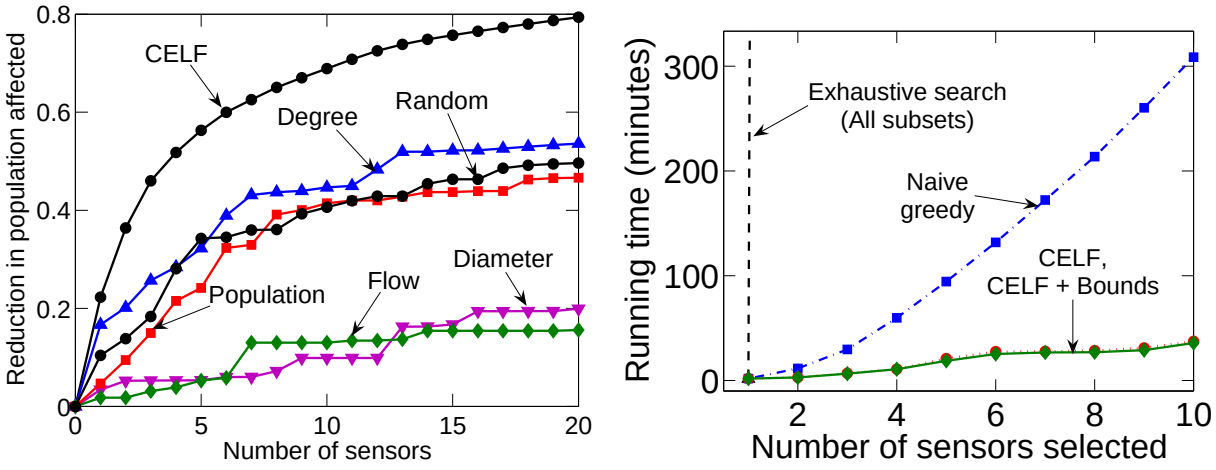
Using the theory developed in Section 8.1.4, we traded-off different objectives for the water distribution application. We selected pairs of objectives, *e.g.*, DL and PA, and varied the weights λ to produce (approximately) Pareto-optimal solutions. In Figure 8.6 (a) we plot the tradeoff curves for different placement sizes k. By adding more sensors, both objectives DL and PA increase. The crves also show, that if we, *e.g.*, optimize for DL, the PA score can be very low. However, there are points which achieve near-optimal scores in both criteria (the *knee* in the curve). This sweet spot is what we aim for in multi-criteria optimization.

We also traded off the affected population PA and a fourth criterion defined by BWSN, the *expected consumption of contaminated water*. Figure 8.6 (b) shows the trade-off curve for this experiment. Notice that the curves (almost) collapse to points, indicating that these criteria are highly correlated, which we expect for this pair of objective functions. Again, the efficiency of our implementation allows to quickly generate and explore these trade-off curves, while maintaining strong guarantees about near-optimality of the results.

We submitted our placements with equally weighted objective functions to BWSN, where Ostfeld et al. [2008] independently evaluated the contributed solutions of 15 participants. Since many solutions were non-dominated (and hence incomparable), the organizers of BWSN did not select a winner. However, in the conclusion of their analysis, for a collection of comparisons, they counted the number of non-dominated solutions. Our approach achieved the highest number with 26 out of 30 non-dominated solutions. The next best set of placements according to their evaluation using this metric was the one by Berry et al. [2006a] with 21 out of 30 non-dominated solutions.

8.2.5 Scalability

In the water distribution setting, we need to simulate 3.6 million contamination scenarios, each of which takes approximately 7 seconds and produces 14KB of data. Since most of the computer cluster scheduling



(a) Comparison with random

(b) Runtime

Figure 8.7: (a) Solutions of CELF outperform heuristic selections. (b) Running time of exhaustive search, greedy and CELF.

systems break if one would submit 3.6 million jobs into the queue, we developed a distributed architecture, where the clients obtain simulation parameters and then confirm the successful completion of the simulation. We run the simulation for a month on a cluster of around 40 machines. This produced 152GB of outbreak simulation data. By exploiting the properties of the problem described in Section 8.1.5, the size of the inverted index (which represents the relevant information for evaluating placement scores) is reduced to 16 GB which we were able to fit into main memory of a server. The fact that we could fit the data into main memory alone sped up the algorithms by at least a factor of 1000.

Figure 8.7 (b) presents the running times of CELF, the naive greedy algorithm and exhaustive search (extrapolated). When placing 20 sensors, the CELF algorithm returns a solution within 1 hour, as opposed to 30 hours for the naive greedy algorithm.

8.3 Case study: Blog Network

8.3.1 Experimental setup

In this chapter, we are not explicitly modeling the spread of information over the network, but rather consider cascades as *input* to our algorithms.

Here we are interested in blogs that actively participate in discussions, we biased the dataset towards the active part of the blogosphere, and selected a subset from the larger set of 2.5 million blogs of [Glance et al., 2005]. We considered all blogs that received at least 3 in-links in the first 6 months of 2006, and then took all their posts for the full year 2006. So, the dataset that we use has 45,000 blogs, 10.5 million posts, and 16.2 million links (30 GB of data). However, only 1 million links point inside the set of 45,000 blogs.

Posts have rich metadata, including time stamps, which allows us to extract information cascades, *i.e.*, subgraphs induced by directed edges representing the temporal flow of information. We adopt the following definition of a cascade [Leskovec et al., 2007a]: every cascade has a single starting post, and other

posts recursively join by linking to posts within the cascade, whereby the links obey time order. We detect cascades by first identifying starting post and then following in-links. We discover 346,209 non-trivial cascades having at least 2 nodes. Since the cascade size distribution is heavy-tailed, we further limit our analysis to only cascades that had at least 10 nodes. The final dataset has 17,589 cascades, where each blog participates in 9.4 different cascades on average.

8.3.2 Objective functions

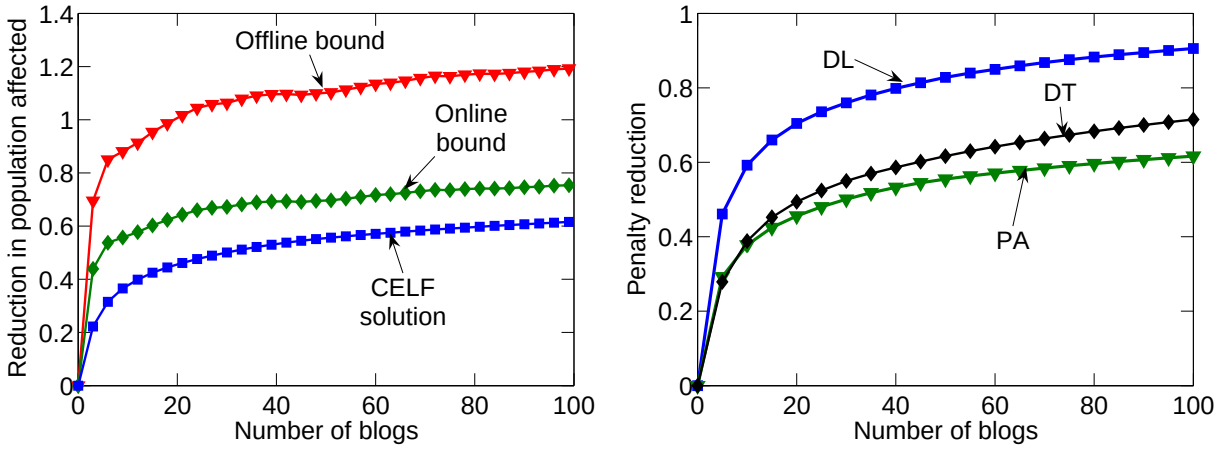
We use the penalty reduction objectives DL, DT and PA as introduced in Section 8.1.2. We normalize the scores of the solution to be between 0 and 1. For the DL (detection likelihood) criterion, the quality of the solution is the fraction of all detected cascades (regardless of when we detect it). The PA (population affected) criterion measures what fraction of the population included in the cascade after we detect it, *i.e.*, if we would be reading all the blogs initiating the cascades, then the quality of the solution is 1. In PA our reward depends on which fraction of the cascades we detect, and big cascades count more than small cascades.

8.3.3 Solution quality

First, we evaluate the performance of CELF, and estimate how far from optimal the solution could be. Note, that obtaining the optimal solution would require enumeration of $2^{45,000}$ subsets. Since this is impractical, we compare our algorithm to the bounds we developed in Section 5.3. Figure 8.8(a) shows scores for increasing budgets when optimized the PA (population affected) criterion. As we select more blogs to read, the proportion of cascades we catch increases (bottom line). We also plot the two bounds. The off-line bound (*c.f.*, Theorem 5.5) shows that the unknown optimal solution lies between our solution (bottom line) and the bound (top line). Notice the discrepancy between the lines is big, which means the bound is very loose. On the other hand, the middle line shows the online bound (*c.f.*, Theorem 5.10), which again tells us that the optimal solution is somewhere between our current solution and the bound. Notice, the gap is much smaller. This means (a) that the our on-line bound is much tighter than the traditional off-line bound. And, (b) that our CELF algorithm performs very close to the optimum.

In contrast to off-line bound, the on-line bound is *algorithm independent*, and thus can be computed regardless of the algorithm used to obtain the solution. Since it is tighter, it gives a much better worst case estimate of the solution quality. For this particular experiment, we see that CELF works very well: after selecting 100 blogs, we are at most **13.8**% away from the optimal solution.

Figure 8.8(b) shows the performance using various objective functions (from top to bottom: DL, DT, PA). DL increases the fastest, which means that one only needs to read a few blogs to detect most of the cascades, or equivalently that most cascades hit one of the big blogs. However, the population affected (PA) increases much slower, which means that one needs many more blogs to know about stories before the rest of population does. By using the on-line bound we also calculated that all objective functions are at most 5% to 15% from optimal.



(a) Performance of CELF (b) Objective functions

Figure 8.8: (a) Performance of CELF algorithm and off-line and on-line bounds for PA objective function. (b) Compares objective functions.

8.3.4 Cost of a blog

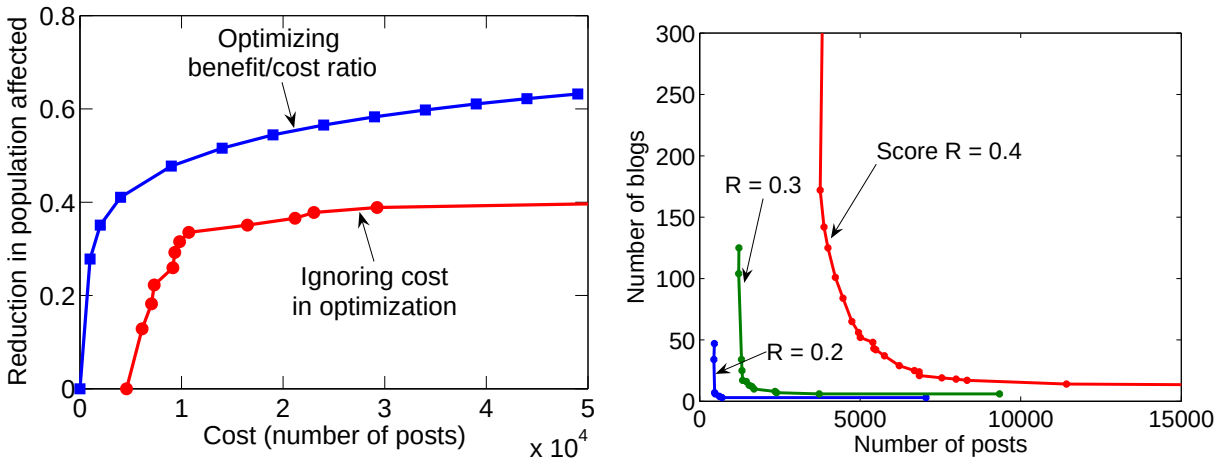
The results presented so far assume that every blog has the same cost. Under this *unit cost* model, the algorithm tends to pick large, influential blogs, that have many posts. For example, `instapundit.com` is the best blog when optimizing PA, but it has 4,593 posts. Interestingly, most of the blogs among the top 10 are politics blogs: `instapundit.com`, `michellemalkin.com`, `blogometer.nationaljournal.com`, and `sciencepolitics.blogspot.com`. Some popular aggregators of interesting things and trends on the blogosphere are also selected: `boingboing.net`, `themodulator.org` and `bloggersblog.com`. The top 10 PA blogs had more than 21,000 thousand posts in 2006. They account for 0.2% of all posts, 3.5% of all in-links, 1.7% of out-links inside the dataset, and 0.37% of all out-links.

Under unit cost model large blogs are important, but reading a blog with many posts is time consuming. This motivates the *number of posts (NP)* cost model, where we set the cost of a blog to the number of posts it had in 2006.

First, we compare the NP cost model with the unit cost in Figure 8.9(a). The top curve shows the value of the PA criterion for budgets of B posts, *i.e.*, we optimize PA such that the selected blogs can have at most B posts total. Note, that under the unit cost model, CELF chooses expensive blogs with many posts. For example, to obtain the same PA objective value, one needs to read 10,710 posts under unit cost model. The NP cost model achieves the same score while reading just 1,500 posts. Thus, optimizing the benefit cost ratio (PA/cost) leads to drastically improved performance.

Interestingly, the solutions obtained under the NP cost model are very different from the unit cost model. Under NP, political blogs are not chosen anymore, but rather summarizers (*e.g.*, `themodulator.org`, `watcherofweasels.com`, `anglican.tk`) are important. Blogs selected under NP cost appear about 3 days later in the cascade as those selected under unit cost, which further suggests that that summarizer blogs tend to be chosen under NP model.

In practice, the cost of reading a blog is not simply proportional to the number of posts, since we also



(a) Cost of a blog

(b) Cost tradeoff

Figure 8.9: (a) Comparison of the unit and the number of posts cost models. (b) For fixed value of PA F, we get multiple solutions varying in costs.

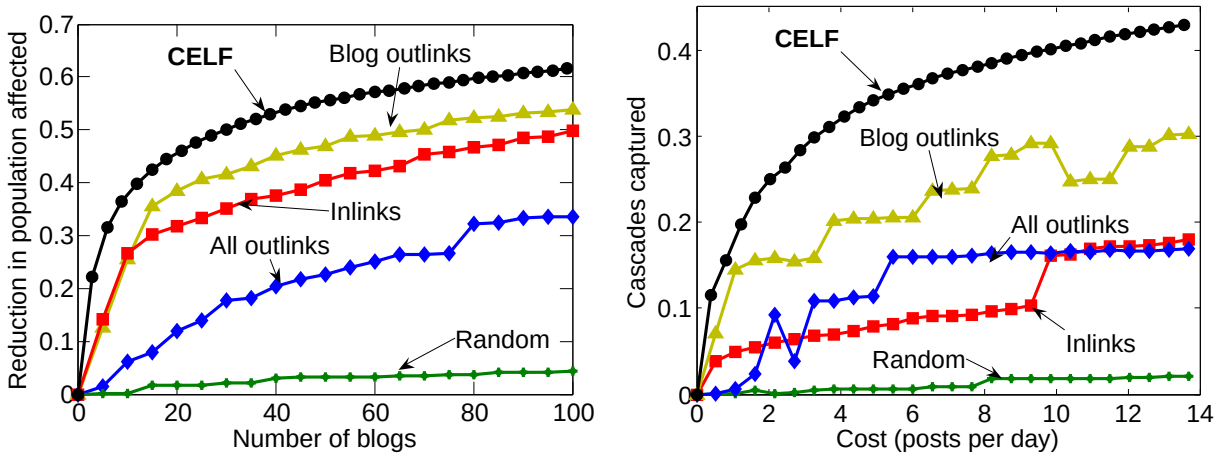
need to navigate to the blog (which takes constant effort per blog). Hence, a combination of unit and NP cost is more realistic. Figure 8.9(b) interpolates between these two cost models. Each curve shows the solutions with the same value F of the PA objective, but using a different number of posts (x-axis) and blogs (y-axis) each. For a given F, the ideal spot is the one closest to origin, which means that we want to read the least number of posts from least blogs to obtain desired score F. Only at the end points does CELF tend to pick extreme solutions: few blogs with many posts, or many blogs with few posts. Note, there is a clear knee on plots of Figure 8.9(b), which means that by only slightly increasing the number of blogs we allow ourselves to read, the number of posts needed decreases drastically, while still maintaining the same value F of the objective function.

8.3.5 Comparison to heuristic blog selection

Next, we compare our method with several intuitive heuristic selection techniques. For example, instead of optimizing the DT, DL or PA objective function using CELF, we may just want to select the most popular blogs and hope to detect many cascades. We considered several such heuristics, where we order blogs by some "goodness" criteria, and then pick top blogs (until the budget is exhausted). We consider the following criteria: the cumulative number of out-links of blog's posts, the number of in-links the blog received from other blogs in the dataset, and the number of out-links to other blogs in the dataset.

As Figure 8.10(a) shows, the CELF algorithm greatly outperforms all the heuristic selection techniques. More interestingly, the best heuristics (doing 45% worse than CELF) pick blogs by the number of in- or out-links from/to other blogs in the dataset. The total number of out-links and random blog selection do not perform well.

Number of in-links is the indicator of a blog's tendency to create cascades, while number of out-links (to other blogs) indicates blog's tendency to summarize the blogosphere. We also note, that the surprisingly good performance of the number of out-links to blogs in the dataset is an artefact of our "closed-world" dataset, and in real-life we can not estimate this. The results also agree well with our intuition that the



(a) Unit cost (b) Number of posts cost

Figure 8.10: Heuristic blog selection methods. (a) unit cost model, (b) number of posts cost model.

number of in-links is a good heuristic, since it directly indicates the of propagation of information.

Figure 8.10(b) explores the same setting under the NP cost model. Here, given a budget of B posts, we select a set of blogs to optimize PA objective. For the heuristics, we select a set of blogs to optimize chosen heuristic, *e.g.*, the total number of in-links of selected blogs while still fitting inside the budget of B posts. Again, CELF outperforms the next best heuristics by 41%, and again the number of in- and out-links are the best heuristics.

These results show that simple heuristics that one could use to identify blogs to read do not really work well. There are good summarizer blogs that may not be very popular, but which, by using few posts, catch most of the important stories propagating over the blogosphere.

8.3.6 Scalability

Figure 8.9(b) plots the running time of selecting k blogs. We see that exhaustively enumerating all possible subsets of k elements is infeasible (the line jumps out of the plot for $k = 3$). The simple greedy algorithm scales as $\Omega(k|\mathcal{V}|)$, since for every increment of k we need to consider selecting all remaining $|\mathcal{V}| - k$ blogs. The bottom line overlapping the x-axis of Figure 8.9(b) shows the performance of our CELF algorithm. For example, for selecting 100 blogs, greedy algorithm runs 4.5h, while CELF takes 23 seconds (700 times faster). Calculation of the on-line bounds while running CELF takes 54s.

Exploiting the sparsity of the problem (*c.f.*, Section 8.1.5) allowed us to reduce the size of the inverted index from originally 3.5 GB to 50 MB, easily fitting it in main memory.

8.4 Discussion and related work

8.4.1 Relationship to Influence Maximization

In Kempe et al. [2003] introduced a *Triggering Model* for modeling the spread of influence in a social network. As the authors show, this model generalizes the Independent Cascade, Linear Threshold and Listen-once models commonly used for modeling the spread of influence. Essentially, this model describes a probability distribution over directed graphs, and the influence is defined as the expected number of nodes reachable from a set of nodes, with respect to this distribution. Kempe et al. [2003] showed that the problem of selecting a set of nodes with maximum influence is submodular, satisfying the conditions of Theorem 5.2, and hence the greedy algorithm provides a $(1-1/e)$ approximation. The problem addressed in this chapter generalizes this Triggering model:

Theorem 8.2. *The Triggering Model of Kempe et al. [2003] is a special case of our network outbreak detection problem.*

In order to prove Theorem 8.2, we consider fixed directed graphs sampled from the Triggering distribution. If we revert the arcs in any such graph, then our PA objective corresponds exactly to the influence function of Kempe et al. [2003] applied to the original graph.

Theorem 8.2 shows that spreading influence under the general Triggering Model can be considered a special case of our outbreak detection formalism. The problems are fundamentally related since, when spreading influence, one tries to affect as many nodes as possible, while when detecting outbreak, one wants to minimize the effect of an outbreak in the network. Secondly, note that in the example of reading blogs, it is not necessarily a good strategy to affect nodes which are very influential, as these tend to have many posts, and hence are expensive to read. In contrast to influence maximization, the notion of cost-benefit analysis is crucial to our applications.

8.4.2 Related work

Virus propagation and outbreak detection

Work on spread of diseases in networks and immunization mostly focuses on determining the value of the *epidemic threshold* [Bailey, 1975], a critical value of the virus transmission probability above which the virus creates an epidemic. Several strategies for immunization have also been proposed: uniform node immunization, targeted immunization of high degree nodes [Pastor-Satorras and Vespignani, 2002], acquaintance immunization, which focuses on highly connected nodes [Cohen et al., 2003], and immunization on based on spectral properties of the network [Giakkoupis et al., 2005]. In the context of our work, uniform immunization strategy corresponds to randomly placing sensors in a water network. Similarly, targeted immunization corresponds to selecting blogs based on their in- or out-degree. As we have seen in Figures 8.10 and 8.7, both strategies perform much worse than direct optimization of the *population affected* criterion.

Information cascades and blog networks

Cascades have been studied for many years by sociologists concerned with the *diffusion of innovation* [Rogers, 1995]; more recently, cascades we used for studying viral marketing [Goldenberg et al.,

2001, Kempe et al., 2003, Leskovec et al., 2006], selecting trendsetters in social networks [Richardson and Domingos, 2002], and explaining trends in blogspace [Gruhl et al., 2004, Kumar et al., 2003]. Studies of blogspace either spend effort mining topics from posts [Gruhl et al., 2004] or consider only the properties of blogspace as a graph of unlabeled URLs [Kumar et al., 2003]. Recently, [Leskovec et al., 2007a] studied the properties and models of information cascades in blogs. While previous work either focused on empirical analyses of information propagation and/or provided models for it, we develop a general methodology for node selection in networks while optimizing a given criterion.

Water distribution network monitoring

A large number of approaches have been proposed for optimizing water sensor networks (*c.f.*, [Berry et al., 2006b] for a concise overview of the prior literature). Most of these approaches are only applicable to small networks up to approximately 500 nodes. Many approaches are based on heuristics (such as genetic algorithms [Ostfeld and Salomons, 2004], cross-entropy selection [Dorini et al., 2006], predator-prey heuristics [Gueli, 2006], etc.) that cannot provide provable performance guarantees about the solutions. Closest to ours is an approach by [Berry et al., 2006b], who equate the placement problem with a p-median problem, and make use of a large toolset of existing algorithms for this problem. The problem instances solved by [Berry et al., 2006b] are a factor 72 smaller than the instances considered in this chapter. In order to obtain bounds for the quality of the generated placements, the approach in [Berry et al., 2006b] needs to solve a complex ($\mathbf{NP}$-hard) mixed-integer program. Our approach is the first algorithm for the water network placement problem, which is guaranteed to provide solutions which achieve at least a constant fraction of the optimal solution within polynomial time. Additionally, it handles orders of magnitude larger problem instances than previously considered.

Facility location

Closely related to the adversarial outbreak detection problem is the *k-median* problem, which is a widely studied clustering problem. In this problem, one is given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ along with a distance function defined over pairs of nodes in $\mathcal{V}$. The goal is to select a subset $\mathcal{A} \subseteq \mathcal{V}$ of size at most k, such that the average distance between any unselected node $s \in \mathcal{V} \setminus \mathcal{A}$ and its nearest center $s' \in \mathcal{A}$ is minimized. For the case where the distance function is symmetric and satisfies the triangle inequality, a constant-factor 1.728 approximation can be obtained [Charikar and Guha, 2005].

If we attempt to minimize the penalty, instead of maximizing the penalty reduction (as considered in this chapter), and if we ignoring the temporal aspect of the outbreak detection problem (i.e., the time of it takes a contaminant to travel an edge in the network varies with time, as it does in our setting), outbreak detection could be modeled as a k-median problem. However, in the water network setting, there is a strong directionality in the water flow, making the problem asymmetric. Archer [2000] showed that for asymmetric distance metrics, k-median is $\Omega(\log |\mathcal{V}|)$-hard to approximate under reasonable complexity theoretic assumptions.

For the setting considered in this chapter, we can in fact show that it is not possible to obtain *any* approximation:

Proposition 8.3. *Unless $\mathbf{P} = \mathbf{NP}$, no polynomial time algorithm can give any approximation guarantee on the penalty minimization problem.*

8.5 Summary

In this chapter, we presented a novel methodology for selecting nodes to detect outbreaks of dynamic processes spreading over a graph. We showed that many important objective functions, such as detection time, likelihood and affected population are submodular. Our CELF algorithm (*c.f.*, Chapter 5) exploits this submodularity to find *near-optimal* node selections – the obtained solutions are guaranteed to achieve at least a fraction of $\frac{1}{2}(1 - 1/e)$ of the optimal solution, even in the more complex case where every node can have an arbitrary additive cost. Our CELF algorithm is up to 700 times faster than standard greedy algorithm. We also developed novel online bounds on the quality of the solution obtained by *any* algorithm. We used these bounds to prove that the solutions we obtained in our experiments achieve 90% of the optimal score (which is intractable to compute).

We extensively evaluated our methodology on two large real-world problems: (a) detection of contaminations in the largest *city water distribution* network considered so far in the literature, and (b) selection of *informative blogs* in a network of more than 10 million posts. We showed how our CELF algorithm greatly outperforms intuitive heuristics. We also demonstrated that our methodology can be used to study complex application-specific questions such as multicriteria tradeoff, cost-sensitivity analyses and generalization behavior. In addition to demonstrating the effectiveness of our method, we obtained some counterintuitive results about the problem domains, such as the fact that the popular blogs might not be the most effective way to catch relevant information cascades.

Part III

Non-greedy Algorithms for Complex Sensing Problems

Chapter 9

Overview of Part III

In Part II of the Thesis, we studied several classes of sensing problems, like spatial monitoring and outbreak detection. We realized that the natural objective functions in these problems satisfy *submodularity*. We saw that for the budgeted maximization of these objective functions, the *myopic* greedy algorithm, which plans ahead only on the next best observation to make, actually obtains near-optimal *nonmyopic* results.

Robust sensor placements. However, in many practical problems, additional constraints or requirements are present. For example, often, we want to select observations which are robust against a number of possible objective functions. Examples include minimizing the maximum posterior variance in Gaussian Process regression, robust experimental design, and sensor placement for outbreak detection. In Chapter 10, we will show that in the case of adversarial noise, the myopic greedy algorithm performs arbitrarily poorly. We then present the *Submodular Saturation* algorithm, a simple and efficient algorithm with strong theoretical approximation guarantees for cases where all objective functions exhibit submodularity. Moreover, we prove that better approximation algorithms do not exist unless $\mathbf{NP}$-complete problems admit efficient algorithms. We also show how the algorithm can be used to near-optimally trade off expected-case (e.g., the Mean Square Prediction Error in Gaussian Process regression) and worst-case (e.g., maximum predictive variance) performance. We show that many important machine learning problems fit our robust submodular observation selection formalism, and provide extensive empirical evaluation on several real-world problems. For Gaussian Process regression, our algorithm compares favorably with state-of-the-art heuristics described in the geostatistics literature, while being simpler, faster and providing theoretical guarantees. For robust experimental design, our algorithm performs favorably compared to existing algorithms based on Semidefinite Programming.

Sensor placement under complex cost functions. When using wireless sensors to monitor spatial phenomena, we face two competing requirements: Not only should the sensors be informative, but they should also be able to communicate efficiently. In previous chapters, we have seen how one can measure the predictive quality of a set of sensor locations. In Chapter 11, we show how we can also predicting the communication cost involved with arbitrary sensor placements. Based on our models for sensing quality and communication cost, we then develop an algorithm, pSPIEL, which selects Sensor Placements at Informative and cost-Effective Locations. Our algorithm has strong approximation guarantees for optimizing the $\mathbf{NP}$-hard tradeoff. In addition, we show how pSPIEL can be combined with SATURATE, to

obtain placements which are robust against changes in the environment. We also show how pSPIEL can be used to solve other combinatorial problems, such as planning informative paths for multiple robots. We also provide extensive experimental validation of our practical approach on several real-world placement problems.

Simultaneous placement and scheduling of sensors. So far, we have mainly considered the problem of locating a small set of sensors such that they provide as much information as possible. However, in many applications, given power constraints due to limited batteries, we also need to determine when to selectively activate these sensors in order to maximize the performance while satisfying lifetime requirements. Traditionally, these two problems of sensor placement and scheduling have been considered separately from each other; one first decides where to place the sensors, and then when to activate them. In Chapter 12, we present an efficient algorithm, eSPASS, that simultaneously optimizes the placement and the schedule. We prove that eSPASS provides a constant-factor approximation to the optimal solution of this **NP**-hard optimization problem. A salient feature of our approach is that it obtains "balanced" schedules that perform uniformly well over time, rather than only on average. We present extensive empirical studies on several sensing tasks, and our results show that simultaneously placing and scheduling gives drastically improved performance compared to separate placement and scheduling.

Summary of contributions. The key contributions of this part of the Thesis are:

1. We consider complex optimization tasks arising in sensing problems. In all these problems, existing greedy approaches perform arbitrarily badly. We develop novel, efficient approximation algorithms:
   - SATURATE for solving robust sensing problems,
   - pSPIEL for optimizing sensor placements under complex cost functions and
   - eSPASS for simultaneously optimizing sensor placements and schedules.
2. We prove hardness of sensing problems. Under reasonable complexity theoretic assumptions, optimizing robust sensing cannot be approximated by any factor unless $\mathbf{P} = \mathbf{NP}$. Our algorithm, SATURATE, provides the best possible bicriterion guarantees (allowing relaxation of certain constraints) for this problem.
3. We demonstrate our algorithms on several real-world case studies, including
   - sensor placement for spatial prediction in Gaussian Processes, demonstrated on real-world sensing data sets and a proof-of-concept case study of a deployed sensor network,
   - sensor placement for outbreak detection in a metropolitan-area drinking water distribution network, where our algorithms obtained a top score in an international competition and
   - sensor scheduling for improving the lifetime of wireless traffic sensors, and for light sensors in a building management prototype testbed.

Chapter 10

Robust Sensing Problems

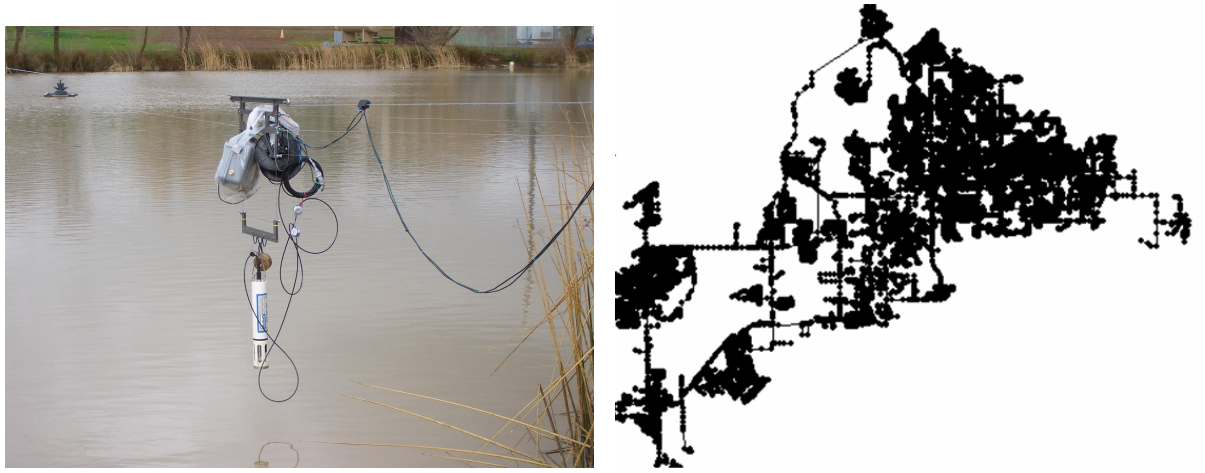
In many of the observation selection problems considered in this Thesis, observations are made with physical devices, which can fail. Similarly, in practice, the probabilistic model (which we assumed to exist so far) is often not fully known and has to be estimated from limited data. In such settings, there are important robustness requirements associated with the observation selection methodology: For example, the chosen observations should still be informative in case a sensor fails, or in case the model is inaccurately estimated from data.

Previously, we have seen that many observation selection problems can be seen as the problem of maximizing a nondecreasing submodular sensing quality function F (*c.f.*, Chapter 5). As we show in this chapter, we can model robust observation selection problems by requiring that a *set* of submodular functions $F_1, \ldots, F_m$ simultaneously be maximized. Each of the functions could, for example, correspond to the case where a particular sensor is failing, or correspond to one particular probabilistic model which could describe the world.

While $\mathbf{NP}$-hard, the problem of selecting an optimal set of k observations maximizing a single submodular objective can be approximately solved using a simple greedy forward-selection algorithm, which is guaranteed to perform near-optimally (Nemhauser et al. [1978], see also Chapter 5). However, as we show, this simple *myopic* algorithm performs arbitrarily badly in the case of a worst-case objective function. In this chapter, we address the fundamental problem of nonmyopically selecting observations which are robust against such an adversarially chosen submodular objective function. In particular:

- We present SATURATE, an efficient algorithm for the robust submodular observation selection problem. Our algorithm guarantees solutions which are at least as informative as the optimal solution, at only a slightly higher cost.
- We prove that our approximation guarantee is the best possible, i.e., the guarantee cannot be improved unless $\mathbf{NP}$-complete problems admit efficient algorithms.
- We discuss several extensions of our approach, handling complex cost functions and trading off worst-case and average-case performance.
- We extensively evaluate our algorithm on several real-world tasks, including minimizing the maximum posterior variance in Gaussian Process regression, finding experiment designs which are robust with respect to parameter uncertainty, and sensor placement for outbreak detection.

This chapter is organized as follows. In Section 10.1, we formulate the robust submodular observation selection problem, and in Section 10.2, we analyze its hardness. We subsequently present SATURATE, an efficient approximation algorithm for this problem (Section 10.3), and show that our approximation guarantees are best possible, unless **NP**-complete problems admit efficient algorithms (Section 10.4). In Section 10.5, we discuss how many important machine learning problems are instances of our robust submodular observation selection formalism. We then discuss extensions (Section 10.6) and evaluate the performance of SATURATE on several real-world observation selection problems (Section 10.7). Section 10.9 presents heuristics to improve the computational performance of our algorithm, Section 10.10 reviews related work, and Section 10.12 presents our conclusions.



(a) *NIMS deployed at UC Merced* (b) *Water distribution network*

Figure 10.1: (a) Deployment of the Networked Infomechanical System (NIMS, Harmon et al. 2006) to monitor a lake near UC Merced. (b) Illustration of the municipal water distribution network considered in the Battle of the Water Sensor Networks challenge [*c.f.* Ostfeld et al., 2008].

10.1 Robust submodular observation selection

In this section, we first review the concept of submodularity (Section 10.1.1), and then introduce the *robust submodular observation selection (*RSOS*)* problem (Section 10.1.2).

10.1.1 Submodular observation selection

Let us consider a spatial prediction problem, where we want to estimate the pH values across a horizontal transect of a river, e.g., using the NIMS robot shown in Figure 10.1(a). As discussed in Chapter 6, we can discretize the space into a finite number of locations $\mathcal{V}$, where we can obtain measurements, and model a joint distribution $P(\mathcal{X}_\mathcal{V})$ over variables $\mathcal{X}_\mathcal{V}$ associated with these locations, for example, by using a Gaussian Process. While in Chapter 14 we have considered the mutual information criterion, an alternative goal in spatial monitoring is to select a subset of locations $\mathcal{A} \subseteq \mathcal{V}$ to observe, such that the

average predictive variance,

$$V(\mathcal{A}) = \frac{1}{n}\sum_i \sigma^2_{i|\mathcal{A}},$$

is minimized (*c.f.*Section 10.5.1 for more details). Hereby, $\sigma^2_{i|\mathcal{A}}$ denotes the predictive variance at location i after observing locations $\mathcal{A}$, i.e.,

$$\sigma^2_{i|\mathcal{A}} = \int P(\mathbf{x}_\mathcal{A})\mathbb{E}\left[(\mathcal{X}_i - \mathbb{E}\left[\mathcal{X}_i \mid \mathbf{x}_\mathcal{A}\right])^2 \mid \mathbf{x}_\mathcal{A}\right] d\mathbf{x}_\mathcal{A}.$$

Unfortunately, the problem

$$\mathcal{A}^* = \operatorname*{argmin}_{|\mathcal{A}|\leq k} V(\mathcal{A})$$

is **NP**-hard in general [Das and Kempe, 2008], and the number of candidate solutions is very large, so generally we cannot expect to efficiently find the optimal solution. Fortunately, as Das and Kempe [2008] show, in many cases, the *variance reduction*

$$F_s(\mathcal{A}) = \sigma^2_s - \sigma^2_{s|\mathcal{A}}$$

at any particular location s is submodular. In the terminology of Chapter 2, this criterion corresponds to choosing the sensing quality function:

$$u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = \sigma^2_s - \sigma^2_{s|\mathcal{A}}.$$

Since each of the variance reduction functions F_s is submodular, the *average variance reduction*

$$F(\mathcal{A}) = V(\emptyset) - V(\mathcal{A}) = \frac{1}{n}\sum_s F_s(\mathcal{A})$$

is also submodular. The average variance reduction is also *nondecreasing*, i.e., for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ it holds that $F(\mathcal{A}) \leq F(\mathcal{B})$, and *normalized* ($F(\emptyset) = 0$).

Hence, the problem of picking the best k locations to minimize the average variance is an instance of Problem (2.4) with $C(\mathcal{A}) = |\mathcal{A}|$ and $B = k$, i.e.,

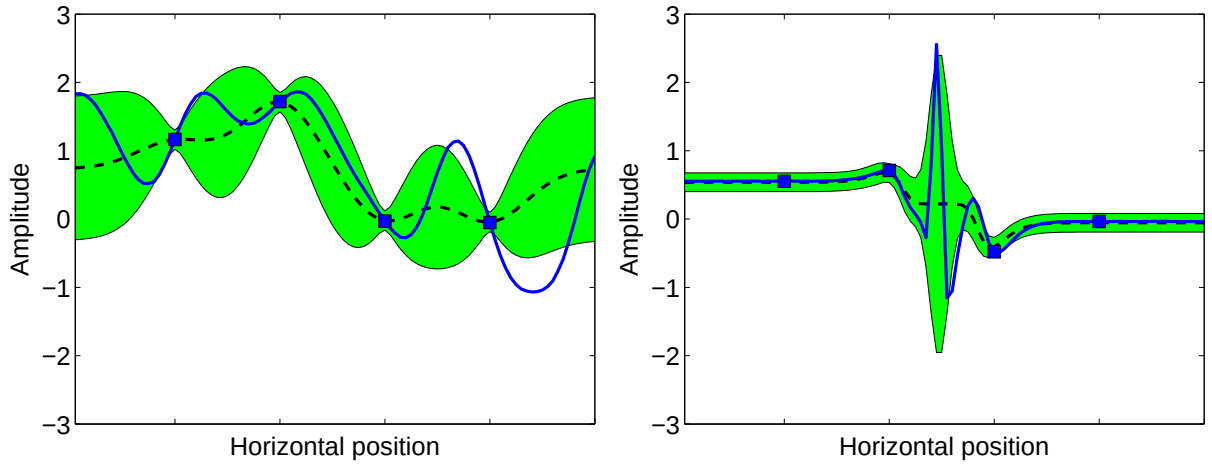
$$\max_{\mathcal{A}\subseteq\mathcal{V}} F(\mathcal{A}), \quad \text{subject to} \quad |\mathcal{A}| \leq k, \tag{10.1}$$

where F is normalized, nondecreasing and submodular, and k is a bound on the number of observations we can make. Hence, Theorem 5.2 can be applied, guaranteeing that the greedy algorithm obtains a solution that achieves at least a constant fraction $(1 - 1/e)$ of the objective value obtained by the optimal solution, i.e.,

$$F(\mathcal{A}_G) \geq (1 - 1/e) \max_{|\mathcal{A}|\leq k} F(\mathcal{A}).$$

10.1.2 The robust submodular observation selection (RSOS) problem

For phenomena, such as the one indicated in Figure 10.2(a), which are spatially homogeneous (isotropic), maximizing this average variance reduction leads to effective variance reduction everywhere in the space. However, many spatial phenomena are nonstationary, being smooth in certain areas and highly variable in



(a) *High average variance* (b) *High maximum variance*

Figure 10.2: Spatial predictions using Gaussian Processes with a small number of observations. The blue solid line indicates the unobserved latent function, and blue squares indicate observations. The plots also show confidence bands (green). Dashed line indicates the prediction. (b) shows an example with high maximum predictive variance, but low average variance, whereas (a) shows an example with high average variance, but lower maximum variance. Note, that in (b) we are most uncertain about the most variable (and interesting, since it is hard to predict) part of the curve, suggesting that the maximum variance should be optimized.

others, such as the example indicated in Figure 10.2(b). In such a case, maximizing the average variance reduction will typically put only few examples in the areas highly variable areas. However, those regions are typically the most interesting, since they are most difficult to predict. In such cases, we might want to simultaneously minimize the variance everywhere in the space.

More generally, in many applications (such as the spatial monitoring problem discussed above, and several other examples which we present in Section 10.5), we want to perform equally well with respect to *multiple* objectives. We will hence consider settings where we are given a *collection* of normalized nondecreasing submodular functions $F_1, \ldots, F_m$, and we want to solve

$$\max_{\mathcal{A} \subseteq \mathcal{V}} \min_i F_i(\mathcal{A}), \quad \text{subject to} \quad |\mathcal{A}| \leq k. \tag{10.2}$$

The goal of Problem (10.2) is to find a set $\mathcal{A}$ of observations, which is robust against the worst possible objective, $\min_i F_i$, from our set of possible objectives. Consider the spatial monitoring setting for example, and assume that the prior variance σ_i^2 is constant (we will relax this assumption in Section 10.6.2) over all locations i. Then, the problem of minimizing the maximum variance, as motivated by the example in Figure 10.2, is equivalent to maximizing the minimum variance reduction, i.e., solving Problem (10.2) where F_i is the variance reduction at location i.

We call Problem (10.2) the *Robust Submodular Observation Selection (*RSOS*)* problem. Note, that even if the F_i are all submodular, $F_{wc}(\mathcal{A}) = \min_i F_i(\mathcal{A})$ is generally *not* submodular. In fact, we show below that, in this setting, the simple greedy algorithm (which performs near-optimally in the single-criterion setting) can perform arbitrarily badly. While the example in Table 10.1 might seem artificial, as we show in Section 10.7 (especially Section 10.7.3), the greedy algorithm exhibits very poor performance when applied to practical problems.

$\mathcal{A}$	$F_1(\mathcal{A})$	$F_2(\mathcal{A})$	$\min_i F_i(\mathcal{A})$
$\emptyset$	0	0	0
$\{s_1\}$	n	0	0
$\{s_2\}$	0	n	0
$\{t_1\}$	1	1	1
$\{t_2\}$	1	1	1
$\{s_1, s_2\}$	n	n	n
$\{s_1, t_1\}$	$n+1$	1	1
$\{s_1, t_2\}$	$n+1$	1	1
$\{s_2, t_1\}$	1	$n+1$	1
$\{s_2, t_2\}$	1	$n+1$	1
$\{t_1, t_2\}$	2	2	2

Table 10.1: Functions F_1 and F_2 used in the counterexample.

10.2 Hardness of the robust submodular observation selection problem

Given the near-optimal performance of the greedy algorithm for the single-objective problem, a natural question is if the performance guarantee generalizes to the more complex robust optimization setting. Unfortunately, this hope is far from true, even in the simpler case of *modular* (additive) functions F_i. Consider a case with two submodular functions, F_1 and F_2, where the set of observations is $\mathcal{V} = \{s_1, s_2, t_1, t_2\}$. The functions take values as indicated in Table 10.1. Optimizing for a set of 2 elements, the greedy algorithm maximizing $F_{wc}(\mathcal{A}) = \min\{F_1(\mathcal{A}), F_2(\mathcal{A})\}$ would first choose t_1 (or t_2), as this choice increases the objective $\min\{F_1, F_2\}$ by 1, as opposed to 0 for s_1 and s_2. The greedy solution for $k = 2$ would then be the set $\{t_1, t_2\}$, obtaining a score of 2. However, the optimal solution with $k = 2$ is $\{s_1, s_2\}$, with a score of n. Hence, as $n \to \infty$, the greedy algorithm performs arbitrarily worse than the optimal solution.

Given that the greedy algorithm performs arbitrarily badly, our next hope would be to obtain a different good approximation algorithm. However, we can show that most likely this is not possible:

Theorem 10.1. *Unless* $\mathbf{P} = \mathbf{NP}$*, there cannot exist* any *polynomial time approximation algorithm for Problem* (10.2)*. More precisely: If there exists a positive function* $\gamma(\cdot) > 0$ *and an algorithm that, for all* n *and* k*, in time polynomial in the size of the problem instance* n*, is guaranteed to find a set* $\mathcal{A}'$ *of size* k *such that* $\min_i F_i(\mathcal{A}') \geq \gamma(n) \max_{|\mathcal{A}| \leq k} \min_i F_i(\mathcal{A})$*, then* $\mathbf{P} = \mathbf{NP}$*.*

Thus, unless $\mathbf{P} = \mathbf{NP}$, there cannot exist any algorithm which is guaranteed to provide, e.g., even an exponentially small fraction ($\gamma(n) = 2^{-n}$) of the optimal solution. All proofs can be found in the Appendix.

10.3 SATURATE: The submodular saturation algorithm

We now present an algorithm that finds a set of observations which perform at least as well as the optimal set, but at slightly increased cost; moreover, we show that no efficient algorithm can provide better guarantees (under reasonable complexity-theoretic assumptions).

10.3.1 Algorithm overview

For now we assume that all F_i take only integral values; this assumption is relaxed in Section 10.6.1. The key idea is to consider the following alternative problem formulation:

$$\max_{c,\mathcal{A}} c, \qquad \text{subject to} \quad F_i(\mathcal{A}) \geq c \text{ for } 1 \leq i \leq m \text{ and } |\mathcal{A}| \leq k. \tag{10.3}$$

We want a set $\mathcal{A}$ of size at most k, such that $F_i(\mathcal{A}) \geq c$ for all i, and c is as large as possible. Note that Problem (10.3) is equivalent to the original Problem (10.2): Maximizing c subject to the existence of a set $\mathcal{A}$, $|\mathcal{A}| \leq k$ such that $F_i(\mathcal{A}) \geq c$ for all i is equivalent to maximizing $\min_i F_i(\mathcal{A})$.

Now suppose we had an algorithm that, for any given value c, solves the following optimization problem:

$$\mathcal{A}_c = \operatorname*{argmin}_{\mathcal{A}} |\mathcal{A}| \qquad \text{subject to} \quad F_i(\mathcal{A}) \geq c \text{ for } 1 \leq i \leq m \tag{10.4}$$

i.e., finds the smallest set $\mathcal{A}$ with $F_i(\mathcal{A}) \geq c$ for all i. If this set has at most k elements, then c (and the set $\mathcal{A}$) is feasible for the RSOS Problem (10.3). If we cannot find a set $\mathcal{A}$ satisfying $F_i(\mathcal{A}) \geq c$ for all i and containing at most k elements, then c is infeasible. A binary search on c would then allow us to find the optimal solution with the maximum feasible c. We call Problem (10.4) the MINCOVER$_c$ problem, as it requires to find the smallest set guaranteeing an equal amount of coverage, c, for all objective functions F_i.

Since Theorem 10.1 rules out *any* approximation algorithm which respects the constraint k on the size of the set $\mathcal{A}$, our only hope for non-trivial guarantees requires us to relax this constraint. Our algorithm is based on the following approach:

- We define a relaxed version of the RSOS problem with a superset of feasible solutions that we call RELRSOS.
- We will maintain a lower bound (a feasible solution) for RELRSOS, and an upper bound for RSOS.
- We will then successively improve the upper and lower bounds using a binary search procedure. Upon convergence, we are thus guaranteed a feasible solution to RELRSOS, that performs at least as well as the optimal solution to the RSOS problem.

We now define the RELRSOS problem, the relaxed version of the RSOS Problem (10.3).

$$\max_{c,\mathcal{A}} c, \qquad \text{subject to} \quad F_i(\mathcal{A}) \geq c \text{ for } 1 \leq i \leq m \text{ and } |\mathcal{A}| \leq \alpha k. \tag{10.5}$$

Hereby, $\alpha \geq 1$ is a parameter relaxing the constraint on $|\mathcal{A}|$. If $\alpha = 1$, we recover the RSOS Problem (10.3).

As described above, our goal will be to *approximately* solve the RELRSOS Problem (10.5) for a fixed constant α. More formally, we will develop an efficient algorithm, SATURATE, which returns a solution $(c', \mathcal{A}')$ that is feasible for the RELRSOS Problem (10.5), and achieves a score that is at least as good as an optimal solution $(c^*, \mathcal{A}^*)$ to the RSOS Problem (10.3), i.e., $c' \geq c^*$ and $|\mathcal{A}'| \leq \alpha|\mathcal{A}^*| \leq \alpha k$.

The basic idea of SATURATE is to use the binary search procedure (maintaining a search interval $[c_{\min}, c_{\max}]$) as described above, but using an *approximate* algorithm, GPC (for *Greedy Partial Coverage*) that we will

develop below, for the MINCOVER$_c$ Problem (10.4). When invoked with a fixed value c, the GPC algorithm will return a feasible solution $|\mathcal{A}'_c|$ to the MINCOVER$_c$ Problem (10.4). We will furthermore guarantee that

- $|\mathcal{A}'_c| > \alpha k$ implies that $c > c^*$, i.e., c is an upper bound to the RSOS Problem (10.3), and hence it is safe to set $c_{\max} = c$, and
- $|\mathcal{A}'_c| \leq \alpha k$ implies that $\mathcal{A}'_c$ is a feasible solution (lower bound) to the RELRSOS Problem (10.5). $\mathcal{A}'_c$ is then kept as best current solution and we can set $c_{\min} = c$.

The binary search procedure will hence always maintain an upper bound $c_{\max}$ to the RSOS Problem (10.3), and a lower bound $c_{\min}$ to the RELRSOS Problem (10.5). Upon termination, it is thus guaranteed to find a solution $\mathcal{A}_S$ for which it holds that $\min_i F_i(\mathcal{A}_S) \geq c^*$ (since $\mathcal{A}_S$ is an upper bound to the RSOS Problem (10.3)) and $|\mathcal{A}_S| \leq \alpha k$ (since $\mathcal{A}_S$ is feasible for the RELRSOS Problem (10.5)). Hence, the approximate solution $\mathcal{A}_S$ obtains minimum value at least as high as the best possible score obtainable using k elements, but using slightly more (at most αk) elements than k elements. Figure 10.3 illustrates the feasible regions of the RSOS and RELRSOS problems, as well as the binary search procedure.

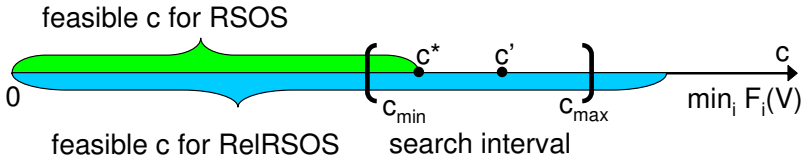


Figure 10.3: Illustration of feasible regions for the RSOS and RELRSOS problems. $[c_{\min}, c_{\max}]$ is the search interval during some iteration of SATURATE. c^* is the optimal solution to the RSOS problem, and c' is the solution that will eventually be returned by SATURATE.

10.3.2 Algorithm details

We will now provide formal details for the algorithm sketched in Section 10.3.1. As trivial lower and upper bounds for the RSOS problem we can initially set $c_{\min} = 0 \leq \min_i F_i(\emptyset)$, and $c_{\max} = \min_i F_i(\mathcal{V})$, due to nondecreasingness of the F_i.

First, we will develop the efficient algorithm GPC which approximately solves the MINCOVER$_c$ Problem (10.4). For any value c that could possibly be feasible (i.e., $0 \leq c \leq \min_i F_i(\mathcal{V})$), define $\widehat{F}_{i,c}(\mathcal{A}) = \min\{F_i(\mathcal{A}), c\}$, the original function F_i truncated at score level c. The key insight is that these truncated functions $\widehat{F}_{i,c}$ remain nondecreasing and submodular [Fujito, 2000]. Figure 10.4 illustrates this truncation concept. Let $\overline{F}_c(\mathcal{A}) = \frac{1}{m}\sum_i \widehat{F}_{i,c}(\mathcal{A})$ be their average value. Since nondecreasing submodular functions are closed under convex combinations, $\overline{F}_c$ is also submodular and nondecreasing. Furthermore, $F_i(\mathcal{A}) \geq c$ for all $1 \leq i \leq m$ if and only if $\overline{F}_c(\mathcal{A}) = c$. Hence, in order to determine whether some c is feasible for Problem (10.5), we need to determine whether there exists a set of size at most αk such that $\overline{F}_c(\mathcal{A}) = c$. Note, that due to nondecreasingness of $\overline{F}_c$ and the choice of c it holds that that $c = \overline{F}_c(\mathcal{V})$. We hence need to solve the following optimization problem:

$$\mathcal{A}^*_c = \operatorname*{argmin}_{\mathcal{A} \subseteq \mathcal{V}} |\mathcal{A}|, \quad \text{such that} \quad \overline{F}_c(\mathcal{A}) = \overline{F}_c(\mathcal{V}). \tag{10.6}$$

Algorithm 10.1: The greedy submodular partial cover (GPC) algorithm.

GPC ($\overline{F}_c$, c)
$\mathcal{A} \leftarrow \emptyset$;
while $\overline{F}_c(\mathcal{A}) < c$ **do**
  foreach $s \in \mathcal{V} \setminus \mathcal{A}$ **do** $\delta_s \leftarrow \overline{F}_c(\mathcal{A} \cup \{s\}) - \overline{F}_c(\mathcal{A})$;
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{\text{argmax}_s\, \delta_s\}$;
end

Problems of the form $\min_{\mathcal{A}} |\mathcal{A}|$ such that $F(\mathcal{A}) = F(\mathcal{V})$, where F is a nondecreasing submodular function, are called *submodular covering problems*. Since $\overline{F}_c$ satisfies these requirements, the MINCOVER$_c$ Problem (10.6) is an instance of such a submodular covering problem. While such problems are **NP**-hard in general [Feige, 1998], Wolsey [1982a] shows that the greedy algorithm, that starts with the empty set ($\mathcal{A} = \emptyset$) and iteratively adds the element s increasing the score the most until $F(\mathcal{A}) = F(\mathcal{V})$, achieves near-optimal performance on this problem. We can hence use the greedy algorithm applied to the truncated functions $\overline{F}_c$ as our approximate algorithm GPC, which is formalized in Algorithm 10.1. Using Wolsey's result and the observation that α can be chosen independently of the truncation threshold c, we find:

Lemma 10.2. *Given integral valued*[1] *nondecreasing submodular functions $F_1, \ldots, F_m$ and a (feasible) constant c, Algorithm 10.1 (with input $\overline{F}_c$) finds a set $\mathcal{A}_G$ such that $F_i(\mathcal{A}_G) \geq c$ for all i, and $|\mathcal{A}_G| \leq \alpha |\mathcal{A}_c^*|$, where $\mathcal{A}_c^*$ is an optimal solution to Problem* (10.6)*, and*

$$\alpha = 1 + \log\left(\max_{s \in \mathcal{V}} \sum_i F_i(\{s\})\right).$$

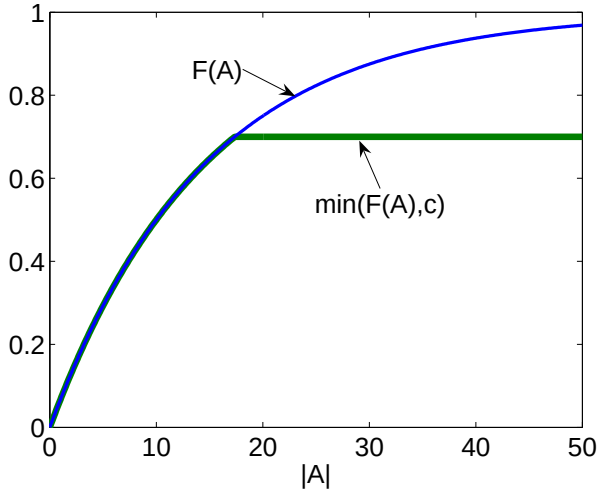


Figure 10.4: Truncating an objective function F preserves submodularity and nondecreasingness.

We can compute this approximation guarantee α for any given instance of the RSOS problem. Hence, if for a given value of c the greedy algorithm returns a set of size greater than αk, there cannot exist a solu-

[1]This bound is only meaningful for integral F_i, otherwise it could be arbitrarily improved by scaling the F_i. We relax the constraint on integrality of the F_i in Section 10.6.1.

Algorithm 10.2: The Submodular Saturation algorithm.

SATURATE $(F_1, \ldots, F_m, k, \alpha)$
$c_{\min} \leftarrow 0$; $c_{\max} \leftarrow \min_i F_i(\mathcal{V})$; $\mathcal{A}_{best} \leftarrow \emptyset$;
while $(c_{\max} - c_{\min}) \geq \frac{1}{m}$ **do**
  $c \leftarrow (c_{\min} + c_{\max})/2$;
  Define $\overline{F}_c(\mathcal{A}) \leftarrow \frac{1}{m} \sum_i \min\{F_i(\mathcal{A}), c\}$;
  $\widehat{\mathcal{A}} \leftarrow GPC(\overline{F}_c, c)$;
  if $|\widehat{\mathcal{A}}| > \alpha k$ **then**
    $c_{\max} \leftarrow c$;
  else
    $c_{\min} \leftarrow c$; $\mathcal{A}_{best} = \widehat{\mathcal{A}}$
  end
end

tion $\mathcal{A}'$ with $|\mathcal{A}'| \leq k$ with $F_i(\mathcal{A}') \geq c$ for all i. Thus, c is an upper bound to the RSOS Problem (10.3). We can use this argument to conduct the binary search discussed in Section 10.3.1 to find the optimal value of c. The binary search procedure maintains an interval $[c_{\min}, c_{\max}]$, initialized $[0, \min_i F_i(\mathcal{V})]$. At every iteration, we test the current center of the interval, $c = (c_{\min} + c_{\max})/2$, and check feasibility of c using the greedy algorithm. If c is feasible, we retain the current best feasible solution and set $c_{\min} = c$. If c is infeasible (which we detect by comparing the number of elements picked by the greedy algorithm with αk), we set $c_{\max} = c$.

We call Algorithm 10.2, which formalizes this procedure, the *submodular saturation algorithm* (SATURATE), as the algorithm considers the truncated objectives $\widehat{F}_{i,c}$, and chooses sets which *saturate* all these objectives. In the pseudo-code of Algorithm 10.2 we pass α as a parameter. Theorem 10.3 (given below) states that SATURATE, when applied with α chosen as in Lemma 10.2, is guaranteed to find a set which achieves worst-case score $\min_i F_i$ at least as high as the optimal solution, if we allow the set to be logarithmically (a factor α) larger than the optimal solution.

Theorem 10.3. *For any integer k,* SATURATE *finds a solution $\mathcal{A}_S$ such that*

$$\min_i F_i(\mathcal{A}_S) \geq \max_{|\mathcal{A}| \leq k} \min_i F_i(\mathcal{A}) \quad \textit{and} \quad |\mathcal{A}_S| \leq \alpha k,$$

for $\alpha = 1 + \log\left(\max_{s \in \mathcal{V}} \sum_i F_i(\{s\})\right)$. The total number of submodular function evaluations is

$$\mathcal{O}\left(|\mathcal{V}|^2 m \log\left(m \min_i F_i(\mathcal{V})\right)\right).$$

Note, that the algorithm still makes sense for any value of α. However, if $\alpha < 1 + \log\left(\max_{s \in \mathcal{V}} \sum_i F_i(\{s\})\right)$, the guarantee of Theorem 10.3 does not hold. As argued in Section 10.3.1, if we had an exact algorithm for submodular coverage, then we would set $\alpha = 1$, and SATURATE would return the optimal solution to the RSOS problem. Since, in our experience, the greedy algorithm for optimizing submodular functions works very effectively (*c.f.*, Chapter 6), in our experiments, we call SATURATE with $\alpha = 1$. This choice empirically performs very well, as demonstrated in Section 10.7.

If we apply SATURATE to the example problem described in Section 10.2, we would start with $c_{\max} = n$. Running the coverage algorithm (GPC) with $c = n/2$ would first pick element s_1 (or s_2), since $\overline{F}_c(\{s_1\}) = n/2$, and, next, pick s_2 (or s_1 resp.), hence finding the optimal solution.

The worst-case running time guarantee is quite pessimistic, and in practice the algorithm is much faster: Using a priority queue and lazy evaluations, Algorithm 10.1 can be sped up drastically. Lazy evaluations exploit the fact that, due to submodularity, the differences $\delta_s(\mathcal{A}) = \overline{F}_c(\mathcal{X}_{\mathcal{A}\cup s}) - \overline{F}_c(\mathcal{X}_{\mathcal{A}})$ that are computed by GPC are monotonically decreasing in $\mathcal{A}$, which allows to avoid a large number of function evaluations (*c.f.*Robertazzi and Schwartz 1989 for details). In addition, for many submodular functions F_i, such as the variance reduction, it is often cheaper to compute $\overline{F}_c(\mathcal{X}_{\mathcal{A}\cup s}) - \overline{F}_c(\mathcal{X}_{\mathcal{A}})$ instead of $\overline{F}_c(\mathcal{X}_{\mathcal{A}\cup s})$. This observation can be exploited to drastically speed up GPC. Furthermore, in practical implementations, one would stop GPC once $\alpha k + 1$ elements have been selected, which already proves that the optimal solution with k elements cannot achieve score c. Also, Algorithm 10.2 can be terminated once $c_{\max} - c_{\min}$ is sufficiently small; in our experiments, 10-15 iterations usually sufficed.

10.4 Hardness of bicriterion approximation

Guarantees of the form presented in Theorem 10.3 are often called *bicriterion* guarantees. Instead of requiring that the obtained objective score is close to the optimal score *and all* constraints are exactly met, a bicriterion guarantee requires a bound on the suboptimality of the objective, as well as bounds on how much the constraints are violated. Theorem 10.1 showed that – unless $\mathbf{P} = \mathbf{NP}$ – no approximation guarantees can be obtained which do not violate the constraint on the cost k, thereby necessitating the bricriterion analysis.

One might ask, whether the guarantee on the size of the set, α, can be improved. Unfortunately, this is not likely, as the following result shows:

Theorem 10.4. *If there were a polynomial time algorithm which, for any integer k, is guaranteed to find a solution $\mathcal{A}_S$ such that $\min_i F_i(\mathcal{A}_S) \geq \max_{|\mathcal{A}|\leq k} \min_i F_i(\mathcal{A})$ and $|\mathcal{A}_S| \leq \beta k$, where $\beta \leq (1-\varepsilon)(1 + \log \max_{s\in\mathcal{V}} \sum_i F_i(\{s\}))$ for some fixed $\varepsilon > 0$, then $\mathbf{NP} \subseteq \mathrm{DTIME}(n^{\log\log n})$.*

Hereby, $\mathrm{DTIME}(n^{\log\log n})$ is a class of deterministic, slightly superpolynomial (but sub-exponential) algorithms [Feige, 1998]; the inclusion $\mathbf{NP} \subseteq \mathrm{DTIME}(n^{\log\log n})$ is considered unlikely [Feige, 1998]. Taken together, Theorem 10.1 and Theorem 10.4, provide strong theoretical evidence that SATURATE achieves best possible theoretical guarantees for the problem of maximizing the minimum over a set of submodular functions.

10.5 Examples of robust submodular observation selection problems

We now demonstrate that many important machine learning problems can be phrased as RSOS problems. Section 10.7 provides more details and experimental results for these domains.

10.5.1 Minimizing the maximum Kriging variance

Consider a Gaussian Process (GP) [*c.f.* Rasmussen and Williams, 2006] $\mathcal{X}_\mathcal{V}$ defined over a finite set of locations (indices) $\mathcal{V}$. Hereby, $\mathcal{X}_\mathcal{V}$ is a set of random variables, one variable $\mathcal{X}_s$ for each location $s \in \mathcal{V}$. Given a set of locations $\mathcal{A} \subseteq \mathcal{V}$ which we observe, we can compute the predictive distribution $P(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$, i.e., the distribution of the variables $\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}$ at the unobserved locations $\mathcal{V} \setminus \mathcal{A}$, conditioned on the measurements at the selected locations, $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$. Let $\sigma^2_{s|\mathcal{A}}$ be the residual variance

after making observations at $\mathcal{A}$. Let $\Sigma_{\mathcal{A}\mathcal{A}}$ be the covariance matrix of the measurements at the chosen locations $\mathcal{A}$, and $\Sigma_{s\mathcal{A}}$ be the vector of cross-covariances between the measurements at s and $\mathcal{A}$. Then, the predictive variance (often called Kriging variance in the geostatistics literature), given by

$$\sigma^2_{s|\mathcal{A}} = \sigma^2_s - \Sigma_{s\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}s},$$

depends only on the set $\mathcal{A}$, and *not* on the observed values $\mathbf{x}_{\mathcal{A}}$[2]. As argued in Section 10.1, an often (especially in the case of nonstationary phenomena) appropriate criterion is to select locations $\mathcal{A}$ such that the maximum marginal variance is as small as possible, i.e., we want to select a subset $\mathcal{A}^* \subseteq \mathcal{V}$ of locations to observe such that

$$\mathcal{A}^* = \operatorname*{argmin}_{|\mathcal{A}|\leq k} \max_{s\in\mathcal{V}} \sigma^2_{s|\mathcal{A}}. \tag{10.7}$$

Let us assume for now that the a priori variance σ^2_s is constant for all locations s (in Section 10.6, we show how our approach generalizes to non-constant marginal variances). Furthermore, let us define the *variance reduction* $F_s(\mathcal{A}) = \sigma^2_s - \sigma^2_{s|\mathcal{A}}$. Solving Problem (10.7) is then equivalent to maximizing the minimum variance reduction over all locations s. For a particular location s, Das and Kempe [2008] show that the variance reduction F_s (often) is a nondecreasing submodular function. Hence the problem

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} \min_{s\in\mathcal{V}} F_s(\mathcal{A}) = \operatorname*{argmax}_{|\mathcal{A}|\leq k} \min_{s\in\mathcal{V}} \sigma^2_s - \sigma^2_{s|\mathcal{A}}$$

is an instance of the RSOS problem.

10.5.2 Variable selection under parameter uncertainty

Consider an application, where we want to diagnose a failure of a complex system, by performing a number of tests. We can model this problem by using a set of discrete random variables $\mathcal{X}_{\mathcal{V}} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ indexed by $\mathcal{V} = \{1, \ldots, n\}$, which model both the hidden state of the system and the outcomes of the diagnostic tests. The interaction between these variables is modeled by a joint distribution $P(\mathcal{X}_{\mathcal{V}} \mid \theta)$ with parameters θ. In Chapters 6 and 7, we have seen that many variable selection problems can be formulated as the problem of optimizing a submodular sensing quality function (measuring, e.g., the information gain $I(\mathcal{X}_{\mathcal{U}}, \mathcal{X}_{\mathcal{A}})$ with respect to some variables of interest $\mathcal{U}$, or the mutual information $I(\mathcal{X}_{\mathcal{A}}; \mathcal{X}_{\mathcal{V}\setminus\mathcal{A}})$ between the observed and unobserved variables, etc.). However, the informativeness of a chosen set $\mathcal{A}$ typically depends on the particular parameters θ, and these parameters might be uncertain. In some applications, it might not be reasonable to impose a prior distribution over θ, and we may want to perform well even under the worst-case parameters. In these cases, we can associate, with each parameter setting θ, a different submodular objective function F_θ, for example,

$$F_\theta(\mathcal{A}) = I(\mathcal{X}_{\mathcal{A}}; \mathcal{X}_{\mathcal{U}} \mid \theta),$$

and we might want to select a set $\mathcal{A}$ which simultaneously performs well for all possible parameter values. In practice, we can discretize the set of possible parameter values θ (for example around a 95% confidence interval estimated from initial data) and optimize the worst case F_θ over the resulting discrete set of parameters.

[2]This independence is a particular property of the Gaussian distribution. It implies that there is no benefit of sequentially selecting observations. For more details see Chapter 14.

10.5.3 Robust experimental designs

Another application is experimental design under nonlinear dynamics [Flaherty et al., 2006]. The goal is to estimate a set of parameters θ of a nonlinear function $y = f(\mathbf{x}, \theta) + w$, by providing a set of experimental stimuli $\mathbf{x}$, and measuring the (noisy) response y. In many cases, experimental design for linear models (where $y = A(\mathbf{x})^T\theta + w$ with Gaussian noise w) can be efficiently solved by semidefinite programming [Boyd and Vandenberghe, 2004]. In the nonlinear case, a common approach [*c.f.* Chaloner and Verdinelli, 1995] is to *linearize* f around an initial parameter estimate θ_0, i.e.,

$$y = f(\mathbf{x}, \theta_0) + V(\mathbf{x})(\theta - \theta_0) + w, \tag{10.8}$$

where $V(\mathbf{x})$ is the Jacobian of f with respect to the parameters θ, evaluated at θ_0. Subsequently, a *locally-optimal* design is sought, which is optimal for the linear design Problem (10.8) for initial parameter estimates θ_0. Flaherty et al. [2006] show that the efficiency of such a locally optimal design can be very sensitive with respect to the initial parameter estimates θ_0. Consequently, they develop an efficient semidefinite program (SDP) for E-optimal design (i.e., the goal is to minimize the maximum eigenvalue of the error covariance) which is robust against perturbations of the Jacobian V. However, it might be more natural to directly consider robustness with respect to perturbation of the initial parameter estimates θ_0, around which the linearization is performed. We show how to find (Bayesian A-optimal) designs which are robust against uncertainty in these parameter estimates. In this setting, the objectives $F_{\theta_0}(\mathcal{A})$ are the reductions of the trace of the parameter covariance,

$$F_{\theta_0}(\mathcal{A}) = \operatorname{tr}\left(\Sigma_{\theta}^{(\theta_0)}\right) - \operatorname{tr}\left(\Sigma_{\theta|\mathcal{A}}^{(\theta_0)}\right),$$

where $\Sigma^{(\theta_0)}$ is the joint covariance of observations and parameters after linearization around θ_0; thus, F_{θ_0} is the sum of marginal parameter variance reductions, which are (often) individually nondecreasing and submodular [Das and Kempe, 2008], and so F_{θ_0} is nondecreasing and submodular as well. Hence, in order to find a robust design, we maximize the minimum variance reduction, where the minimum is taken over (a discretization into a finite subset of) all initial parameter values θ_0.

10.5.4 Sensor placement for outbreak detection

Another class of examples are outbreak detection problems on graphs, such as the ones studied in Chapter 8. Here, we are given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and a phenomenon spreading dynamically over the graph. We define a set of *intrusion scenarios* $\mathcal{I}$; each scenario $i \in \mathcal{I}$ models an outbreak (e.g., spreading of contamination) starting from a given node $s \in \mathcal{V}$ in the network. By placing sensors at a set of locations $\mathcal{A} \subseteq \mathcal{V}$, we can detect such an outbreak, and thereby minimize the adverse effects on the network.

More formally, for each possible outbreak scenario $i \in \mathcal{I}$ and for each node $v \in \mathcal{V}$ we define the detection time $T_i(v)$ as the time when the outbreak affects node v (and $T_i(v) = \infty$ if node v is never affected). We furthermore define a penalty function $\pi_i(t)$ which models the penalty incurred for detecting outbreak i at time t. We require $\pi_i(t)$ to be monotonically nondecreasing in t (i.e., we never prefer late over early detection), and bounded above by $\pi_i(\infty) \in \mathbb{R}$. Our goal is to minimize the worst-case penalty: We extend π_i to observation sets $\mathcal{A}$ as $\pi_i(\mathcal{A}) = \pi_i\left(\min_{s\in\mathcal{A}} T_i(s)\right)$. Then, our goal is to solve

$$\mathcal{A}^* = \operatorname*{argmin}_{|\mathcal{A}|\leq k} \max_{i\in\mathcal{I}} \pi_i(\mathcal{A}).$$

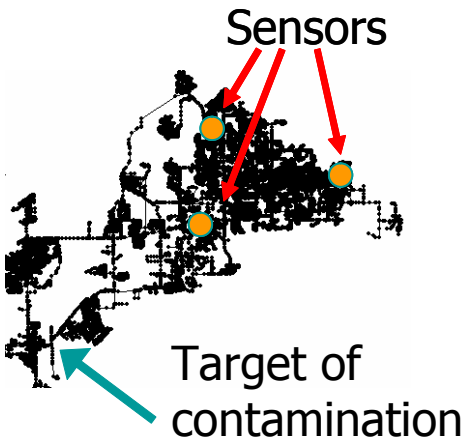


Figure 10.5: Securing a municipal water distribution network against contaminations performed under knowledge of the sensor placement is another instance of the RSOS problem.

Equivalently, we can define the *penalty reduction* $F_i(\mathcal{A}) = \pi_i(\infty) - \pi_i(\mathcal{A})$. Clearly, $F_i(\emptyset) = 0$, F_i is nondecreasing. In Chapter 8, we showed that F_i is also guaranteed to be submodular. For now, let us assume that $\pi_i(\infty)$ is constant for all i (we will relax this assumption in Section 10.6.2). Our goal in sensor placement is then to select a set of sensors $\mathcal{A}$ such that the minimum penalty reduction is as large as possible, i.e., we want to select

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} \min_{i\in\mathcal{I}} F_i(\mathcal{A}).$$

In other words, an adversary observes our sensor placement $\mathcal{A}$, and then decides on an intrusion i for which our sensing quality $F_i(\mathcal{A})$ is as small as possible. Hence, our goal is to find a placement $\mathcal{A}$ which performs well against such an adversarial opponent.

10.5.5 Robustness against sensor failures and feature deletion

Another interesting instance of the RSOS problem arises in the context of robust sensor placements. For example, in the outbreak detection problem, sensors might *fail*, due to hardware problems or manipulation by an adversary. We can model this problem in the following way: Consider the case where all sensors at a subset $\mathcal{B} \subseteq \mathcal{V}$ of locations fail. Given a submodular function F (e.g., the utility for placing a set of sensors), and the set $\mathcal{B} \subseteq \mathcal{V}$ of failing sensors, we can define a new function $F_{\mathcal{B}}(\mathcal{A}) = F(\mathcal{A} \setminus \mathcal{B})$, corresponding to the (reduced) sensing quality of placement $\mathcal{A}$ after the sensor failures. It is easy to show that if F is nondecreasing and submodular, so is $F_{\mathcal{B}}$. Hence, the problem of optimizing sensor placements which are robust to sensor failures results in a problem of simultaneously maximizing a collection of submodular functions, e.g., for the worst-case failure of $k' < k$ sensors we solve

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} \min_{|\mathcal{B}|\leq k'} F_{\mathcal{B}}(\mathcal{A}).$$

We can also combine the optimization against adversarial contamination scenarios as discussed in Section 10.5.3 with adversarial sensor failures, and optimize

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} \min_{i\in\mathcal{I}} \min_{|\mathcal{B}|\leq k'} F_i(\mathcal{A} \setminus \mathcal{B}).$$

Another important problem in machine learning is feature selection. In feature selection, the goal is to select a subset of features which are informative with respect to, e.g., a given classification task. One objective frequently considered is the problem of selecting a set of features which maximize the information gained about the class variable $\mathcal{X}_Y$ after observing the features $\mathcal{X}_\mathcal{A}$, $F(\mathcal{A}) = H(\mathcal{X}_Y) - H(\mathcal{X}_Y \mid \mathcal{X}_\mathcal{A})$, where H denotes the Shannon entropy. As shown in Chapter 7, in a large class of graphical models, the information gain $F(\mathcal{A})$ is in fact a submodular function. Now we can consider a setting, where an adversary can delete features which we selected (as considered, e.g., by Globerson and Roweis 2006). The problem of selecting features robustly against such arbitrary deletion of, e.g., m features, is hence equivalent to the problem of maximizing $\min_{|\mathcal{B}|\leq m} F_\mathcal{B}(\mathcal{A})$, where $\mathcal{B}$ are the deleted features.

Improved guarantees for sensor failures

As discussed above, in principle, we could find a placement robust to single sensor failures by using SATURATE to (approximately) solve

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} \min_s F_s(\mathcal{A}).$$

However, since $|\mathcal{V}|$ can be very large, and the approximation guarantee α depends logarithmically on $|\mathcal{V}|$, such a direct approach might not be desirable. We can improve the guarantee from $\mathcal{O}(\log|\mathcal{V}|)$ to $\mathcal{O}(\log(k\log|\mathcal{V}|))$, which typically is much tighter, if $k \ll |\mathcal{V}|/\log|\mathcal{V}|$ (i.e., we place far fewer sensors than we have possible sensor locations). We can improve the approximation guarantee drastically by noticing that $F_s(\mathcal{A}) = F(\mathcal{A})$ if $s \notin \mathcal{A}$. Hence,

$$\overline{F}_c(\mathcal{A}) = \frac{|\mathcal{V}| - |\mathcal{A}|}{|\mathcal{V}|} \min\{F(\mathcal{A}), c\} + \frac{1}{|\mathcal{V}|} \sum_{s\in\mathcal{A}} \widehat{F}_{s,c}(\mathcal{A}).$$

We can replace this objective by a new objective function,

$$\overline{F}'_c(\mathcal{A}) = \frac{k' - |\mathcal{A}|}{k'} \min\{F(\mathcal{A}), c\} + \frac{1}{k'} \sum_{s\in\mathcal{A}} \widehat{F}_{s,c}(\mathcal{A})$$

for some constant $\hat{k}$ to be specified below. This modified objective is still nondecreasing and submodular when restricted to sets of size at most $\hat{k}$. It still holds that, for all subsets $|\mathcal{A}| \leq \hat{k}$, that

$$\overline{F}'_c(\mathcal{A}) \geq c \Leftrightarrow F_s(\mathcal{A}) \geq c \text{ for all } s \in \mathcal{V}.$$

How large should we choose $\hat{k}$? We have to choose $\hat{k}$ large enough such that SATURATE will never choose sets larger than $\hat{k}$. A sufficient choice for $\hat{k}$ is hence $\lceil \alpha k \rceil$, where $\alpha = 1 + \log\left(|\mathcal{V}| \max_{s\in\mathcal{V}} F(\{s\})\right)$. For this choice of $\hat{k}$, our new approximation guarantee will be

$$\begin{aligned}\alpha' &= 1 + \log\left(\alpha k \max_{s\in\mathcal{V}} F(\{s\})\right) = 1 + \log\left(\left(1 + \log\left(|\mathcal{V}| \max_{s\in\mathcal{V}} F(\{s\})\right)\right) k \max_{s\in\mathcal{V}} F(\{s\})\right) \\ &\leq 1 + 2\log\left(k\log(|\mathcal{V}|) \max_{s\in\mathcal{V}} F(\{s\})\right)\end{aligned}$$

Hence, for the new objective $\overline{F}'_c$, we get a tighter approximation guarantee,

$$\alpha' = 1 + 2\log\left(k\log(|\mathcal{V}|) \max_{s\in\mathcal{V}} F(\{s\})\right),$$

which now depends logarithmically on $k \log |\mathcal{V}|$, instead of the number of available locations $|\mathcal{V}|$. Note that this same approach can also provide tighter approximation guarantees in the case of multiple sensor failures.

10.6 Extensions

We now show how some of the assumptions made in our presentation above can be relaxed. We also discuss several extensions, allowing more complex cost functions, and the tradeoff between worst-case and average-case scores.

10.6.1 Non-integral objectives

In our analysis of SATURATE (Section 10.3), we have assumed, that each of the objective functions F_i only take values in the positive integers. However, most objective functions of interest in observation selection (such as those discussed in Section 10.5) typically do not meet this assumption. If the F_i take on rational numbers, we can scale the objectives by multiplying by their common denominator.

If we allow small additive approximation error (i.e., are indifferent if the approximate solution differs from the optimal solution in low order bits), we can also approximate the values assumed by the functions F_i by their highest order bits. In this case, we replace the functions $F_i(\mathcal{A})$ by the approximations

$$F'_i(\mathcal{A}) = \frac{\lceil 2^j F_i(\mathcal{A}) \rceil}{2^j}.$$

By construction, $F'_i(\mathcal{A}) \leq F_i(\mathcal{A}) \leq F'_i(\mathcal{A})(1 + 2^{-j})$, i.e., F'_i is within a factor of $(1 + 2^{-j})$ of F_i. Also, $2^j F'_i(\mathcal{A})$ is integral. However, $F'_i(\mathcal{A})$ is not guaranteed to be submodular. Nevertheless, an analysis similar to the one presented in Chapter 6 can be used to bound the effect of this approximation on the theoretical guarantees α obtained by the algorithm, which will now scale linearly with the number j of high order bits considered. In practice, as we show in Section 10.7, SATURATE provides state-of-the-art performance, even without rounding the objectives to the highest order bits.

10.6.2 Non-constant thresholds

Consider the example of minimizing the maximum variance in Gaussian Process regression. Here, the $F_i(\mathcal{A}) = \sigma_i^2 - \sigma_{i|\mathcal{A}}^2$ denote the variance reductions at location i. However, rather than guaranteeing that $F_i(\mathcal{A}) \geq c$ for all i (which, in this example, means that the *minimum variance reduction* is c), we want to guarantee that $\sigma_{i|\mathcal{A}}^2 \leq c$ for all i. We can easily adapt our approach to handle this case: Instead of defining $\widehat{F}_{i,c}(\mathcal{A}) = \min\{F_i(\mathcal{A}), c\}$, we define $\widehat{F}_{i,c}(\mathcal{A}) = \min\{F_i(\mathcal{A}), \sigma_i^2 - c\}$, and then again perform binary search over c, but searching for the smallest c instead. The algorithm, using objectives modified in this way, will bear the same approximation guarantees.

10.6.3 Non-uniform observation costs

We can extend SATURATE to the setting where different observations have different costs. In the spatial monitoring setting for example, certain locations might be more expensive to acquire a measurement from.

Suppose a cost function $\mathrm{C} : \mathcal{V} \to \mathbb{R}^+$ assigns each element $s \in \mathcal{V}$ a positive cost $\mathrm{C}(s)$; the cost of a set of observations is then $\mathrm{C}(\mathcal{A}) = \sum_{s \in \mathcal{A}} \mathrm{C}(s)$. The problem is to find $\mathcal{A}^* = \operatorname{argmax}_{\mathcal{A} \subset \mathcal{V}} \min_i F_i(\mathcal{A})$ subject to $\mathrm{C}(\mathcal{A}) \leq B$, where $B > 0$ is a *budget* we can spend on making observations. In this case, we use the rule

$$\delta_s \leftarrow \frac{\overline{F}_c(\mathcal{A} \cup \{s\}) - \overline{F}_c(\mathcal{A})}{\mathrm{C}(s)}$$

in Algorithm 10.1. For this modified algorithm, Theorem 10.3 still holds, with $|\mathcal{A}|$ replaced by $\mathrm{C}(\mathcal{A})$ and k replaced by B. This more general result holds, since the analysis of the greedy algorithm for submodular covering of Wolsey [1982a], which we used to prove Lemma 10.2, applies to the more general setting of non-uniform cost functions.

10.6.4 Handling more complex cost functions

So far, we considered problems where we are given an *additive* cost function $\mathrm{C}(\mathcal{A})$ over the possible sets $\mathcal{A}$ of observations. In some applications, more complex cost functions arise. For example, when placing wireless sensor networks, the placements $\mathcal{A}$ should not only be informative (i.e., $F_i(\mathcal{A})$ should be high for all sensing quality functions F_i), but the placement should also have *low communication cost*. In Chapter 11 we describe such an approach, where the cost $\mathrm{C}(\mathcal{A})$ measures the *expected number of retransmissions* required for sending messages across an optimal routing tree connecting the sensors $\mathcal{A}$. Formally, the observations s are considered to be nodes in a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, with edge weights $w(e)$ for each edge $e \in \mathcal{E}$. The cost $\mathrm{C}(\mathcal{A})$ is the cost of a minimum Steiner Tree [*c.f.*, Vazirani, 2003] connecting the observations $\mathcal{A}$ in the graph $\mathcal{G}$.

More generally, we want to solve problems of the form

$$\operatorname*{argmax}_{\mathcal{A}} \min_i F_i(\mathcal{A}) \text{ subject to } \mathrm{C}(\mathcal{A}) \leq B, \tag{10.9}$$

where $\mathrm{C}(\mathcal{A})$ is a complex cost function. The key insight of the SATURATE algorithm is that the non-submodular robust optimization problem can be approximately solved by solving a submodular covering problem. In the case where $\mathrm{C}(\mathcal{A}) = |\mathcal{A}|$ this problem requires solving (10.6). More generally, we can apply SATURATE to any problem where we can (approximately) solve

$$\mathcal{A}_c = \operatorname*{argmin}_{\mathcal{A} \subseteq \mathcal{V}} g(\mathcal{A}), \quad \text{such that} \quad \overline{F}_c(\mathcal{A}) = c. \tag{10.10}$$

Problem (10.10) can be (approximately) solved for a variety of cost functions, such as those arising from communication and path constraints (*c.f.*, Chapter 11, Meliou et al. [2007], Singh et al. [2007]).

Let us summarize our analysis as follows:

Proposition 10.5. *Assume we have an algorithm which, given a nondecreasing submodular function F and a cost function* C*, returns a solution $\mathcal{A}'$ such that $F(\mathcal{A}') = F(\mathcal{V})$ and*

$$\mathrm{C}(\mathcal{A}') \leq \alpha_F \min_{\mathcal{A}: F(\mathcal{A}) = F(\mathcal{V})} \mathrm{C}(\mathcal{A}),$$

where α_F depends on the function F. SATURATE*, using this covering algorithm, can obtain a solution $\mathcal{A}_S$ to the* RSOS *problem such that*

$$\min_i F_i(\mathcal{A}_S) \geq \max_{\mathrm{C}(\mathcal{A}) \leq B} \min_i F_i(\mathcal{A}),$$

and

$$\mathrm{C}(\mathcal{A}_S) \leq \alpha_{\overline{F}} B,$$

where $\alpha_{\overline{F}}$ *is the approximation factor of the covering algorithm, when applied to* $\overline{F} = \frac{1}{m}\sum_i F_i$.

Note that the formalism developed in this section also allows to handle robust versions of combinatorial optimization problems such as the *Knapsack* [*c.f.* Martello and Toth, 1990], Orienteering [*c.f.* Blum et al., 2003, Laporte and Martello, 1990] and *Budgeted Steiner Tree* [*c.f.* Johnson et al., 2000] problems. In these problems, instead of a general *submodular* objective function, the special case of a *modular* (additive) function F is optimized:

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathrm{C}(\mathcal{A}) \leq B} F(\mathcal{A}).$$

The problems differ only in the choice of the complex cost function. In *Knapsack* for example, C is additive, in the *Budgeted Steiner Tree* problem, $\mathrm{C}(\mathcal{A})$ is the cost of a minimum Steiner tree connecting the nodes $\mathcal{A}$ in a graph, and in *Orienteering*, $\mathrm{C}(\mathcal{A})$ is the cost of a shortest path connecting the nodes $\mathcal{A}$ in a graph. In practice, often the sensing quality function $F(\mathcal{A})$ is not exactly known, and a solution is desired which is robust against worst-case choice of the sensing quality function. Since modular functions are a special case of submodular functions, such problems can be approximately solved using Proposition 10.5.

10.6.5 Trading off average-case and worst-case scores

In some applications, optimizing the worst-case score $F_{wc}(\mathcal{A}) = \min_i F_i(\mathcal{A})$ might be a too pessimistic approach. On the other hand, ignoring the worst-case and only optimizing the average-case (the expected score under a distribution over the objectives) $F_{ac}(\mathcal{A}) = \frac{1}{m}\sum_i F_i(\mathcal{A})$ might be too optimistic. In fact, in Section 10.7 we show that optimizing the average-case score F_{ac} can often lead to drastically poor worst-case scores. In general, we might be interested in solutions, which perform well both in the average- and worst-case scores.

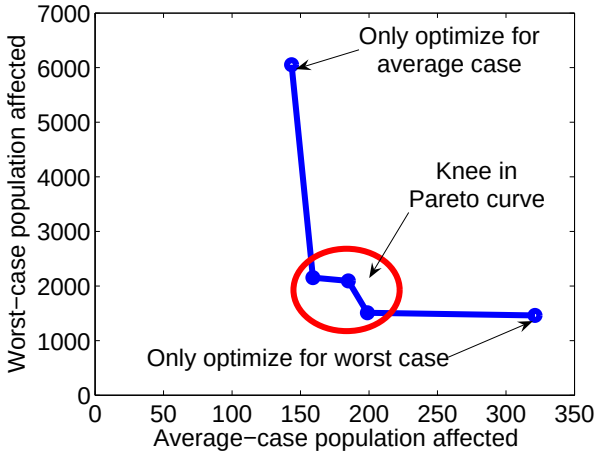


Figure 10.6: Tradeoff curve for simultaneously optimizing the average- and worst-case score in the water distribution network monitoring application. Notice the knee in the tradeoff curve, indicating that by performing multi-criterion optimization, solutions performing well for both average- and worst-case scores can be obtained.

Formally, we can define a multicriterion optimization problem, where we intend to optimize the *vector* $[F_{ac}(\mathcal{A}), F_{wc}(\mathcal{A})]$. In this setting, we can only hope for *Pareto-optimal* solutions [*c.f.* Boyd and Vandenberghe, 2004, in the context of convex functions]. A set $\mathcal{A}^*$, $|\mathcal{A}^*| \leq k$ is called *Pareto-optimal*, if it is not *dominated*, i.e., there does not exist another set $\mathcal{B}$, $|\mathcal{B}| \leq k$ with $F_{ac}(\mathcal{B}) > F_{ac}(\mathcal{A}^*)$ and $F_{wc}(\mathcal{B}) \geq F_{wc}(\mathcal{A}^*)$ (or $F_{ac}(\mathcal{B}) \geq F_{ac}(\mathcal{A}^*)$ and $F_{wc}(\mathcal{B}) > F_{wc}(\mathcal{A}^*)$).

One possible approach to find such Pareto-optimal solutions is constrained optimization[3]: for a specified value of c_{ac}, we desire a solution to

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} F_{wc}(\mathcal{A}) \text{ such that } F_{ac}(\mathcal{A}) \geq c_{ac}. \tag{10.11}$$

By specifying different values of c_{ac} in (10.11), we would obtain different Pareto-optimal solutions[4]. Figure 10.6 presents an example of several Pareto-optimal solutions, based on data from the outbreak detection problem (Details will be discussed in Section 10.7.3). This curve shows that, using the techniques described below, multicriterion solutions can be found which combine the advantages of worst-case and average-case solutions.

We can modify SATURATE to solve Problem (10.11) in the following way. Let us again assume we know the optimal value c_{wc} achievable for Problem (10.11). Then, Problem (10.11) is equivalent to solving

$$\mathcal{A}^* = \operatorname*{argmin}_{\mathcal{A}} |\mathcal{A}| \text{ subject to } F_{wc}(\mathcal{A}) \geq c_{wc} \text{ and } F_{ac}(\mathcal{A}) \geq c_{ac}. \tag{10.12}$$

Now, using our notation from Section 10.3, this problem is again equivalent to

$$\mathcal{A}^* = \operatorname*{argmin}_{\mathcal{A}} |\mathcal{A}| \text{ subject to } \overline{F}_{c_{wc},c_{ac}} = c_{wc} + c_{ac}, \tag{10.13}$$

where

$$\overline{F}_{c_{wc},c_{ac}}(\mathcal{A}) = \overline{F}_{c_{wc}}(\mathcal{A}) + \min\{F_{ac}(\mathcal{A}), c_{ac}\}.$$

Note that $\overline{F}_{c_{wc},c_{ac}}$ is a submodular function, and hence (10.13) is a submodular covering problem, which can be approximately solved using the greedy algorithm.

For any choice of c_{ac}, we can find the optimal value of c_{wc} by performing binary search on c_{wc}. We summarize our analysis in the following Theorem:

Theorem 10.6. *For any integer k and constraint c_{ac},* SATURATE *finds a solution $\mathcal{A}_S$ (if it exists) such that*

$$F_{wc}(\mathcal{A}_S) \geq \max_{|\mathcal{A}| \leq k, F_{ac}(\mathcal{A}) \geq c_{ac}} F_{wc}(\mathcal{A}),$$

$F_{ac}(\mathcal{A}_S) \geq c_{ac}$, and $|\mathcal{A}_S| \leq \alpha k$, for $\alpha = 1 + \log\left(2 \max_{s \in \mathcal{V}} \sum_i F_i(\{s\})\right)$. Each such solution $\mathcal{A}_S$ is approximately *Pareto-optimal, i.e., there does not exist a set $\mathcal{B}$, $|\mathcal{B}| \leq k$ such that $\mathcal{B}$ dominates $\mathcal{A}_S$. The total number of submodular function evaluations is $\mathcal{O}\left(|\mathcal{V}|^2 m \log(\sum_i F_i(\mathcal{V}))\right)$.*

[3]Another approach is *scalarization*, where we optimize $F_\lambda(\mathcal{A}) = \lambda F_{wc}(\mathcal{A}) + (1-\lambda) F_{ac}(\mathcal{A})$ for some λ, $0 < \lambda < 1$. SATURATE can be modified to handle such scalarized objectives as well (*c.f.*Section 10.6.5).

[4]In fact, *all* Pareto-optimal solutions can be found in this way [Papadimitriou and Yannakakis, 2000].

Scalarization

In the scalarized objective version of the problem we optimize the objective

$$F_\lambda(\mathcal{A}) = \lambda F_{wc}(\mathcal{A}) + (1-\lambda)F_{ac}(\mathcal{A})$$

for a fixed parameter $\lambda \in [0,1]$. The scalarization parameter λ can be though of as a prior: with λ probability we expect to face an adversarial objective, while with probability $1-\lambda$ we expect an objective drawn from the uniform distribution[5] F_{ac}.

It turns out that this formulation is equivalent to facing an adversary who, rather than being able to pick any F_i arbitrarily, instead chooses a probability distribution on the F_i from some convex set of probability distributions $\mathcal{P}_{\text{wc}}$. If $\mathcal{P}_{\text{wc}} = \Delta(m)$, the set of all possible probability distributions on the m functions F_i, then we recover the fully adversarial problem as the adversary can choose a probability distribution that puts probability 1 on the best response to our chosen sensor placements:

$$F_{wc}(\mathcal{A}) = \min_{p \in \Delta(m)} \sum_i p_i F_i(\mathcal{A}) \tag{10.14}$$

The scalarized multi-criterion objective F_λ corresponds to a more interesting convex set,

$$\mathcal{P}^\lambda_{\text{wc}} = \left\{ q \mid q_i = \lambda p_i + \frac{1-\lambda}{m} \text{ for some } p \in \Delta(m) \right\},$$

which can be derived as follows:

$$\begin{aligned}
F_\lambda(\mathcal{A}) &= \lambda F_{wc}(\mathcal{A}) + (1-\lambda)F_{ac}(\mathcal{A}) \\
&= \lambda \min_{p \in \Delta(m)} \sum_i p_i F_i(\mathcal{A}) + (1-\lambda) \sum_i \frac{1}{m} F_i(\mathcal{A}) && \text{by (10.14)} \\
&= \min_{p \in \Delta(m)} \sum_i \left(\lambda p_i + \frac{1-\lambda}{m} \mathbf{1} \right) F_i(\mathcal{A}) \\
&= \min_{q \in \mathcal{P}^\lambda_{\text{wc}}} \sum_i q_i F_i(\mathcal{A}).
\end{aligned}$$

It is straightforward to show $\mathcal{P}^\lambda_{\text{wc}} = \{q \mid q_i \geq (1-\lambda)/m,\ \sum_i q_i = 1\}$, and so the scalarized multi-criterion objective is equivalent to solving the adversarial problem where the adversary must put at least probability $(1-\lambda)/m$ on every scenario.

In order to solve this problem, we recast F_λ again as

$$F_\lambda(\mathcal{A}) = \sum_i \left(\lambda F_i(\mathcal{A}) + (1-\lambda)F_{ac}(\mathcal{A}) \right).$$

The functions $F^\lambda_i = \lambda F_i(\mathcal{A}) + (1-\lambda)F_{ac}(\mathcal{A})$ are submodular, and so we can solve the scalarized version of the multi-criterion optimization by running an unmodified SATURATE algorithm on the set of objective functions F^λ_i.

In fact, this algorithmic technique can be extended to an arbitrary convex set of probability distributions $\mathcal{P}_{\text{wc}}$ with a finite number of extreme points. An extreme point of $\mathcal{P}_{\text{wc}}$ is a distribution that cannot be

[5] In fact, this argument easily generalizes to an arbitrary fixed distribution on the F_i

expressed as a convex combination of other distributions in $\mathcal{P}_{\text{wc}}$. Since the adversary knows our choice $\mathcal{A}$, the optimization over $\mathcal{P}_{\text{wc}}$ to find the best-response is the optimization of a linear objective over a bounded convex set, and so there always exists an extreme point that is optimal.

An extreme point q is a distribution over the original F_i, and in fact corresponds exactly to the submodular expected cost function

$$\sum_i q_i F_i(\mathcal{A}).$$

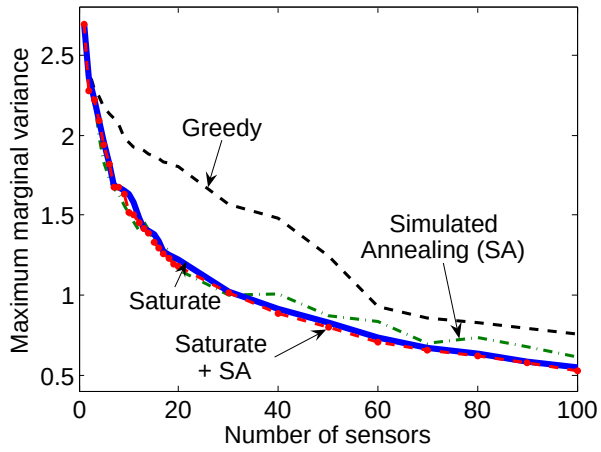
Hence, if we have a finite number of extreme points we can solve the set-selection problem against an adversary constrained to play from $\mathcal{P}_{\text{wc}}$ by running SATURATE on the set of derived expected-cost functions corresponding to the extreme points.

10.7 Experimental results

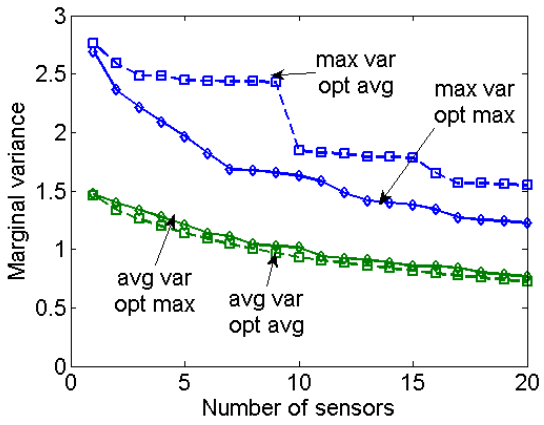
10.7.1 Minimizing the maximum Kriging variance

First, we use SATURATE to select observations in a GP to minimize the maximum posterior variance (*c.f.*Section 10.5.1). We consider three data sets: [T] temperature data from a deployment of 52 sensors at Intel Research Berkeley, [P] Precipitation data from the Pacific Northwest of the United States [Widmann and Bretherton, 1999] and [L] temperature data from the NIMS sensor node [Harmon et al., 2006] deployed at a lake near the University of California, Merced. For the three monitoring problems, [T], [P], and [L], we discretize the space into 46, 167 and 86 locations each, respectively. For [T], we consider the empirical covariance matrix of temperature sensor measurements obtained over a period of 5 days. For [P], we consider the empirical covariance of 50 years of data, which we preprocessed as described in Chapter 6. For [L], we train a nonstationary Gaussian Process using data from a single scan of the lake by the NIMS sensor node, using a method described in more detail in Chapter 14.

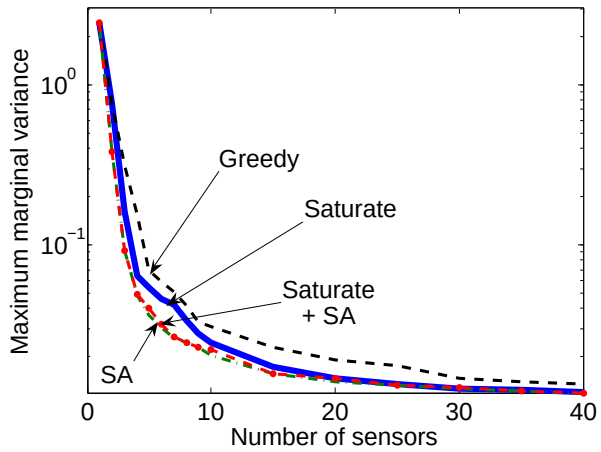
In the geostatistics literature, the predominant choice of optimization algorithms for selecting locations in a GP to minimize the (maximum and average) predictive variance are carefully tuned local search procedures, prominently simulated annealing (*c.f.*Sacks and Schiller 1988, van Groenigen and Stein 1998, Wiens 2005). We compare our SATURATE algorithm against a state-of-the-art implementation of such a simulated annealing (SA) algorithm, first proposed by Sacks and Schiller [1988]. We use an optimized implementation described recently by Wiens [2005]. This algorithm has 7 parameters which need to be tuned, describing the annealing schedule, distribution of iterations among several inner loops, etc. We use the parameter settings as reported by Wiens [2005], and present the best result of the algorithm among 10 random trials. In order to compare observation sets of the same size, we called SATURATE with $\alpha = 1$.



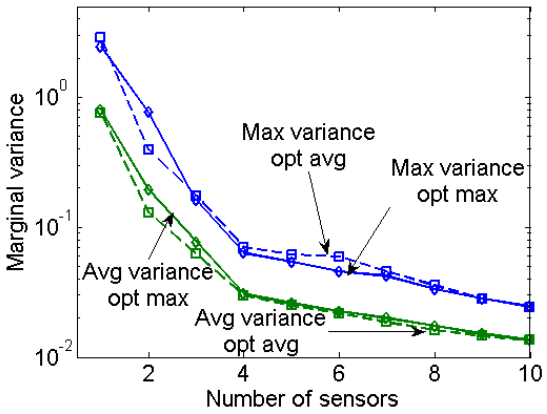
(a) *[P] Algorithm comparison*



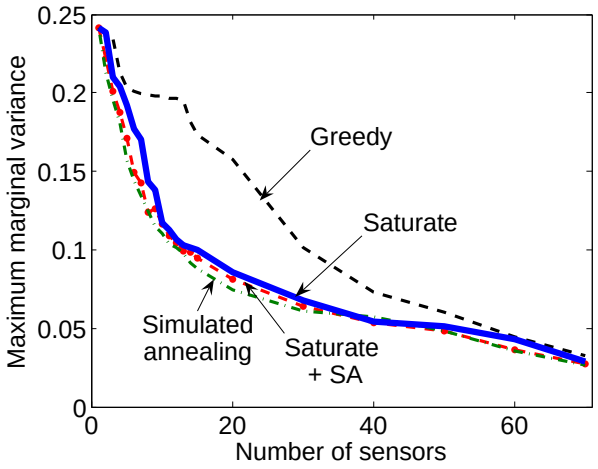
(b) *[P] Avg. vs max. variance*



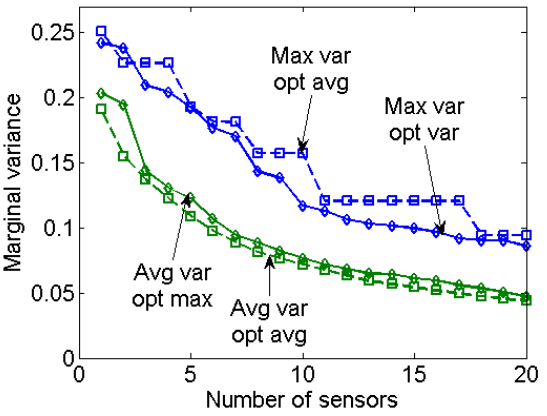
(c) *[T] Algorithm comparison*



(d) *[T] Avg. vs max. variance*



(e) *[L] Algorithm comparison*



(f) *[L] Avg. vs max. variance*

Figure 10.7: (a,c,e) SATURATE, greedy and SA on the (a) precipitation, (b) building temperature and (c) lake temperature data. SATURATE performs comparably with the fine-tuned SA algorithm, and outperforms it for larger placements. (b,d,f) Optimizing for the maximum variance (using SATURATE) leads to low average variance, but optimizing for average variance (using greedy) does not lead to low maximum variance.

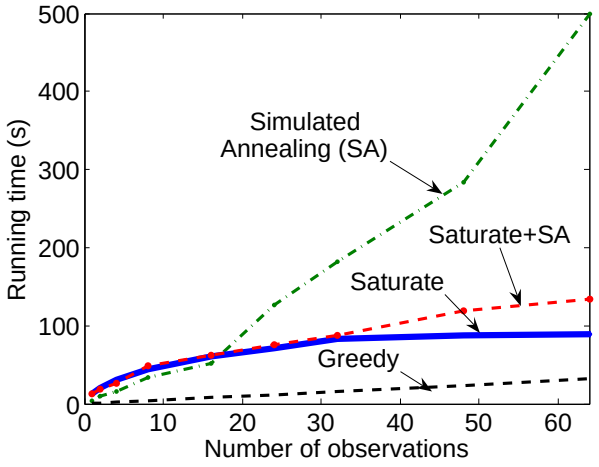


Figure 10.8: Running time for algorithms on the precipitation data set [P].

Figures 10.7(a), 10.7(c) and 10.7(e) compare simulated annealing, SATURATE, and the greedy algorithm which greedily selects elements which decrease the maximum variance the most on the three data sets. We also used SATURATE to initialize the simulated annealing algorithm (using only a single run of simulated annealing, as opposed to 10 random trials). In all three data sets, SATURATE obtains placements which are drastically better than the placements obtained by the greedy algorithm. Furthermore, the performance is very close to the performance of the simulated annealing algorithm. In our largest monitoring dataset [P], SATURATE even strictly outperforms the simulated annealing algorithm when selecting 30 and more sensors. Furthermore, as Figure 10.8 shows, SATURATE is significantly faster than simulated annealing, by factors of 5-10 for larger problems. When using SATURATE in order to initialize the simulated annealing algorithm, the resulting performance almost always resulted in the best solutions we were able to find with any method, while still executing faster than simulated annealing with 10 random restarts as proposed by Wiens [2005]. These results indicate that SATURATE compares favorably to state-of-the-art local search heuristics, while being faster, requiring no parameters to tune, and providing theoretical approximation guarantees.

Optimizing for the maximum variance could potentially be considered too pessimistic. Hence we compared placements obtained by SATURATE, minimizing the maximum marginal posterior variance, with placements obtained by the greedy algorithm, where we minimize the *average* marginal variance. Note, that, whereas the maximum variance reduction is non-submodular, the *average* variance reduction is (often) submodular [Das and Kempe, 2008], and hence the greedy algorithm can be expected to provide near-optimal placements. Figures 10.7(b), 10.7(d) and 10.7(f) present the maximum and average marginal variances for both algorithms. On all three data sets, our results show that if we optimize for the maximum variance we still achieve comparable average variance. If we optimize for average variance however, the maximum posterior variance remains much higher.

10.7.2 Robust experimental design

We consider the robust design of experiments (*c.f.*Section 10.5.3) for the Michaelis-Menten mass-action kinetics model, as discussed by Flaherty et al. [2006]. The goal is least-square parameter estimation for

a function $y = f(x, \theta)$, where x is the chosen experimental stimulus (the initial substrate concentration S_0), and $\theta = (\theta_1, \theta_2)$ are two parameters as described by Flaherty et al. [2006]. The stimulus x is chosen from a menu of six options, $x \in \{1/8, 1, 2, 4, 8, 16\}$, each of which can be repeatedly chosen. The goal is to produce a fractional design $\mathbf{w} = (w_1, \ldots, w_6)$, where each component w_i measures the relative frequency according to which the stimulus x_i is chosen. Since f is nonlinear, f is linearized around an initial parameter estimate $\theta_0 = (\theta_{01}, \theta_{02})$, and approximated by its Jacobian V_{θ_0}. Classical experimental design considers the error covariance of the least squares estimate $\hat{\theta}$, $\mathrm{Cov}(\hat{\theta} \mid \theta_0, \mathbf{w}) = \sigma^2 (V_{\theta_0}^T W V_{\theta_0})^{-1}$, where $W = \mathrm{diag}(\mathbf{w})$, and aims to find designs $\mathbf{w}$ which minimize this error covariance. E-optimality, the criterion adopted by Flaherty et al. [2006], measures smallness in terms of the maximum eigenvalue of the error covariance matrix. The optimal $\mathbf{w}$ can be found using Semidefinite Programming (SDP) [Boyd and Vandenberghe, 2004].

The estimate $\mathrm{Cov}(\hat{\theta} \mid \theta_0, \mathbf{w})$ depends on the initial parameter estimate θ_0, where linearization is performed. However, since the goal is parameter estimation, a "certain circularity is involved" [Flaherty et al., 2006]. To avoid this problem, Flaherty et al. [2006] find a design $\mathbf{w}_\rho(\theta_0)$ by solving a robust SDP which minimizes the error size, subject to a worst-case perturbation Δ on the Jacobian V_{θ_0}; the robustness parameter ρ bounds the spectral norm of Δ. As evaluation criterion, Flaherty et al. [2006] define a notion of *efficiency*, which is the error size of the optimal design with correct initial parameter estimate, divided by the error when using a robust design obtained at the wrong initial parameter estimates, i.e.,

$$\text{efficiency} \equiv \frac{\lambda_{\max}[\mathrm{Cov}(\hat{\theta} \mid \theta_{true}, \mathbf{w}_{opt}(\theta_{true}))]}{\lambda_{\max}[\mathrm{Cov}(\hat{\theta} \mid \theta_{true}, \mathbf{w}_\rho(\theta_0))]},$$

where $\mathbf{w}_{opt}(\theta)$ is the E-optimal design for parameter θ. They show that for appropriately chosen values of ρ, the robust design is more *efficient* than the optimal design, if the initial parameter θ_0 does not equal the true parameter.

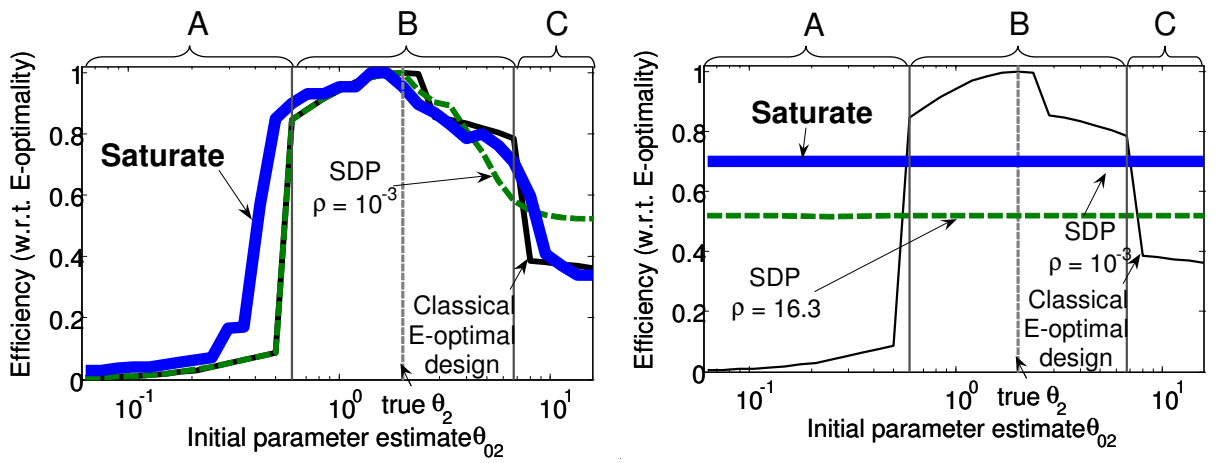
While their results are very promising, an arguably more natural approach than perturbing the Jacobian would be to perturb the initial parameter estimate, around which linearization is performed. For example, if the function f describes a process which behaves characteristically differently in different "phases", and the parameter θ controls which of the phases the process is in, then a robust design should intuitively "hedge" the design against the behavior in each possible phase. In such a case, the uniform distribution (which the robust SDP chooses for large ρ) would not be the most robust design.

If we discretize the space of possible parameter perturbations (within a reasonably chosen interval), we can use SATURATE to find robust experimental designs. While the classical E-optimality is not submodular [Krause et al., 2008b], Bayesian A-optimality is (usually) submodular [Das and Kempe, 2008, Krause et al., 2008b]. Here, the goal is to minimize the *trace* instead of maximum eigenvalue size of the covariance matrix. Furthermore, we equip the parameters θ with an uninformative normal prior (which we chose as $\mathrm{diag}([20^2, 20^2])$) as typically done in Bayesian experimental design. We then minimize the expected trace of the posterior error covariance, $\mathrm{tr}(\Sigma_{\theta|\mathcal{A}})$. Hereby, $\mathcal{A}$ is a discrete design of 20 experiments, where each option x_i can be chosen repeatedly. In order to apply SATURATE, for each θ_0, we define $F_{\theta_0}(\mathcal{A})$ as the normalized variance reduction

$$F_{\theta_0}(\mathcal{A}) = \frac{1}{Z_{\theta_0}} \left(\mathrm{tr}\left(\Sigma_\theta^{(\theta_0)}\right) - \mathrm{tr}\left(\Sigma_{\theta|\mathcal{A}}^{(\theta_0)}\right) \right).$$

The normalization Z_{θ_0} is chosen such that $F_{\theta_0}(\mathcal{A}) = 1$ if

$$\mathcal{A} = \operatorname*{argmax}_{|\mathcal{A}'|=20} F_{\theta_0}(\mathcal{A}'),$$



(a) *Low uncertainty in* θ_0

(b) *High uncertainty in* θ_0

Figure 10.9: Efficiency of robust SDP of Flaherty et al. [2006] and SATURATE on a biological experimental design problem. (a) Low assumed uncertainty in initial parameter estimates: SDP performs better in region C, SATURATE performs better in region A. (b) High assumed uncertainty in initial parameter estimates: SATURATE outperforms the SDP solutions.

i.e., if $\mathcal{A}$ is chosen to maximize only F_{θ_0}. SATURATE is then used to maximize the worst-case normalized variance reduction.

We reproduced the experiment of Flaherty et al. [2006], where the initial estimate of the second component θ_{02} of θ_0 was varied between 0 and 16, the "true" value being $\theta_2 = 2$. For each initial estimate of θ_{02}, we computed a robust design, using the SDP approach and using SATURATE, and compared them using the efficiency metric of Flaherty et al. [2006]. Note that this efficiency metric is defined with respect to E-optimality, even though we optimize Bayesian A-optimality, hence potentially putting SATURATE at a disadvantage. We first optimized designs which are robust against a small perturbation of the initial parameter estimate. For the SDP, we chose a robustness parameter $\rho = 10^{-3}$, as reported in Flaherty et al. [2006]. For SATURATE, we considered an interval around $[\theta\frac{1}{1+\varepsilon}, \theta(1+\varepsilon)]$, discretized in a 5×5 grid, with $\varepsilon = .1$.

Figure 10.9(a) shows three characteristically different regions, A, B, C, separated by vertical lines. In region B which contains the true parameter setting, the E-optimal design (which is optimal if the true parameter is known, i.e., $\theta_{02} = \theta_2$) performs similar to both robust methods. Hence, in region B (i.e., small deviation from the true parameter), robustness is not really necessary. Outside of region B however, where the standard E-optimal design performs badly, both robust designs do not perform well either. This is an intuitive result, as they were optimized to be robust only to small parameter perturbations.

Consequently, we compared designs which are robust against a *large* parameter range. For SDP, we chose $\rho = 16.3$, which is the maximum spectral variation of the Jacobian when we consider all initial estimates from θ_{02} varying between 0 and 16. For SATURATE, we optimized a single design which achieves the maximum normalized variance reduction over all values of θ_{02} between 0 and 16. Figure 10.9(b) shows, that in this case, the design obtained by SATURATE achieves an efficiency of 69%, whereas the efficiency of the SDP design is only 52%. In the regions A and C, the SATURATE design strictly outperforms the other robust designs. This experiment indicates that designs which are robust against a large range

of initial parameter estimates, as provided by SATURATE, can be more efficient than designs which are robust against perturbations of the Jacobian (the SDP approach).

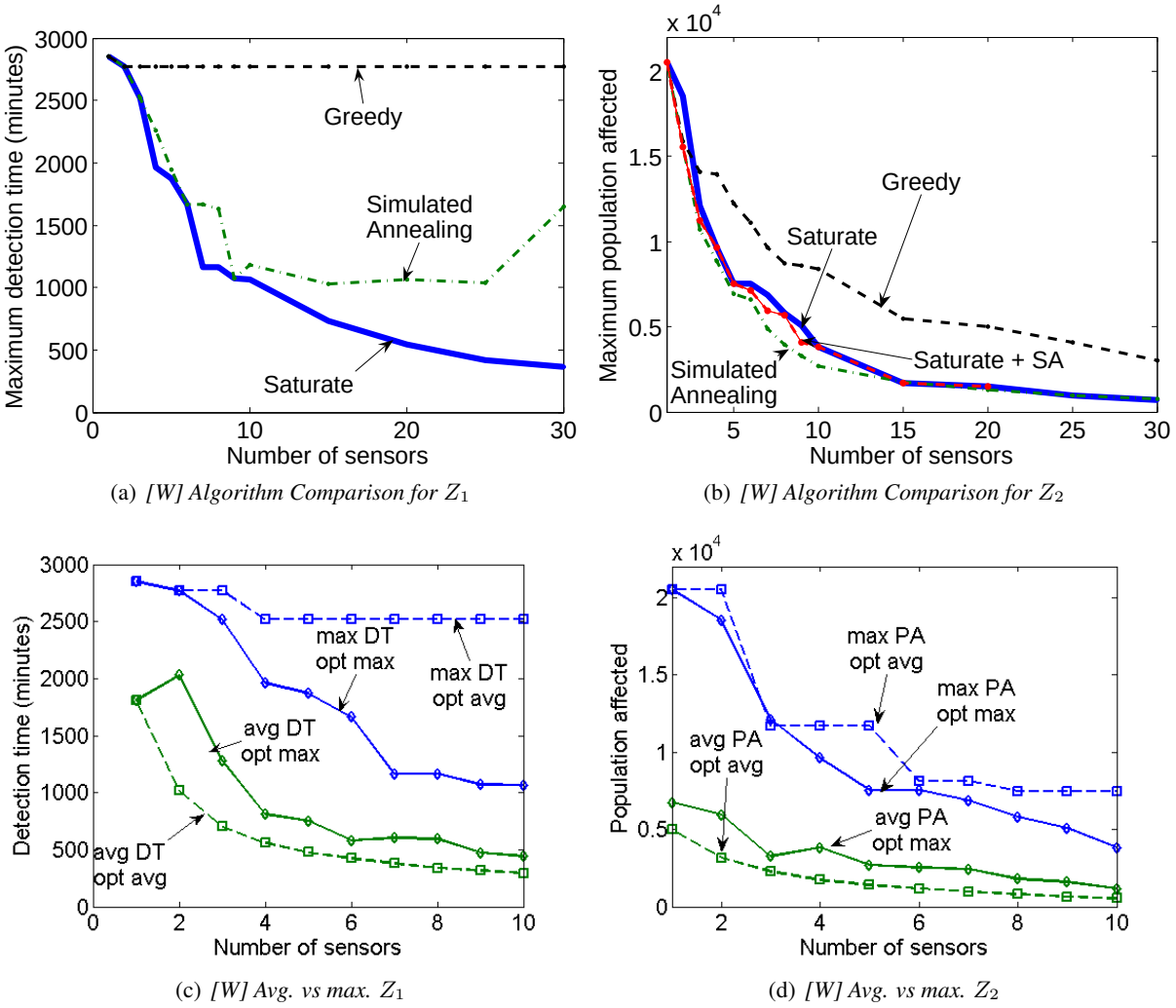
10.7.3 Outbreak detection

We also apply our approach to the problem of deploying sensors for contamination detection in municipal water distribution networks, as described in Chapter 8. Recall from Chapter 8, that in August 2006, the Battle of Water Sensor Networks (BWSN) [Ostfeld et al., 2006] was organized as an international challenge to find the best sensor placements for a real (but anonymized) metropolitan water distribution network, consisting of 12,527 nodes. In this challenge, a set of intrusion scenarios is specified, and for each scenario a realistic simulator provided by the EPA [Rossman, 1999] is used to simulate the spread of the contaminant for a 48 hour period. An intrusion is considered detected when one selected node shows positive contaminant concentration. BWSN considered a variety of impact measures, including the time to detection (called Z_1), and the size of the affected population calculated using a realistic disease model (Z_2). The goal of BWSN was to minimize the *expectation* of the impact measures Z_1 and Z_2 given a *uniform distribution* over intrusion scenarios.

In this chapter, we consider the *adversarial* setting, where an opponent chooses the contamination scenario with knowledge of the sensor locations. The objective functions Z_1 and Z_2 are in fact submodular for a fixed intrusion scenario [Leskovec et al., 2007b], and so the robust optimization problem of minimizing the impact of the worst possible intrusion fits into our formalism. For these experiments, we consider scenarios which affect at least 10% of the network, resulting in a total of 3424 scenarios. Figures 10.10(a) and 10.10(b) compare the greedy algorithm, SATURATE and the simulated annealing (SA) algorithm for the problem of maximizing the worst-case detection time (Z_1) and worst-case affected population (Z_2).

Interestingly, the behavior is very different for the two objectives. For the affected population (Z_2), greedy performs reasonably, and SA sometimes even outperforms SATURATE. For the detection time (Z_1), however, the greedy algorithm did not improve the objective at all, and SA performs poorly. The reason is that for Z_2, the maximum achievable scores, $F_i(\mathcal{V})$, vary drastically, since some scenarios have much higher impact than others. Hence, there is a strong "gradient", as the worst-case objective changes quickly when the high impact scenarios are covered. This gradient allows greedy and SA to work well. On other hand, for Z_1, the maximum achievable scores, $F_i(\mathcal{V})$, are constant, since all scenarios have the same simulation duration. Unless *all* scenarios are detected, the worst-case detection time stays constant at the simulation length. Hence, many node exchange proposals considered by SA, as well as the addition of a new sensor location by greedy, do not change the worst-case objective, and the algorithms have no useful performance metric.

Figures 10.10(c) and 10.10(d) compare the placements of SATURATE (when optimizing the worst-case penalty), and greedy (when optimizing the average-case penalty, which is submodular). Similarly to the results in the GP setting, optimizing the worst-case score leads to reasonable performance in the average case score, but not necessarily vice versa (especially when considering the detection time).



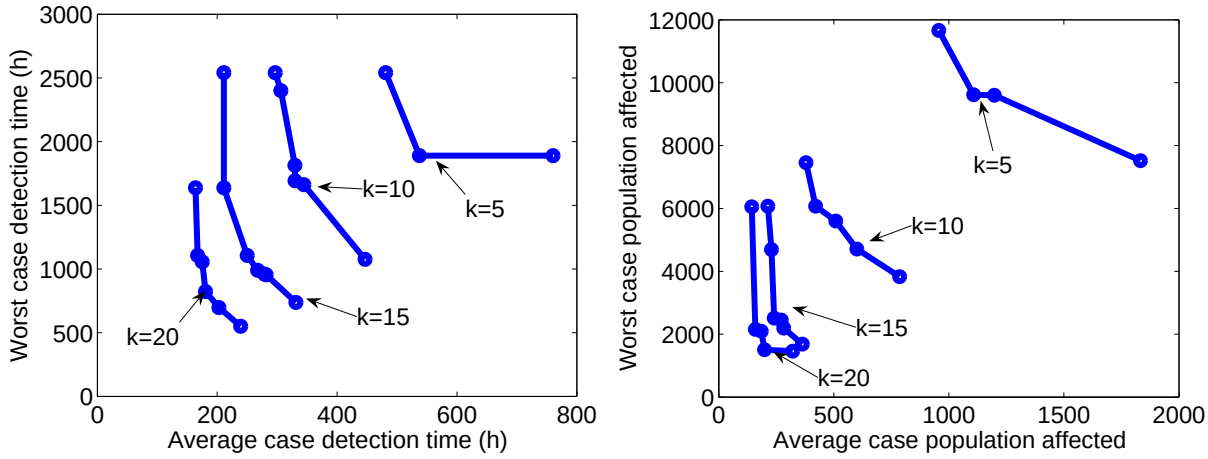
(a) *[W] Algorithm Comparison for* Z_1

(b) *[W] Algorithm Comparison for* Z_2

(c) *[W] Avg. vs max.* Z_1

(d) *[W] Avg. vs max.* Z_2

Figure 10.10: (a,b) compare SATURATE, greedy and SA in the water network setting, when optimizing worst-case detection time (Z_1, (a)) and affected population (Z_2, (b)). SATURATE performs comparably to SA for Z_2 and strictly outperforms SA for Z_1. (c,d) compare optimizing for the worst-case vs. average-case objectives. Optimizing for the worst-case leads to good average case performs, but not vice versa.



(a) *[W] Pareto curve for Z_1*

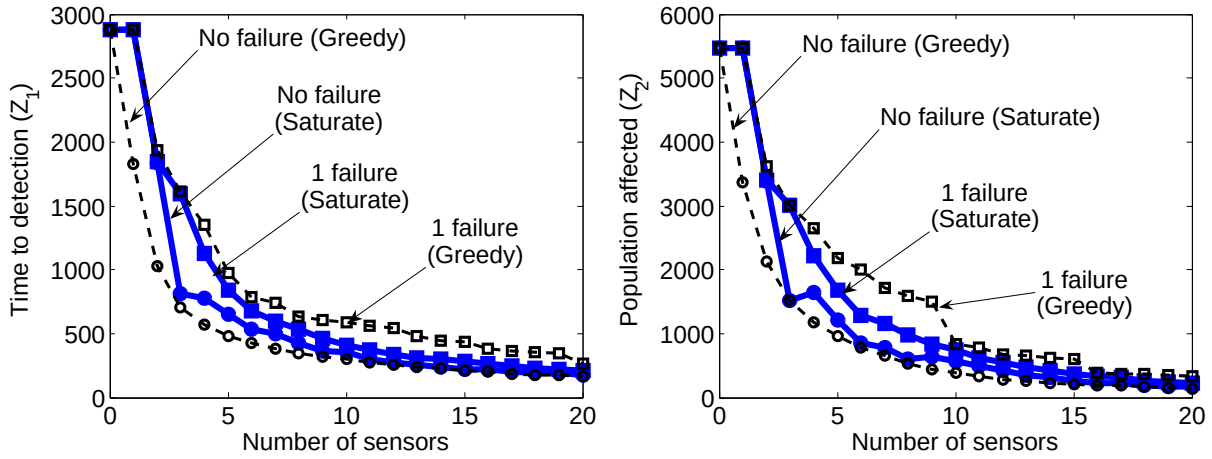
(b) *[W] Pareto curve for Z_2*

Figure 10.11: Experiments on trading off worst-case and average-case penalties on the water network [W] data, minimizing detection time (a) and affected population (b).

We also performed experiments trading off the worst-case and average-case penalty reductions, using the approach discussed in Section 10.6.5. We first ran the greedy algorithm to optimize the average-case score, and then ran SATURATE to optimize the worst-case score. We considered the average-case scores c_{ac}^{greedy} and $c_{ac}^{Saturate}$ obtained by both algorithms, and uniformly discretized the interval bounded by these average-case scores. For each score level c_{ac} in the discretization, we use the modified SATURATE algorithm as described in Section 10.6.5, maximizing the worst-case score, subject to a constraint on the average-case score. Each possible value of the constraint on c_{ac} can lead to a different solution, trading off average- and worst-case scores. Figure 10.11(a) presents the tradeoff curve obtained in this fashion for the detection time (Z_1) metric, for different numbers k of placed sensors. We generally observe that there is more variability in the worst-case score than in the average-case score. We can also see that when placing 5 sensors, there is a prominent knee in the tradeoff curve, effectively achieving the minimum worst-case penalty but drastically reducing the average-case penalty incurred when compared to only optimizing for the worst-case score. The other tradeoff curves do not exhibit quite such prominent knees, but nevertheless allow flexibility in trading off worst- and average-case scores. Figure 10.11(b) presents the same experiment for the population affected (Z_2) metric. Here, we notice prominent knees when placing $k = 15$ and 20 sensors. We can generally conclude that trading off average- and worst-case scores allows to effectively achieve a compromise between too pessimistic (only optimizing for the worst case) and optimistic (only optimizing for the average case) objectives.

10.7.4 Sensor failures

We also performed experiments on analyzing worst-case sensor failures (*c.f.*Section 10.5.5). We consider the outbreak detection application, and optimize the average score, i.e., $F(\mathcal{A}) = \frac{1}{m}\sum_i F_i(\mathcal{A})$ (modeling, e.g., accidental contaminations). We use SATURATE in order to optimize the modified objective function $\overline{F}'_c$ described in Section 10.5.5, for increasing numbers of sensors k. We also use the greedy algorithm to optimize sensor placements, ignoring possible sensor failures. For both algorithms, we compute the



(a) *[W] Sensor failures for Z_1*

(b) *[W] Sensor failures for Z_2*

Figure 10.12: (a,b) compare Greedy (ignoring sensor failures) and SATURATE (optimizing for the worst-case sensor failure) on water network data with detection time (a) and population affected (b) scores.

expected scores (penalty reductions Z_1 and Z_2) in the case of no sensor failure, and in the case of a single, worst-case sensor failure. Figure 10.12(a) presents the results for the time to detection objective (Z_1). We can see, that initially, with small numbers of sensors, failures can strongly diminish the Z_1 score. However, as the number of sensors increases, the placement scores optimized using SATURATE for sensor failures quickly approach those of Greedy in the case of no sensor failures. Hence, even if only a small number of sensors are placed, SATURATE can quickly exploit redundancy and find sensor placements, which perform well both with and without sensor failures. On the other hand, when not taking sensor failures into account, such failures can drastically diminish the sensing quality of a placed set of sensors. Figure 10.12(b) presents analogous results when minimizing the affected population (Z_2).

10.7.5 Parameter uncertainty

We also conducted experiments on selecting variables under parameter uncertainty (*c.f.*Section 10.5.2). More specifically, we consider a sensor placement problem for monitoring temperature in a building. In such a problem, we would like to place sensors in order to get accurate predictions at various times of the day. However, since phenomena such as temperature in buildings change over time, at different times of the day, different placements would be most informative.

In our experiment, we consider the temperature data set [T], and learn four models, described by parameters $\theta_1, \ldots, \theta_4$, during four six-hour time periods over the day: 12am-6am, 6am-12pm, 12pm-6pm and 6pm-12am. As models, we use the empirical covariances $\Sigma^{(\theta_i)}$ from the corresponding time periods of the 5 day historical training data. We also use the single model Σ for the entire day, as described in Section 10.7.1. We then use the greedy algorithm to optimize sensor placement of increasing sizes for the single model Σ, optimizing the average variance reduction objective function. Similarly, we use SATURATE to optimize the minimum variance reduction over the four models $\Sigma^{(i)}$, normalized by the average variance over the entire space.

Subsequently, we used both placements to compute the average Root Mean Squared (RMS) prediction

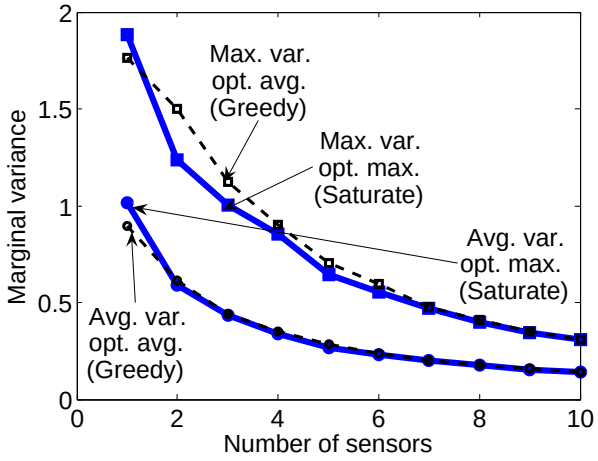


Figure 10.13: [T] Average and Maximum variance when optimizing for four different covariance models obtained during different parts of the day.

error over the entire day on 2 days of held out test data. We also computed the maximum RMS error over the four six-hour time periods. Figure 10.13 presents the results of this experiment. While the average RMS error is roughly equal for both placements, the maximum RMS error is larger for the greedy sensor placement, as compared to the robust placement of SATURATE, especially for small numbers of sensors (six and less sensors).

10.8 Robustness and regularization

There is an interesting connection between robustness and regularization in machine learning. For example, recent results by Xu et al. [2008] show that ℓ_1-regularized regression is equivalent to finding a solution to an ordinary regression problem which is worst-case robust against perturbation of the design matrix. In this section, we show how robust optimization can lead to regularization (and thereby improved generalization) for optimization problems under uncertain data.

Consider a setting where we are given a ground set $\mathcal{V}$ of elements, and "training data" in the form of a collection of submodular functions $F_1, \ldots, F_t$. We call each of these functions a "scenario". An example of a scenario could be a contamination event in the water networks domain (Section 10.5.3), or a cascade in the blog domain (Section 8.3). Hereby, $F_i(\mathcal{A})$ measures the performance of a set of chosen elements $\mathcal{A}$ in scenario i (e.g., in the water networks domain, $F_i(\mathcal{A})$ measures the reduction in population affected by contamination of node i when placing sensors at locations $\mathcal{A}$). Our goal is to select a subset $\mathcal{A}$, $|\mathcal{A}| \leq k$, which performs well on a separate set of "test" scenarios, which comprises functions $F'_1, \ldots, F'_m$ defined over the same ground set $\mathcal{V}$. I.e., we would like to find a set $\mathcal{A}$ maximizing

$$\max_{|\mathcal{A}| \leq k} \sum_{i=1}^{m} F'_i(\mathcal{A}).$$

$\mathcal{A}$	$F_1(\mathcal{A})$	$F_2(\mathcal{A})$	$F_1'(\mathcal{A})$
$\emptyset$	0	0	0
$\{s_1\}$	1	0	0
$\{s_2\}$	.4	.4	.5

Table 10.2: Functions F_1 and F_2 used in the generalization counterexample.

Note that simply optimizing the training set performance

$$\max_{|\mathcal{A}|\leq k} \sum_{i=1}^{t} F_i(\mathcal{A})$$

can lead to overfitting. As an example, suppose $\mathcal{V} = \{s_1, s_2\}$, $k = 1$ and there are two training scenarios, F_1, F_2, and one test scenario F_1' as defined in Table 10.2. When optimizing for the training set, one optimal solution would be to pick $\mathcal{A} = \{s_1\}$, with performance 1 on the training set, but 0 on the testing set. Hence, optimizing purely on the training set leads to arbitrary poor generalization performance. However, intuitively, the test scenario F_1' is "similar" to F_2, one of the training examples, and hence we would hope for non-trivial generalization performance. In many applications, such as the water networks and blog domains, it is unrealistic to assume that the functions F_i and F_i' are sampled i.i.d. from some fixed distribution, and hence standard generalization bounds do not apply.

To have any chance of generalizing from the training to the test scenarios, we need to make some assumptions about how the test functions relate to the training data. We will assume that the test set is "similar" to some part of the training set. More specifically, we will assume that there is a mapping ν that maps the indices $\{1, \ldots, m\}$ to a subset of the indices $\{1, \ldots, t\}$, and it holds that

$$|F_i'(\mathcal{B}) - F_{\nu(i)}(\mathcal{B})| \leq \varepsilon$$

for all subsets $\mathcal{B} \subseteq \mathcal{V}$. Hence, each function F_i' is similar (with additive error ε) to some function $F_{\nu(i)}$. This assumption is sensible, for example, if we believe that the training data "approximately covers" the space of all possible scenarios. The assumption applies, for example, in a setting where the utility functions change over time (i.e., the indices $1, \ldots, t$ correspond to a linear ordering), and the test functions F_i' are most similar to the last functions in the training set with respect to the temporal ordering. In our example from Table 10.2, $\nu(1) = 2$, and $\varepsilon = 0.1$.

If we knew the mapping ν, we could optimize the test set performance directly by optimizing

$$\max_{|\mathcal{A}|\leq k} \sum_{i=1}^{m} F_{\nu(i)}(\mathcal{A}).$$

Since in practice the mapping ν is not known, we can consider the performance for any particular mapping ν' as

$$F_{\nu'}(\mathcal{A}) = \sum_{i=1}^{m} F_{\nu(i)}(\mathcal{A}),$$

and optimize

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}|\leq k} \min_{\nu'} F_{\nu'}(\mathcal{A}).$$

This formulation simultaneously optimizes the performance over all possible test sets that are similar (according to a mapping ν') to the training set. In our example, the optimal solution to this optimization problem would be to pick the set $\mathcal{A}' = \{s_2\}$ which obtains optimal testing performance $1/2$.

Note that the worst case $\min_{\nu'}$ will always be achieved by mapping all elements $1, \ldots, m$ to a single training scenario. This assumption might be too pessimistic, and lead to solutions which "underfit" the training data. Hence, in practice, we would choose a regularization parameter L, and only consider mappings ν that map to at least L different training scenarios.

In this case, there will be a large number of possible mappings ν that map to at least L different training scenarios, and evaluation of $\min_\nu F_\nu(\mathcal{A})$, respectively the average truncated function $\sum_\nu \min\{F_\nu(\mathcal{A}), c\}$ as needed in the SATURATE algorithm will be intractable. In practice, we have at least three different options for dealing with this intractability:

- We restrict ourselves to particular classes of functions ν. One example of this approach will be provided in Section 10.8.1.
- We use techniques for dealing with a large number of objective functions as discussed in Section 10.9.
- We approximate the average truncated function $\sum_\nu \min\{F_\nu(\mathcal{A}), c\}$ by a sample average with a small number of samples.

10.8.1 Results on blog data

As an example, consider the problem of selecting informative blogs, as discussed in Chapter 8. When designing a web service to suggest blogs to read, it is important to select blogs that remain informative in the future.

We perform the following experiment. We split our blog data set (as described in Section 8.3) into two parts: a "historic" training set, containing the cascades starting January 2006 and concluding by May 2006, and a testing set of "future" data, containing the cascades starting June 2006-November 2006. For each cascade i in the training set, we define a separate objective function F_i using the population affected (PA) objective function from Chapter 8. Similarly, we define functions F_i' on the test set cascades.

We first use the CELF algorithm (Algorithm 5.4) to select blogs from the training set, using the "number of posts" cost metric as specified in Chapter 8. In order to assess the generalization ability of the selected blogs, we evaluate the performance of the obtained solutions on the test set. As a (utopic) baseline, we also perform the same optimization on the test set. Figure 10.14 shows a big gap between the performance of both approaches, indicating a poor generalization.

In order to understand this poor generalization behavior, we analyze how many cascades are detected over time by the blogs selected using the CELF algorithm. Figure 10.15(a) shows that while many cascades are detected during the first half of the training data set, this detection performance degrades in the second half of the data set. Hence, the blog selection "overfits" to the initial part of the training data set.

In order to improve generalization, we would like to obtain a selection $\mathcal{A}$ of blogs that continues to perform well over time. We can achieve this goal by considering mappings ν that map into different, subsequent time intervals. In our experiments, we partition the training set into bins with one month of

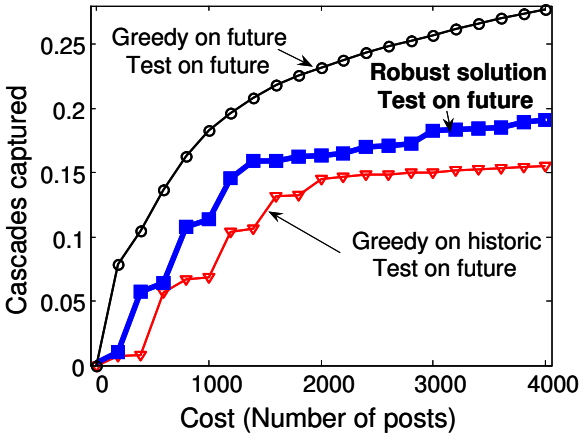


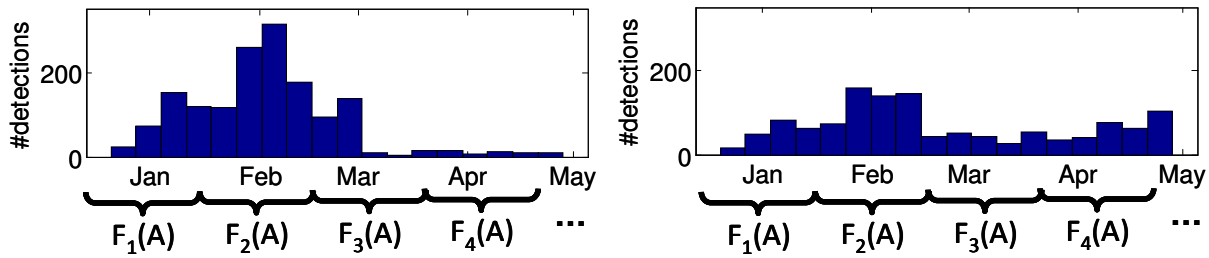
Figure 10.14: [B] Blog data. Optimizing on training data leads to overfitting. Robust optimization using SATURATE acts as a regularizer and leads to improved generalization performance.

length each, and use an objective function F_{ν_t} to quantify the performance of the blog selection in time interval t. To obtain a blog selection that performs uniformly well over time, we can then formulate the robust observation selection problem

$$\mathcal{A}^* = \operatorname*{argmax}_{|\mathcal{A}| \leq k} \min_t F_{\nu_t}(\mathcal{A}),$$

which we can approximately solve using the SATURATE algorithm.

Using such a robust solution, the detection performance is now uniformly high over time, as shown by Figure 10.15(b). Robust optimization also leads to improved performance on future data as shown by Figure 10.14, which demonstrates significantly improved generalization especially in the "low cost" regime of the performance curve. Hence, robust optimization acts as an effective regularizer for the blog selection problem. We consider it a promising area of future work to further explore to interpretation of robust solution of optimization problems as regularization to improve performance on unseen test data.



(a) *Detections using* CELF

(b) *Detections using* SATURATE

Figure 10.15: (a) Cascades detected over time using the CELF algorithm for selecting blogs. (b) Detections over time using the SATURATE algorithm for selecting blogs.

10.9 Reducing the number of objective functions

In many of the examples considered in Section 10.5, the number m of objective functions F_i can be quite large (e.g., one F_i per parameter setting, or outbreak scenario), which impacts both the running time (which depends linearly on m) and the approximation guarantees (which depend logarithmically on m) of SATURATE. Hence, showing that we can work with a smaller set of objectives has both computational and theoretical advantages.

10.9.1 Removal of dominated strategies

One direct approach to eliminate objective functions (and hence speed up computation and improve the approximation guarantee) is to remove dominated objectives. An objective function F_i is dominated by another objective F_j, if $F_i(\mathcal{A}) \geq F_j(\mathcal{A})$ for all sets $\mathcal{A} \subseteq \mathcal{V}$. Hence, an F_i is dominated by F_j if an adversary can always reduce our score by choosing F_j instead of F_i. For example, when considering sensor failures or feature deletion (as discussed in Section 10.5.5), for two sets $\mathcal{B} \subseteq \mathcal{B}'$, the objective $F_{\mathcal{B}}$ is dominated by the objective $F_{\mathcal{B}'}$, i.e., the score decreases more if more sensors fail. Similarly, in the case of outbreak detection, some outbreak scenarios have much more impact on the network than others. Even though objective functions measuring the impact reductions F_i for scenarios $i \in \mathcal{I}$ might not be *exactly* dominated, they might be ε-dominated, i.e., $F_i(\mathcal{A}) \geq F_j(\mathcal{A}) - \varepsilon$ for some $\varepsilon > 0$ and all $\mathcal{A} \subseteq \mathcal{V}$. In such cases, these approximately dominated scenarios can be removed, incurring at most an error of ε in the quality of the approximate solution.

10.9.2 Constraint generation

Another possible approach to reduce the number m of objective functions is constraint generation [*c.f.*, Benders, 1962]. In this approach, one starts with an arbitrary single objective function, F_1. In iteration $j+1$, ($j \geq 1$), after functions $F_1, \ldots, F_j$ have been considered, one searches for set $\mathcal{A}_j$ maximizing $\max_{\mathcal{A}} \min_{1 \leq i \leq j} F_j(\mathcal{A})$. Subsequently, one selects F_{j+1} minimizing $\min_i F_i(\mathcal{A}_j)$. The iteration terminates once F_{j+1} is contained in the already selected objectives $F_1, \ldots, F_j$. Another option is to terminate once the new objective F_{j+1} is ε-dominated by some objective F_i, $1 \leq i \leq j$. In this case, the approximate solution is guaranteed to incur at most an absolute error of ε as compared to the optimal solution.

In order to implement this constraint generation scheme, one must be able to efficiently solve problem $\min_i F_i(\mathcal{A}_j)$. In some settings, this problem might admit an efficient (perhaps approximate) solution. In many problems, such as the experimental design setting, one actually wants to perform well against an (uncountably) infinite set of possible objective functions, corresponding to parameters $\theta \in D$ in some (typically compact and convex set D). In such a setting, $\min_\theta F_\theta(\mathcal{A}_j)$ could potentially be (at least heuristically) solved using a numerical optimization approach such as a conjugate gradient method.

10.10 Related work

10.10.1 Robust discrete optimization

Robust optimization of submodular functions is an instance of a robust discrete optimization problem. In such problems, the goal generally is to perform well with respect to a worst-case choice of evaluation scenario. Other instances of robust discrete problems have been studied by a number of authors. Kouvelis and Yu [1997] introduce several notions of robust discrete problems, presents hardness results and a class of robust problems that can be optimally solved. Averbakh [2001] shows that a class of robust optimization problems (selecting a k-element subset of elements of minimum cost) is solvable in polynomial time if the uncertain cost coefficients are contained in an interval, but $\mathbf{NP}$-hard under an arbitrary (finite) set of adversarially chosen scenarios. Bertsimas and Sim [2003] proposes a class of robust mixed integer programs, accommodating uncertainty both in cost and data coefficients. They show that in certain cases (robust matching, spanning tree, etc.), the robust formulations are solvable in polynomial time if the non-robust problem instances are solvable in polynomial time. In the case of $\mathbf{NP}$-hard but α-approximable non-robust problems, they show that the corresponding robust formulations also remain α-approximable. However, their results do not transfer to our setting of robust submodular optimization, since in this case, even though non-robust solutions are $(1 - 1/e)$ approximable, the non-robust formulation does not admit any approximation guarantees (*c.f.*Section 10.2).

10.10.2 Robust methods in statistics

Robust experimental design

Experimental design under parameter uncertainty has been studied in statistics; most of the earlier work is reviewed in the excellent survey of Chaloner and Verdinelli [1995]. In the survey, the authors discuss Bayesian approaches to handling parameter uncertainty, as well as robust Bayesian [*c.f.*, Berger, 1984] approaches, which perform worst-case analyses over prior and likelihood functions. In experimental design, most approaches have focused on *locally* optimal designs, i.e., those selecting an optimal design based on a linearization around an initial parameter estimate, for reasons of computational tractability. In order to cope with uncertainty in the initial parameter estimates around which linearization is performed, heuristic techniques have been developed, such as the SDP based approach of Flaherty et al. [2006], or a clustering heuristic described by Dror and Steinberg [2006]. We are not aware of approaches which allow to find designs in the context of such parameter uncertainty that bear theoretical guarantees similar to the approach described in this chapter.

Minimax Kriging

Minimizing the maximum predictive variance in Gaussian Process regression has been proposed as a design criterion by Burgess et al. [1981] and since then extensively used. [*c.f.* Sacks and Schiller, 1988, van Groenigen and Stein, 1998]. To our knowledge, prior to this work, no algorithms with approximation guarantees are known for this criterion.

Several authors consider the problem of spatial prediction under unknown covariance parameters. Pilz et al. [1996] describes an approach for selecting – for a fixed set of observed sites – the Kriging estimate

minimizing the maximum prediction error, where the worst-case over a fixed class of covariance functions is assumed. Wiens [2005] consider a similar setting but also addresses the design problem of choosing locations in order to minimize the mean squared prediction error against the worst-case covariance function. Algorithmically, Wiens [2005] use the simulated annealing algorithm described in Section 10.7.1 with 7 tuned parameter settings. Note that the SATURATE algorithm can be used in this context as well.

10.10.3 Robust sensor placement and facility location

Carr et al. [2006] consider the problem of robust sensor placements in water distribution networks. They formulate Mixed Integer Programs for selecting sensor placements robust against uncertainty in adversarial strategies and in water demands. Due to computational complexity of Mixed Integer Programming, in their experiments, they used only small networks of at most 470 nodes. SATURATE can potentially be applied to handle uncertainty in demands as well, which is an interesting direction for future work. Watson et al. [2006] consider different notions of robustness in the context of water distribution networks, intended to remove some of the pessimistic assumptions of purely robust sensor placements. They develop integer programs, as well as heuristics, and apply them to networks of similar size as the one considered in this chapter. Their local search heuristic performs a sequence of local moves similar to those performed by the simulated annealing algorithm considered in Section 10.7.3, and does not provide any theoretical guarantees.

Closely related to the adversarial outbreak detection problem is the k-*center* problem. In this problem, one is given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ along with a distance function defined over pairs of nodes in $\mathcal{V}$. The goal is to select a subset $\mathcal{A} \subseteq \mathcal{V}$ of size at most k, such that the maximum distance between any unselected node $s \in \mathcal{V} \setminus \mathcal{A}$ and its nearest center $s' \in \mathcal{A}$ is minimized. For this problem, Minieka [1970] discuss a technique reducing the solution of this problem to a sequence of set cover problems combined in a binary search, similar in spirit to SATURATE. However, they do not discuss any implications regarding approximation guarantees, and do not consider the case of arbitrary submodular functions. Berger-Wolf et al. [2005] shows that certain contamination detection problem formulations can be modeled as asymmetric k-center problems. Mladenovic et al. [2003] presents a Tabu search heuristic for k-center, also without theoretical guarantees. Gonzalez [1985] and Hochbaum and Shmoys [1985] present a 2 approximation for the k-center problem in the case of symmetric distance functions satisfying the triangle inequality. Panigrahy and Vishwanathan [1998] present a $\log^*(n)$ approximation in the case of distance functions satisfying the asymmetric triangle inequality, which is shown to be best possible by Chuzhoy et al. [2005]. Chuzhoy et al. [2005] also show that even for bicriterion algorithms (such as SATURATE), k-center is $\log^*(n)$ hard to approximate, even if $\mathcal{O}(k)$ additional centers can be selected. Note that SATURATE can be used to solve k-center problems (without any requirements on symmetry or on the triangle inequality), hence the bicriterion hardness result of Chuzhoy et al. [2005] gives further evidence on the tightness of the guarantees described in Section 10.4.

Anthony et al. [2008] consider robust and stochastic notions of facility location problems (such as k-center and k-median, where, instead of the maximum distance the average distance is optimized). In contrast to the robust problems in this chapter which want to select k elements to *maximize* the minimum value achieved by these k elements over the m scenarios, the problems in Anthony et al. [2008] try to select k "centers" in a metric space to *minimize* the maximum cost incurred over the m scenarios—where the cost is some function of the distances between non-selected vertices to the selected centers. For several such robust cost-minimization problems in cases where distances satisfy the symmetric triangle inequality, they present an algorithm that opens k "centers" and achieves an approximation ratio of $\mathcal{O}(\log n + \log m)$

(where n is the number of nodes in the graph, and m is the number of scenarios): this should be compared to the impossibility results for approximating robust value-maximization problems presented in this chapter.

10.10.4 Relationship to game theory and allocation problems

The RSOS problem can be viewed as the problem of finding an optimal pure strategy for a zero-sum matrix game with player ordering. In this matrix game, the rows would correspond to the possible sensor placements, and the columns would correspond to the objective functions F_i. The entry for cell $(\mathcal{A}, F_i)$ is our payoff $F_i(\mathcal{A})$. In the RSOS problem, we want to select a row of the matrix, our adversary selects a column F_i (knowing our choice $\mathcal{A}$, hence the player ordering) minimizing our score $F_i(\mathcal{A})$. A very related class of game theoretic problems are *allocation problems*. In these problems, one is typically given a set $\mathcal{V}$ of objects, and the goal is to allocate the objects to m agents (bidders), each of whom has a (potentially different) valuation function $F_i(\mathcal{A}_i)$ defined over subsets of received items $\mathcal{A}_i$. The problem of finding the best such allocation (partition) is **NP**-hard, but recently, several approximation algorithms have been proposed. The allocation problem most similar to the RSOS problem is

$$\pi^* = \operatorname*{argmax}_{\text{partition } \pi=(\mathcal{A}_1,\ldots,\mathcal{A}_m)} \min_i F_i(\mathcal{A}_i).$$

The main difference is that in the allocation problem, the full set $\mathcal{V}$ is partitioned into subsets $\mathcal{A}_1, \ldots, \mathcal{A}_m$, and the functions F_i are evaluated on the respective subset $\mathcal{A}_i$ each. In the case of *additive* objective functions F_i, Asadpour and Saberi [2007] provide an $\mathcal{O}(\sqrt{k}\log^3 k)$ approximation algorithm. In the case of the function being *subadditive* (which is implied by, and is more general than, submodularity), Ponnuswami and Khot [2007] present an $\mathcal{O}(2k-1)$ approximation algorithm. For settings where the *sum* of the valuations is optimized, i.e.,

$$\pi^* = \operatorname*{argmax}_{\text{partition } \pi=(\mathcal{A}_1,\ldots,\mathcal{A}_m)} \sum_i F_i(\mathcal{A}_i),$$

Feige et al. [2007] develop a randomized 2-approximation for subadditive and $1 - 1/e$ approximation for submodular valuation functions.

The problem of trading off safety (i.e., improvements in worst-case scores) and average case performance has been studied by several authors. Johanson et al. [2007] consider the problem of opponent modeling in games, and develop an algorithm which can exploit opponents which it can accurately model, and falls back to a safe (Nash) strategy in case the models do not capture the opponents behavior. Their algorithm has a tradeoff parameter which controls the eagerness of exploiting, and they present Pareto-curves similar to those presented in Section 10.6.5. However, their approach does not apply to our robust submodular observation selection setting. Watson et al. [2006] consider different optimization problem formulations allowing to control risk in the water distribution network monitoring application, but they only present heuristic algorithms without guarantees for coping with large networks.

10.10.5 Relationship to Machine Learning

Submodular function optimization has found increasing use in machine learning. The algorithm of Queyranne [1995] for minimizing symmetric submodular functions has been used for learning graphical models by

Narasimhan and Bilmes [2004] and for clustering by Narasimhan et al. [2005]. We are not aware of any work on optimizing the minimum over a collection of submodular functions.

Observation selection approaches have been used in the context of active learning [*c.f.* Axelrod et al., 2001, Cohn, 1994, Freund et al., 1997, MacKay, 1992, Sollich, 1996]. Test point selection has been used to minimize average predictive variance in Gaussian Processes regression by Seo et al. [2000], and to speed up Gaussian Process inference in the Informative Vector Machine (IVM) by Lawrence et al. [2003], Seeger et al. [2003]. In these approaches, the sequential setting is considered, where previous measurements are taken into account when deciding on the next observation to make. A note by Seeger [2004b] proves that the greedy algorithm in the IVM optimizes a submodular function. The extension of the robust techniques discussed in this chapter, which address the a priori selection problem (i.e., observations are selected before measurements are obtained), to the sequential setting is an important direction for future research.

Balcan et al. [2006] consider the problem of active learning in the presence of *adversarial* noise. While their method is very different, our results potentially generalize to active learning settings, since, as Hoi et al. [2006] show, certain active learning objectives are (approximately) submodular.

Price and Messinger [2005] consider the problem of constructing recommendation sets, and show that this problem is an instance of a k-median problem (*c.f.*Section 10.10.3). The analogue of the k-center problem in the preference set construction would be to construct a preference set which maximizes the utility of displayed items under worst-case instantiation of the parameters. This analogue seems natural, and an interesting direction for future work would be to explore the use of SATURATE in the recommendation set context.

10.11 Existing work on submodular function maximization

Lovasz [1983] discusses the relationship between submodular functions and convexity. He also shows that under certain conditions, the minimum of two submodular functions remains submodular (and hence can be efficiently optimized using the greedy algorithm). The objective functions resulting from observation selection problems typically do not satisfy these properties, and, as we have shown, the greedy algorithm can perform arbitrarily badly. Fujito [2000] uses submodularity of truncated functions to find sets with partial submodular coverage; however, they do not consider the case of multiple objectives, which we address in this chapter. Bar-Ilan et al. [2001] consider covering problems for a generalization of submodular functions; they use a similar binary search technique combined with multiple applications of the greedy algorithm. Their approach does not apply to maximizing the minimum over a set of submodular functions. Golovin and Streeter [2008] present an algorithm for online maximization of a single submodular set function. An interesting question for future work would be to investigate whether our approach for maximizing the minimum over a collection of submodular functions can be generalized to an online setting as well.

10.12 Summary

In this chapter, we considered the RSOS problem of robustly selecting observations which are informative with respect to a worst-case submodular objective function. We demonstrated the generality of this prob-

lem, and showed how it encompasses the problem of sensor placements which minimize the maximum posterior variance in Gaussian Process regression, variable selection under parameter uncertainty, robust experimental design, and detecting events spreading over graphs, even in the case of adversarial sensor failures. In each of these settings, the individual objectives are submodular and can be approximated well using, e.g., the greedy algorithm; the robust objective, however, is not submodular.

We proved that there cannot exist any approximation algorithm for the robust optimization problem if the constraint on the observation set size must be exactly met, unless $\mathbf{P} = \mathbf{NP}$. Consequently, we presented an efficient approximation algorithm, SATURATE, which finds observation sets which are guaranteed to be least as informative as the optimal solution, and only logarithmically more expensive. In a strong sense, this guarantee is the best possible under reasonable complexity theoretic assumptions.

We provided several extensions to our methodology, accommodating more complex cost functions (non-uniform observation costs, communication and path costs). Additionally, we described how a compromise between worst-case and average-case performance can be achieved. We also discussed several approaches for reducing the number of objective functions, improving both running times and theoretical guarantees.

We extensively evaluated our algorithm on several real-world problems. For Gaussian Process regression, for example, we showed that SATURATE compares favorably to state-of-the-art heuristics, while being simpler, faster, and providing theoretical guarantees. For robust experimental design, SATURATE performs favorably compared to SDP based approaches. We believe that the ideas developed in this chapter will help the development of robust monitoring systems and provide new insights for adapting machine learning algorithms to cope with adversarial environments.

Chapter 11

Sensing under Complex Cost Functions

Networks of small, wireless sensors are becoming increasingly popular for monitoring spatial phenomena, such as the temperature distribution in a building [Deshpande et al., 2004]. In the previous chapters of this Thesis, we have studied the problem of selecting the most informative locations to deploy the sensors at. However, the use of wireless communication to collect data leads to additional challenges. Poor link qualities, which can arise if sensors are too far apart, or due to obstacles such as walls or radiation from appliances, can cause message loss and hence require a large number of retransmissions in order to collect the data effectively. Such retransmissions can consume battery power drastically, and hence decrease the overall deployment lifetime of the sensor network. This suggests that the communication cost is a fundamental constraint which must be taken into account when placing wireless sensors.

Existing work on sensor placement under communication constraints [Funke et al., 04, Gupta et al., 2003b, Kar and Banerjee, 2003] has considered the problem mainly from a geometric perspective: Sensors have a fixed *sensing region*, such as a disc with a certain radius, and can only communicate with other sensors which are at most a specified distance apart. In Chapter 6, we have argued that this sensing region assumption is often violated in practice, due to nonstationarity in the environment, and the disregard of the fact that multiple sensors can combine their measurements to improve their predictions.

The assumption that two sensors at fixed locations can either perfectly communicate (i.e., they are "connected") or not communicate at all (and are "disconnected") is equally unreasonable, as it does not take into account variabilities in the link quality due to moving obstacles (e.g., doors), interference with other radio transmissions, and packet loss due to reflections [Cerpa et al., 2005].

In this chapter, we address the problem of selecting sensor placements that are simultaneously informative, and achieve low communication cost. Note that this problem cannot be solved merely by first finding the most informative locations, and then connecting them up with the least cost—indeed, it is easy to construct examples where such a two-phase strategy performs very poorly. We also avoid the *connectedness* assumption (sensors are "connected" iff they can perfectly communicate): in this chapter, we use the *expected number of retransmissions* as a cost metric on the communication between two sensors. This cost metric directly translates to the deployment lifetime of the wireless sensor network. We propose to use the probabilistic framework of *Gaussian Processes* not only to model the monitored phenomena (as done in Chapter 6), but also to predict communication costs.

Given such a model for predicting communication cost, we can associate all sensing locations $\mathcal{V}$ with a graph, where the edge between node s and t encodes the expected communication cost between these

locations. The goal then is to pick nodes $\mathcal{A} \subseteq \mathcal{V}$ in the graph that simultaneously achieve high sensing quality $F(\mathcal{A})$, and low *connection cost* in the graph.

This problem formulation not just allows us to model sensor placement under communication constraints. For example, when using mobile robots to make observations, the chosen locations need to lie on a path, the length of which is bounded by fuel and time constraints. In this case, we can annotate the edges in the graph by the traveling cost, and attempt to find a path with bounded length that is informative as possible.

In this chapter, we present a novel algorithm for the problem of selecting locations that simultaneously achieve high sensing quality and low connection cost in a specified graph. More specifically, in this chapter, we present:

- A method for learning a probabilistic model of the expected communication cost between any two locations, in addition to the probabilistic model of the environment (as in Chapter 6) from a small, short-term initial deployment. These models, based on *Gaussian Processes*, allow to avoid strong assumptions previously made in the literature.
- A novel and efficient algorithm for *Sensor Placements at Informative and cost-Effective Locations* (pSPIEL), which is guaranteed to provide near-optimal placements for this hard problem.
- Extensive evaluations of our proposed approach on temperature and light prediction tasks, using data from real-world sensor network deployments, as well as on a precipitation prediction task in the Pacific Northwest.

11.1 Problem statement

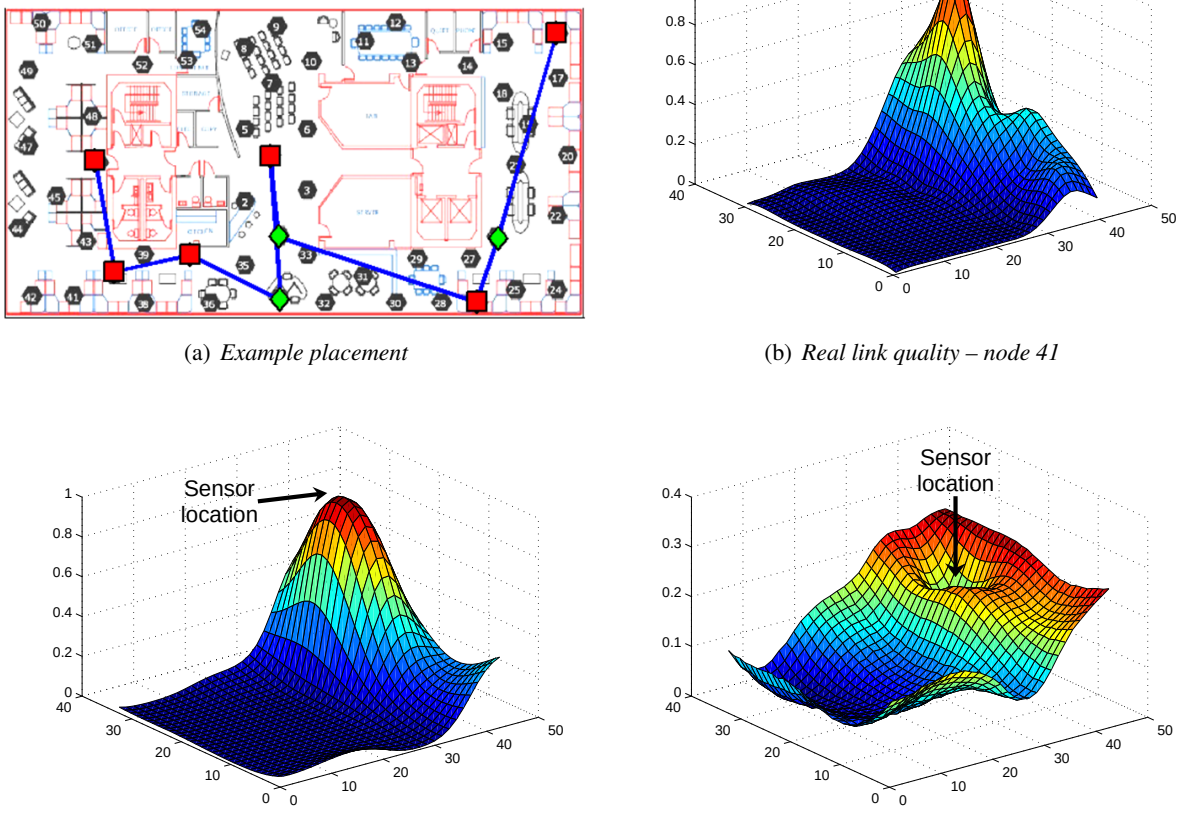
In this section, we briefly introduce the two fundamental quantities involved in optimizing sensor placements. A *sensor placement* is a finite subset of locations $\mathcal{A}$ from a ground set of possible sensor locations $\mathcal{V}$. Any possible placement is assigned a *sensing quality* $F(\mathcal{A}) \geq 0$, and a *communication cost* $\mathrm{C}(\mathcal{A}) \geq 0$, where the functions F and C will be defined presently. We will use a temperature prediction task as a running example: In this example, our goal is to deploy a network of wireless sensors in a building in order to monitor the temperature field, e.g., to actuate the air conditioning or heating system. Here, the sensing quality refers to our temperature prediction accuracy, or any other function considered in Part II of this Thesis. The communication cost depends on how efficiently the sensors communicate with each other. More generally, we investigate the problem of solving optimization problems of the form

$$\min_{\mathcal{A} \subseteq \mathcal{V}} \mathrm{C}(\mathcal{A}) \text{ subject to } F(\mathcal{A}) \geq Q, \tag{11.1}$$

for some *quota* $Q > 0$, which denotes the required amount of certainty achieved by any sensor placement. This optimization problem aims at finding the minimum cost placement that provides a specified amount of certainty Q, and is called the *covering problem*. We also address the dual problem of solving

$$\max_{\mathcal{A} \subseteq \mathcal{V}} F(\mathcal{A}) \text{ subject to } \mathrm{C}(\mathcal{A}) \leq B, \tag{11.2}$$

for some *budget* $B > 0$. This optimization problem aims at finding the most informative placement subject to a budget on the communication cost, and is called the *maximization problem*. In practice, one



(a) *Example placement*

(b) *Real link quality – node 41*

(c) *GP link quality – node 41*

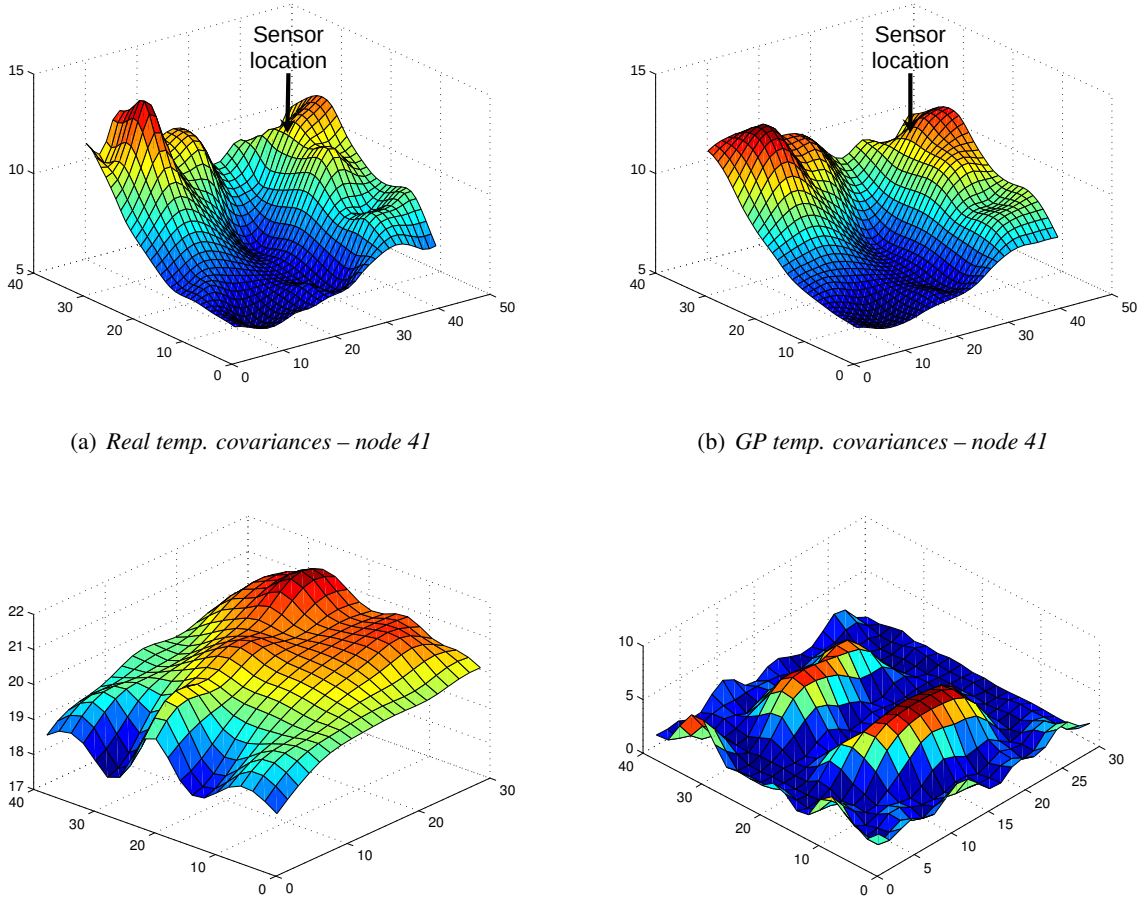
(d) *Link quality variance – node 41*

Figure 11.1: (a) Indoor deployment of 54 nodes and an example placement of six sensors (squares) and three relay nodes (diamonds); (b) measured trasmission link qualities for node 41; (c) GP fit of link quality for the same node and (d) shows variance of this GP estimate.

would often want to specify a particular location $s \in \mathcal{V}$ that must be contained in the solution $\mathcal{A}$. This requirement arises, for example, if the deployed network needs to be connected to a base station that is positioned at a fixed location s. We call the optimization problem (11.1) (resp. (11.2)) that includes such an additional constraint a *rooted* covering (resp. *rooted* maximization) problem. In this chapter, we present efficient approximation algorithms for both the covering and maximization problems, in both the rooted and unrooted formulations.

11.1.1 What is communication cost?

Since each transmission drains battery of the deployed sensors, we have to ensure that our sensor placements have reliable communication links, and the number of unnecessary retransmissions is minimized. If the probability for a successful transmission between two sensor locations s and t is $\theta_{s,t}$, the expected number of retransmissions is $1/\theta_{s,t}$. Since we have to predict the success probability between any two



(a) *Real temp. covariances – node 41*

(b) *GP temp. covariances – node 41*

(c) *GP prediction of temp. surface*

(d) *Variance of GP temp. surface*

Figure 11.2: (a) Measured temperature covariance between node 41 and other nodes in the deployment; (b) predicted covariance using non-stationary GP; (c) predicted temperatures for sensor readings taken at noon on February 28th 2004, and (d) shows the variance of this prediction.

locations s, $t \in \mathcal{V}$, we will in general only have a distribution $P(\theta_{s,t})$ with density $p(\theta_{s,t})$ instead of a fixed value for $\theta_{s,t}$. Surprisingly, this uncertainty has a fundamental effect on the expected number of retransmissions. For a simple example, assume that with probability $\frac{1}{2}$ we predict that our transmission success rate is $\frac{3}{4}$, and with probability $\frac{1}{2}$, it is $\frac{1}{4}$. Then, the mean transmission rate would be $\frac{1}{2}$, leading us to assume that the expected number of retransmissions might be 2. In expectation over the success rate however, our expected number of retransmissions becomes $\frac{1}{2} \cdot 4 + \frac{1}{2} \cdot \frac{4}{3} = 2 + \frac{2}{3} > 2$. More generally, the expected number is

$$\mathrm{C}(\{s,t\}) = \int_{\theta} \frac{1}{\theta_{s,t}} p(\theta_{s,t}) d\theta_{s,t}. \tag{11.3}$$

Using this formula, we can compute the expected number of retransmissions for any pair of locations. If $\mathcal{V}$ is finite, we can model all locations in $\mathcal{V}$ as nodes in a graph $\mathcal{G} = (\mathcal{V}, E)$, with the edges E labeled by their communication costs. We call this graph the *communication graph* of $\mathcal{V}$. For any sensor placement $\mathcal{A} \subseteq \mathcal{V}$, we define its cost by the minimum cost tree $\mathcal{T}$, $\mathcal{A} \subseteq \mathcal{T} \subseteq \mathcal{V}$, connecting all sensors $\mathcal{A}$ in the

communication graph for $\mathcal{V}$. (In general, the locations $\mathcal{A}$ may include distant sensors, requiring us to place *relay nodes*, which do not sense but only aid communication.) Finding this minimum cost tree $\mathcal{T}$ to evaluate the cost function $\mathrm{C}(\mathcal{A})$ is called the *Steiner tree* problem; an $\mathbf{NP}$-complete problem that has very good approximation algorithms [Vazirani, 2003]. Our algorithm, pSPIEL, will however not just find an informative placement and then simply add relay nodes, since the resulting cost may be exorbitant. Instead, it *simultaneously* optimizes sensing quality and communication cost.

Note that it if we threshold all link qualities at some specified cut-off point, and define the edge costs between two locations in the communication graph as 1 if the link quality is above the cut-off point, and infinite if the link quality is below the cut-off point, then the communication cost of a sensor placement is exactly (one less than) the number of placed sensors. Hence, in this special case, we can interpret the maximization problem (11.2) as the problem of finding the most informative connected sensor placement of at most $B + 1$ nodes.

11.1.2 Overview of our approach

Having established the notions of sensing quality and communication cost, we now present an outline of our proposed approach.

1. We collect sensor and link quality data from an initial deployment of sensors. From this data, we learn probabilistic models for the sensor data (such as Gaussian Process models as considered in Chapter 6) and the communication cost (which we will discuss below). Alternatively, we can use expert knowledge to design such models.

2. These models allow us to predict the sensing quality $F(\mathcal{A})$ and communication cost $\mathrm{C}(\mathcal{A})$ for any candidate placement $\mathcal{A} \subseteq \mathcal{V}$.

3. Using pSPIEL, our proposed algorithm, we then find highly informative placements which (approximately) minimize communication cost. We can approximately solve both the covering and maximization problems.

4. After deploying the sensors, we then possibly add sensors or redeploy the existing sensors, by restarting from Step 2), until we achieve a satisfactory placement. (This step is optional.)

Consider our temperature prediction example. Here, in step 1), we would place a set of motes throughout the building, based on geometrical or other intuitive criteria. After collecting training data consisting of temperature measurements and packet transmission logs, in step 2), we learn probabilistic models from the data. This process is explained in the following sections. Figure 11.2(c) and Figure 11.2(d) present examples of the mean and variance of our model learned during this step. As expected, the variance is high in areas where no sensors are located. In step 3), we would then explore the sensing quality tradeoff for different placements proposed by pSPIEL, and select an appropriate one. This placement automatically suggests if relay nodes should be deployed. After deployment, we can collect more data, and, if the placement is not satisfactory, iterate by repeating from step 2).

11.2 Predicting communication cost

As discussed in Section 11.1.1, an appropriate measure for communication cost is the expected number of retransmissions. If we have a probability distribution $P(\theta_{s,t})$ over transmission success probabilities $\theta_{s,t}$, Equation (11.3) can be used in a Bayesian approach to compute the expected number of retransmissions. The problem of determining such predictive distributions for transmission success probabilities is very similar to the problem of estimating predictive distributions for the sensor values as discussed in Chapter 6, suggesting the use of GPs for predicting link qualities. A closer look however shows several qualitative differences: When learning a model for sensor values, samples from the actual values can be obtained. In the link quality case however, we can only determine whether certain messages between nodes were successfully transmitted or not. Additionally, transmission success probabilities are constrained to be between 0 and 1. Fortunately, GPs can be extended to handle this case as well [Csato et al., 2000]. In this *classification* setting, the predictions of the GP are transformed by the sigmoid, also called link function, $f(x) = \frac{1}{1+\exp(-x)}$. For large positive values of x, $f(x)$ is close to 1, for large negative values it is close to 0 and $f(0) = \frac{1}{2}$.

Since we want to predict link qualities for every *pair* of locations in $\mathcal{V}$, we define a random process $\Theta(s,t) = f(W(s,t))$, where $W(s,t)$ is a GP over $(s,t) \in \mathcal{V}^2$. We call $\Theta(s,t)$ the *link quality process*. This process can be learned the following way. In our initial deployment, we let each sensor broadcast a message once every epoch, containing its identification number. Each sensor also records, from which other sensors it has received messages this epoch. This leads to a collection of samples of the form $(s_{i,k}, s_{j,k}, \theta_k(s_i, s_j))_{i,j,k}$, where i, j range over the deployed sensors, k ranges over the epochs of data collection, and $\theta_k(s_i, s_j)$ is 1 if node i received the message from node j in epoch k, and 0 otherwise. We will interpret $\theta_k(s_i, s_j)$ as samples from the link quality process $\Theta(\cdot, \cdot)$. Using these samples, we want to compute predictive distributions similar to those described in Equations (6.1) and (6.2). Unfortunately, in the classification setting, the predictive distributions cannot be computed in closed form anymore, but one can resort to approximate techniques [*c.f.*, Csato et al., 2000]. Using these techniques, we infer the link qualities by modeling the underlying GP $W(s,t)$. Intuitively, the binary observations will be converted to "hallucinated" observations of $W(s,t)$, such that $\Theta(s,t) = f(W(s,t))$ will correspond to the empirical transmission probabilities between locations s and t. We now can use Equations (6.1) and (6.2) to compute the predictive distributions $W(s,t)$ for *any* pair of locations $(s,t) \in \mathcal{V}^2$. Applying the sigmoid transform will then result in a probability distribution over transmission success probabilities. In our implementation, instead of parameterizing $W(s,t)$ by pairs of coordinates, we use the parametrization $W(t-s, s)$. The first component of this parametrization is the displacement the successful or unsuccessful message has traveled, and the second component is the actual set of physical coordinates of the transmitting sensor. This parametrization tends to exhibit better generalization behavior, since the distance to the receiver (component 1) is the dominating feature, when compared to the spatial variation in link quality. Figure 11.1(c) shows an example of the predicted link qualities using a GP for our indoors deployment, Figure 11.1(d) shows the variance in this estimate.

What is left to do is to compute the expected number of retransmissions, as described in formula (11.3). Assuming the predictive distribution for $W(s,t)$ is normal with mean μ and variance σ^2, we compute $\int \frac{1}{f(x)} \mathcal{N}(x; \mu, \sigma^2) dx = 1 + \exp(-\mu + \sigma^2)$, where $\mathcal{N}(\cdot; \mu, \sigma^2)$ is the normal density with mean μ and variance σ^2. Hence we have a closed form solution for this integral. If $\sigma^2 = 0$, we simply retain that the expected number of retransmissions is the inverse of the transmission success probability. If σ^2 is very large however, the expected number of retransmission drastically increases. This implies that even if we predict the transmission success probability to be reasonably high, e.g., $2/3$, if we do not have enough

Algorithm 11.1: Greedy algorithm for maximizing mutual information.

Input: Locations $\mathcal{C} \subseteq \mathcal{V}$
Output: Greedy sequence $g_1, g_2, \ldots, g_{|\mathcal{C}|}$, $\mathcal{C}_i = \{g_1, \ldots, g_i\}$
begin
  $\mathcal{C}_0 \leftarrow \emptyset$;
  for *j = 1* **to** $|\mathcal{C}|$ **do**
    $g_j \leftarrow \underset{g \in \mathcal{C} \setminus \mathcal{C}_{j-1}}{\operatorname{argmax}} F(\mathcal{C}_{j-1} \cup \{g\}) - F(\mathcal{C}_{j-1})$;
    $\mathcal{C}_j \leftarrow \mathcal{C}_{j-1} \cup g_j$;
  end
end

samples to back up this prediction and hence our predictive variance σ^2 is very large, we necessarily have to expect the worst for the number of retransmissions. So, using this GP model, we may determine that it is better to select a link with success probability $1/3$, about which we are very certain, to a link with a higher success probability, but about which we are very uncertain. Enabling this tradeoff is a great strength of using GPs for predicting communication costs.

11.3 Problem structure in sensor placement under complex cost functions

We now address the covering and maximization problems described in Section 11.1. In Part II of this Thesis, we have seen that many interesting sensing quality functions, such as mutual information, satisfy submodularity, an intuitive diminishing returns property. Sensing quality is also (approximately) nondecreasing. We have also seen that for such functions, when attempting to select the best set of k observations, the simple greedy algorithm (Algorithm 5.1) finds a solution that obtains at least a constant fraction of $(1 - 1/e)$ of the optimal sensing quality.

Unfortunately, the near-optimality of the greedy algorithm only holds when we do not take communication cost into account, and does not generalize to the covering and maximization problems (11.1) and (11.2) we study in this chapter. Indeed, since the greedy algorithm does not take distances into account, it would prefer to place two highly informative sensors very far apart (in order to achieve the quota Q), whereas a cheaper solution may select three sensors which are slightly less informative (still satisfying the quota), but which are closer together. In Section 11.5 we show that even a modified version of the greedy algorithm naturally taking into account communication cost can provide very poor solutions.

In addition to *submodularity*, the mutual information criterion empirically (*c.f.*Figure 11.6(d)) exhibits another important *locality* property: Sensors which are very far apart are approximately independent. This implies that if we consider placing a subset of sensors $\mathcal{A}_1$ in one area of the building, and $\mathcal{A}_2$ in another area, then $F(\mathcal{A}_1 \cup \mathcal{A}_2) \approx F(\mathcal{A}_1) + F(\mathcal{A}_2)$. Here, we will abstract out this property to assume that there are constants $r > 0$ and $0 < \gamma \leq 1$, such that for any subsets of nodes $\mathcal{A}_1$ and $\mathcal{A}_2$ which are at least distance r apart, $F(\mathcal{A}_1 \cup \mathcal{A}_2) \geq F(\mathcal{A}_1) + \gamma F(\mathcal{A}_2)$. Such a submodular function F will be called (r, γ)-*local*.

11.3.1 Examples of (r, γ)-local submodular functions

There are several important examples of monotonic, submodular and (r, γ)-local objective functions:

Geometric coverage Suppose that with each location $s \in \mathcal{V}$, we associate a sensing region $B_s \subseteq \mathcal{V}$. Then the function

$$F(\mathcal{A}) = \left| \bigcup_{s \in \mathcal{A}} B_s \right|$$

measures the size of the region covered by placing sensors at locations $\mathcal{A}$. In this case, F is monotonic, submodular and $(r, 1)$-local, where

$$r = 2 \max_s \max_{i,j \in B_s} d(i, j)$$

is twice the maximum diameter of the sensing regions. This objective function F captures the commonly used disk model.

Probabilistic detections Suppose, we want to place sensors for event detection (e.g., detecting fires in buildings), and that a sensor placed at a location s can detect events at distance d with probability $0 \leq \varphi_s(d) \leq 1$, where φ_s is a monotonically decreasing function. Further suppose that we place sensors at locations $\mathcal{A}$. Then, if each sensor detects independently of the others, the probability of detecting an event happening at location $t \in \mathcal{V}$ is

$$F_t(\mathcal{A}) = 1 - \prod_{s \in \mathcal{A}} (1 - \varphi_s(d(s, t))).$$

Now let $r = 2\varphi^{-1}(1/2)$, i.e., the distance at which detection happens with probability $1/2$. Then F_t is a monotonic, submodular and $(r, 1/2)$-local. Now suppose we have a distribution $Q(t)$ over the possible outbreak locations. Then the expected detection performance

$$F(\mathcal{A}) = \sum_t Q(t) F_t(\mathcal{A})$$

is, as a convex combination of monotonic, submodular and $(r, 1/2)$-local objectives also monotonic, submodular and $(r, 1/2)$-local.

Mutual information Suppose $P(\mathcal{X}_\mathcal{V})$ is a GP with compact kernel, i.e., there is a constant r such that $\mathcal{K}(s, t) = 0$ whenever $d(s, t) \geq r/2$. Then the mutual information $F(\mathcal{A}) = H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}}) - H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}} \mid \mathcal{X}_\mathcal{A})$ is $(r, 1)$-local. Even if the kernel is not compact, mutual information empirically exhibits (r, γ)-locality (*c.f.*, Figure 11.6(d)).

The theory developed in this chapter applies to any objective function which satisfies nondecreasing submodularity and (r, γ)-locality. In our experiments, we use the mutual information criterion

$$F(\mathcal{A}) = F_{MI}(\mathcal{A}) = H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}}) - H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}} \mid \mathcal{X}_\mathcal{A}).$$

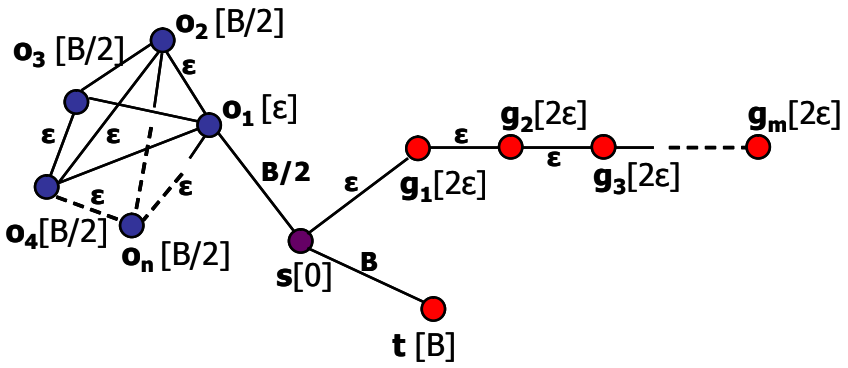


Figure 11.3: Example demonstrating the poor performance of the greedy algorithm.

11.3.2 The greedy algorithm

If we are interested in finding the set of k locations maximizing the sensing quality $F(\mathcal{A})$, irrespective of the cost $\mathrm{C}(\mathcal{A})$, we could use the greedy algorithm discussed in Chapter 5, which gives a near-optimal solution to the problem of finding a set of k locations maximizing the sensing quality.

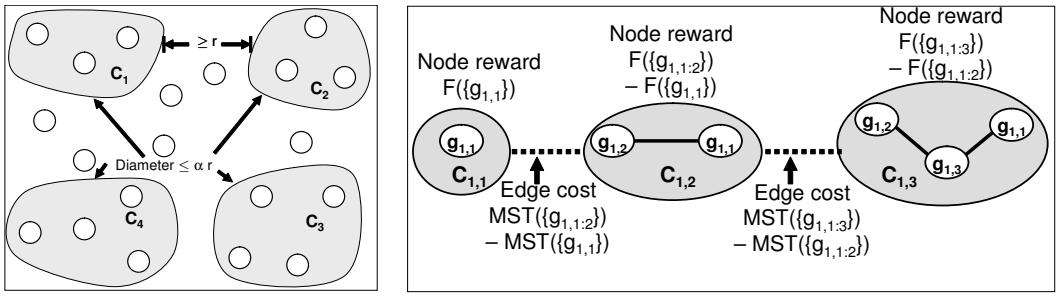
Unfortunately, the near-optimality of the greedy algorithm only holds when we do not take communication cost into account, and does not generalize to the covering and maximization problems (11.1) and (11.2) which we study in this chapter. For an illustration, consider Figure 11.3. In this illustration, we consider an additive (a special case of a submodular) sensing quality function F defined on the ground set $\mathcal{V} = \{s, t, o_1, \ldots, o_n, g_1, \ldots, g_m\}$, where the sensing quality of a selected set $\mathcal{A}$ of nodes is the sum of the values in squared brackets associated with the selected nodes (e.g., $F(\{s, o_1, g_1\}) = 3\varepsilon$). Consider the setting where we want to solve the maximization problem with root s and budget B. The optimal solution would be to choose the set $\mathcal{A}^* = \{s, o_1, \ldots, o_{B/(2\varepsilon)}\}$, with value $F(\mathcal{A}^*) = (B/\varepsilon - 1)(B/2) + \varepsilon$. The simple greedy algorithm that ignores cost would first pick t, and hence immediately run out of budget, returning set $\mathcal{A}_G = \{s, t\}$ with total sensing quality B. A greedy algorithm that takes the cost into account greedily selecting the element

$$s^* = \operatorname*{argmax}_{s \in \mathcal{V} \setminus \mathcal{A}} \frac{F(\mathcal{A} \cup \{s\} - F(\mathcal{A})}{\mathrm{C}(\mathcal{A} \cup \{s\}) - \mathrm{C}(\mathcal{A})}$$

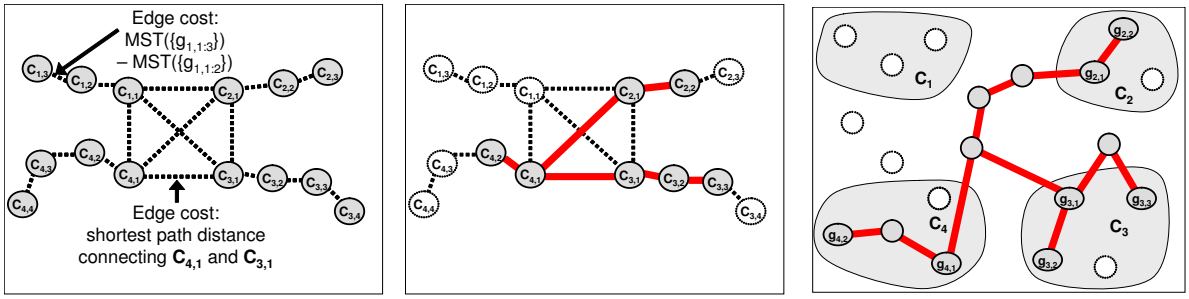
and hence optimizing the benefit/cost ratio of the chosen element s^* would select the set $\mathcal{A}_{GCB} = \{s, g_1, \ldots, g_{B/\varepsilon}\}$ with total value $2B$. Hence, as $\varepsilon \to 0$, the greedy algorithm performs arbitrarily worse than the optimal solution. In Section 11.5 we show that this poor performance of the greedy algorithm actually occurs in practice.

11.4 Approximation algorithm

In this section, we propose an efficient, non-greedy, approximation algorithm for selecting Padded Sensor Placements at Informative and cost-Effective Locations (PSPIEL). Our algorithm exploits problem



(a) *Padded decomposition* (b) *Chain in modular approximation graph for cluster 1*



(c) *Modular approximation graph* (d) *Quota-MST solution on MAG* (e) *Final solution*

Figure 11.4: Illustration of our algorithm: (a) presents a padded decomposition into four clusters; (b) displays the chain in the modular approximation graph associated with cluster 1; (c) shows the modular approximation graph with chains induced by greedy algorithm and the complete "core"; (d) the solution of the Quota-MST problem on the modular approximation graph; and (e) is the final solution after expanding the Quota-MST edges representing shortest paths.

structure via *submodularity* and *locality*, both properties described in Section 11.3. Before presenting our results and performance guarantees, here is an overview of our algorithm.

1. We randomly select a decomposition of the possible locations $\mathcal{V}$ into *small* clusters using Algorithm 11.2 (*c.f.*Figure 11.4(a), Section 11.4.1, Gupta et al. [2003a]). Nodes close to the "boundary" of their clusters are stripped away and hence the remaining clusters are "well-separated". (We prove that not too many nodes are stripped away). The well-separatedness and the locality property of F ensure the clusters are approximately independent, and hence very informative. Since the clusters are small, we are not concerned about communication cost within the clusters.

2. We use the greedy algorithm (Algorithm 11.1) within each cluster i to get an order $g_{i,1}, g_{i,2}, \ldots g_{i,n_i}$ on the n_i nodes in cluster i. We call $z_i = g_{i,1}$ the *center* of cluster i. Create a chain for this cluster by connecting the vertices in this order, with suitably chosen costs for each edge $(g_{i,j}, g_{i,j+1})$, as in Figure 11.4(b). The submodularity of F ensures that the first k nodes in this chain are almost as informative as the best subset of k nodes in the cluster (*c.f.*, Chapter 6).

3. Create a "modular approximation graph" $\mathcal{G}'$ from $\mathcal{G}$ by taking all these chains, and creating a fully connected graph on the cluster centers $z_1, z_2, \ldots, z_m$, the first nodes of each chain. The edge costs $(z_i, z_{i'})$ correspond to the shortest path distances between z_i and $z_{i'}$, as in Figure 11.4(c).

4. We now need to decide how to distribute the desired quota to the clusters. Hence, we approxi-

Algorithm 11.2: Algorithm for computing padded decompositions.

Input: Graph $(\mathcal{V}, E)$, shortest path distance $d(\cdot,\cdot)$, $r > 0$, $\alpha \geq 64 \dim(\mathcal{V}, E)$
Output: (α, r)-padded decomposition $\mathcal{C} = \{\mathcal{C}_u : u \in \mathcal{U}\}$
begin
  repeat
    $\mathcal{C} \leftarrow \emptyset$; $r' \leftarrow \frac{\alpha r}{4}$; $\mathcal{U} \leftarrow \{$a random element in $\mathcal{V}\}$;
    while $\exists\, v \in \mathcal{V} :\ \forall u \in \mathcal{U}\ d(u,v) > r'$ **do** $\mathcal{U} \leftarrow \mathcal{U} \cup \{v\}$;
    $\pi \leftarrow$ random permutation on $\mathcal{U}$;
    $R \leftarrow$ uniform at random in $(r', 2r']$;
    foreach $u \in \mathcal{U}$ *according to* π **do**
      $\mathcal{C}_u \leftarrow \{v \in \mathcal{V} : d(u,v) < R,$ and $\forall u' \in \mathcal{U}$ appearing earlier that u in π , $d(u',v) \geq R\}$;
    end
  until *at least* $\frac{1}{2}$ *nodes* r*-padded* ;
end

mately solve the Quota-MST problem (for the covering version) or the Budget-MST problem (for the maximization problem) on $\mathcal{G}'$ [Garg, 2005, Johnson et al., 2000] (Figure 11.4(d)).

5. Expand the chosen edges of $\mathcal{G}'$ in terms of the shortest paths they represent in $\mathcal{G}$, as in Figure 11.4(e).

Suppose $n = |\mathcal{V}|$ is the number of nodes in $\mathcal{V}$, and $\mathcal{A}^*$ denotes the optimal set (for the covering or maximization problem), with cost ℓ^*. Finally, let $\dim(\mathcal{V}, E)$ be the *doubling dimension* of the data, which is constant for many graphs (and for costs that can be embedded in low-dimensional spaces), and is $\mathcal{O}(\log n)$ for arbitrary graphs [*c.f.*, Gupta et al., 2003a]. We prove the following guarantee:

Theorem 11.1. *Let $\mathcal{G} = (\mathcal{V}, E)$ be a graph and a F be a (r,γ)-local monotone submodular function on $\mathcal{V}$. Suppose $\mathcal{G}$ contains a tree $\mathcal{T}^*$ with cost ℓ^*, spanning a set $\mathcal{A}^*$. Then* pSPIEL *can find a tree $\mathcal{T}$ with cost $\mathcal{O}(r \dim(\mathcal{V}, E)) \times \ell^*$, spanning a set $\mathcal{A}$ with expected sensing quality $F(\mathcal{A}) \geq \Omega(\gamma) \times F(\mathcal{A}^*)$. The algorithm is randomized and runs in polynomial-time.* □

In other words, Theorem 11.1 shows that we can solve the covering and maximization problems (11.1) and (11.2) to provide a sensor placement for which the communication cost is at most a small factor (at worst logarithmic) larger, and for which the sensing quality is at most a constant factor worse than the optimal solution. The proof can be found in the Appendix. While the actual guarantee of our algorithm holds in expectation, running the algorithm a small (polynomial) number of times will lead to appropriate solutions with arbitrarily high probability. Details on this procedure can be found in Section 11.6. In the rest of this section, we flesh out the details of the algorithm, giving more technical insight and intuition about the performance of our approach.

11.4.1 Padded decompositions

To exploit the locality property, we would like to decompose our space into "well-separated" clusters; loosely, an r-padded decomposition is a way to do this so that most vertices of $\mathcal{V}$ lie in clusters $\mathcal{C}_i$ that are at least r apart. Intuitively, *padded decompositions* allow us to split the original placement problem into approximately independent placement problems, one for each cluster $\mathcal{C}_i$. This padding and the locality

property of the objective function F guarantee that, if we compute selections $\mathcal{A}_1, \ldots, \mathcal{A}_m$ for each of the m clusters separately, then it holds that $F(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_m) \geq \gamma \sum_i F(\mathcal{A}_i)$, i.e., we only lose a constant factor. An example is presented in Figure 11.4(a).

If we put all nodes into a single cluster, we obtain a padded decomposition that is not very useful. To exploit our locality property, we want clusters of size about r that are at least r apart. It is difficult to obtain separated clusters of size exactly r, but padded decompositions exist for arbitrary graphs for cluster sizes a constant α larger, where α is $\Omega(\dim(\mathcal{V}, E))$ [Gupta et al., 2003a]. We want small clusters, since we can then ignore communication cost within each cluster.

Formally, an (α, r)-padded decomposition is a probability distribution over partitions of $\mathcal{V}$ into *clusters* $\mathcal{C}_1, \ldots, \mathcal{C}_m$, such that:

(i) Every cluster $\mathcal{C}_i$ in the partition is guaranteed to have bounded diameter, i.e., $\operatorname{diam}(\mathcal{C}_i) \leq \alpha r$.

(ii) Each node $s \in \mathcal{V}$ is r-padded in the partition with probability at least ρ. (A node s is *r-padded* if all nodes t at distance at most r from s are contained in the same cluster as s.)

The parameter ρ can be chosen as a constant (in our implementation, $\rho = \frac{1}{2}$). In this chapter, we use the term padded decomposition to refer both to the distribution, as well as samples from the distribution, which can be obtained efficiently using Algorithm 11.2 [Gupta et al., 2003a]. In pSPIEL, for a fixed value of the locality parameter r, we gradually increase α, stopping when we achieve a partition, in which at least half the nodes are r-padded. This rejection sampling is the only randomized part of our algorithm, and, in expectation, the number of required samples is polynomial.

Our algorithm strips away nodes that are not r-padded, suggesting a risk of missing informative locations. The following Lemma proves that we will not lose significant information in expectation.

Lemma 11.2. *Consider a submodular function $F(\cdot)$ on a ground set $\mathcal{V}$, a set $\mathcal{B} \subseteq \mathcal{V}$, and a probability distribution over subsets $\mathcal{A}$ of $\mathcal{B}$ with the property that, for some constant ρ, we have $\Pr[v \in \mathcal{A}] \geq \rho$ for all $v \in \mathcal{B}$. Then $\mathbb{E}[F(\mathcal{A})] \geq \rho F(\mathcal{B})$.* □

The proof of this Lemma appears in the Appendix. Let $\mathcal{A}^*$ be the optimal solution for the covering or maximization problem, and let $\mathcal{A}^*_r$ denote a subset of nodes in $\mathcal{A}^*$ that are r-padded. Lemma 11.2 proves that, in expectation, the information provided by $\mathcal{A}^*_r$ is at most a constant factor ρ worse than $\mathcal{A}^*$. Since the cost of collecting data from $\mathcal{A}^*_r$ is no larger than that of $\mathcal{A}^*$, this lemma shows that our padded decomposition preserves near-optimal solutions.

11.4.2 The greedy algorithm

After having sampled a padded decomposition, we run the greedy algorithm as presented in Algorithm 11.1 on the r-padded nodes in each cluster $\mathcal{C}_i$, with k set to n_i, the number of padded elements in cluster $\mathcal{C}_i$. Let us label the nodes as $g_{i,1}, g_{i,2}, \ldots, g_{i_{n_i}}$ in the order they are chosen by the greedy algorithm, and let $\mathcal{C}_{i,j} = \{g_{i,1}, \ldots, g_{i,j}\}$ denote the greedy set after iteration j. From Chapter 6 we know that each set $\mathcal{C}_{i,j}$ is at most a factor $(1 - 1/e)$ worse than the optimal set of j padded elements in that cluster. Furthermore, from (r, γ)-locality and using the fact that the nodes are r-padded, we can prove that

$$F(\mathcal{C}_{1,j_1} \cup \cdots \cup \mathcal{C}_{m,j_m}) \geq \gamma \sum_{k=1}^{m} F(\mathcal{C}_{k,j_k}) \geq \gamma \left(1 - \tfrac{1}{e}\right) \sum_{k=1}^{m} F(\mathcal{C}^*_{k,j_k})$$

for any collection of indices $j_1, \ldots, j_m$, where $\mathcal{C}^*_{k,j_k}$ denotes the optimal selection of j_k nodes within cluster k.

11.4.3 The modular approximation graph $\mathcal{G}'$

In step 3), pSPIEL creates the auxiliary *modular approximation graph* (MAG) $\mathcal{G}'$ from $\mathcal{G}$. Intuitively, this MAG will approximate $\mathcal{G}$, such that running the Quota-MST algorithm on it will decide how many nodes should be picked from each cluster. The nodes of $\mathcal{G}'$ are the greedy sets $\mathcal{C}_{i,j}$. The greedy sets for cluster i are arranged in a chain with edge $e_{i,j}$ connecting $\mathcal{C}_{i,j}$ and $\mathcal{C}_{i,j+1}$ for every i and j. For a set of nodes $\mathcal{B}$, if $\mathrm{C}_{MST}(\mathcal{B})$ is the cost of a minimum spanning tree (MST) connecting the nodes in $\mathcal{B}$ by their shortest paths, the weight of $e_{i,j}$ in $\mathcal{G}'$ is the difference in costs of the MSTs of $\mathcal{C}_{i,j}$ and $\mathcal{C}_{i,j+1}$ (or 0 if this difference becomes negative), i.e., $\mathrm{C}(e_{i,j}) = \max\left[\mathrm{C}_{MST}(\mathcal{C}_{i,j+1}) - \mathrm{C}_{MST}(\mathcal{C}_{i,j}), 0\right]$. We also associate a "reward" $\mathrm{reward}(\mathcal{C}_{i,j}) = F(\mathcal{C}_{i,j}) - F(\mathcal{C}_{i,j-1})$ with each node, where $F(\mathcal{C}_{i,0}) \triangleq 0$. Note that, by telescopic sum, the total reward of the first k elements in chain i is $F(\mathcal{C}_{i,k})$, and the total cost of the edges connecting them is $\mathrm{C}_{MST}(\mathcal{C}_{i,k})$, which is at most 2 times the the cost of a minimum Steiner tree connecting the nodes in $\mathcal{C}_{i,k}$ in the original graph $\mathcal{G}$. By property (i) of the padded decomposition, $\mathrm{C}_{MST}(\mathcal{C}_{i,k}) \leq \alpha\ r\ k$. By associating these rewards with each node, we define a *modular* set function F' on $\mathcal{G}'$, such that for a set $\mathcal{B}$ of nodes in $\mathcal{G}'$, its value $F'(\mathcal{B})$ is the sum of the rewards of all elements in $\mathcal{B}$. Figure 11.4(b) presents an example of a chain associated with cluster 1 in Figure 11.4(a). Additionally, we connect every pair of nodes $\mathcal{C}_{i,1}, \mathcal{C}_{j,1}$ with an edge with cost being the shortest path distance between $g_{i,1}$ and $g_{j,1}$ in $\mathcal{G}$. This fully connected subgraph is called the *core* of $\mathcal{G}'$. Figure 11.4(c) presents the modular approximation graph associated with the padded decomposition of Figure 11.4(a).

11.4.4 Solving the covering and maximization problems in $\mathcal{G}'$

The modular approximation graph $\mathcal{G}'$ reduces the problem of optimizing a submodular set function in $\mathcal{G}$ to one of optimizing a *modular* set function F' (where the value of a set is the sum of rewards of its elements) in $\mathcal{G}'$ to minimize communication costs. This is a well studied problem, and constant factor approximation algorithms have been found for the covering and maximization problems. The (rooted) *Quota-MST* problem asks for a minimum weight tree $\mathcal{T}$ (with a specified root), in which the sum of rewards exceeds the specified quota. Conversely, the *Budget-MST* problem desires a tree of maximum reward, subject to the constraint that the sum of edge costs is bounded by a budget. The best known approximation factors for these problems is 2 for rooted Quota-MST [Garg, 2005], and $3 + \varepsilon$ (for any $\varepsilon > 0$) for unrooted Budget-MST [Levin, 2004]. We can use these algorithms to get an approximate solution for the covering and maximization problems in $\mathcal{G}'$. From Section 11.4.3, we know that it suffices to decide which chains to connect, and how deep to descend into each chain; any such choice will give a subtree of $\mathcal{G}'$. To find this tree, we consider all $\mathcal{C}_{i,1}$ for each i as possible roots, and choose the best tree as an approximate solution. (For the Budget-MST problem, we only have an unrooted algorithm, but we can use the structure of our modular approximation graph to get an approximately optimal solution.) Figure 11.4(d) illustrates such a Quota-MST solution.

11.4.5 Transferring the solution from $\mathcal{G}'$ back to $\mathcal{G}$

The Quota- or Budget-MST algorithms select a tree $\mathcal{T}'$ in $\mathcal{G}'$, which is at most a constant factor worse than the optimal such tree. We use this solution $\mathcal{T}'$ obtained for $\mathcal{G}'$ to select a tree $\mathcal{T} \subseteq \mathcal{G}$: For every cluster i, if $\mathcal{C}_{i,j} \in \mathcal{T}'$ we mark $g_{i,1}, \ldots, g_{i,j}$ in $\mathcal{G}$. We then select $\mathcal{T}$ to be an approximately optimal Steiner tree connecting all marked nodes in $\mathcal{G}$, obtained, e.g., by computing an MST for the fully connected graph over all marked vertices, where the cost of an edge between s and t is the shortest path distance between these

nodes in $\mathcal{G}$. This tree $\mathcal{T}$ is the approximate solution promised in Theorem 11.1. (Figure 11.4(e) presents the expansion of the Quota-MST from Figure 11.4(d).)

11.4.6 Additional implementation details

pSPIEL relies heavily on the nondecreasing submodularity and locality assumptions. In practice, since we may not know the constants r and γ, we run the algorithm multiple times with different choice for r. Since the algorithm is randomized, we repeat it several times to achieve a good solution with high probability. Finally, since we do not know γ, we cannot directly specify the desired quota when solving the covering problem. To alleviate all these intricacies, we use the following strategy to select a good placement: For a fixed number of iterations, randomly sample an r between 0 and the diameter of $\mathcal{G}$. Also sample a quota Q between 0 and $Q_{\max}$, the maximum submodular function value achieved by the unconstrained greedy algorithm. Run pSPIEL with these parameters r and Q, and record the actual placement, as well as the communication cost and sensing quality achieved by the proposed placement. After N iterations, these values result in a cost-benefit curve, which can be used to identify a good cost-benefit tradeoff as done in Section 11.5.

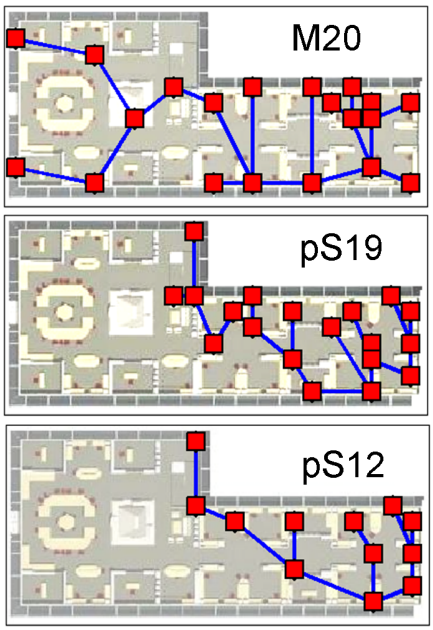
Also, note that a key step of pSPIEL is to run the greedy algorithm in each cluster. Using the technique of *lazy evaluations* as discussed in Section 5.4, this step can often be drastically sped up.

11.5 Experiments

In order to evaluate our method, we computed sensor placements for three real-world problems: Indoor illumination measurement, the temperature prediction task as described in our running example, and the prediction of precipitation in the United States' Pacific Northwest.

11.5.1 System implementation

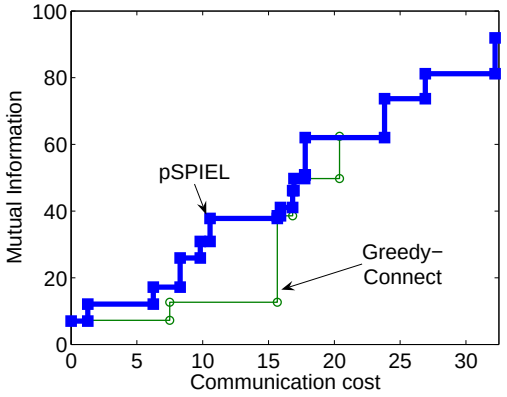
We developed a complete system implementation of our sensor placement approach, based on Tmote Sky motes. The data collection from the pilot deployment is based on the TinyOS SurgeTelos application, which we extended to collect link quality information. Once per epoch, every sensor sends out a broadcast message containing its unique identifier. Upon receipt of these messages, every sensor will compile a bit-string, indicating from which neighbor it has heard in the current epoch. This transmission log information will then be transmitted, along with the current sensor readings, via multi-hop routing to the base station. After enough data has been collected, we learn GP models for sensing quality and communication cost, which are subsequently used by the pSPIEL algorithm. Our implementation of pSPIEL uses a heuristically improved approximate k-MST algorithm as described by Johnson et al. [2000]. Using pSPIEL, we generate multiple placements and plot them in a trade-off curve as described in Section 11.4.6. We then identify an appropriate trade-off by selecting good placements from this trade-off curve.



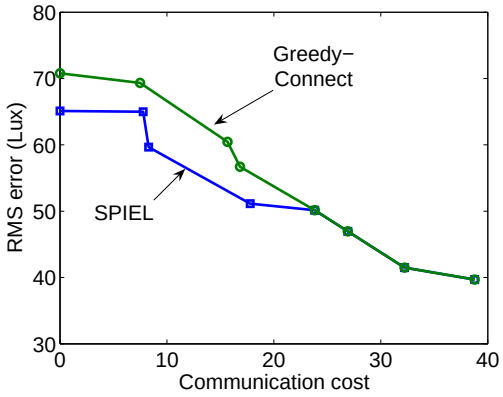
(a) *Placements*

Metric	M20	pS19	pS12
RMS	91.0	**51.2**	71.5
MAD	67.0	**31.3**	45.1
Pred. c.	24.4	19.9	**15.3**
Real c.	22.9	21.8	**15.0**

(b) *Costs and prediction qualities*



(c) *Cost-benefit for light data*



(d) *RMS error for light data*

Figure 11.5: Experimental results. (a) shows the expert placement (M20) as well as two placements proposed by PSPIEL, (pS19) and (pS12). (b) presents root-mean-squares (RMS) and mean-absolute-deviation (MAD) prediction errors for the manual placement and two placements from PSPIEL. (c) compares the cost-benefit tradeoff curves for the light data GP on a 187 points grid. (d) compares the root-mean-squares error for the light data.

11.5.2 Proof-of-concept study

As a proof-of-concept experiment, we deployed a network of 46 Tmote Sky motes in the Intelligent Workplace at CMU. As a baseline deployment, we selected 20 locations (M20) that seemed to capture the overall variation in light intensity. After collecting the total solar radiation data for 20 hours, we learned GP models, and used PSPIEL to propose a placement of 19 motes (pS19). Figure 11.5(a) shows

the 20 and 19 motes deployments. After deploying the competing placements, we collected data for 6 hours starting at 12 PM and compared the prediction accuracy for all placements, on validation data from 41 evenly distributed motes. Figure 11.5(b) presents the results. Interestingly, the proposed placement (pS19) drastically reduces the prediction error by about 50%. This reduction can be explained by the fact that there are two components in lighting: natural and artificial. Our baseline deployment placed sensors spread throughout the environment, and in many intuitive locations near the windows. On the other hand, pSPIEL decided not to explore the large western area, a part of the lab that was not occupied during the night, and thus had little fluctuation with artificial lighting. Focusing on the eastern part, pSPIEL was able to make sufficiently good natural light predictions throughout the lab, and better focus of the sources of variation in artificial light. We repeated the evaluation for a 12 motes subsample (pS12), also proposed by pSPIEL, which still provides better prediction than the manual placement of 20 nodes (M20), and significantly lower communication cost. We also compared the predicted communication cost using the GPs with the measured communication cost. Figure 11.5(b) shows that the prediction matches well to the measurement. Figs. 11.5(c) and 11.5(d) show that pSPIEL outperforms the Greedy heuristic explained below, both in the sensing quality and communication cost tradeoff and in predictive RMS error.

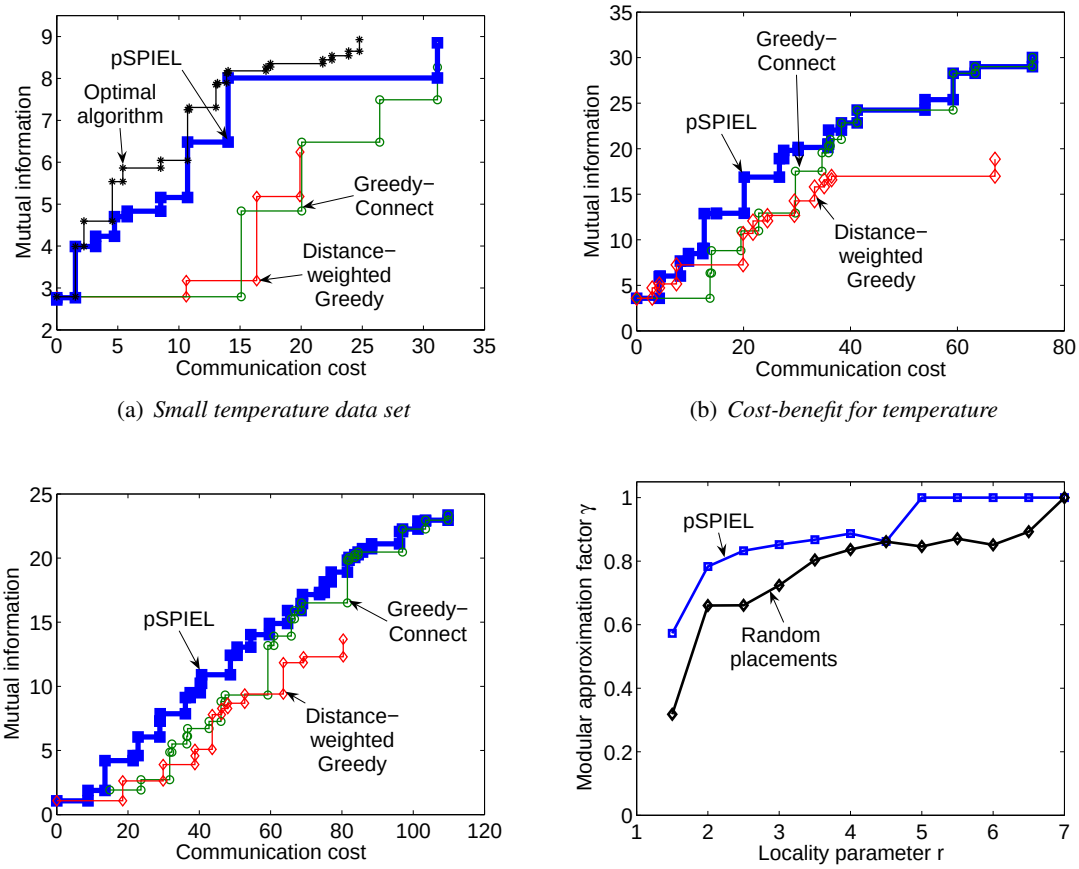
11.5.3 Indoor temperature measurements

In our second set of experiments, we used data from the deployment of 52 wireless sensor motes at Intel Research, Berkeley, as introduced in Chapter 6. In addition to learning a model for the temperature, we also learned a model for predicting communication costs. After learning the GP models from five days of data, we used pSPIEL to propose improved sensor placements. We compared pSPIEL to two heuristics, and—for small problems—with the optimal algorithm which exhaustively searches through all possible deployments. The first heuristic, *Greedy-Connect*, runs the unconstrained greedy algorithm (Algorithm 11.1), and then connects the selected sensors using a Steiner tree approximation. The second heuristic, *Distance-weighted Greedy*, is inspired by an algorithm that provides near-optimal solutions to the Quota-MST problem [Awerbuch et al., 1999]. This heuristic initially starts with all nodes in separate clusters, and iteratively merges – using the shortest path – clusters maximizing the following greedy criterion:

$$\text{gain}(\mathcal{C}_1, \mathcal{C}_2) = \frac{\min_{i \in 1,2}(F(\mathcal{C}_1 \cup \mathcal{C}_2) - F(\mathcal{C}_i))}{\text{dist}(\mathcal{C}_1, \mathcal{C}_2)}.$$

The intuition for this greedy rule is that it tries to maximize the benefit-cost ratio for merging two clusters. Since it works near-optimally in the modular case, we would hope it performs well in the submodular case also. The algorithm stops after sufficiently large components are generated [*c.f.*, Awerbuch et al., 1999].

Figure 11.6(a) compares the performance of pSPIEL with the other algorithms on a small problem with only 16 candidate locations. We used the empirical covariance and link qualities measured from 16 selected sensors. In this small problem, we could explicitly compute the optimal solution by exhaustive search. Figure 11.6(a) indicates that the performance of pSPIEL is significantly closer to the optimal solution than any of the two heuristics. Figure 11.6(b) presents a comparison of the algorithms for selecting placements on a 10×10 grid. We used our GP models to predict the covariance and communication cost for this discretization. From Figure 11.6(b) we can see that for very low quotas (less than 25% of the maximum), the algorithms performed very similarly. Also, for very large quotas (greater than 80%), pSPIEL does not significantly outperform not *Greedy-Connect*, since, when the environment is densely covered,



(a) *Small temperature data set*

(b) *Cost-benefit for temperature*

(c) *Cost-benefit for precipitation*

(d) (r, γ) *for temperature data*

Figure 11.6: Experimental results. (a) compares trade-off curves for a small subset of the temperature data. (b) shows tradeoff curves for the temperature GPs on a 10x10 grid. (c) compares tradeoffs for precipitation data from 167 weather stations. (d) compares the locality parameter r and the loss γ incurred by the modular approximation for the temperature GPs.

communication is not an issue. In fact, if the information quota requires a very dense deployments, the padded decomposition tends to strip away many nodes, leading pSPIEL to increase the locality constant r, until r is large enough to include all nodes are in a single cluster. In this case, pSPIEL essentially reverts back to the *Greedy-Connect* algorithm. In the important region between 25% and 80% however, pSPIEL clearly outperforms the heuristics. Our results also indicate that in this region the steepest drop in out-of-sample root mean squares (RMS) prediction accuracy occurs. This region corresponds to placements of approximately $10 - 20$ sensors, an appropriate number for the target deployment Figure 11.1(a).

In order to study the effect of the locality parameter r, we generated padded decompositions for increasing values of r. For random subsets of the padded nodes, and for placements from pSPIEL, we then compared the modular approximation, i.e., the sum of the local objective values per cluster, with the mutual information for the entire set of selected nodes. As r increases to values close to 2, the approximation factor γ drastically increases from .3 to .7 and then flattens as r encompasses the the entire graph $\mathcal{G}$. This suggests that the value $r = 2$ is an appropriate choice for the locality parameter, since it only incurs a small approximation loss, but guarantees small diameters of the padded clusters, thereby keeping communication

cost small. For placements proposed by pSPIEL, the approximation factor is even better.

11.5.4 Precipitation data

In our third application, our goal was to place sensors for the precipitation prediction task as introduced in Chapter 6. Since we did not have communication costs for this data set, we assumed that the link quality decayed as the inverse square of the distance, based on physical considerations. Figure 11.6(c) compares the sensing quality – communication cost tradeoff curves for selecting placements from all 167 locations. pSPIEL outperforms the heuristics up to very large quotas.

11.6 Robust Sensor Placements

In Section 11.5, we have seen that optimized placements can lead to much higher prediction accuracy and lower communication cost as compared to manual placements. However, such optimization can lead to negative effects if the model that the optimization is based on changes. For example, in our proof-of-concept study, it is conceivable that the building usage patterns change, and the western part of the building becomes occupied. In this case, the optimized placement will fail to capture important variations in light distribution. Intuitively, the manual placement (M20) should be able to capture such change of the environment better, as the sensors are more uniformly spread out. This intuitive assessment comes from our prior assumptions that, since lights spreads uniformly over space, a regularly-spaced distributed placement should be quite informative.

How can we place sensors that perform well both according to the current state of the world, as well as to possible future changes? One possibility is to require the sensor placement to perform well both according to our prior assumptions (i.e., favoring uniform placements) and to the data we collected so far. We can formalize this idea by defining two separate sensing quality functions, F_1 and F_2. $F_1(\mathcal{A})$ measures the informativeness of placement $\mathcal{A}$ under the isotropic prior. F_2 measures the informativeness according to the collected data, as described before. Assuming a priori that the phenomenon will always uniformly spread in the environment, we could choose F_1 to be the mutual information of an isotropic Gaussian process, as in Equation (6.4). Optimizing according to F_1 only would lead to sensor placements that are (asymptotically) distributed on a regular grid, and we would hence expect such placements to be robust against changes in the environment. F_2 is the mutual information according to the complex, data-dependent, nonstationary Gaussian process as considered in the earlier parts of this chapter. Optimizing for F_2 would lead to placements that exploit the correlations in the data collected from the pilot deployment. Based on these two objective functions, we would then like to find a placement $\mathcal{A}$ that jointly optimizes F_1 and F_2, i.e., which is both robust, and exploits correlations estimated from data.

More generally, we would like to solve the robust optimization problem

$$\min_{\mathcal{A}} \mathrm{C}(\mathcal{A}) \text{ such that for all } i, F_i(\mathcal{A}) \geq Q, \tag{11.4}$$

where $F_1, \ldots, F_m$ is a collection of monotonic submodular functions. Note that, unlike problem (11.1), in problem (11.4) there are now multiple submodular constraints $F_1(\mathcal{A}) \geq Q, \ldots, F_m(\mathcal{A}) \geq Q$. We can use the same idea that we used to develop the SATURATE algorithm in Chapter 10, which allows us to reduce problem (11.4) to problem (11.1): For each function F_i, define a new truncated objective function

$$\widehat{F}_{i,Q}(\mathcal{A}) = \min\{F_i(\mathcal{A}), Q\}.$$

It holds that whenever F_i is monotonic and submodular, $\widehat{F}_{i,Q}$ is monotonic and submodular as well [Fujito, 2000]. As a nonnegative linear combination of monotonic submodular functions, the function

$$\overline{F}_Q(\mathcal{A}) = \frac{1}{m}\sum_i \widehat{F}_{i,Q}(\mathcal{A})$$

is monotonic submodular as well. Furthermore, it holds that $\overline{F}_Q(\mathcal{A}) = Q$ if and only if $F_i(\mathcal{A}) \geq Q$ for all i. Hence, instead of problem (11.4), we can equivalently solve

$$\min_{\mathcal{A}} \mathrm{C}(\mathcal{A}) \text{ such that } \overline{F}_Q(\mathcal{A}) \geq Q,$$

which is an instance of problem (11.1). However, we cannot readily apply PSPIEL to this problem, since it is only guaranteed to return a solution such that, in expectation, $\overline{F}_Q(\mathcal{A}) \geq \beta Q$, for $\beta = (1-1/e)\gamma/2$ (from Theorem 11.1). Unfortunately, $\overline{F}_Q(\mathcal{A}) \geq \beta Q$ does not guarantee that $F_i(\mathcal{A}) \geq \beta Q$ for each i.

However, we can nevertheless use PSPIEL to solve problem (11.4). We first present an overview of our algorithm, and then discuss the details in Section 11.6.1.

1. We first convert PSPIEL into an algorithm, for which Theorem 11.1 holds not just in expectation, but with high probability. For arbitrary $\delta > 0$, this new algorithm will be guaranteed to provide, with probability at least $1-\delta$, a solution $\mathcal{A}$ such that $\overline{F}_Q(\mathcal{A}) \geq \beta Q/2$.
2. Call $Q_{togo} = Q - \overline{F}_Q(\mathcal{A})$ the remaining quota that still needs to be covered. When applying PSPIEL once, $Q_{togo} \leq Q(1-\beta/2)$. We show how we can iteratively apply PSPIEL to a modified objective function to obtain larger and larger sets $\mathcal{A}$ such that after k iterations $Q_{togo} \leq Q(1-\beta/2)^k$. Hence, after logarithmically many iterations, $Q_{togo} \leq \varepsilon/m$. This implies that $F_i(\mathcal{A}) \geq Q(1-\varepsilon)$ for all i.

11.6.1 Algorithm details

We first need to turn PSPIEL into an algorithm that solves problem (11.1) with high probability $1-\delta$. Let M be an upper bound on the optimal value. Then each run of PSPIEL returns a solution $\mathcal{A}_i$ with $0 \leq F(\mathcal{A}_i) \leq M$.

Lemma 11.3. *We need*

$$N = \left\lceil \frac{1}{2}\left(\frac{2M}{Q\beta}\right)^2 \log\frac{1}{\delta} \right\rceil$$

samples to guarantee a sample $\mathcal{A}_i$ with $\overline{F}_Q(\mathcal{A}_i) \geq \frac{\beta Q}{2}$ with probability $1-\delta$.

In order to guarantee that $\min_i F_i(\mathcal{A}_i) \geq (1-\varepsilon)Q$, we follow the following strategy: Let $\widetilde{F}^{(1)} = \overline{F}_Q$. We invoke random sampling of PSPIEL applied to $\widetilde{F}^{(1)}$ until we obtain a solution $\mathcal{A}_1$ such that $\widetilde{F}^{(1)}(\mathcal{A}_1) \geq \frac{\beta Q}{2}$. We then define a new monotonic submodular function, $\widetilde{F}^{(2)}(\mathcal{A}) = \overline{F}_Q(\mathcal{A}\cup\mathcal{A}_1) - \overline{F}_Q(\mathcal{A}_1)$. $\widetilde{F}^{(2)}$ is called a *residual submodular function*. We then repeatedly run PSPIEL to obtain a solution $\mathcal{A}_2$ such that $\widetilde{F}^{(2)}(\mathcal{A}_2) \geq \frac{\beta(Q-\widetilde{F}_1(\mathcal{A}_1))}{2}$. After k steps, we define the function

$$\widetilde{F}^{(k+1)}(\mathcal{A}) = \overline{F}_Q(\mathcal{A}\cup\mathcal{A}_1\cup\cdots\cup\mathcal{A}_k) - \overline{F}_Q(\mathcal{A}_1\cup\cdots\cup\mathcal{A}_k),$$

and use PSPIEL to obtain a solution such that

$$\widetilde{F}^{(k+1)}(\mathcal{A}_{k+1}) \geq \frac{\beta(Q - \overline{F}_Q(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k))}{2}.$$

Note that after k steps, it holds that $(Q - \overline{F}_Q(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k) \leq Q(1 - \frac{1}{2}\beta)^k$, and hence after

$$k = \left\lceil \frac{\log \frac{m}{\varepsilon}}{\log \frac{2}{2-\beta}} \right\rceil$$

iterations it holds that

$$\overline{F}_q(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k) \geq Q(1 - \frac{\varepsilon}{m}),$$

and hence $F_i(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k) \geq Q(1 - \varepsilon)$. We choose δ small enough to apply the union bound over all k trials. Algorithm 11.3 presents pseudo-code for our approach.

We summarize our analysis in the following Theorem:

Theorem 11.4. *Given a graph $\mathcal{G} = (\mathcal{V}, E)$, constants ε, δ, Q and (r, γ)-local monotone submodular functions $F_1, \ldots, F_m$ bounded above by M, we can find a tree $\mathcal{T}$ with cost*

$$\mathrm{C}(\mathcal{T}) = \mathcal{O}(r \dim(\mathcal{V},E)) \times \left\lceil \frac{\log \frac{m}{\varepsilon}}{\log \frac{2}{2-\Omega(\gamma)}} \right\rceil \times \ell^*,$$

spanning a set $\mathcal{A}$ with $F_i(\mathcal{A}) \geq Q(1 - \varepsilon)$ for all i. The algorithm is randomized and runs in expected time polynomial in the size of the problem instance and polynomial in $\frac{M}{Q}$. □

Hence, for an arbitrary $\varepsilon > 0$ we can, in expected polynomial time, find a sensor placement with sensing quality $F_i(\mathcal{A}) \geq Q(1 - \varepsilon)$. The cost of the solution $\mathrm{C}(\mathcal{T})$ grows logarithmically in $\frac{m}{\varepsilon}$ which depends on the number m of objective functions.

11.6.2 Experiments on robust optimization

We use our robust version of PSPIEL to make the sensor placement in our proof-of-concept study more robust. We choose $F_1(\mathcal{A})$ as the mutual information obtained in an isotropic Gaussian process with kernel (6.4) and fixed bandwidth h. As F_2, we choose the nonstationary GP learned from the pilot deployment, as in Section 11.5.

In order to model F_1 using an isotropic GP (modeling the uniform spreading of light), we need to specify the bandwidth parameter h in (6.4). This bandwidth parameter encodes our smoothness assumptions about the world. The smaller h, the quicker the assumed correlation decays with distance, and hence the more rough we assume the phenomenon to be. In addition, the smaller h, the more sensors we need in order to obtain a high level of mutual information. This means, that for small values of h, the uninformed sensing quality F_1 dominates the optimization. For large values of h however, the fewer sensors we need to obtain high sensing quality F_1, and hence F_2 dominates the objective value. Hence, by varying h, we can vary the amount of robustness. Figure 11.7 shows different sensor placements obtained by choosing an increasing bandwidth h. For small bandwidths, the placements are basically uniformly spread out over the space. For high bandwidths, the robust places resemble the non-robust placement pS19 (from Section 11.5).

Algorithm 11.3: Algorithm for robust sensor placements.

Input: Graph $(\mathcal{V}, E)$, $F_1, \ldots, F_m$, Q, M, β, ε, δ
Output: Placement $\mathcal{A}$ such that $F_i(\mathcal{A}) \geq Q$ for all i with probability $1 - \delta$.
begin
  $\overline{F}_Q(\mathcal{A}) \leftarrow \frac{1}{m} \sum_{i=1}^{m} \min\{F_i(\mathcal{A}), Q\}$;
  $\widetilde{F}^{(1)} \leftarrow \overline{F}_Q$;
  $\mathcal{A} \leftarrow \emptyset$;
  $k \leftarrow 0$;
  while $\overline{F}_Q(\mathcal{A}) \leq Q(1 - \varepsilon/m)$ **do**
    $k \leftarrow k + 1$;
    $\widetilde{F}^{(k)}(\mathcal{A}') \leftarrow \overline{F}_Q(\mathcal{A}' \cup \mathcal{A}) - \overline{F}_Q(\mathcal{A})$;
    $Q_{togo} \leftarrow Q - \overline{F}_Q(\mathcal{A})$;
    repeat
      $\mathcal{A}' \leftarrow pSPIEL(\widetilde{F}^{(k)}, Q_{togo})$
    until $\widetilde{F}^{(k)}(\mathcal{A}') \geq Q_{togo}\beta/2$;
    $\mathcal{A} \leftarrow \mathcal{A} \cup \mathcal{A}'$;
  end
end

We also compare the manual placement (M20) and the non-robust pSPIEL placement (pS19) of Section 11.5 with the robust solutions. For each robust placement $\mathcal{A}$, we compute $\min_i F_i(\mathcal{A})$, which measures how well the placement performs with respect to both the data-driven model F_2 and the uniform prior F_1. The placement in Figure 11.7(b) maximizes this score over all the robust placements, indicating that Figure 11.7(b) is a good compromise between the data-driven model and prior assumptions. Figure 11.8 compares the manual placement (M20) with both optimized placements. Note that while the robust placement obtains higher RMS error and communication cost than the non-robust placement (pS19), it still performs drastically better than the manual placement (M20). Also note that the robustness scores $\min_i F_i(\mathcal{A})$ of both the robust and the manual placement are higher than for the non-robust placement (pS19).

11.7 Modular approximation for other combinatorial problems: Informative path planning

The key idea behind pSPIEL was to reduce the problem of maximizing sensing quality, a submodular function, to the problem of maximizing a *modular* function on the Modular Approximation Graph, which we can solve using existing combinatorial algorithms for modular functions. This idea of reducing a local-submodular optimization to a modular optimization is quite powerful. For example, we can use the same algorithmic idea to solve other local-submodular combinatorial optimization problems. In the following, we will discuss one such example: Applying pSPIEL for informative path planning.

Consider the setting where, instead of deploying a wireless sensor network, we use a robot to collect observations. In this case, we also want to identify locations $\mathcal{A}$ of high sensing quality, but the robot needs to travel between successive sensing location. We can model this setting by specifying a graph $\mathcal{G} = (\mathcal{V}, E)$

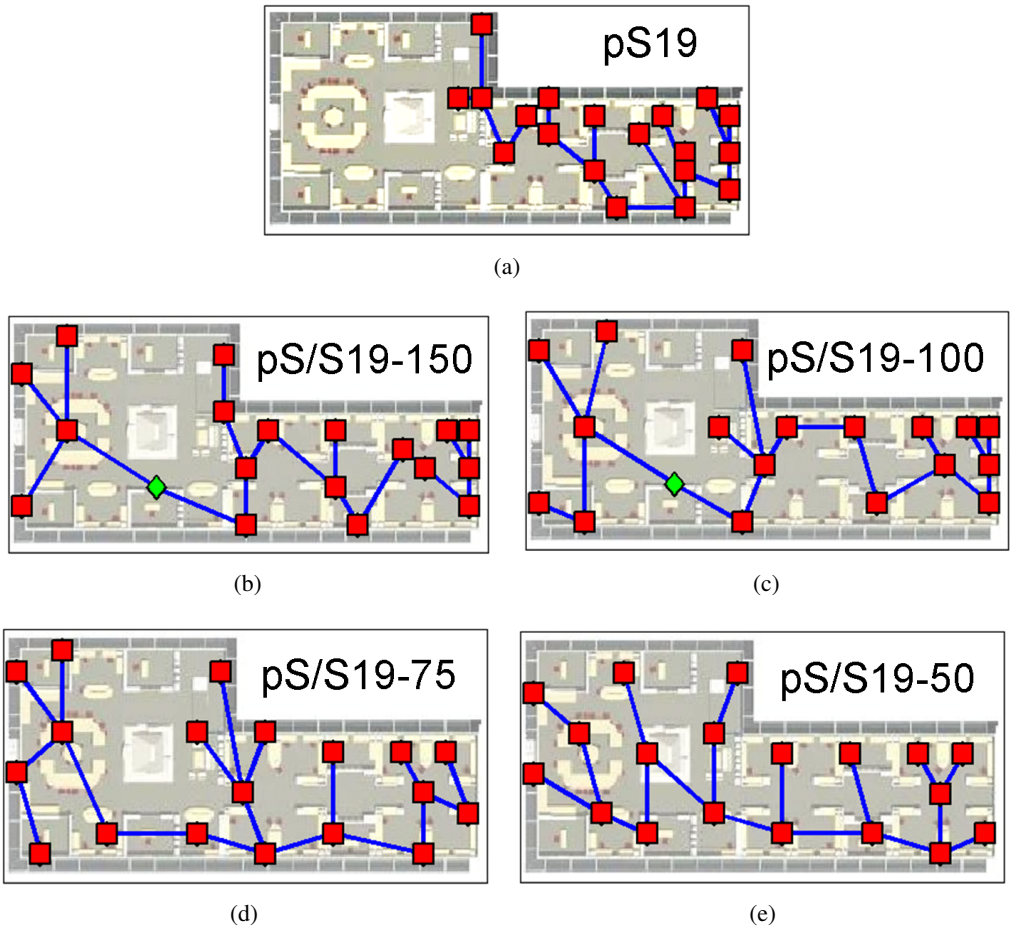


Figure 11.7: Experimental results. (a) shows the non-robust placement. (b-e) show placements optimized using Algorithm 11.3 where F_2 is the mutual information according to the pilot deployment, and F_1 is the mutual information w.r.t. an isotropic GP with different bandwidths: pS/S19-x refers to the result when using a bandwidth proportional to x. Notice that with decreasing bandwidths, the placements become more and more regularly-spaced.

over the sensing locations, and our goal will be to find an informative path $\mathcal{P} = (a_1, \ldots, a_k)$ spanning the locations $\mathcal{A} \subseteq \mathcal{P}$. In this setting, rather than modeling the communication cost, the cost $\mathrm{C}(\mathcal{P})$ is the length of the path $\mathcal{P}$, i.e.,

$$\mathrm{C}(\mathcal{P}) = \sum_{i=1}^{k-1} \mathrm{C}(\{a_i, a_{i+1}\}).$$

Similarly, $\mathrm{C}(\mathcal{A})$ is the cost of the shortest path spanning nodes $\mathcal{A}$. Using this modified notion of cost, we designate specific nodes $s, t \in \mathcal{V}$ as starting and ending locations, i.e., require that $a_1 = s$ and $a_k = t$, and solve

$$\max_{\mathcal{P}} F(\mathcal{P}) \text{ subject to } \mathrm{C}(\mathcal{P}) \leq B \text{ and } \mathcal{P} \text{ is an } s - t \text{ path in } \mathcal{G} \tag{11.5}$$

For modular functions F, this problem is known as the $s - t$ *orienteering* problem [*c.f.*, Chekuri et al., 2008]. For the more general case where F is submodular, so far for this problem only an algorithm with

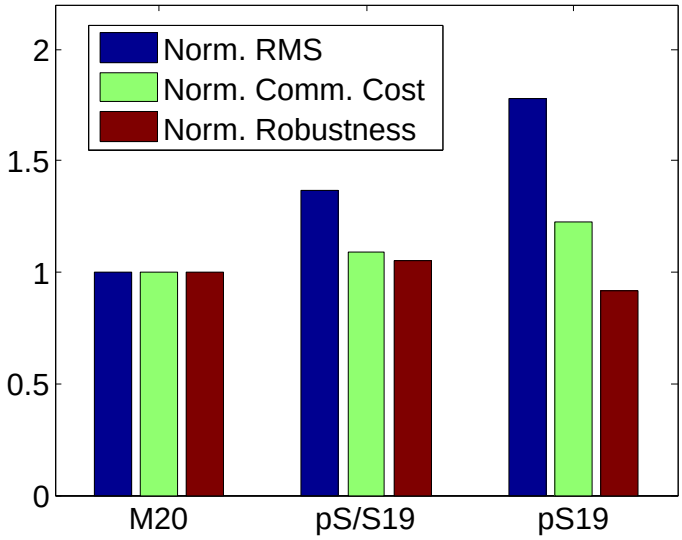


Figure 11.8: Experimental results comparing the manual placement with 20 sensors (left) to the robust (center) and non-robust (right) placements obtained using PSPIEL. For each placement, three bars are shown, measuring improvement in RMS error, communication cost and robustness compared to the manual solution as baseline (which is normalized to 1). Higher values are better. Both the robust and non-robust placements outperform the manual deployment in communication cost and prediction error. The robust placement also obtains higher robustness score than the manual deployment. The robustness score of the non-robust placement is even lower than that of the manual placement.

quasipolynomial running time was proposed by Chekuri and Pal [2005], which has been extended and used for informative path planning by Singh et al. [2007].

Using an approximate algorithm for $s-t$ orienteering on the modular approximation graph, we obtain the first polynomial time approximation algorithm for (r,γ)-local submodular orienteering.

Theorem 11.5. *Given a graph $\mathcal{G}=(\mathcal{V},E)$, $s,t\in\mathcal{V}$ and an (r,γ)-local monotone submodular function F,* PSPIEL *will find an $s-t$ path $\mathcal{P}$ with cost $\mathcal{O}(r\dim(\mathcal{V},E))\times\ell^*$, spanning a set $\mathcal{A}$ with expected sensing quality $F(\mathcal{A})\geq\Omega(\gamma)\times F(\mathcal{A}^*)$. The algorithm is randomized and runs in polynomial-time.* □

Singh et al. [2007] proved that any κ-approximate algorithm for submodular orienteering can be extended to an efficient $\kappa+1$-approximate algorithm for the more complex problem planning *multiple* paths (for multiple robots). This result can immediately be used to extend PSPIEL to the setting of multiple robot informative path planning.

11.8 Related work

In this section, we relate our approach to work in several areas.

11.8.1 Sensor placement to monitor spatial phenomena

The problem of selecting observations for monitoring spatial phenomena has been investigated extensively in geostatistics (*c.f.*, Cressie [1991] for an overview), and more generally (Bayesian) experimental design [*c.f.*, Chaloner and Verdinelli, 1995]. Heuristics for actively selecting observations in GPs in order to achieve high mutual information have been proposed by Caselton and Zidek [1984]. Sensor selection considering both the value of information together with the cost of acquiring the information in the context of sensor networks was first formalized by Zhao et al. [2002]. These approaches however do not consider communication cost as done in this chapter.

11.8.2 Sensor placement under communication constraints

Existing work on sensor placement under communication constraints [Funke et al., 04, Gupta et al., 2003b, Kar and Banerjee, 2003] has considered the problem mainly from a geometric perspective: Sensors have a fixed *sensing region*, such as a disc with a certain radius, and can only communicate with other sensors that are at most a specified distance apart. In addition, it is assumed that two sensors at fixed locations can either perfectly communicate or not communicate at all. As argued in the introduction these assumptions are problematic. By using probabilistic models for both the phenomenon and the link qualities, our pSPIEL approach avoids these strong assumptions.

11.8.3 Statistical models for modeling link quality

Often, it is assumed that a transmitting node has perfect connectivity with all nodes inside a given radius and zero connectivity with nodes outside that disk [Bai et al., 2006, Cerpa et al., 2005]. However, depending on how the disk radius is chosen, such disk models may not capture actual communication patterns in one particular network. In order to allow more flexibility, Cerpa et al. [2005] use a probabilistic disk model. Like the regular disk model, it assumes connectivity is a function only of geometric distance between motes but unlike that model, it can predict a real-valued connectivity value, that is, a probability of packet reception that is not zero or one. However, their isotropic approach does not adapt to a specific environment (containing obstacles like walls, doors, furniture, etc.).

In order to account for more complex behavior, physical models like radio propagation or path loss equations were obtained from real-data and describes the signal quality fall off away from a transmitting sensor [Friis, 1946, Rappaport, 2000]. These equations can model complex communication behaviors with parameters encoding for the number of walls, the construction materials of the clutter, multipath signal effects, and microwave interference [Morrow, 2004, Rappaport, 2000]. Unfortunately, this deployment-specific information can be as hard to model and obtain as the packet transmission data needed for data-driven approaches.

Our link quality model described in Section 11.1.1 allows to both model complex, environment dependent behavior, and is completely data driven (i.e., no deployment-specific information needs to be manually supplied). Extending our link quality model, Ertin [2007] proposed an approach for learning Gaussian Process models for link quality estimation, explicitly taking into account that fact that sensor measurements are lost (censored). Note that our pSPIEL approach can use such alternative approaches for estimating link quality as well.

11.8.4 Related work on submodular optimization

Problem (11.1) for an arbitrary integer valued monotonic submodular function F is called the polymatroid Steiner tree problem [Calinescu and Zelikovsky, 2005]. Calinescu and Zelikovsky [2005] developed a polylogarithmic approximation algorithm for this problem. However, their approach does not exploit locality, and hence leads to approximation guarantees that are worse than those obtained by pSPIEL (which solves the problem for all (r, γ)-local submodular functions F) if locality is present. The submodular orienteering problem, i.e., the problem of finding a path of bounded length maximizing a submodular sensing quality function, was first considered by Chekuri and Pal [2005], who developed an algorithm with quasipolynomial running time. While providing important theoretical insights, their approach does not scale to practical sensing problems. Singh et al. [2007] proposed a spatial decomposition approach as well as branch and bound techniques to significantly speed up the approach of Chekuri and Pal [2005]. They also applied it to informative path planning in the context of environmental monitoring problems. However, their approach still has worst-case quasipolynomial running time. The approach presented in Section 11.7 is the first efficient (polynomial-time) algorithm for submodular orienteering, in the case where the objective function F is (r, γ)-local.

11.9 Summary

In this chapter, we proposed a unified approach for robust placement of wireless sensor networks. As in Chapter 6, our approach uses Gaussian Processes, which can be chosen from expert knowledge or learned from an initial deployment. We propose to use GPs not only to model the monitored phenomena, but also for predicting communication costs. We presented a polynomial time algorithm – pSPIEL– selecting Sensor Placements at Informative and cost-Effective Locations. Our algorithm provides strong theoretical performance guarantees. Our algorithm is based on a new technique, the modular approximation graph, that is more general and can also be used, for example, to plan informative paths for robots. We extended our pSPIEL approach to obtain sensor placements that are robust against changes in the environment. We built a complete implementation on Tmote Sky motes and extensively evaluated our approach on real-world placement problems. Our empirical evaluation shows that pSPIEL significantly outperforms existing methods.

Chapter 12

Simultaneous Placement and Scheduling of Sensors

As argued in the previous chapters, when monitoring spatial phenomena, such as road speeds on a highway, deciding where to *place* a small number of sensors to obtain best prediction accuracy is an important task. Figure 12.1 shows a Sensys Networks wireless traffic sensor [Haoui et al., 2008], that provides 30 second aggregate speed, flow and vehicle density measurements. Currently the system is being deployed by Caltrans at different sites in California, including highways and arterial roads. When using such wireless sensor networks, power consumption is a key constraint, since every measurement drains the battery. For applications such as road speed monitoring, a minimum battery lifetime is required to ensure feasibility of the sensor network deployment. One approach to meeting such lifetime requirements is to deploy few nodes with large batteries. However, such an approach can be sensitive to node failures. Additionally, packaging constraints can limit the size of the battery deployed with the nodes. For these and other reasons, it can be more effective to deploy a larger number of nodes with smaller batteries, that are activated only a fraction of the time. Hence, to improve the lifetime of such a sensor network, the problem of *scheduling* becomes of crucial importance: Given a fixed placement of sensors, when should we turn each sensor on in order to obtain high monitoring performance over all time steps? One approach that has been found effective in the past is to partition the sensors into k groups [Abrams et al., 2004, Deshpande et al., 2008, Koushanfary et al., 2006]. By activating a different group of sensors at each time step and cyclicly shifting through these groups, the network lifetime can effectively be increased by a factor of k. In the traffic network application, current studies indicate that an increase by a factor of $k = 4$ would be required to make sensor deployment an economically feasible option (*c.f.*, Section 12.5 for more details).

Traditionally, sensor placement and sensor scheduling have been considered separately from each other – one first decides where to place the sensors, and then when to activate them. In this chapter, we present an efficient algorithm, ESPASS (for *efficient Simultaneous Placement and Scheduling of Sensors*), that jointly optimizes the sensor placement and the sensor schedule. We prove that our algorithm provides a constant factor approximation to the optimal solution of this **NP**-hard optimization problem.

Most existing approaches to sensor placement and scheduling associate a fixed sensing region with every sensor, and then attempt to maximize the number of regions covered in every group of sensors [*c.f.*, Abrams et al., 2004, Deshpande et al., 2008, Hochbaum and Maas, 1985]. In complex applications such as traffic or environmental monitoring however, the goal of sensor placement is a prediction problem,

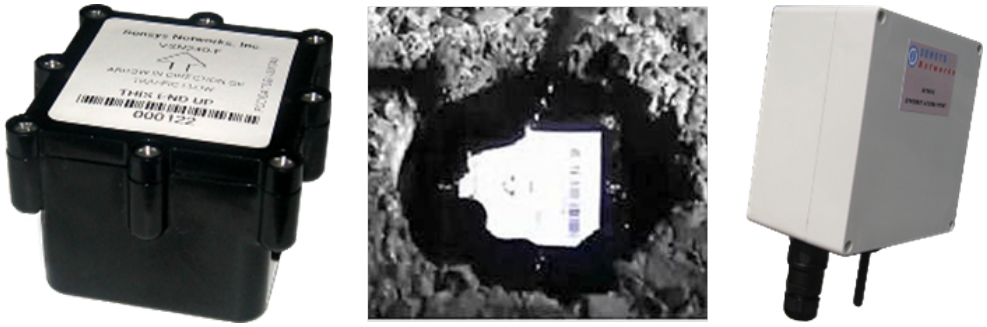


Figure 12.1: Sensys Networks wireless traffic sensor. (left) encased unit, (middle) sensor deployed in pavement, (right) GPRS/CDMA base station.

where one intends to predict the sensed phenomenon at the locations where no sensors are placed. Our algorithm applies to such settings where the sensing quality of a set of sensors is measured, e.g., in the improvement of prediction accuracy (more formally, our algorithm applies whenever the sensing quality function satisfies *submodularity*, an intuitive diminishing returns property).

In contrast to most existing algorithms that optimize scheduling for average case performance, our approach furthermore provides a schedule that performs uniformly well over time, hence leading to a well-balanced performance of the sensor network. For security-critical applications such as outbreak detection, such balanced performance is a crucial requirement not met by existing algorithms. In fact, our experimental results show that average-case optimal solutions can lead to arbitrarily unbalanced performance, but optimizing for balanced performance (using ESPASS) typically leads to good average-case performance.

Deploying a large number of scheduled sensors has the additional benefit that it allows trading off power and accuracy. The deployed network might have several modes of operation: a scheduled mode of operation, where only a small fraction of sensors is turned on, and a "high density" mode where all (or a larger fraction of) sensors are activated. For example, in traffic monitoring, once a traffic congestion is detected (during scheduled mode), the high density mode could be used to accurately identify the boundary of the congestion. We show how our algorithm can be extended to support such a power-accuracy tradeoff.

We present extensive empirical studies on several case studies, illustrating the versatility of our algorithm. These case studies include sensing tasks such as traffic and environmental monitoring and placing sensors for outbreak detection and selecting informative weblogs to read on the Internet. Our results show that simultaneously placing and scheduling results in drastically improved performance compared to the setting where optimization over the placement and the scheduling are performed separately.

In summary, our main contributions are:

- We study the problem of simultaneously placing and scheduling sensors as a novel optimization problem.
- We develop ESPASS, an efficient approximation algorithm for this problem, that applies to a variety of realistic sensing quality functions (such as area coverage, variance reduction, outbreak detection, etc.). Our algorithm is guaranteed to provide a near-optimal solution, that obtains at least a constant fraction of the optimal sensing quality. ESPASS furthermore allows to trade off power consumption

and accuracy.

- We perform several extensive case studies on real sensing problems in traffic and environmental monitoring as well as outbreak detection, demonstrating the effectiveness of our approach.

12.1 Problem statement

We will first separately introduce the sensor placement and scheduling problems, and then formalize the problem of simultaneously placing and scheduling sensors.

12.1.1 Sensor placement

As discussed in Chapter 6, in *sensor placement*, we are given a finite set $\mathcal{V}$ of possible locations where sensors can be placed. Our goal is to select a small subset $\mathcal{A} \subseteq \mathcal{V}$ of locations to place sensors at, that maximizes a sensing quality function $F(\mathcal{A})$. There are several different notions of sensing quality that we might want to optimize, each depending on the particular sensing task. For example, we can associate sensing regions with every sensor, and $F(\mathcal{A})$ can measure the total area covered when placing sensors at locations $\mathcal{A}$. In complex applications such as the traffic monitoring problem, we are interested in optimizing the prediction accuracy when obtaining measurements from locations $\mathcal{A}$. In this setting, we can model the state of the world (e.g., the traffic condition at different locations) using a collection of random variables $\mathcal{X}_\mathcal{V}$, one variable $\mathcal{X}_s$ for each location $s \in \mathcal{V}$. We can then use a probabilistic model (such as a Gaussian Process which is frequently used in geostatistics, *c.f.*, Cressie [1991]) that models a joint probability distribution $P(\mathcal{X}_\mathcal{V})$ over the possible locations. Upon acquiring measurements $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$ at a subset of locations $\mathcal{A}$, we can then predict the phenomenon at the unobserved locations using the conditional distribution $P(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$. We can then use the expected mean squared error,

$$\mathrm{Var}(\mathcal{X}_\mathcal{V} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \frac{1}{|\mathcal{V}|} \sum_{s \in \mathcal{V}} \mathbb{E}\left[(\mathcal{X}_s - \mathbb{E}[\mathcal{X}_s \mid \mathbf{x}_\mathcal{A}])^2 \mid \mathbf{x}_\mathcal{A}\right]$$

to quantify the uncertainty in this prediction.

In this chapter, we will mainly focus on the *expected reduction in variance* at the unobserved locations,

$$F(\mathcal{A}) = \mathrm{Var}(\mathcal{X}_\mathcal{V}) - \int P(\mathbf{x}_\mathcal{A}) \, \mathrm{Var}(\mathcal{X}_\mathcal{V} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) d\mathbf{x}_\mathcal{A},$$

as introduced in Section 2.3. This sensing quality function has been found useful for sensor selection [*c.f.*, Deshpande et al., 2004, Krause et al., 2008c] and experimental design [*c.f.*, Chaloner and Verdinelli, 1995].

As we have seen in Part II of this Thesis, many practical notions of sensing quality, such as the area covered are submodular (*c.f.*, Chapter 5). As recently shown by Das and Kempe [2008], this property is satisfied by the variance reduction objective for Gaussian distributions under certain assumptions about the covariance. Recall that a set function F is called submodular, if for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $s \in \mathcal{V} \setminus \mathcal{B}$

$$F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{s\}) - F(\mathcal{B}),$$

i.e., adding s to a small set $\mathcal{A}$ helps more than adding s to the superset $\mathcal{B}$. In addition, these sensing quality functions are *nondecreasing*: For all $\mathcal{A} \subseteq \mathcal{B}$ it holds that $F(\mathcal{A}) \leq F(\mathcal{B})$, i.e., adding more sensors can only improve the sensing quality.

Based on this notion of a nondecreasing submodular sensing quality function, the sensor placement problem then is

$$\max_{\mathcal{A}} F(\mathcal{A}) \text{ such that } |\mathcal{A}| \leq m,$$

i.e., we want to find a set $\mathcal{A}$ of at most m locations to place sensors maximizing the sensing quality F.

12.1.2 Sensor scheduling

In *sensor scheduling*, we are given a sensor placement (i.e., locations $\mathcal{A}$), and our goal is to assign each sensor $s \in \mathcal{A}$ one of k time slots. This assignment partitions the set $\mathcal{A}$ into disjoint sets $\mathcal{A}_1, \ldots, \mathcal{A}_k$, where $\mathcal{A}_t \subseteq \mathcal{A}$ is the subset of sensors that have been assigned slot t. A round-robin schedule can then be applied that cycles through the time slots, and activates sensors $\mathcal{A}_t$ at time t. Since each sensor is active at only one out of k time slots, this procedure effectively increases the lifetime of the network by a factor of k. How can we quantify the value of a schedule $\mathcal{A} = (\mathcal{A}_1, \ldots, \mathcal{A}_k)$? For each group $\mathcal{A}_t$, we can compute the sensing quality $F(\mathcal{A}_t)$[1]. One possibility would then be to optimize for the *average* performance over time,

$$\max_{\mathcal{A}_1,\ldots,\mathcal{A}_k} \frac{1}{k} \sum_{t=1}^{k} F(\mathcal{A}_t).$$

However, as we show in our experiments, if we optimize for the average case performance, it can happen that a few of the time slots are very poorly covered, i.e., there is a time t such that $F(\mathcal{A}_t)$ is very low. For security-critical applications, this can be problematic. Instead, we can also optimize for a *balanced* schedule,

$$\max_{\mathcal{A}_1,\ldots,\mathcal{A}_k} \min_{t} F(\mathcal{A}_t),$$

that performs uniformly well over time.

Note that the above formulation of the scheduling problem allows to handle settings where each sensor can be active at $r \geq 1$ timesteps. In this setting, we simply define a new ground set $\mathcal{A}' = \mathcal{A} \times \{1, \ldots, r\}$ where the pair $(s, i) \in \mathcal{A}'$ refers to the i-th activation of sensor s. The sensing quality function is modified as $F'(\mathcal{A}'_j) = F(\{s : \exists_i (s, i) \in \mathcal{A}'_j\})$.

12.1.3 Simultaneous placement and scheduling

Both sensor placement and sensor scheduling have been studied separately from each other in the past. One approach towards placement and scheduling would be to first use an algorithm (such as the CELF algorithm proposed in Chapter 5) to find a sensor placement $\mathcal{A}$, and then use a separate algorithm (such as the mixed integer approach of Koushanfary et al. [2006]) to find a schedule $\mathcal{A}_1, \ldots, \mathcal{A}_k$. We call this approach a *stage-wise* approach, and illustrate it in Figures 12.2(a) and 12.2(b).

[1]Note that we assume the same sensing quality function F for each time step. This assumption has been made in the past [*c.f.*, Abrams et al., 2004, Koushanfary et al., 2006], and is reasonable for many monitoring tasks.

Instead of separating placement and scheduling, we can *simultaneously* optimize for the placement and the schedule. Suppose we have resources to purchase m sensors, and we would like to extend the network lifetime by a factor of k. Our goal would then be to find k disjoint sets $\mathcal{A}_1, \ldots, \mathcal{A}_k \subseteq \mathcal{V}$, such that together these sets contain at most m locations, i.e., $|\bigcup_t \mathcal{A}_t| \leq m$. We call this problem the SPASS problem, for *simultaneous placement and scheduling of sensors*. Again, we can consider the average-case performance,

$$\max_{\mathcal{A}_1 \ldots \mathcal{A}_k} \frac{1}{k} \sum_{t=1}^{k} F(\mathcal{A}_t) \text{ s.t. } \mathcal{A}_i \cap \mathcal{A}_j = \emptyset \text{ if } i \neq j \text{ and } |\bigcup_t \mathcal{A}_t| \leq m \tag{12.1}$$

and the balanced objective:

$$\max_{\mathcal{A}_1 \ldots \mathcal{A}_k} \min_t F(\mathcal{A}_t) \text{ s.t. } \mathcal{A}_i \cap \mathcal{A}_j = \emptyset \text{ if } i \neq j \text{ and } |\bigcup_t \mathcal{A}_t| \leq m. \tag{12.2}$$

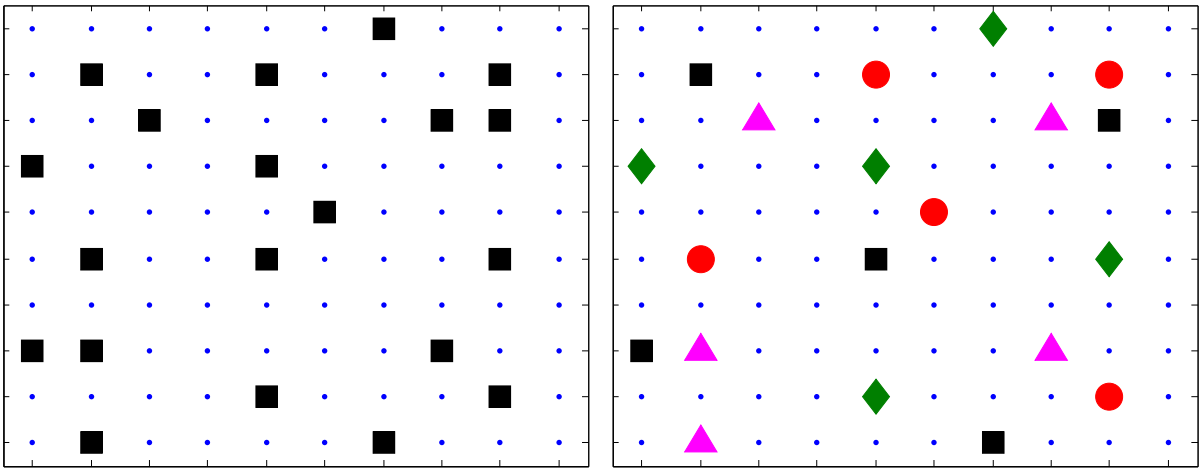
By performing this simultaneous optimization, we can obtain very different solutions, as illustrated in Figure 12.2(c). In Section 12.5, we will show that this simultaneous approach can lead to drastically improved performance as compared to the traditional, stage-wise approach. In this chapter, we present ESPASS, an efficient approximation algorithm with strong theoretical guarantees for this problem.

The placement and schedule in Figure 12.2(c) has the property that the sensors selected at each time step share very similar locations, and hence perform roughly equally well. However, if activated all at the same time, the "high-density" performance $F(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k)$ is much lower than that of the placement in Figure 12.2(a). We also develop an algorithm, MCSPASS, that leads to placements which perform well both in scheduled and in high-density mode. Figure 12.2(d) presents the solution obtained for the MCSPASS algorithm.

Note that instead of fixing the number of time slots, we could also specify a desired accuracy constraint Q and then ask for the maximum lifetime solution, i.e., the largest number k of time slots such that a solution with minimum (or average) sensing quality Q is obtained. Clearly, an algorithm that solves Problem (12.2) (or Problem (12.1)) could be used to solve this alternative problem, by simply binary searching over possible values for k[2].

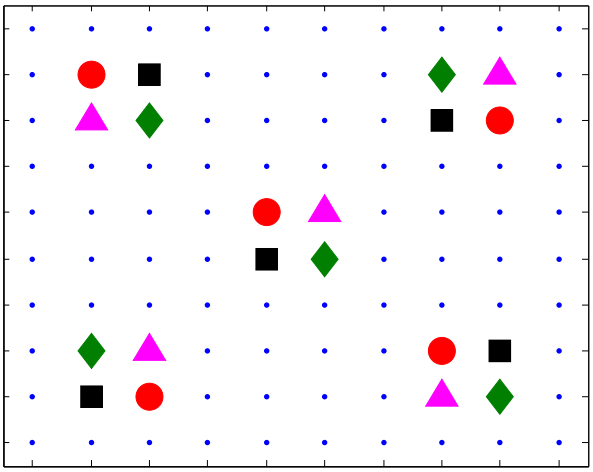
Further note that it is possible to allow each sensor to be active at $r \geq 1$ timesteps by using the modification described in Section 12.1.2.

[2]However, in case an approximate algorithm is used such as the ESPASS algorithm developed in this chapter, its guarantees are not necessarily preserved.

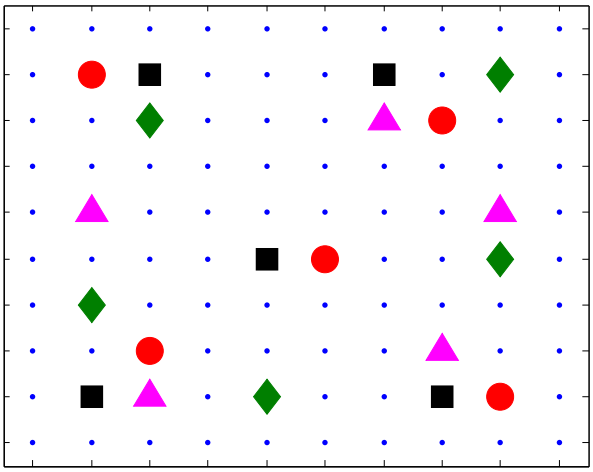


(a) *Optimized sensor placement*

(b) *Optimized schedule for placement (a)*



(c) *Simultaneous placement and schedule*



(d) *Multicriterion solution*

Figure 12.2: In the stage-wise approach, sensors are first deployed (a), and the deployed sensors are then scheduled (b, sensors assigned to the same time slot are drawn using the same color and marker). In the simultaneous approach, we jointly optimize over placement and schedule (c). (d) Multicriterion solution to Problem (12.4) ($\lambda = .25$) that performs well both in scheduled and high-density mode.

12.2 A naive greedy algorithm

We will first study the problem of optimizing the average performance over time, i.e., Problem (12.1), for a fixed nondecreasing submodular sensing quality function F. Considering the fact that simultaneously placing and scheduling is a strict generalization of sensor placement, which itself is **NP**-hard (*c.f.*, Theorem 6.2), we cannot expect to efficiently find the optimal solution to Problem (12.1) in general.

Instead, we will use the following intuitive greedy algorithm that we call GPC for *Greedy Average-case Placement and Scheduling*. At every round, GPC picks a time slot t and location s which increases the total sensing quality the most, until m location/time-slot pairs have been picked. It is formalized as Algorithm 12.1.

Algorithm 12.1: The greedy average-case placement and scheduling (GPC) algorithm.

Algorithm GPC (F, $\mathcal{V}$, k, m)
$\mathcal{A}_t \leftarrow \emptyset$ for all t;
for $i = 1$ **to** m **do**
  foreach $s \in \mathcal{V} \setminus (\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k)$, $1 \leq t \leq k$ **do**
1 $\delta_{t,s} \leftarrow F(\mathcal{A}_t \cup \{s\}) - F(\mathcal{A}_t)$;
  end
  $(t^*, s^*) \leftarrow \operatorname{argmax}_{t,s} \delta_{t,s}$;
  $\mathcal{A}_{t^*} \leftarrow \mathcal{A}_{t^*} \cup \{s^*\}$;
end

12.2.1 Theoretical guarantee

Perhaps surprisingly, we can show that this simple algorithm provides near-optimal solutions for Problem (12.1). In fact, it generalizes the distributed Set-k Cover algorithm proposed by Abrams et al. [2004] to arbitrary submodular sensing quality functions F, and to the setting where at most m sensors can be selected in total.

Theorem 12.1. *For any nondecreasing, and submodular function F,* GPC *returns a solution $\mathcal{A}_1, \ldots, \mathcal{A}_k$ s.t.*

$$\frac{1}{k} \sum_t F(\mathcal{A}_t) \geq \frac{1}{2} \max_{\mathcal{A}'} \frac{1}{k} \sum_t F(\mathcal{A}'_t).$$

GPC *requires at most $\mathcal{O}(kmn)$ evaluations of F.*

The proofs of Lemma 12.3 and all other results are given in the Appendix. The key observation is that Problem (12.1) is an instance of maximizing a submodular function subject to a *matroid* constraint (*c.f.*, the Appendix for details). A fundamental result by Fisher et al. [1978] then proves that the greedy algorithm returns a solution that obtains at least one half of the optimal average-case score. Matroids for sensor scheduling have been considered before by Williams et al. [2007].

12.2.2 The greedy algorithm can lead to unbalanced solutions

If a sensor placement and schedule is sought that performs well "on-average" over time, GPC performs well. However, even though the average performance over time, $\frac{1}{k}\sum_t F(\mathcal{A}_t)$, is high, the performance at some individual timesteps t' can be very poor, and hence the schedule can be *unfair*. In security-critical applications, where high performance is required at all times, this behavior can be problematic. In such settings, we might be interested in optimizing the *balanced* performance over time, $\min_t F(\mathcal{A}_t)$. This optimization task was raised as an open problem by [Abrams et al., 2004].

A first idea would be to try to modify the GPC algorithm to directly optimize this balanced performance, i.e., replace Line 14 in Algorithm 12.1 by

$$\delta_{t,s} \leftarrow \min_j F(\mathcal{A}_j^{+(t,s)}) - \min_j F(\mathcal{A}_j),$$

where $\mathcal{A}^{+(t,s)}$ is the solution obtained by adding location s to time slot $\mathcal{A}_t$ in solution $\mathcal{A} = \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k$. We call this modified algorithm the GBPS algorithm (for Greedy Balanced Placement and Scheduling). Unfortunately, both GPC and GBPS can perform arbitrarily badly. Consider a simple scenario with three locations, $\mathcal{V} = \{a, b, c\}$, and the nondecreasing submodular function $F(\mathcal{A}) = |\mathcal{A}|$. We want to partition $\mathcal{V}$ into three timesteps, i.e., $k = 3$ and $m = 3$. Here, the optimal solution would be to pick $\mathcal{A}_1^* = \{a\}$, $\mathcal{A}_2^* = \{b\}$ and $\mathcal{A}_3^* = \{c\}$. However, both GPC and GBPS would (ties broken unfavorably) pick $\mathcal{A}_1 = \{a, b, c\}$ and $\mathcal{A}_2 = \mathcal{A}_3 = \emptyset$, obtaining a minimum score of 0.

Unfortunately, this poor performance is not just a theoretical example – in Section 12.5 we demonstrate it empirically on real sensing tasks.

12.3 The eSPASS algorithm

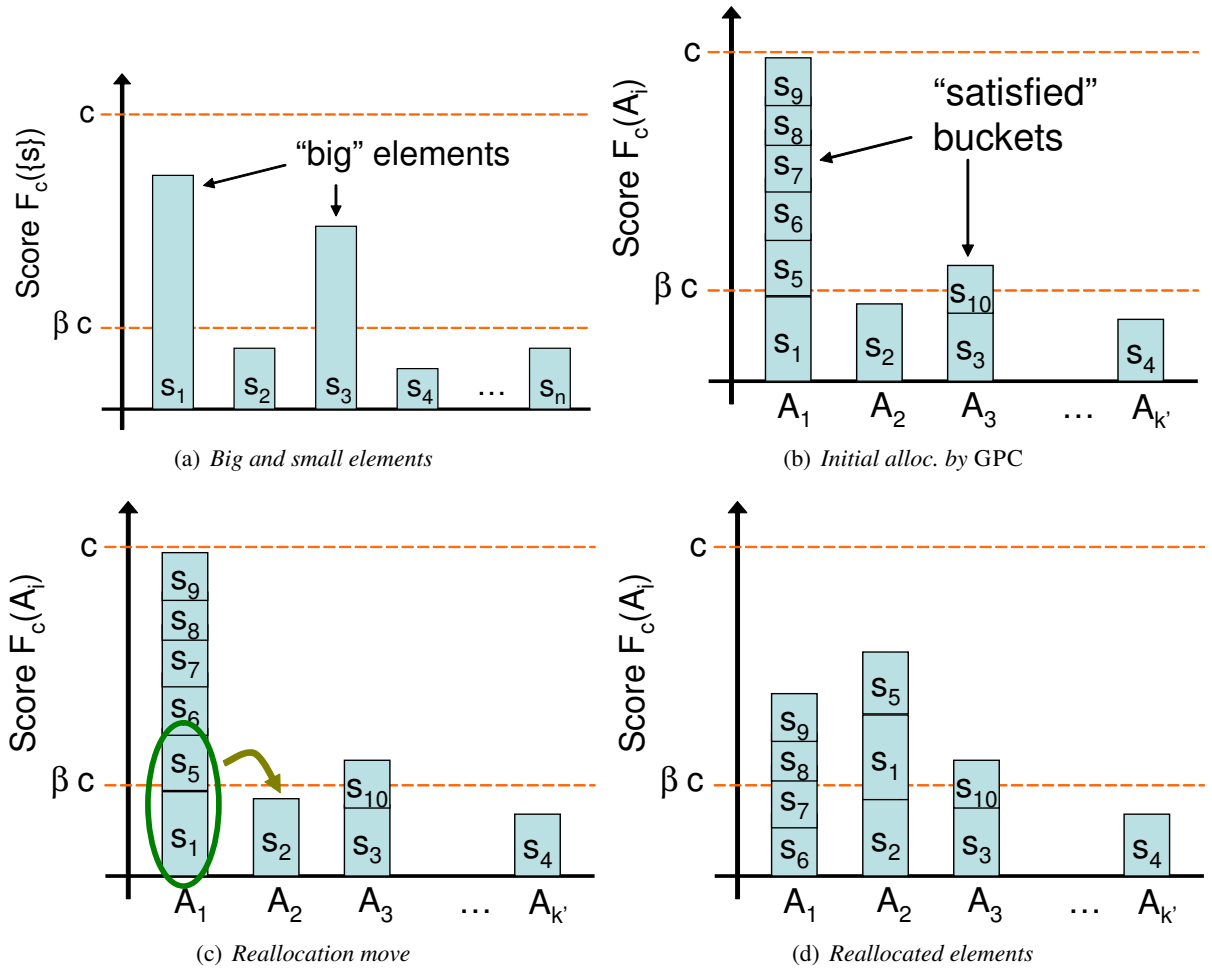
In the following, we will develop an efficient algorithm, eSPASS (for *efficient Simultaneous Placement and Scheduling of Sensors*), that, as we will show in Section 12.3.2, is guaranteed to provide a near-optimal solution to the Problem (12.2). To the best of our knowledge, our algorithm is the first algorithm with theoretical guarantees for this general problem, hence partly resolving the open problem described by Abrams et al. [2004].

12.3.1 Algorithm overview

We start with an outline of our algorithm, and then proceed to discuss each step more formally.

Our high-level goal will be to reduce the problem of optimizing the balanced objective into a sequence of modified optimization problems involving an average-case objective, which we can approximately solve using GPC. This idea is based on the following intuition: Consider a *truncated* objective function $F_c(\mathcal{A}) = \min\{F(\mathcal{A}), c\}$. The key observation is that, for *any* constant c, it holds that

$$\min_t F(\mathcal{A}_t) \geq c \Leftrightarrow \frac{1}{k}\sum_{t=1}^{k} F_c(\mathcal{A}_t) = c,$$



(a) *Big and small elements* (b) *Initial alloc. by* GPC

(c) *Reallocation move* (d) *Reallocated elements*

Figure 12.3: Illustration of our ESPASS algorithm. The algorithm first "guesses" (binary searches for) the optimal value c. (a) Then, big elements s where $F(\{s\}) \geq \beta c$ are allocated to separate buckets. (b) Next, the remaining small elements are allocated to empty buckets using the GPC algorithm. (c,d) Finally, elements are reallocated until all buckets are satisfied.

i.e., the minimum score is greater than or equal to c if and only if the average truncated score is c. In Chapter 10, we have already successfully applied this observation in order to develop the SATURATE algorithm.

Now suppose someone tells us the value c^* attained by an optimal solution, i.e., $\max_{\mathcal{A}} \min_t F(\mathcal{A}_t) = c^*$. By the above observation this problem is equivalent to solving

$$\max_{\mathcal{A}} \frac{1}{k} \sum_{t=1}^{k} F_{c^*}(\mathcal{A}_t). \tag{12.3}$$

It can be shown [*c.f.*, Fujito, 2000] that for $c \geq 0$, the truncated objective function F_c remains nondecreasing and submodular. Hence, Problem (12.3) is an instance of the *average-case* Problem (12.1). Now we face the challenge that we do *not* generally know the optimal value c^*. We could use a simple binary search procedure to find this optimal value. Hence, if we could *optimally* solve the nondecreasing submodular *average-case* Problem (12.1), we would obtain an optimal solution to the *balanced* Problem (12.2).

Unfortunately, as shown in Section 12.2.1, solving the average-case problem is **NP**-hard, and, using GPC, we can only solve it *approximately*, obtaining a solution that achieves at least half of the optimal value. In the following, we will show how we can turn this approximate solution for the average-case problem into a near-optimal solution for the balanced problem.

Our algorithm will maintain one "bucket" $\mathcal{A}_t \subseteq \mathcal{V}$ for each time slot t. Since our goal is to develop an approximation algorithm achieving at least a fraction $\beta > 0$ of the optimal sensing quality, we need to allocate m elements $s \in \mathcal{V}$ to the k buckets such that $F(\mathcal{A}_t) \geq \beta c^*$ for all buckets $\mathcal{A}_t$. Hereby, β is a constant that we will specify later. We call a bucket "satisfied" if $F(\mathcal{A}_t) \geq \beta c^*$, "unsatisfied" otherwise. Here is an outline of our ESPASS algorithm, Figure 12.3 presents an illustration.

1. "Guess" the optimal value c.
2. Call an element $s \in \mathcal{V}$ "big" if $F_c(\{s\}) \geq \beta c$ and "small" otherwise. Put each big element into a separate bucket (*c.f.*, Figure 12.3(a)). From now on, we ignore those satisfied buckets, and focus on the unsatisfied buckets.
3. Run GPC to optimize F_c and allocate the small elements to the unsatisfied buckets (*c.f.*, Figure 12.3(b)).
4. Pick a "satisfied" bucket $\mathcal{A}_t$ that contains sufficiently many elements, and reallocate enough elements to an "unsatisfied" bucket to make it satisfied (*c.f.*, Figures 12.3(c) and 12.3(d)). Repeat step 3 until no more buckets are unsatisfied or no more reallocation is possible. We will show that this reallocation will always terminate.
5. If all buckets are satisfied, return to step 1 with a more optimistic (higher) "guess" for c. If at least one bucket remains unsatisfied, return to step 1 with a more pessimistic (lower) guess for c.

ESPASS terminates with a value for c such that *all* buckets t have been assigned elements $\mathcal{A}_t$ such that $F(\mathcal{A}_t) \geq \beta c$. It guarantees that upon termination, c is an upper bound on the value of the optimal solution, hence providing a β approximation guarantee. In Section 12.3.2 we will show that $\beta = \frac{1}{6}$ suffices. In summary, we have the following guarantee about ESPASS:

Theorem 12.2. *For any nondecreasing submodular function F and constant $\varepsilon > 0$,* ESPASS*, using* GPC *as subroutine, returns a solution $\mathcal{A}_1, \ldots, \mathcal{A}_k$ such that*

$$\min_t F(\mathcal{A}_t) \geq \frac{1}{6} \max_{\mathcal{A}'} \min_t F(\mathcal{A}'_t) - \varepsilon.$$

ESPASS *requires at most $\mathcal{O}\left((1 + \log_2 F(\mathcal{V})/\varepsilon)kmn\right)$ evaluations of F.*

Hereby, ε is an tolerance parameter that can be made arbitrarily small. The number of iterations increases only logarithmically in $1/\varepsilon$.

12.3.2 Algorithm details

We will now analyze each of the steps of ESPASS in detail. The pseudocode is given in Algorithm 12.2.

Removing big elements. The main challenge when applying the GPC algorithm to the truncated Problem (12.3) is exemplified by the following pathological example. Suppose the optimal value is c. GPC,

Algorithm 12.2: The ESPASS algorithm for simultaneously placing and scheduling sensors.

Algorithm ESPASS (F, $\mathcal{V}$, k, m, ε)
$c_{\min} \leftarrow 0$; $c_{\max} \leftarrow F(\mathcal{V})$; $\beta \leftarrow 1/6$;
while $c_{\max} - c_{\min} \geq \varepsilon$ **do**
  $c \leftarrow (c_{\max} + c_{\min})/2$;
1 $\mathcal{B} \leftarrow \{s \in \mathcal{V} : F_c(\{s\}) \geq \beta c\}$;
  $k' \leftarrow k$;
2 **foreach** $s \in \mathcal{B}$ **do**
    $\mathcal{A}_{k'} \leftarrow \{s\}$; $k' \leftarrow k' - 1$;
    if $k' = 0$ **then** $c_{\min} \leftarrow c$;
      $\mathcal{A}_{best} \leftarrow (\mathcal{A}_1, \ldots, \mathcal{A}_k)$;
      continue with while loop;
  end
  $\mathcal{V}' \leftarrow \mathcal{V} \setminus \mathcal{B}$; $m' \leftarrow m - |\mathcal{B}|$;
3 $\mathcal{A}_{1:k'} \leftarrow GPC(F_c, \mathcal{V}', k', m')$;
4 **if** $\sum_t F(\mathcal{A}_t) < k'c/2$ **then** $c_{\max} \leftarrow c$; **continue**;
  else
5   **while** $\exists i, j \leq k'$: $F_c(\mathcal{A}_j) \leq \beta c$, $F_c(\mathcal{A}_i) \geq 3\beta c$ **do**
      foreach $s \in \mathcal{A}_i$ **do**
        $\mathcal{A}_j \leftarrow \mathcal{A}_j \cup \{s\}$; $\mathcal{A}_i \leftarrow \mathcal{A}_i \setminus \{s\}$;
        if $F_c(\mathcal{A}_j) \geq \beta c$ **then break**;
      end
    end
    $c_{\min} \leftarrow c$; $\mathcal{A}_{best} \leftarrow (\mathcal{A}_1, \ldots, \mathcal{A}_k)$;
  end
end

when applied to the truncated function F_c, could pick $k/2$ elements $s_1, \ldots, s_{k/2}$, with $F(\{s_i\}) = c$ each. While this solution obtains an average-case score of $c/2$ (one half of optimal as guaranteed by Theorem 12.1), there is no possibility to reallocate these $k/2$ elements into k buckets, and hence some buckets will remain empty, giving a balanced score of 0.

To avoid this pathological case, we would like to eliminate such elements $s \in \mathcal{V}$ with high individual scores $F(\{s\})$, to make sure that we can rearrange the solution of GPC to obtain high balanced score. Hence, we distinguish two kinds of elements: Big elements s with $F(\{s\}) \geq \beta c$, and small elements s with $F(\{s\}) < \beta c$. If we intend to obtain a β approximation to the optimal score c, we realize that big elements have high enough value to each satisfy an individual bucket. Let $\mathcal{B}$ be the set of big elements (this set is determined in Line 15). If $|\mathcal{B}| \geq k$, we already have a β-approximate solution: Just put one big element in each bucket. If $|\mathcal{B}| < k$, put each element in $\mathcal{B}$ in a separate bucket $\mathcal{A}_1, \ldots, \mathcal{A}_{|\mathcal{B}|}$ (*c.f.*, Line 15). We can now set these satisfied buckets aside, and look at the reduced problem instance with elements $\mathcal{V}' = \mathcal{V} \setminus \mathcal{B}$, $m' = m - |\mathcal{B}|$ and $k' = k - |\mathcal{B}|$. Our first lemma shows that if the original problem instance $(F, \mathcal{V}, k, m)$ has optimal value c, the reduced problem instance $(F, \mathcal{V}', k', m')$ still has optimal value c.

Lemma 12.3. *The optimal value on the new problem instance $(F, \mathcal{V}', k', m')$ is still c.*

Hence, without loss of generality, we can now assume that for all $s \in \mathcal{V}$, $F(\{s\}) \leq \beta c$.

Solving the average-case problem. In the next step of the algorithm, we run an α-approximate algorithm (such as GPC where $\alpha = \frac{1}{2}$), using the truncated objective F_c, on the reduced problem instance containing only small elements (*c.f.*, Line 15). This application results in an allocation $\mathcal{A}_1, \ldots, \mathcal{A}_{k'}$ of elements into buckets. If $\sum_t F_c(\mathcal{A}_t) < \alpha c k'$, then we know that c is an upper bound to the optimal solution, and it is safe to set c_{max} to c (*c.f.*, Line 15) and continue with the binary search. Otherwise, we have a solution where $\sum_t F_c(\mathcal{A}_t) \geq \alpha c k'$. However, as argued in Section 12.2.2, this α-approximate solution could still have balanced score 0, if all the elements are allocated to only the first $\alpha k'$ buckets. Hence, we need to reallocate elements from satisfied into unsatisfied buckets to obtain a balanced solution.

Reallocation. We will transfer elements from satisfied buckets to unsatisfied buckets, until all buckets are satisfied. Let us define a "reallocation move" as follows (*c.f.*, Line 15). Pick a bucket $\mathcal{A}_i = \{a_1, \ldots, a_l\}$ for which $F_c(\mathcal{A}_i) \geq 3\beta c$ (we will guarantee that such a bucket always exists), and a bucket $\mathcal{A}_j$ that is not satisfied, i.e., $F_c(\mathcal{A}_j) < \beta c$. Choose ℓ such that $F_c(\{a_1, \ldots, a_{\ell-1}\}) < \beta c$ and $F_c(\{a_1, \ldots, a_\ell\}) \geq \beta c$. Let $\Delta = \{a_1, \ldots, a_\ell\}$. Note that Δ is not empty since each a_i is small (i.e., $F_c(\{a_i\}) < \beta c$). We reallocate the elements Δ by removing Δ from $\mathcal{A}_i$ and adding Δ to $\mathcal{A}_j$.

Lemma 12.4. *It holds that*

$$F_c(\mathcal{A}_j \cup \Delta) \geq \beta c, \text{ and } F_c(\mathcal{A}_i \setminus \Delta) \geq F_c(\mathcal{A}_i) - 2\beta c.$$

Hence, removing elements Δ does not decrease the value of $\mathcal{A}_i$ by more than $2\beta c$, and thus $\mathcal{A}_i$ remains satisfied. On the other hand, the previously unsatisfied bucket $\mathcal{A}_j$ becomes satisfied by adding the elements Δ. We want to make sure that we can always execute our reallocation move, until all buckets are satisfied. The following result shows that if we choose $\beta = \frac{\alpha}{3}$, this will always be the case:

Lemma 12.5. *If we set $\beta = \frac{\alpha}{3}$, then, after at most k reallocation moves, all buckets will be satisfied, i.e., $F_c(\mathcal{A}_i) \geq \beta c$ for all i.*

Binary search. Since the optimal value c is generally not known, we have to search for it. This is done using a simple binary search strategy, starting with the interval $[0, F(\mathcal{V})]$ which is guaranteed the optimal value due to nondecreasingness. At every step, we test the center c of the current interval. If all buckets can be filled to βc, then the truncation threshold c can be increased. If the algorithm for maximizing the average-case score (such as GPC) does not return a solution of value at least αc, then that implies that the optimal value has to be less than c, and the truncation threshold is decreased (*c.f.*, Line 15). If F takes only integral values, then after at most $\lceil \log_2 F(\mathcal{V}) \rceil + 1$ iterations, the binary search terminates.

12.3.3 Improving the bounds

The bound in Theorem 12.2 is "offline" – we can state it independently of the specified problem instance. While guaranteeing that the obtained solutions cannot be arbitrarily bad, the constant factor 6 bound for ESPASS is typically rather weak, and we can show that our obtained solutions are typically much closer to the optimal value.

We can do this by computing the following data-dependent bounds on the optimal value. Let $\mathcal{A}' = (\mathcal{A}'_1, \ldots, \mathcal{A}'_k)$ be a candidate solution to Problem (12.2) (e.g., obtained using the ESPASS algorithm or any other algorithm). For every $1 \leq \ell \leq k$ and $s \in \mathcal{V}$ let $\delta_{\ell,s} = F(\mathcal{A}'_\ell \cup \{s\}) - F(\mathcal{A}'_\ell)$ be the increment in function value when adding sensor s to time slot ℓ.

Theorem 12.6. *The optimal value $c^* = \max_{\mathcal{A}} \min_t F(\mathcal{A}_t)$ is bounded by the solution c to the following linear program:*

$$\max_{\lambda_{i,s},c} c \text{ s.t.}$$
$$c \leq F(\mathcal{A}_i) + \sum_s \lambda_{i,s}\delta_{i,s} \text{ for all } i$$
$$\sum_i \lambda_{i,s} \leq 1 \text{ for all } s \text{ and } \sum_{i,s} \lambda_{i,s} \leq m$$

Theorem 12.6 states that for any given instance of the SPASS problem, and for a candidate solution $\mathcal{A}$ obtained by using *any* algorithm (not necessarily using ESPASS), we can solve a linear program to efficiently get an upper bound on the optimal solution. In Section 12.5 we will show that these bounds prove that our solutions obtained using ESPASS are often much closer to the optimal solution than guaranteed by the bound of Theorem 12.2.

12.4 Trading off power and accuracy

As argued in Section 12, deploying a larger number of sensors and scheduling them has the advantage over deploying a small number of sensors with large batteries that a high density mode can be supported. In contrast to scheduled mode, where the sensors are activated according to the schedule, in high-density mode all sensors are active, to provide higher resolution sensor data (e.g., to localize the boundary of a traffic congestion in our running example). For a fixed solution $\mathcal{A}_1, \ldots, \mathcal{A}_k$ to the SPASS problem, the (balanced) scheduled-mode sensing quality is $\min_i F(\mathcal{A}_i)$, whereas the high-density sensing quality is $F(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k)$. Note that optimizing for the scheduled sensing quality does not necessarily lead to good high-density sensing quality. Hence, if both modes of operation should be supported, then we

should simultaneously optimize for both performance measures. One such approach to this multicriterion optimization problem is to define the scalarized objective

$$\widehat{F}_\lambda(\mathcal{A}_1, \ldots, \mathcal{A}_k) = \lambda \min_i F(\mathcal{A}_i) + (1-\lambda) F(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k),$$

and then solve the problem

$$\max_{\mathcal{A}_1 \ldots \mathcal{A}_k} \widehat{F}_\lambda(\mathcal{A}_1, \ldots, \mathcal{A}_k) \text{ s.t. } \mathcal{A}_i \cap \mathcal{A}_j = \emptyset \text{ if } i \neq j, |\bigcup_t \mathcal{A}_t| \leq m. \tag{12.4}$$

Note that if $\lambda = 1$, we recover the SPASS problem. Furthermore, as $\lambda \to 0$, the high-density sensing quality $F(\mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k)$ dominates, and the chosen solution will converge to the stage-wise approach, where first the set $\mathcal{A}$ of all sensors is optimized, and then this placement is partitioned into $\mathcal{A} = \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k$. Hence, by varying λ between 1 and 0, we can interpolate between the simultaneous and the stage-wise placement and scheduling.

We modify ESPASS to approximately solve Problem (12.4), and call the modified algorithm MCSPASS. The basic strategy is still a binary search procedure. However, instead of simply picking all available big elements (as done by ESPASS), MCSPASS will also guess (search for) the number ℓ of big elements used in the optimal solution. It will pick these big elements in a greedy fashion, resulting in a set $\mathcal{A}_{big} \subseteq \mathcal{V}$. For a fixed guess of c and ℓ, MCSPASS will again use GPC as a subroutine. However, the objective function used by GPC will be modified to account for the high-density performance:

$$G(\mathcal{A}_1, \ldots, \mathcal{A}_{k'}) = \lambda \sum_i F_c(\mathcal{A}_i) + (1-\lambda) F(\mathcal{A} \cup \mathcal{A}_{big}),$$

where $\mathcal{A} = \mathcal{A}_1 \cup \ldots \mathcal{A}_{k'}$, and $k' = k - \ell$. This modified objective function combines a component (weighted by λ) that measures the scheduled performance, as well as a component (weighted by $1 - \lambda$) that measures the improvement in high-density performance, taking into account the set $\mathcal{A}_{big}$ of big elements that have already been selected. The reallocation procedure remains the same as in ESPASS. The remaining details of our MCSPASS approach are presented in the proof to the following theorem, which can be found in the Appendix.

Theorem 12.7. *For any nondecreasing submodular function F and constants $\varepsilon > 0$ and $0 \leq \lambda \leq 1$,* MCSPASS *will efficiently find a solution $\mathcal{A}_1, \ldots, \mathcal{A}_k$ such that*

$$\widehat{F}_\lambda(\mathcal{A}_1, \ldots, \mathcal{A}_k) \geq \frac{1}{8} \max_{\mathcal{A}'} \widehat{F}_\lambda(\mathcal{A}'_1, \ldots, \mathcal{A}'_k) - \varepsilon.$$

In Section 12.5 we will see that we can use this extension to obtain placements and schedules that perform well both in scheduled and high-density mode.

12.5 Experiments

In our experiments, we analyze several case studies on different data sets. Each case study presents a different justification of the SPASS optimization problem.

12.5.1 Case study I: Highway monitoring

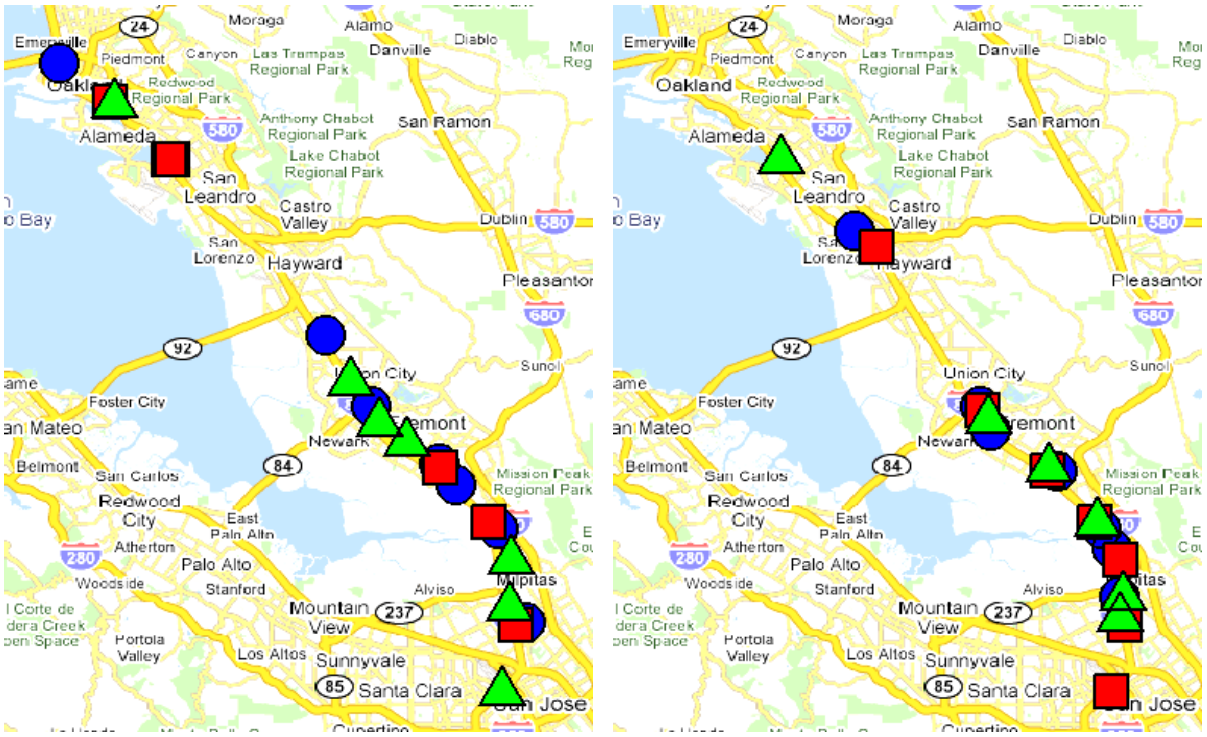
The California highways are currently monitored by over 10,000 traffic sensors based on older technologies. As these loops fail, they are being replaced by novel wireless sensor networks technologies, and it is an important problem to identify economic deployment strategies. PeMS [UC Berkeley, PATH and Caltrans, 2005] is a website and project that integrates, cleanses and tracks real time traffic information for the whole state, computing key performance indicators. The sensors typically report speed, flow and vehicle counts every 30 seconds, and PeMS aggregates the data further into 5 minute blocks. For this case study, we use data from highway I-880 South, which extends for 35 miles in northern California (Figure 12.4) and has between 3 and 5 lanes. This highway experiences heavy traffic, and accurate measurements are essential for proper resource management. Measurement variation is mainly due to congestion and events such as accidents and road closures. There are 88 measurement sites along the highway, on average every 2 miles, which comprise 357 sensors covering all lanes. We use speed information from lanes, for all days of the week in a single month, excluding weekends and holidays for the period from 6AM to 11AM, which is the time when the highway is congested. This is the most difficult time for making predictions, as when there is no congestion, even a free flow speed prediction of 60 mph is accurate.

The number and locations of sensors are limited by costs and physical deployment constraints. Typically at each location, it is only possible to place one sensor at each lane. Furthermore, lane closures for sensor installations are very costly. Given these constraints, California requires that sensor technologies have a target lifetime of 10 years. This implies that most wireless sensor solutions require intelligent scheduling in order to extend the lifetime by four times, since most sensor network solutions batteries are expected to last 2 to 3 years. Including more batteries in a single sensor is not viable, as sensors have physical constraints to avoid disrupting the existing pavement structure and keep installation costs at a minimum.

As wireless sensors displace existing loop technologies, it is desirable to place as few sensors as possible, without trading off too much sensing quality. To achieve these goals in a principled manner, historical loop data from the current deployment should be used. ESPASS provides a solution which can balance these conflicting requirements, by combining scheduling and placement: more sensors are placed initially, still keeping road closures at a minimum, and scheduling is used to extend the lifetime of the network, keeping sensing quality balanced. In this section we explore this solution and compare ESPASS to other simultaneous placement and scheduling solutions.

Simultaneous vs. stage-wise optimization. In our first experiment, we study the benefit of simultaneously placing and scheduling sensors. For varying numbers m of sensors and k of time slots, we use different strategies to find k disjoint sets $\mathcal{A}_1, \ldots, \mathcal{A}_k$, where $\mathcal{A}_i$ is the sensors active at time slot i. We compare the simultaneous placement and schedule (optimized using ESPASS and GPC) with solutions obtained by first placing sensors at a fixed set of locations, and then scheduling them. We consider both optimized and random sensor placements, followed by optimized and random scheduling, amounting to four stage-wise strategies. For random placements and schedules, we report the mean and standard error over 20 random trials.

Figure 12.5(a) presents the performance of the five strategies when optimizing the average-case performance, for a fixed number of $m = 50$ sensors and a number of time slots k varying from 1 to 20. GPC performs best, followed by the stage-wise optimized placement and schedule (OP/OS). Of the two strategies where one component (either the placement or the schedule) is randomized (OP/RS and RP/OS),



(a) *[T] stage-wise* (b) *[T] Simultaneous*

Figure 12.4: Placements and schedules for the traffic data. (a) Stage-wise approach, (b) SPASS solution.

for small numbers ($\leq$ 3) of time slots OP/RS performs slightly better, and for large numbers of time slots ($\geq$ 10), RP/OS performs slightly better. The completely randomized solution performs significantly worse.

Figure 12.5(b) presents the same results when optimizing the balanced criterion. ESPASS outperforms the stage-wise strategies and the completely randomized strategy RP/RS performs worst as expected. Interestingly, for the balanced criterion, OP/RS performs drastically worse than RP/OS for $k \geq 4$ time slots. We hypothesize this to be due to the fact that a poor random placement can more easily be compensated for by using a good schedule than vice versa: When partitioning a sensor placement of 50 sensors randomly into a large number of time slots, it is fairly likely that at least one of the timeslots exhibits poor performance, hence leading to a poor balanced score. This insight also suggests that the larger the intended improvement in network lifetime (number of time slots), the more important it is to optimize for a balanced schedule.

To summarize this analysis, we see that simultaneous placement and scheduling drastically outperforms the stage-wise strategies. For example, if we place 50 sensors at random, and then use ESPASS to schedule them into 4 time slots, we achieve an estimated minimum reduction in Mean Squared error by 58%. If we first optimize the placement and then use ESPASS for scheduling, we can achieve the same amount of variance reduction by scheduling 6 time slots (hence obtaining a 50% increase in network lifetime). If instead of stage-wise optimization we simultaneously optimize the placement and the schedule using ESPASS, we can obtain the same variance reduction by scheduling 8 time slots, hence an increase in

network lifetime by 100%.

Average vs. balanced performance. We have seen that simultaneously placing and scheduling can drastically outperform stage-wise strategies, for both the average-case and the balanced objective. But which of the objectives should we use? In order to gain insight into this question, we performed the following experiment. For varying k and m, we obtain solutions to the SPASS problem using both the ESPASS and the GPC algorithm. We then evaluate the respective solutions both using the average-case and the balanced criterion. Figure 12.5(c) presents the results of this experiment for k varying from 1 to 10, and fixed ratio of 5 sensors per time slot. As expected, ESPASS outperforms GPC with respect to the balanced criterion, and GPC outperforms ESPASS according to the average-case criterion. However, while ESPASS achieves average-case score very close to the solution obtained by GPC, the balanced score of the GPC solutions are far worse than those obtained by ESPASS. Hence, optimizing for the balanced criterion performs well for the average case, but not vice versa.

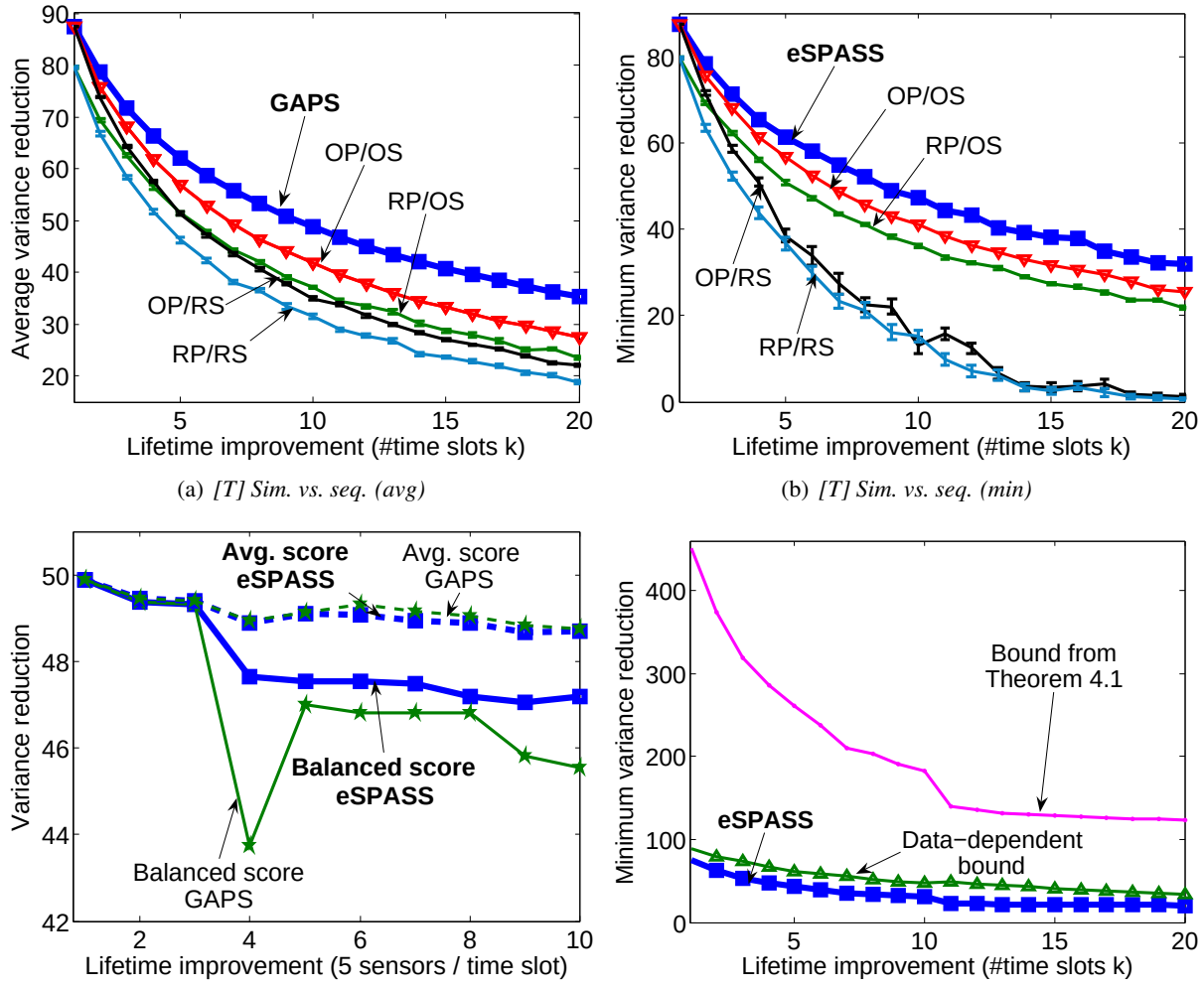
Online bounds. In order to see how close the solutions obtained by ESPASS are to the optimal solution, we also compute the bounds from Theorem 12.6. Figure 12.5(d) presents the bounds on the maximum variance reduction achievable when placing 50 sensors and partitioning them into an increasing number of groups. We plot both the factor 6 bound due to Theorem 12.2, as well as the data-dependent bound due to Theorem 12.6. We can see that the data dependent bounds are much tighter. For example, if we partition the sensors into 2 groups, our solution is at least 78% of optimum, for 5 groups it is at least 70% of optimum (rather than the 17% of Theorem 12.2).

12.5.2 Case study II: Community Sensing

While the static deployment of sensors has become an important means for monitoring traffic on highways and arterial roads, due to high deployment and maintenance cost it is difficult to extend sensor coverage to urban, side-streets. However, in order to optimize road-network utilization, accurate estimates of side-street conditions is essential.

Instead of (or in addition to) statically deploying sensors, a promising approach, studied by Krause et al. [2008c], would be to utilize cars as traffic sensors: An increasing number of vehicles nowadays are equipped with GPS and Personal Navigation Devices, which can accurately localize a car on a road network. Furthermore, these devices are becoming connected to wireless networks, using, e.g., GPRS or Edge connectivity, through which they could report their location and speed. Hence, in principle, it is possible to access accurate sensor data through the network of cars.

However, there are significant challenges when dealing with such a network of non-centrally owned sensors. While users may generally consider sharing their sensor data, they have reasonable concerns about their privacy. Krause et al. [2008c] provide methods for *community sensing*, describing strategies for selectively querying a community sensor network while maintaining preferences about privacy. They demonstrated how the selective querying of such a community sensor network can be modeled as the problem of optimizing a nondecreasing submodular sensing quality function that considers demand based on road usage. Preferences about privacy map to constraints in the optimization problem.



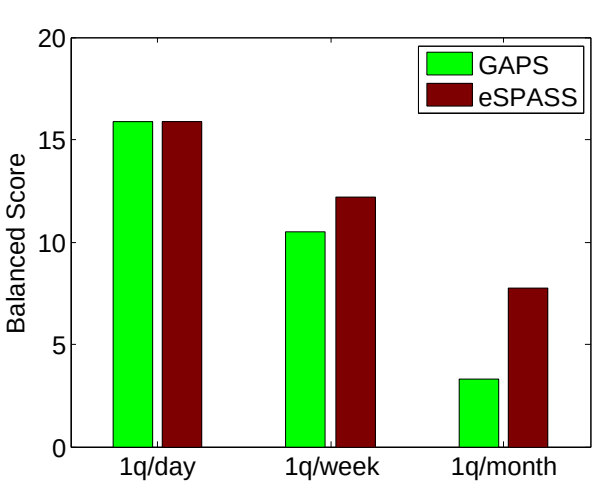
(a) *[T] Sim. vs. seq. (avg)*

(b) *[T] Sim. vs. seq. (min)*

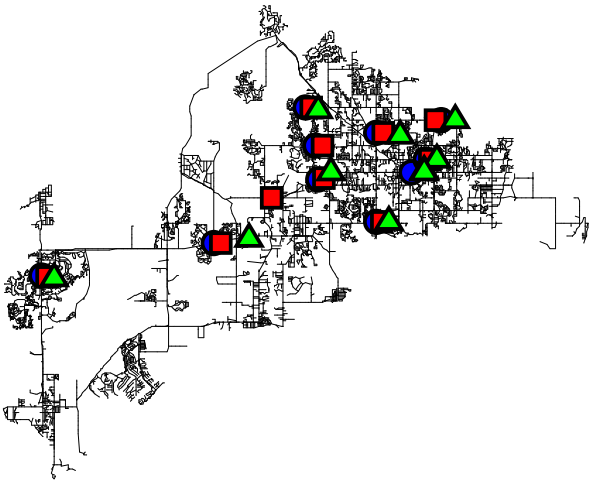
(c) *[T] Avg. vs. balanced*

(d) *[T] Bounds*

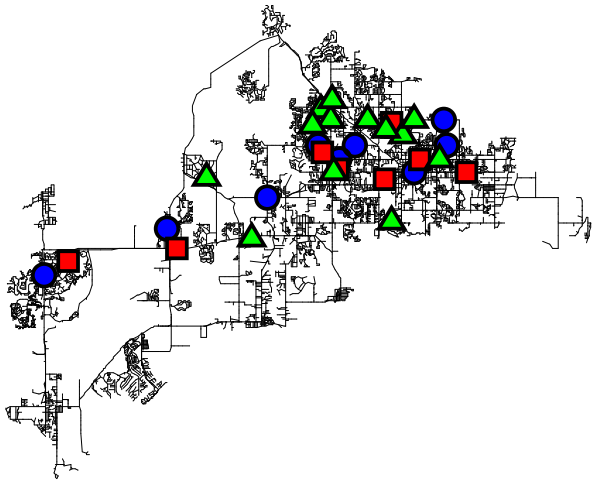
Figure 12.5: Results for traffic monitoring [T]. (a,b) compare simultaneous placement and scheduling to stage-wise strategies on (a) average-case and (b) balanced performance ($m = 50$, k varies). (c) compare average-case and balanced performance, when optimizing for average-case (using GPC) and balanced (using ESPASS) performance. (d) "Online" (data-dependent) bounds show that the ESPASS solutions are closer to optimal than the factor 6 "offline" bound from Theorem 12.2 suggests.



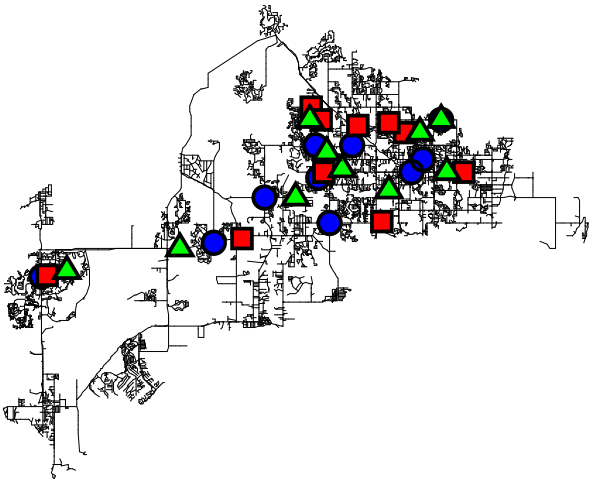
(a) *[C] Avg. vs. balanced*



(b) *[W] Simultaneous*

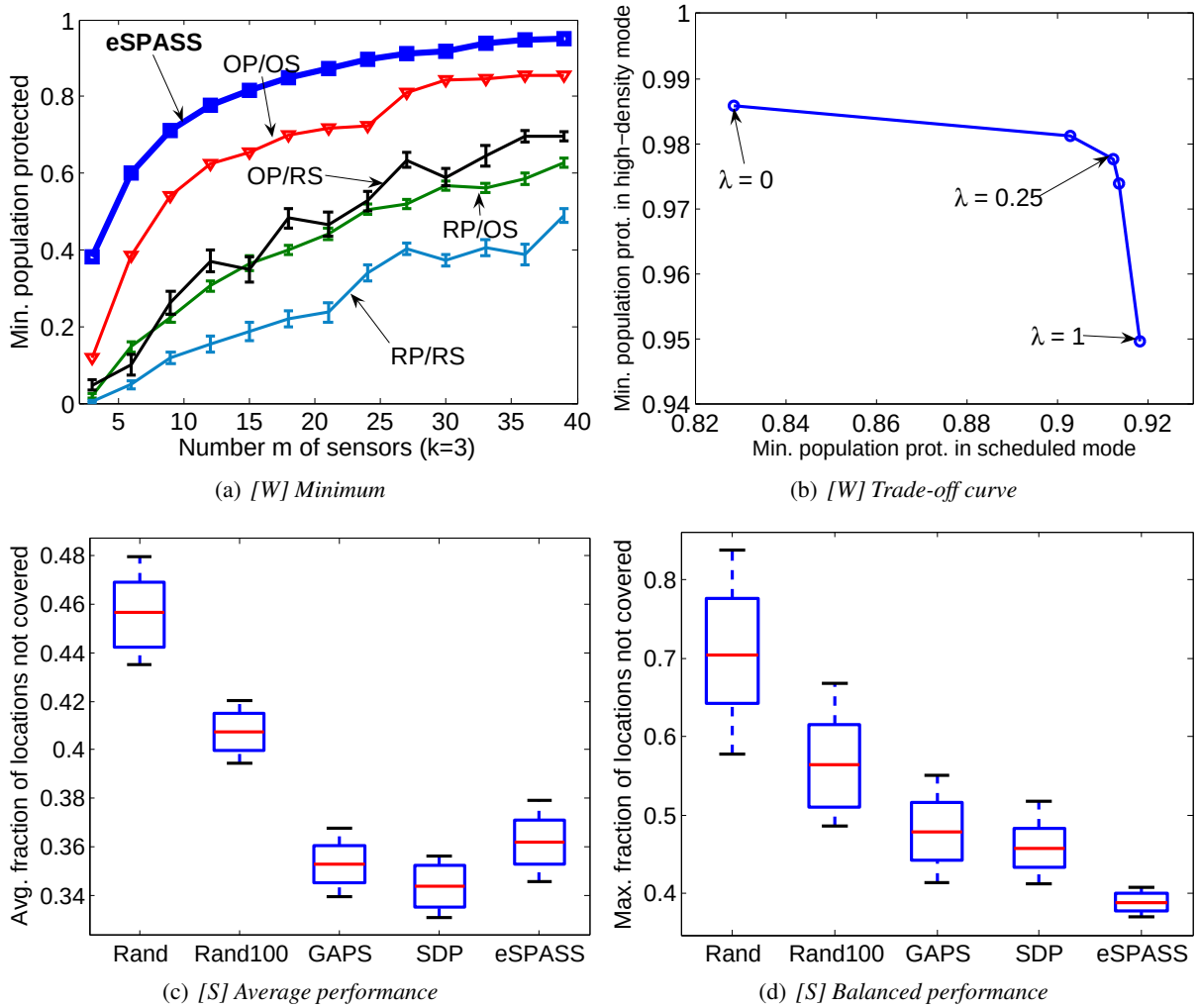


(c) *[W] Stage-wise*



(d) *[W] Multicrit. ($\lambda = 0.25$)*

Figure 12.6: (a) Results for community sensing [C]. When querying each car only once each week (using the ESPASS schedule), the sensing quality is only 23% lower than when querying every day. (b-d) Example placements and schedules for water networks [W].



(a) *[W] Minimum*

(b) *[W] Trade-off curve*

(c) *[S] Average performance*

(d) *[S] Balanced performance*

Figure 12.7: (a,b) Contamination detection in water networks [W]. (a) compares simultaneous and stage-wise solutions. (b) power/accuracy tradeoff curve with strong knee. (c,d) compares ESPASS with existing solutions on synthetic data [S].

One basic preference that needs to be supported is the preference that each user is queried at most once in a specified time interval (e.g., queried at most once each week or month). Let $\mathcal{V}$ be the set of users subscribing to the community sensing service. In order to query each user at most one in k time steps, one strategy would be to partition the users into k sets $\mathcal{A}_1, \ldots, \mathcal{A}_k$, such that at time step t, users $\mathcal{A}_t$ are queried. In order to obtain continuously high performance of the monitoring service, we want to make sure that the performance $F(\mathcal{A}_t)$ is maximized simultaneously over all time steps. This is exactly an instance of the SPASS problem.

In order to evaluate the performance of the ESPASS algorithm, we used the experimental setup of Krause et al. [2008c], using real traffic data from 534 detector loops deployed underneath highways, GPS traces from 85 volunteer drivers and demand data based on directions generated in response to requests to a traffic prediction and route planning prototype named ClearFlow, developed at Microsoft Research[3]. Details about the data sets are described by Krause et al. [2008c]. Based on this experimental setup, we compare the performance of the ESPASS and GPC algorithms (latter of which was described as a candidate scheduling approach by Krause et al. [2008c]). Using each algorithm, we partition the users into 7 respective 31 sets (i.e., querying each user at most once each week respective month, both of which are possible options for privacy preferences). We then evaluate the performance based on the worst case prediction error over all test time steps. Figure 12.6(a) presents the results of this experiment. We can see that the ESPASS solutions outperform the GPC solutions. When partitioning into 31 sets, the worst prediction performance of ESPASS is more than twice as good than the worst prediction performance of GPC. Most importantly, this experiment shows that, using ESPASS for scheduling, one can obtain a very high balanced performance, even when querying each individual car only very infrequently. For example, when querying each car only once each week, the balanced sensing quality is only 23% lower than that obtained by the privacy-intrusive continuous (daily) querying. Even if querying only once each month (i.e., a factor 31 more infrequently), the balanced performance is only reduced by approximately a factor of 2. These results indicate that, using ESPASS, even stringent preferences about privacy can be met without losing much prediction accuracy.

12.5.3 Case study III: Contamination detection

We also apply our algorithm to the problem of placing sensors in municipal drinking water distribution networks, in order to detect contaminations (as introduced in Chapter 8).

The goal of this application is to minimize impact measures, such as the expected population affected, which is calculated using a realistic disease model. In Chapter 8, we showed that the function $F(\mathcal{A})$ which measures the expected population protected by placing sensors at location $\mathcal{A}$ is a nondecreasing submodular function. Water quality sondes can operate for a fairly long amount of time on battery power. For example, the YSI 6600 Sonde can sample 15 water quality parameters every 15 minutes for 75 days. However, for the long-term feasibility it is desirable to considerably improve this battery lifetime by sensor scheduling. On the other hand, high sampling rates are desirable to ensure rapid response to possible contaminations. For a security-critical sensing task such as protecting drinking water from contamination, it is important to obtain balanced, uniformly good detection performance over time. In addition, deployment and maintenance cost restrict the number of sensors that can be deployed. Hence, the problem of deploying battery powered sensors for drinking water quality monitoring is another natural instance of the SPASS problem.

[3]The ClearFlow research system, available only to users within Microsoft Corporation, was the prototype for the Clearflow context-sensitive routing service now available publicly for North American cities at http://maps.live.com.

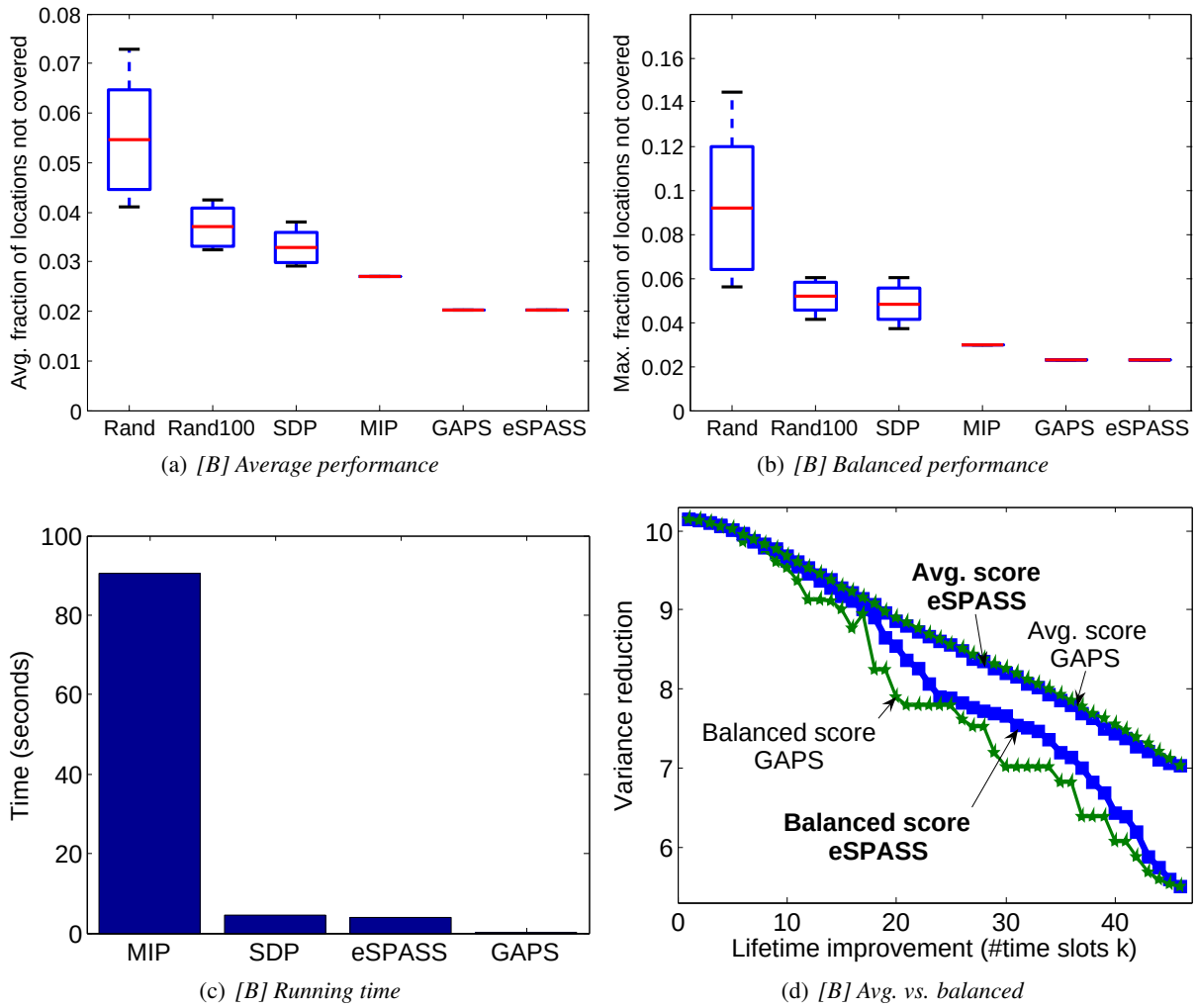
We reproduce the experimental setup detailed in Section 8.2. However, instead of only optimizing for the sensor placement, we simultaneously optimize for placement and schedule using the ESPASS algorithm. Figure 12.7(a) compares ESPASS with the stage-wise approaches. For each algorithm, we report the population protected by placing sensors, normalized by the maximum protection achievable when placing sensors at every node in the network.

Simultaneous vs. stage-wise optimization. ESPASS obtains drastically improved performance when compared to the stage-wise approaches. For example, when scheduling 3 time slots, in order to obtain 85% protection, ESPASS requires 18 sensors. The fully optimized stage-wise approach (OP/OS) requires twice the number of sensors. When placing 36 sensors, the stage-wise approach leaves 3 times more population unprotected as compared to the simultaneous ESPASS solution with the same number of sensors. ESPASS solved this large scale optimization task ($n = 12,527$, $k = 3$, $m = 30$) in 26 minutes using our MATLAB implementation.

Trading off power and accuracy. We also applied our modified ESPASS algorithm in order to trade off scheduled mode and high density mode performance. For a fixed number of $m = 30$ sensors and $k = 3$ time slots, we solve Problem (12.4) for values of λ varying from 0 to 1. For each value of λ, we obtain a different solution, and plot the normalized expected population protected (higher is better) both in scheduled- and in high-density mode in Figure 12.7(b). We can see that this trade-off curve exhibits a prominent knee, where solutions are obtained that perform nearly optimally with respect to both criteria. Figures 12.6(b), 12.6(c) and 12.6(d) show the placements and schedules obtained for $\lambda = 1$ (i.e., ignoring the high-density sensing quality), $\lambda = 0$ (ignoring the schedule, effectively performing a stage-wise approach) and a value $\lambda = 0.25$ (from the knee in the trade-off curve) respectively. Note how the solution for $\lambda = 1$ clusters the sensors closely together, obtaining three very similar placements $\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3$ for each time slot (similar as in Figure 12.2). The solution for $\lambda = 0$ spreads out the sensors more, having to leave, e.g., the Western part of the network uncovered in the time slot indicated by the green triangle. The multicriterion solution ($\lambda = 0.25$) is a compromise between the former two solutions: The sensors are still clustered together, but also spread out more – the Western part of the network can be covered in this solution.

12.5.4 Case study IV: Multiple weblog coverage

We also apply ESPASS to the problem of selecting informative weblogs to read on the Internet, as introduced in Chapter 8. Recall that our approach to quantifying informativeness in this domain is based on the intuitive notion of an information cascade: A blogger writes a posting, and, after some time, other blogs link to it. An information cascade is a directed acyclic graph of vertices (each vertex corresponds to a posting at some blog), where edges are annotated by the time difference between the postings. Based on this notion of an information cascade, we would like to select blogs, that detect big cascades (containing many nodes) as early as possible (i.e., we want to learn about an important event before most other readers). In Chapter 8, we showed how we can formalize this intuition using a nondecreasing submodular function F that measures the informativeness of a subset $\mathcal{A}$ of all blogs $\mathcal{V}$. Optimizing the submodular function F leads to a small set $\mathcal{A}$ of blogs that "covers" most cascades. Since blogs from different topics (such as politics, religion, tech news, etc.) participate in different cascades, the solution set $\mathcal{A}$ will typically contain very *diverse* blogs. Similar blogs that overlap significantly suffer from diminishing returns.



(a) *[B] Average performance*

(b) *[B] Balanced performance*

(c) *[B] Running time*

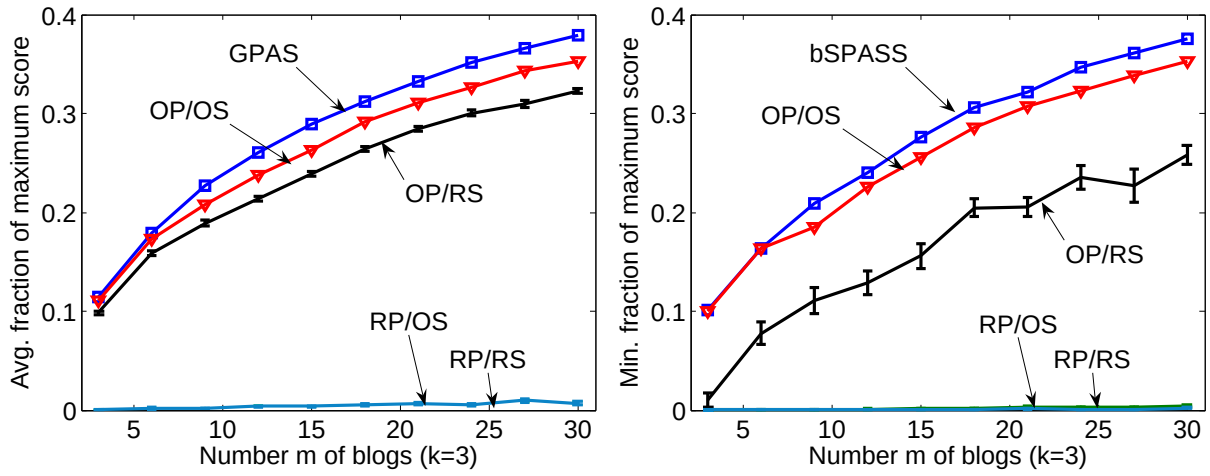
(d) *[B] Avg. vs. balanced*

Figure 12.8: Results on temperature data from Intel Research Berkeley [B]. (a,b) compares ESPASS with existing solutions. (c) compares running time. (d) compares average-case and balanced performance.

While such a notion of coverage is intuitive, sometimes multiple coverage is desired: We might be interested in selecting "multiple representatives" from each topic, whereby we allow some (limited) amount overlap among the covered cascades. For example, there might be political blogs that participate in the same cascades (discussions), and hence have a lot of overlap, but it is nevertheless desirable to select multiple blogs as each of them presents their own perspective.

We can formulate this requirement as an instance of the SPASS problem: We would like to select multiple, disjoint sets of blogs, that each provide a large amount of information.

We reproduce the experimental setup of Leskovec et al. [2007b], that is based on 45,000 blogs, as well as one million posts during 2006. This data set contains 16,000 cascades in which at least 10 blogs participate. We use the *population affected* objective function described in detail in Leskovec et al. [2007b]. Based on this experimental setup, we apply ESPASS to partition an increasing number of blogs into three groups (to allow up to three-fold coverage). We optimize the problem using the ESPASS algorithm. As comparison, we first select the set of blogs (using the approach described in Leskovec et al. [2007b]), and



(a) *[W] Average* (b) *[W] Minimum*

Figure 12.9: Multiple coverage for blogs

then use the ESPASS algorithm to partition the selected set into three groups ($k = 3$). We also compare against the other stage-wise approaches as done in Section 12.5.1.

Figure 12.9 presents the results of this experiment. It shows that blog selections using the ESPASS algorithm obtain 10% better performance than the stage-wise approach. Further note that ESPASS is able to solve the large problem instance with $n = 45,000$ within 26 minutes.

12.5.5 Comparison with existing techniques

We also compare ESPASS with several existing algorithms. Since the existing algorithms apply to the scheduling problem only, we call ESPASS with $m = |\mathcal{V}|$ (i.e., allow it to select all sensors).

Set covering. Most existing algorithms for sensor scheduling assume that sensors are associated with a fixed sensing region that can be perfectly observed by the sensor [*c.f.*, Abrams et al., 2004, Deshpande et al., 2008]. In this setting, we associate with each location $s \in \mathcal{A}$ a set $\mathcal{R}_s \subseteq \mathcal{V}$ of locations that can be monitored by the sensor, and define the sensing quality $F(\mathcal{A}) = |\bigcup_{s \in \mathcal{A}} \mathcal{R}_s|$ to be the total area covered by all sensors. Since set coverage is an example of a nondecreasing submodular function, we can use ESPASS to optimize it.

We compare ESPASS to the greedy approach by Abrams et al. [2004], as well as the the approach by Deshpande et al. [2008] that relies on solving a semidefinite program (SDP). We use the synthetic experimental setup defined by Deshpande et al. [2008] to compare the approaches. A set of n sensors is used to cover M regions. Each sensor s is associated with a set $\mathcal{R}_s$ of regions it covers. The objective is to divide the n sensors into k groups (buckets), such that the minimum or the average number regions covered by each group is maximized.

For the SDP by Deshpande et al. [2008], we solve the SDP using SeDuMi to get a distribution over possible schedules, and then pick the best solution out of 100 random samples drawn from this distribution. For

the random assignment approach (Rand100) of Abrams et al. [2004], we sample 100 random schedules and pick the best one. In addition, we run the GPC and the ESPASS algorithms. We apply those four algorithms to 50 random set cover instances as defined by Deshpande et al. [2008]: for each sensor, a uniform random integer r between 3 and 5 is chosen, and then the first r regions from a random permutation of the set of M regions is assigned to that sensor. The sensor network size is $n = 20$, the number of desired groups $k = 5$ and the number of regions is $M = 50$.

Figure 12.7(c) presents the average performance of the four approaches. In this setting, the SDP performs best, closely followed by GPC and ESPASS. Figure 12.7(d) presents the balanced performance of the four approaches. Here, ESPASS significantly outperforms both the SDP and the GPC solution.

Building monitoring. As argued in the introduction, for complex spatial monitoring problems, the sensing region assumption is unrealistic, and we would rather like to optimize prediction accuracy directly. The approach by Koushanfary et al. [2006] is designed to schedule sensors under constraints on the prediction accuracy. Their approach, given a required prediction accuracy, constructs a prediction graph that encodes which sensors can predict which other sensors. They then solve a domatic partitioning problem, selecting a maximal number of disjoint subsets that can predict all other sensors with the desired accuracy. In order to determine the domatic partitioning, their algorithm relies on the solution of a Mixed Integer Program (MIP). However, solving MIPs is $\mathbf{NP}$-hard in general, and unfortunately, we were not able to scale their approach to the traffic data application. Instead, we use data from 46 temperature sensors deployed at Intel Research, Berkeley [*c.f.*, Deshpande et al., 2004].

On this smaller data set, we first apply the MIP for domatic partitioning, with a specified accuracy constraint. The MIP was very sensitive with respect to this accuracy constraint. For just slightly too small values of ε, the MIP returned a trivial solution consisting of only a single set. For slightly too large values, the MIP had to consider partitions into a large number of possible time slots, increasing the size of the MIP such that the solver ran out of memory. Requiring that sensors can predict each other with a Root Mean Squared (RMS) error of 1.25 Kelvin leads to a selection of $m = 19$ sensors, partitioned into $k = 3$ time slots. Using this setting for m and k, we run the GPC and ESPASS algorithms, which happen to return the same solution for this example. In order to compare these solutions with the SDP and random selection from the previous section, we apply them to the prediction graph induced by the required prediction accuracy. We first randomly select 19 locations, and then partition them into 3 groups using the SDP and Rand100 approach, respectively. As a baseline, we randomly select 3 groups totaling 19 sensors (Rand). For these randomized techniques, we report the distribution over 20 trials. All approaches are evaluated based on the variance reduction objective function.

Figure 12.8(a) presents the result for optimizing the average variance reduction, and Figure 12.8(b) for the minimum variance reduction. In both settings, GPC and ESPASS perform best, obtaining 23% less remaining maximum variance when compared to the MIP solution of Koushanfary et al. [2006]. Furthermore, using YalMIP in Matlab, solving the MIP requires 95 seconds, as compared to 4 seconds for the SDP and 3.8 seconds for ESPASS (Figure 12.8(c)). Even though the MIP returns an optimum solution for the domatic partition of the prediction graph, ESPASS performs better since it uses the fact that the combination of multiple sensors can lead to better prediction accuracy than only using single sensors for prediction. Even the best out of 20 random trials for the SDP performs worse than the MIP, due to the approximate nature of the algorithm and the random selection of the initial 19 sensors. The Rand100 approach does performs only slightly (not significantly) worse than the SDP based approach.

12.6 Related work

In the context of wireless sensor networks, where sensor nodes have limited battery and can hence only enable a small number of measurements, optimally placing and scheduling sensors is of key importance.

12.6.1 Sensor placement

Many approaches for optimizing sensor placements assume that sensors have a fixed region [Bai et al., 2006, Gonzalez-Banos and Latombe, 2001, Hochbaum and Maas, 1985]. These regions are usually convex or even circular. Furthermore, it is assumed that everything within this region can be perfectly observed, and everything outside cannot be measured by the sensors. For complex applications such as traffic monitoring however, such assumptions are unrealistic, and the direct optimization of prediction accuracy is desired. The problem of selecting observations for monitoring spatial phenomena has been investigated extensively in geostatistics (*c.f.*, Cressie [1991] for an overview), and more generally (Bayesian) experimental design [*c.f.*, Chaloner and Verdinelli, 1995]. These approaches however only consider the sensor placement problem, and not the scheduling aspect.

12.6.2 Sensor Scheduling

The problem of deciding when to selectively turn on sensors in order to conserve power was first discussed by Slijepcevic and Potkonjak [2001] and Zhao et al. [2002]. Typically, it is assumed that sensors are associated with a fixed sensing region, and a spatial domain needs to be covered by the regions associated with the selected sensors. Abrams et al. [2004] presents an efficient approximation algorithm with theoretical guarantees for this problem. Deshpande et al. [2008] presents an approach for this problem based on semidefinite programming (SDP), handling more general constraints and providing tighter approximations. They also provide a randomized rounding based approach for scheduling under the balanced objective (which they call min-coverage (time)). However, in contrast to ESPASS (when specialized to scheduling) their algorithm requires to relax the constraint that each sensor location can only be selected once. Also, their guarantee only holds with high probability, whereas ESPASS is deterministic. The approaches described above do not apply to the problem of optimizing sensor schedules for more complex sensing quality functions such as, e.g., the increase in prediction accuracy and other sensing quality functions considered in this Thesis. To address these shortcomings, Koushanfary et al. [2006] developed an approach for sensor scheduling that guarantees a specified prediction accuracy based on a regression model. However, their approach relies on the solution of a Mixed Integer Program, which is intractable in general. Zhao et al. [2002] proposed heuristics for selectively querying nodes in a sensor network in order to reduce the entropy of the prediction. Unlike the algorithms presented in this chapter, their approaches do not have any performance guarantees.

12.6.3 Submodular optimization

The problem of maximizing a submodular function subject to a matroid constraint, of which Problem (12.1) is an instance, has been studied by Fisher et al. [1978], who proved that the greedy algorithm gives a factor 2 approximation. Recently, Vondrak [2008] showed that a more complex algorithm achieves a (1-1/e)

approximation to this problem. Note that this algorithm could be applied to the Problem (12.1) instead of GPC. Furthermore note that using this algorithm as a subroutine, the analysis of ESPASS can be improved to give a $\frac{e-1}{3e} \geq \frac{1}{5}$ guarantee. Comparing this algorithm is an interesting direction for future work. A related version of optimization Problem (12.2), where for each time step t a different function F_t is used has been studied in the context of combinatorial allocation problems. Ponnuswami and Khot [2007] present an algorithm that guarantees a $1/(2k-1)$ approximation. For the special case where the objective functions F_t are additive (modular), Asadpour and Saberi [2007] developed an algorithm that guarantees an improved $\Omega(\frac{1}{\sqrt{k}\log^3 k})$ approximation. Both algorithms however only apply for the scheduling setting (i.e., they require that $m = |\mathcal{V}|$). Furthermore, note that the approximation performance of these algorithms very quickly decreases with k, in contrast to our ESPASS approach that provides an approximation guarantee that is independent of the number of time steps.

12.7 Summary

When deploying sensor networks for monitoring tasks, both placing and scheduling the sensors are of key importance, in order to ensure informative measurements and long deployment lifetime. Traditionally, the problems of sensor placement and scheduling have been considered separately from each other. In this chapter, we presented an efficient algorithm, ESPASS, that simultaneously optimizes the sensor placement and the schedule. We considered both the setting where the average-case performance over time is optimized, as well as the balanced setting, where uniformly good performance is required. Such balanced performance is crucial for security-critical applications such as contamination detection. Our results indicate that optimizing for balanced performance often yields good average-case performance, but not necessarily vice versa. We proved that our ESPASS algorithm provides a constant factor 6 approximation to the optimal balanced solution. To the best of our knowledge, ESPASS is the first algorithm that provides strong guarantees for this problem, partly resolving an open problem raised by Abrams et al. [2004]. Furthermore, our algorithm applies to any setting where the sensing quality function is submodular, which allows to address complex sensing tasks where one intends to optimize prediction accuracy or optimizes detection performance.

We also considered complex sensor placement scenarios, where the deployed sensor network must be able to function well both in a scheduled, low power mode, but also in a high accuracy mode, where all sensors are activated simultaneously. We developed an algorithm, MCSPASS, that directly optimizes this power-accuracy tradeoff. Our results show that MCSPASS yields solutions which perform near-optimally with respect to both the scheduled and the high-density performance.

We extensively evaluated our approach on several real-world sensing case studies, including traffic and building monitoring as well as contamination detection in metropolitan area drinking water networks. When applied to the simpler special case of sensor scheduling (i.e., ignoring the placement aspect), ESPASS outperforms existing sensor scheduling algorithms on standard data sets. For the more complex, general case, our algorithm performs provably near-optimal (as demonstrated by tight, data-dependent bounds). Our results show that, for fixed deployment budget, drastic improvements in sensor network lifetime can be achieved by simultaneously optimizing the placement and the schedule, as compared to the traditional, stage-wise approach. For example, for traffic prediction, ESPASS achieves a 33% improvement in network lifetime compared to the setting where placement and scheduled are optimized separately, and a 100% improvement when compared to the traditional setting where sensors are first randomly deployed and then optimally scheduled.

Part IV

Sequential Sensing

Chapter 13

Overview of Part IV

In Part II and Part III of the Thesis, we studied subset selection problems, where all observations are decided upon before any measurements are obtained. In this part, we will consider *sequential* selection problems, where the observations made can depend on the measured values.

Active learning in Gaussian processes. In Chapter 14, we will study the sequential selection problem in the context of Gaussian Processes for spatial prediction. In this setting, a fundamental question is when an active learning, or sequential design, strategy, where locations are selected based on previous measurements, will perform significantly better than sensing at an a priori specified set of locations. We present an analysis and efficient algorithms that address this question. Central to our analysis is a theoretical bound which quantifies the performance difference between active and a priori design strategies. We consider GPs with unknown kernel parameters and present a nonmyopic approach for trading off exploration, i.e., decreasing uncertainty about the model parameters, and exploitation, i.e., near-optimally selecting observations when the parameters are (approximately) known. We call our approach EEGP for *Exploration–Exploitation sampling in Gaussian Processes*. We discuss several exploration strategies, and present logarithmic sample complexity bounds for the exploration phase of EEGP. We then extend our EEGP algorithm to handle nonstationary GPs exploiting local structure in the model. A variational approach allows us to perform efficient inference in this class of nonstationary models.

Optimizing decision-theoretic value of information. In the previous chapters we have focused on settings where the sensing quality is submodular. However, this requirement is not met in all practical applications. When using the decision theoretic value of information, for example, in order to decide on optimal medical treatment plans, this requirement is violated. In Chapter 15, we present VOIDP, the first efficient optimal algorithm for optimizing *arbitrary* decomposable sensing quality functions that are not necessarily submodular, for a class of probabilistic graphical models containing Hidden Markov Models (HMMs). VOIDP applies to both selecting the optimal subset of observations, and for obtaining an optimal conditional observation plan. Furthermore we prove a surprising result: In most graphical models tasks, if one designs an efficient algorithm for chain graphs, such as HMMs, this procedure can be generalized to polytree graphical models. We prove that the optimizing value of information is $\mathbf{NP}^{\mathbf{PP}}$-hard even for polytrees. It also follows from our results that just computing decision theoretic value of information objective functions, which are commonly used in practice, is a $\#\mathbf{P}$-complete problem even on Naive Bayes models (a simple special case of polytrees). It is $\mathbf{NP}$-hard to approximate value of

information to any factor. In addition, we consider several extensions, such as using our algorithms for scheduling observation selection for multiple sensors. We demonstrate the effectiveness of our approach on several real-world datasets, including a prototype sensor network deployment for energy conservation in buildings.

Summary of contributions. The key contributions of this part of the Thesis are:

1. We develop novel nonmyopic algorithms for sequential sensing problems, for
   - active learning for spatial prediction in Gaussian Processes and
   - optimal conditional planning in chain graphical models.
2. We prove hardness of sensing problems. Under reasonable complexity theoretic assumptions, exact optimization of value of information is $\#\mathbf{P}$-hard for Naive Bayes models and $\mathbf{NP}^{\mathbf{PP}}$-hard for discrete polytrees.
3. We demonstrate our algorithms on several real-world case studies, including
   - sensor placement for spatial prediction in Gaussian Processes, demonstrated on real-world sensing data sets and a proof-of-concept case study of a deployed sensor network and
   - sensor scheduling for improving the lifetime of light sensors in a building management prototype testbed.

Chapter 14

Nonmyopic Active Learning of GPs

When monitoring spatial phenomena, such as the ecological condition of a river as in Fig. 14.1, it is of fundamental importance to decide on the most informative locations to make observations. However, to find locations which predict the phenomena best, one needs a model of the spatial phenomenon itself. Gaussian processes (GPs) have been shown to be effective models for this purpose [Cressie, 1991, Rasmussen and Williams, 2006].

In the previous parts of this Thesis, we have, as most previous work on observation selection in GPs [*c.f.*, Seo et al., 2000, Zhu and Stein, 2006], considered the *a priori design* problem, in which the locations are selected in advance prior to making observations. Indeed, as already observed in Chapter 6, if the GP model parameters are completely known, the predictive variances do *not* depend on actual observed values, and hence nothing is lost by committing to sampling locations in advance. In the case of unknown parameters however, this independence is no longer true. Key questions we strive to understand in this chapter are *how much* better a *sequential* algorithm, taking into account previous observations, can perform compared to a priori design when the parameters are *unknown*, and how can this understanding lead to better observation selection methods.

The main theoretical result that we present in this chapter is a bound which quantifies the performance difference between sequential and a priori strategies in terms of the parameter entropy of the prior over kernels. The lower the uncertainty about the parameters, the less we can potentially gain by using an active learning (sequential) strategy. This relationship bears a striking resemblance to the exploration–exploitation tradeoff in Reinforcement Learning. If the model parameters are known, we can *exploit* the model by finding a near-optimal policy for sampling using the mutual information criterion (*c.f.*, Chapter 6). If the parameters are unknown, we present several *exploration* strategies for efficiently decreasing the uncertainty about the model. Most approaches for active sampling of GPs have been *myopic* in nature, in each step selecting observations which, e.g., most decrease the predictive variance. Our approach however is *nonmyopic* in nature: we prove logarithmic sample complexity bounds on the duration of the exploration phase, and near optimal performance in the exploitation phase. Based on these insights, we develop EEGP, an efficient algorithm for exploration–exploitation based active learning in Gaussian Processes.

Often, e.g., in spatial interpolation [Rasmussen and Williams, 2006], GP models are assumed to be isotropic, where the covariance of two locations depends only on their distance, and some (unknown) parameters. As we have seen in Chapter 6, and has been found in previous work [Nott and Dunsmuir,

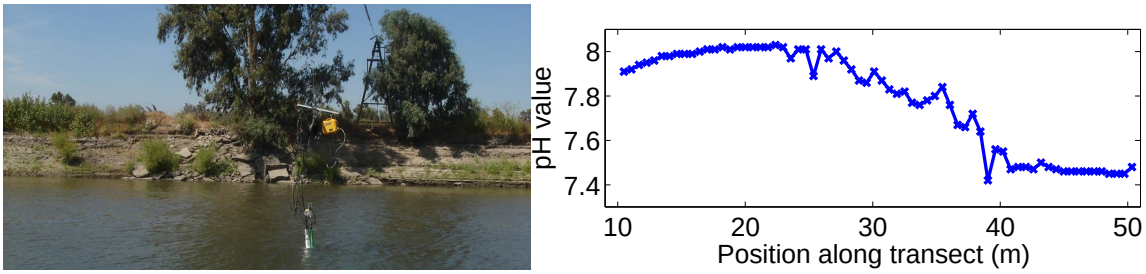


Figure 14.1: Left:Active sampling using the Networked Infomechanical System (NIMS) sensor [Harmon et al., 2006], deployed at the Merced River. The sensor is attached to a wire, which enables horizontal traversal of the transect. On fixed horizontal position, it can vertically lower or raise the sensing unit. Right: Samples of pH acquired along horizontal transect near the confluence of the San Joaquin and Merced rivers.

2002, Paciorek, 2003], many phenomena of interest however are nonstationary. In our river example (*c.f.*, Figure 14.1), the pH values are strongly correlated along the border, but weakly in the turbulent inner region. Our approach is applicable to both stationary and nonstationary processes. However, nonstationary processes are often defined by a much larger number of parameters. To address this issue, we extend our algorithm to handle nonstationary GPs with local structure, providing efficient exploration strategies and computational techniques that handle high dimensional parameter vectors. In summary, our contributions presented in this chapter are:

- A theoretical and empirical investigation of the performance difference between sequential and a priori strategies for sampling in GPs;
- An exploration–exploitation analysis and sample complexity bounds for sequential design;
- EEGP, an efficient, nonmyopic, sequential algorithm for observation selection in isotropic GPs;
- Extension of our method to nonstationary GPs;
- Empirical evaluation on several real-world spatial monitoring problems.

14.1 Gaussian Processes

Consider, for example, the task of monitoring the ecological state of a river using a robotic sensor, such as the one shown in Figure 14.1. We can model the pH values as a random process $\mathcal{X}_\mathcal{V}$ over the locations $\mathcal{V}$, e.g., $\mathcal{V} \subset \mathbb{R}^2$. Hereby, the pH value at every location $y \in \mathcal{V}$ is a random variable $\mathcal{X}_y$. Measurements $\mathbf{x}_\mathcal{A}$ at sensor locations $\mathcal{A} \subset \mathcal{V}$ then allow us to predict the pH value at uninstrumented locations y, by conditioning on the observations, i.e., predicting $\mathbb{E}[\mathcal{X}_y \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}]$.

It has been shown, that pH values, temperatures and many other spatial phenomena, can be effectively modeled using Gaussian processes (GPs) (*c.f.*, Cressie 1991, Shewry and Wynn 1987). Recall from Chapter 6, a GP (*c.f.*, Rasmussen and Williams 2006) is a random process $\mathcal{X}_\mathcal{V}$, such that every finite subset of variables $\mathcal{X}_\mathcal{A} \subseteq \mathcal{X}_\mathcal{V}$ has a (consistent) multivariate normal distribution:

$$P(\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \frac{1}{(2\pi)^{n/2}|\Sigma_{\mathcal{A}\mathcal{A}}|} e^{-\frac{1}{2}(\mathbf{x}_\mathcal{A}-\mu_\mathcal{A})^T \Sigma_{\mathcal{A}\mathcal{A}}^{-1}(\mathbf{x}_\mathcal{A}-\mu_\mathcal{A})},$$

where $\mu_{\mathcal{A}}$ is the mean vector and $\Sigma_{\mathcal{A}\mathcal{A}}$ is the covariance matrix. A GP is fully specified by a *mean function* $\mathcal{M}(\cdot)$, and a symmetric positive-definite *kernel function* $\mathcal{K}(\cdot,\cdot)$, often called the covariance function. For each random variable $\mathcal{X}_u$ with index $u \in \mathcal{V}$, its mean μ_u is given by $\mathcal{M}(u)$, and for each pair of indices $u, v \in \mathcal{V}$, their covariance σ_{uv} is given by $\mathcal{K}(u,v)$. For simplicity of notation, we denote the mean vector of a set of variables $\mathcal{X}_{\mathcal{A}}$ by $\mu_{\mathcal{A}}$, where the entry for element u of $\mu_{\mathcal{A}}$ is $\mathcal{M}(u)$. Similarly, we denote their covariance matrix by $\Sigma_{\mathcal{A}\mathcal{A}}$, where the entry for u, v is $\mathcal{K}(u,v)$. The GP representation allows us to efficiently compute predictive distributions, $P(\mathcal{X}_y \mid x_{\mathcal{A}})$, which, e.g., correspond to the predicted temperature at location y after observing sensor measurements $\mathcal{X}_{\mathcal{A}} = \mathbf{x}_{\mathcal{A}}$. The distribution of $\mathcal{X}_y$ given these observations is a Gaussian whose conditional mean $\mu_{y|\mathcal{A}}$ and variance $\sigma^2_{y|\mathcal{A}}$ are:

$$\mu_{y|\mathcal{A}} = \mu_y + \Sigma_{y\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}(\mathbf{x}_{\mathcal{A}} - \mu_{\mathcal{A}}), \tag{14.1}$$

$$\sigma^2_{y|\mathcal{A}} = \mathcal{K}(y,y) - \Sigma_{y\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}y}, \tag{14.2}$$

where $\Sigma_{y\mathcal{A}}$ is a covariance vector with one entry for each $u \in \mathcal{A}$ with value $\mathcal{K}(y,u)$, and $\Sigma_{\mathcal{A}y} = \Sigma^T_{y\mathcal{A}}$. An important property of GPs is that the posterior variance (14.2) does *not* depend on the observed values $\mathbf{x}_{\mathcal{A}}$.

In order to compute predictive distributions using (14.1) and (14.2), the mean and kernel functions have to be known. The mean function can usually be estimated using regression techniques. Estimating kernel functions is difficult, and usually, strongly limiting assumptions are made. For example, it is commonly assumed that the kernel $\mathcal{K}(u,v)$ is *stationary*, depending only on the difference between the locations, i.e., $\mathcal{K}(u,v) = \mathcal{K}_\theta(u-v)$, where θ is a set of parameters. Very often, the kernel is even assumed to be *isotropic*, which means that the covariance only depends on the distance between locations, i.e., $\mathcal{K}(u,v) = \mathcal{K}_\theta(||u-v||_2)$. A common choice for an isotropic kernel is the exponential kernel,

$$\mathcal{K}_\theta(\delta) = \exp(-\frac{|\delta|}{\theta}),$$

or the Gaussian (squared exponential) kernel,

$$\mathcal{K}_\theta(\delta) = \exp(-\frac{\delta^2}{\theta^2}).$$

Many other parametric forms are possible.

In Section 14.2.1, we address a general form (not necessarily isotropic), where the kernel function is specified by a set of parameters θ. We adopt a hierarchical Bayesian approach and assign a prior $P(\theta)$ to the parameters θ, which we assume to be discretized in our analysis. Hence, $P(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{A}}) = \sum_\theta P(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{A}}, \theta) P(\theta \mid \mathcal{X}_{\mathcal{A}})$. For clarity of presentation, we also assume that the prior mean function $\mathcal{M}(\cdot)$ is zero. This assumption can be relaxed, for example by assigning a normal prior to the mean function.

14.2 Observation selection policies

In Chapter 6, we have studied the problem of selecting a set $\mathcal{A}$ of sensing locations in order to maximize a sensing quality utility function $F(\mathcal{A})$. In this chapter, instead of selecting sets, we will concern ourselves with devising sequential policies π (as introduced in Chapter 2) for selecting observations. Similar as in the non-adaptive setting considered in Chapter 6, we will study both the entropy and the mutual information criteria.

14.2.1 The sequential entropy criterion

In order to select informative observations, the *entropy* criterion has been frequently used (*c.f.*, Gramacy 2005, Seo et al. 2000, Shewry and Wynn 1987). In the non-adaptive setting, as discussed in Chapter 6, this criterion selects observations $\mathcal{A}^* \subseteq \mathcal{V}$ with highest entropy,

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A}\subseteq\mathcal{V}} H(\mathcal{X}_\mathcal{A}), \tag{14.3}$$

where

$$H(\mathcal{X}_\mathcal{A}) = -\int p(\mathbf{x}_\mathcal{A}) \log p(\mathbf{x}_\mathcal{A}) d\mathbf{x}_\mathcal{A}$$

is the joint (differential) entropy of the random variables $\mathcal{X}_\mathcal{A}$. We call (14.3) an *a priori* design criterion, as it does not depend on the actual observed values, and can be optimized in advance. Maximizing (14.3) is **NP**-hard [Ko et al., 1995], so usually, a myopic (greedy) algorithm is used, as introduced in Chapter 6. Starting with the empty set, $\mathcal{A}^{(0)}$, at each step t it adds the location $y_i = \operatorname{argmax}_{y\in\mathcal{V}\setminus\mathcal{A}_{i-1}} H(\mathcal{X}_{y_i} \mid \mathcal{X}_{\mathcal{A}_{i-1}})$ to the set of already selected locations $\mathcal{A}_{i-1}$.

This a priori greedy rule is readily turned into a *sequential* algorithm, selecting

$$y_i = \operatorname*{argmax}_{y\in\mathcal{V}\setminus\mathcal{A}_{i-1}} H(\mathcal{X}_{y_i} \mid \mathcal{X}_{\mathcal{A}_{i-1}} = \mathbf{x}_{\mathcal{A}_{i-1}}).$$

In this sequential setting, the selected location y_i depends on the observations $\mathbf{x}_{\mathcal{A}_{i-1}}$. This sequential algorithm is an example of a *policy* π for selecting variables. In general, such policies do not need to be greedy.

For each instantiation of the process $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$, such a sequential policy π can select a *different* set of observations $\pi(\mathbf{x}_\mathcal{V}) \subseteq \mathcal{V}$. Hereby, the i-th element, π_i, deterministically depends on the observations made in the first $i-1$ steps, i.e., on $\mathbf{x}_{\pi_{1:i-1}}$. Hence, a policy can be considered a decision tree, where after each observation, we decide on the next observation to make. If we apply the greedy policy π_{GH} to our river example, $\pi_{GH,i}$ would select the location which has highest entropy for predicting pH, conditioned on the measurements we have made so far. We write $|\pi| = k$ to indicate that π selects sets $\mathcal{X}_\pi$ of k elements. In analogy to the definition of $H(\mathcal{X}_\mathcal{A})$, we can define the joint entropy of any sequential policy π as

$$H(\mathcal{X}_\pi) \equiv -\int p(\mathbf{x}_\mathcal{V}) \log p(\mathbf{x}_\pi) d\mathbf{x}_\mathcal{V},$$

whereby $\pi = \pi(\mathbf{x}_\mathcal{V})$ denotes the set of observations selected by the policy in the event $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$. $H(\mathcal{X}_\mathcal{A})$ is the entropy of a fixed set of variables $\mathcal{A}$. Since π will typically select different observations in different realizations $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$, $H(\mathcal{X}_\pi)$ will measure the "entropy" of different variables in each realization $\mathbf{x}_\mathcal{V}$.

14.2.2 The sequential mutual information criterion

Caselton and Zidek [1984] proposed the *mutual information* criterion for observation selection,

$$F_{MI}(\mathcal{X}_\mathcal{A}) = H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}}) - H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_\mathcal{A}).$$

In Chapter 6, we showed that in the non-adaptive (a priori) setting, this criterion selects locations which most effectively reduce the uncertainty at the unobserved locations, hence it often leads to better predictions compared to the entropy criterion.

A natural generalization of mutual information to the sequential setting is

$$\begin{aligned} F_{MI}(\mathcal{X}_\pi) &= H(\mathcal{X}_{\mathcal{V}\setminus\pi}) - H(\mathcal{X}_{\mathcal{V}\setminus\pi} \mid \mathcal{X}_\pi) \\ &= -\int p(\mathbf{x}_\mathcal{V})[\log p(\mathbf{x}_{\mathcal{V}\setminus\pi}) - \log p(\mathbf{x}_{\mathcal{V}\setminus\pi} \mid \mathbf{x}_\pi)] d\mathbf{x}_\mathcal{V}. \end{aligned}$$

Hereby, for each realization $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$, $\mathcal{V} \setminus \pi = \mathcal{V} \setminus \pi(\mathbf{x}_\mathcal{V})$ is the set of locations not picked by the policy π. The greedy policy π_{GMI} for mutual information, after some algebraic manipulation, is given by:

$$\pi_i = \operatorname{argmax}_y H(\mathcal{X}_y \mid \mathcal{X}_{\pi_{1:algp_i-1}} = \mathbf{x}_{\pi_{1:i-1}}) - H(\mathcal{X}_y \mid \mathcal{X}_{\bar{\pi}_{1:i-1}}), \tag{14.4}$$

where $\pi_i \equiv \pi_i(\mathbf{x}_{\pi_{1:i-1}})$, and $\bar{\pi} \equiv \mathcal{V} \setminus \{y, \pi(\mathbf{x}_\mathcal{V})\}$ is the set of "unsensed" locations if $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$, excluding y.

In Chapter 6, we showed that mutual information is submodular and approximately nondecreasing, allowing us to use algorithms for optimizing submodular functions in order to find near-optimal sets of observations $\mathcal{A}^*$. In the sequential setting however, these results cannot be readily applied, since, for example, Theorem 5.2 only applies to set functions and not to general sequential policies.

14.3 Bounds on the advantage of active learning strategies

A key question in active learning is to determine the potential of improvement of sequential strategies over a priori designs, e.g., how much greater $\max_{|\pi|=k} H(\mathcal{X}_\pi)$ is than $\max_{|\mathcal{A}|=k} H(\mathcal{X}_\mathcal{A})$. If the GP parameters θ are known, it holds that

$$H(\mathcal{X}_y|\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}, \theta) = \frac{1}{2}\log 2\pi e\sigma^2_{\mathcal{X}_y|\mathcal{X}_\mathcal{A}} = H(\mathcal{X}_y|\mathcal{X}_\mathcal{A}, \theta), \tag{14.5}$$

where $\sigma^2_{\mathcal{X}_y|\mathcal{X}_\mathcal{A}}$, as given by Equation (14.2). Thus, the entropy of a set of variables does not depend on the actual observed values $\mathbf{x}_\mathcal{A}$. Hence, perhaps surprisingly, in this case,

$$\max_{|\pi|=k} H(\mathcal{X}_\pi) = \max_{|\mathcal{A}|=k} H(\mathcal{X}_\mathcal{A}).$$

More generally, any objective function depending only on the predictive variances, cannot benefit from sequential strategies. Note that for non-Gaussian models, sequential strategies can strictly outperform a priori designs, even with known parameters.

With unknown parameters,

$$H(\mathcal{X}_\mathcal{A}) = -\sum_\theta \int P(\mathbf{x}_\mathcal{A}, \theta) \log \left(\sum_{\theta'} \int P(\mathbf{x}_\mathcal{A}, \theta') \right) d\mathbf{x}_\mathcal{A}$$

is the entropy of a mixture of GPs. Since observed values affect the posterior over the parameters $P(\Theta|\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$, the predictive distributions now depend on these values. Intuitively, if we have low uncertainty about our parameters, the predictive distributions should be *almost* independent of the observed values, and there should be *almost* no benefit from sequential strategies. We will now theoretically formalize this intuition.

The following central result achieves this goal, by bounding $H(\mathcal{X}_\pi)$ (and similarly for mutual information) of the optimal *policy* π by a mixture of entropies of *sets* $H(\mathcal{X}_{\mathcal{A}_\theta} \mid \theta)$, whereby the sets are $\mathcal{A}_\theta$ are chosen optimally for each fixed parameter θ (and can thus be selected a priori, without a sequential policy):

Theorem 14.1.

$$\max_{|\pi|=k} H(\mathcal{X}_\pi) \leq \sum_\theta P(\theta) \max_{|A|=k} H(\mathcal{X}_\mathcal{A} \mid \theta) + H(\Theta);$$

$$\max_{|\pi|=k} F_{MI}(\mathcal{X}_\pi) \leq \sum_\theta P(\theta) \max_{|A|=k} F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta) + H(\Theta).$$

The proofs of all theorems can be found in the Appendix.

Theorem 14.1 bounds the advantage of sequential designs by two components: The expected advantage by optimizing sets for known parameters, i.e.,

$$\sum_\theta P(\theta) \max_{|A|=k} F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta),$$

and the parameter entropy, $H(\Theta)$. This result implies, that if we are able to (approximately) find the best set of observations $\mathcal{A}_\theta$ for a GP with known parameters θ, we can bound the advantage of using a sequential design. If this advantage is small, we select the set of observations ahead of time, without having to wait for the measurements.

14.4 Exploration–exploitation approach towards learning GPs

Theorem 14.1 allows two conclusions: Firstly, if the parameter distribution $P(\Theta)$ is very peaked, we cannot expect active learning strategies to drastically outperform a priori designs. More importantly however, it motivates an exploration–exploitation approach towards active learning of GPs: If the bound provided by Theorem 14.1 is close to our current mutual information, we can exploit our current model, and optimize the sampling without having to wait for further measurements. If the bound is very loose, we explore, by making observations to improve the bound from Theorem 14.1. We can compute the bound while running the algorithm to decide when to stop exploring.

14.4.1 Exploitation using submodularity

Theorem 14.1 shows that in order to bound the value of the optimal policy, it suffices to bound the value of the optimal set. This insight allows us to apply the results we derived in Chapter 6 for mutual information, using the concept of *submodularity*.

Recall from Chapter 5 that a fundamental result about nondecreasing submodular functions F is the guarantee that the greedy algorithm, which greedily adds the element x to $\mathcal{A}$ such that $F(\mathcal{A} \cup \{x\}) - F(\mathcal{A})$ is largest, selects a set $\mathcal{A}_G$ of k elements which is at most a constant factor $(1 - 1/e)$ worse than the set of k elements of maximal value, i.e., $F(\mathcal{A}_G) \geq (1 - 1/e) \max_{|\mathcal{A}|=k} F(\mathcal{A})$ (*c.f.*, Theorem 5.2). Also recall from Chapter 6 that we have the following

Theorem 14.2. *Let $\mathcal{X}_\mathcal{V}$ be a Gaussian process. Under sufficiently fine discretization $\mathcal{V}$, the greedy algorithm for mutual information is guaranteed to select a set $\mathcal{A}_G$ of k sensors for which*

$$F_{MI}(\mathcal{X}_{\mathcal{A}_G}) \geq (1 - 1/e)(\mathrm{OPT} - k\varepsilon),$$

where OPT *is the mutual information achieved by the optimal placement, and ε depends polynomially on the discretization.*

Hence, we have the following result about exploitation using the mutual information criterion:

Corollary 14.3. *Choose the discretization of the GP such that Theorem 14.2 holds for all discrete values of* Θ*. Then*

$$F_{MI}(\mathcal{X}_{\mathcal{A}_G} \mid \Theta) \leq \max_{|\pi|=k} F_{MI}(\mathcal{X}_\pi) \leq (1-1/e)^{-1} \sum_\theta P(\theta) F_{MI}(\mathcal{X}_{\mathcal{A}_G}^{(\theta)} \mid \theta) + k\varepsilon + H(\Theta),$$

where $\mathcal{A}_G$ *is the greedy set for*

$$F_{MI}(\mathcal{X}_\mathcal{A} \mid \Theta) = \sum_\theta P(\theta) F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta),$$

and $\mathcal{A}_G^{(\theta)}$ *is the greedy set for* $F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta)$*.*

This result allows us to *efficiently compute* online bounds on how much can be gained by following a sequential active learning strategy. Intuitively, it states that if this bound is close to our current mutual information, we can stop exploring, and exploit our current knowledge about the model by near-optimally finding the best set of observations. We can also use Corollary 14.3 as a *stopping criterion*: We can use exploration techniques (as described in the next section) until the bound on the advantage of the sequential strategy drops below a specified threshold η, i.e., we stop if

$$\frac{(1-1/e)^{-1} \sum_\theta P(\theta) F_{MI}(\mathcal{X}_{\mathcal{A}_G}^{(\theta)} \mid \theta) + k\varepsilon + H(\Theta) - F_{MI}(\mathcal{X}_{\mathcal{A}_G} \mid \Theta)}{F_{MI}(\mathcal{X}_{\mathcal{A}_G} \mid \Theta)} \leq \eta. \tag{14.6}$$

In this case, we can use the greedy a priori design to achieve near-optimal mutual information, and obtain performance comparable to the optimal sequential policy. This a priori design is logistically simpler and easier to analyze. Hence, the stopping criterion interpretation of Corollary 14.3 has strong practical value, and we are not aware of any other approach for actively learning GPs which allow to compute such a stopping criterion. We use the acronym EEGP (for *Exploration–Exploitation for Actively learning GPs*, *c.f.*, Algorithm 14.1) to refer to the general class of algorithms that uses one of the exploration strategies that we will introduce in the next section, then stops exploring once the stopping criterion (14.6) signals the fact that sequential selection cannot exceed the a priori sensing quality by more than η, and then switches to the a priori greedy algorithm for selecting the remaining observations.

14.4.2 Implicit and explicit exploration

In order to practically use Corollary 14.3 as a stopping criterion for exploration, we have to, for each parameter θ, solve the optimization problem $\max_\mathcal{A} H(\mathcal{X}_\mathcal{A} \mid \theta)$. The following theorem shows, that if the parameter entropy is small enough, the contribution of the term $\sum_\theta P(\theta) \max_{|A|=k} F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta)$ to the bound diminishes quickly, and hence, we should concentrate solely on minimizing the parameter entropy $H(\Theta)$.

Theorem 14.4. *Let*

$$M = \max_A \max_{\theta_1,\theta_2} \frac{F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta_1)}{F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta_2)} < \infty, \quad K = \max_\theta \max_\mathcal{A} F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta),$$

and $H(\Theta) < 1$*. Then*

$$F_{MI}(\mathcal{X}_{\mathcal{A}^*} \mid \Theta) - H(\Theta) \leq F_{MI}(\mathcal{X}_{\pi^*}) \leq F_{MI}(\mathcal{X}_{\mathcal{A}^*} \mid \Theta) + CH(\Theta),$$

Algorithm 14.1: The EEGP algorithm for active learning in Gaussian Processes using Exploration–Exploitation analysis.

Algorithm EEGP (F_θ, $P(\theta)$, $\mathcal{V}$, k, exploration policy $\pi_{explore}$, η)
$i \leftarrow 0$;
$\mathcal{A} = \emptyset$;
while $i < k$ **do**
  $i \leftarrow i + 1$;
  Use $\pi_{explore}$ to choose next exploration location s;
  Obtain measurement x_s;
  $\mathcal{A} \leftarrow \mathcal{A} \cup \{s\}$;
  Compute posterior $P(\Theta \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$;
  if *inequality* (14.6) *holds for* η **then**
    $\mathcal{B} \leftarrow Greedy(F(\cdot \mid \Theta), \mathcal{V}, k - i)$;
    $i \leftarrow k$;
  end
end

where $\mathcal{A}^* = \operatorname{argmax}_\mathcal{A} F_{MI}(\mathcal{X}_\mathcal{A} \mid \Theta)$ *and* $\pi^* = \operatorname{argmax}_\pi F_{MI}(\mathcal{X}_\pi)$*, and* $C = \left(1 + \frac{MK}{\log_2 \frac{1}{H(\Theta)}}\right)$.

As a function of $H(\Theta)$, C converges to 1 very quickly as $H(\Theta)$ decreases. Theorem 14.4 hence provides the computational advantage, that, once the parameter entropy is small enough, we do not need to re-compute the term $\sum_\theta P(\theta) \max_{|A|=k} F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta)$ when using Theorem 14.1 as a criterion for stopping exploration. Hence, in the following, we concentrate on directly decreasing the parameter uncertainty. We describe three natural strategies for this goal. As we show in Section 14.6, none of these strategies dominates the other; whichever is more appropriate depends on the particular application.

Explicit Exploration via Independence Tests (ITE). In many cases, the unknown parameter of an isotropic GP is the bandwidth of the kernel, effectively scaling the kernel over space. Let $\theta_1 < \cdots < \theta_m$ be the possible bandwidths. In the exponential kernel, $\mathcal{K}_\theta(\delta) = \exp(-\frac{|\delta|}{\theta})$, or the Gaussian kernel, $\mathcal{K}_\theta(\delta) = \exp(-\frac{\delta^2}{\theta^2})$, the correlation between two variables at distance δ decreases exponentially with their distance δ. Hence, there is an *exponentially large gap* between the correlation for bandwidths θ_i and θ_{i+1}: There will be a distance $\hat{\delta}$, for which two random variables within this distance will appear dependent if the true bandwidth θ is at least $\theta \geq \theta_{i+1}$, and (roughly) independent if $\theta \leq \theta_i$. Our goal is to exploit this gap to efficiently determine the correct parameter.

First note that if we can separate θ_i from θ_{i+1}, we effectively distinguish any θ_j, for $j \leq i$, from θ_l, for $l \geq i+1$, since the bandwidths scale the kernels. Let I_i be a function of Θ, such that $(I_i \mid \Theta) = 0$ if $\Theta \leq \theta_i$, and $(I_i \mid \Theta) = 1$ if $\Theta \geq \theta_{i+1}$. Assume we have tests T_i, using $\hat{N}$ samples, such that $P(T_i \neq I_i \mid \theta) \leq \alpha$ for all θ. We can now use a *binary search* procedure to identify the true bandwidth with high probability using at most $\hat{N}\lceil \log_2 m \rceil$ samples. Let $\pi_{G \circ ITE}$ be the policy, where we first explore using ITE, and then greedily select the set $\mathcal{A}_G$ maximizing $F_{MI}(\mathcal{X}_{\mathcal{A}_G} \mid \Theta, \mathbf{x}_{\pi_{ITE}})$. Let $\mathbf{x}_{\pi_{ITE}}$ be the observations made by ITE, and let $\mathcal{A}_G^{(\theta)}$ be the solution of the greedy algorithm for optimizing $F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta)$.

Theorem 14.5. *Under the assumptions of Corollary 14.3 for sets of sizes up to* $k + \hat{N}\lceil \log m \rceil$*, if we have*

tests T_i using at most $\hat{N}$ samples, such that for all θ

$$P(T_i \neq I_i \mid \theta) \leq \alpha/(\lceil \log m \rceil^2 (\max_{\theta} |F_{MI}(\mathcal{X}_{\pi_{G \circ ITE}} \mid \Theta) - F_{MI}(\mathcal{X}_{\mathcal{A}_G^{(\theta)}} \mid \theta)|)),$$

then it holds that

$$\mathbb{E}_T[F_{MI}(\mathcal{X}_{\pi_{G \circ ITE}} \mid \Theta)] \geq (1 - 1/e) \max_{|\pi|=k} F_{MI}(\mathcal{X}_\pi) - k\varepsilon - \alpha.$$

In order to make use of Theorem 14.5, we need to find tests T_i such that $P(T_i \neq I_i \mid \theta)$ is sufficiently small for all θ. If only the bandwidth is unknown, we can for example use a test based on Pearson's correlation coefficient. Since this test requires independent samples, let us first assume, that the kernel function has bounded support (*c.f.*, Storkey [1999]), and that the domain of the GP is sufficiently large, such that we can get independent samples by sampling pairs of variables outside the support of the "widest" kernel. The number of samples will depend on the error probability α, and the difference $\hat{\rho}$ between the correlations depending on whether $\Theta \leq \theta_i$ or $\Theta \geq \theta_{i+1}$. This difference will in turn depend on the distance between the two samples. Let

$$\hat{\rho}_i = \max_{\delta} \min_{j \leq i, l \geq i+1} |\mathcal{K}_{\theta_j}(\delta) - \mathcal{K}_{\theta_l}(\delta)|, \text{ and}$$

$$\hat{\delta}_i = \operatorname*{argmax}_{\delta} \min_{j \leq i, l \geq i+1} |\mathcal{K}_{\theta_j}(\delta) - \mathcal{K}_{\theta_l}(\delta)|.$$

$\hat{\rho}_i$ is the maximum "gap" achievable for separating bandwidths at most θ_i from those at least θ_{i+1}. $\hat{\delta}_i$ is the distance at which two samples should be taken to achieve this gap in correlation. If several feasible pairs of locations are avaible, we choose the one which maximizes mutual information.

Theorem 14.6. *We need*

$$\hat{N}_i = \mathcal{O}\left(\frac{1}{\hat{\rho}_i^2} \log^2 \frac{1}{\alpha}\right)$$

independent pairs of samples at distance $\hat{\delta}_i$ to decide between $\theta \leq \theta_i$ or $\theta \geq \theta_{i+1}$ with $P(T_i \neq I_i \mid \theta) \leq \alpha$ for all θ.

In the case of kernels with non-compact support, such as the Gaussian or Exponential kernel, we cannot generate such independent samples, since distant points will have some (exponentially small) correlation. However, these almost independent samples suffice:

Corollary 14.7. *Let $\mathcal{X}$ have variance σ^2, measurement noise σ_n^2 at each location, $\hat{\rho} = \min_i \hat{\rho}_i$, and $\xi < \hat{\rho}$. We can obtain a test T_i with $P(T_i \neq I_i \mid \theta) \leq \alpha$ using*

$$\hat{N} = \mathcal{O}\left(\frac{1}{(\hat{\rho} - \xi)^2} \log^2 \frac{1}{\alpha}\right)$$

pairs of samples $\mathcal{X}_s = (\mathcal{X}_{s_1}, \mathcal{X}_{s_2})$ at distance $\hat{\delta}_i$, if, for every $\mathcal{X}_s$ and $\mathcal{X}_t$ in our sample set,

$$\mathrm{Cor}(\mathcal{X}_{s_i}, \mathcal{X}_{t_j}) \leq \sqrt{\frac{\xi \sigma_n^2}{4\sigma^2 \hat{N} \lceil \log_2 m \rceil}},$$

for $i, j \in \{1, 2\}$.

Hence, since most kernel functions decay exponentially fast, only a small spatial distance has to be guaranteed between the pairs of samples of the independence tests.

Note that for the Gaussian and the Exponential kernel for example, we can compute $\hat{\rho}_i$ analytically: For the Gaussian kernel we have

$$\hat{\delta}_i = \frac{\theta_{i+1}\theta_i\sqrt{(\theta_{i+1}^2 - \theta_i^2)\log(\theta_{i+1}^2/\theta_i^2)}}{\theta_{i+1}^2 - \theta_i^2}, \quad \hat{\rho}_i = \left(\exp(-\frac{\hat{\delta}^2}{\theta_{i+1}^2}) - \exp(-\frac{\hat{\delta}^2}{\theta_i^2})\right)(1 - \sigma_n^2/\sigma^2),$$

and for the Exponential kernel,

$$\hat{\delta}_i = \frac{\theta_{i+1}\theta_i\log(\theta_{i+1}/\theta_i)}{\theta_{i+1} - \theta_i}, \quad \hat{\rho}_i = \left(\exp(-\frac{\hat{\delta}}{\theta_{i+1}}) - \exp(-\frac{\hat{\delta}}{\theta_i})\right)(1 - \sigma_n^2/\sigma^2).$$

Further note that while this discussion focused on detecting bandwidths, the technique is general, and can be used to distinguish other parameters, e.g., variance, as well, as long as appropriate tests are available.

This hypothesis testing exploration strategy gives us sample complexity bounds. It guarantees that with a small number of samples we can decrease the parameter uncertainty enough such that, using Theorem 14.4 as stopping criterion, we can switch to exploitation.

Explicit Exploration based on Information Gain (IGE). As the bound in Theorem 14.4 directly depends on $H(\Theta)$, another natural exploration strategy is to select samples which have highest information gain about the *parameters*, $H(\Theta)$. More formally, this strategy, after observing samples $\mathcal{X}_{\pi_{1:algp_i}} = \mathbf{x}_{\pi_{1:algp_i}}$, selects the location π_{i+1} such that

$$\pi_{i+1} = \operatorname*{argmax}_y H(\Theta \mid \mathbf{x}_{\pi_{1:algp_i}}) - H(\Theta \mid \mathcal{X}_y, \mathbf{x}_{\pi_{1:algp_i}}).$$

Implicit Exploration (IE). The following generalization of the "information never hurts" principle [Cover and Thomas, 1991] to policies shows that *any* exploration strategy will, in expectation, decrease $H(\Theta)$.

Proposition 14.8. *Let $\mathcal{X}_\mathcal{V}$ be a GP with kernel parameters Θ. Let π be a policy for selecting observations. Then $H(\Theta \mid \mathcal{X}_\pi) \leq H(\Theta)$.*

Considering the near-optimal performance of the greedy heuristic in the a priori case, a natural implicit exploration strategy is the sequential greedy algorithm. Using Eq. (14.4), IE considers the previous observations, when deciding on the next observation, and, using Proposition 14.8, *implicitly* decreases $H(\Theta)$.

14.5 Actively learning nonstationary GPs

Many spatial phenomena are nonstationary, being strongly correlated in some areas of the space and very weakly correlated in others. In our river example, we consider the pH values in the region just below the

confluence of the San Joaquin and Merced rivers. The former was dominated by agricultural and wetland drainage, whereas, in contrast, the latter was less saline. The data (*c.f.*, Figure 14.2(a)) is very nonstationary. There is very high correlation and low variance in the outer regions. The turbulent confluence region however exhibits high variance and low correlation.

Modeling nonstationarity has to trade off richness of the model and computational and statistical tractability. Even though the covariance function is a an infinite dimensional object, often a parametric form is chosen. For example, Nott and Dunsmuir [2002] suggest to model nonstationarity by a spatially varying linear combination of isotropic processes. In any such a parametric setting, Corollary 14.3 holds without additional assumptions; the major difference is that $H(\Theta)$ can be much larger, increasing the potential for improvement of the active strategy over the a priori design.

14.5.1 Nonstationary model

Motivated by the river monitoring problem, we partition the space into disjoint regions $\mathcal{V}^{(1)}, \ldots, \mathcal{V}^{(m)}$, which are specified by the user. With each region $\mathcal{V}^{(i)}$, we associate an isotropic process $\mathcal{X}_{\mathcal{V}}^{(i)}$, with parameters $\Theta^{(i)}$, which are assumed to have independent priors. We define our GP prior for the full space $\mathcal{V}$ as a linear combination of the local GPs: $\mathcal{X}_s = \sum_i \lambda_i(s)\mathcal{X}_s^{(i)}$. Note that such a linear combination is still a valid GP. How should we choose the weights $\lambda_i(s)$? We want a model which behaves similar to process $\mathcal{X}_{\mathcal{V}}^{(i)}$ within region i, and interpolates smoothly between regions. In order to achieve that, we associate a weighting function $\nu_i(s)$ with each region. This function should achieve its maximum value in region i and decrease with distance to region i. In our river example, we set the weighting functions as indicated in Figure 14.2(a). We can then set

$$\lambda_i(s) = \sqrt{\frac{\nu_i(s)}{\sum_{i'} \nu_{i'}(s)}},$$

which ensures that the variance at location s is a convex combination of the variances of the local GPs, with contribution proportional to $\nu_i(s)$. If each $\mathcal{X}_{\mathcal{V}}^{(i)}$ has zero mean, and kernel $\mathcal{K}_i(s,t)$, then the new, nonstationary GP $\mathcal{X}_{\mathcal{V}}$ has the kernel

$$\sum_i \lambda_i(s)\lambda_i(t)\,\mathcal{K}_i(s,t).$$

By adding a deterministic function $\mathcal{M}(s)$, one can also modify the prior mean of the GP. While the decomposition into prespecified regions might appear restrictive, in many applications, as in the river monitoring setting, a good decomposition can be provided by an expert. Furthermore, one can control the amount of smoothing by a bandwidth parameter, which can be part of the model. By this approach, the data itself can decide whether two adjacent regions should be joined (high smoothing bandwidth) or almost independent (low smoothing bandwidth).

14.5.2 Efficient nonstationary active learning

Now, in principle we could apply Corollary 14.3 to this model to determine when to switch from exploration (e.g., using information gain) to exploitation. However, even if each $\Theta^{(i)}$ is discretized so that the distribution over $\Theta^{(i)}$ can be exactly maintained, the joint distribution over $\Theta = (\Theta^{(1)}, \ldots, \Theta^{(m)})$ is exponentially large in m. In order to address this problem, let us first consider the special case where each $\nu_{(i)}$ is positive only within region i. In this case, an observation made in region i only affects the prediction

and parameter estimation in region i. The joint distribution over Θ will always stay fully factorized, and efficient inference is possible. We effectively monitor a collection of independent GPs, and our active learning algorithm attempts to optimally allocate the samples to the independent GPs.

Now let us consider the general case, where the weights ν_i take positive values outside region i. In this case, an observation s made with positive weights $\nu_i(s) > 0$ and $\nu_j(s) > 0$ for two regions i and j effectively couples the parameters $\Theta^{(i)}$ and $\Theta^{(j)}$. Eventually, all parameters become dependent, and we need to maintain the full, exponentially large joint distribution. In order to cope with this complexity, we apply a variational approach: After making an observation, we find a fully factorized[1] approximate posterior distribution, which is closest in KL divergence. More formally, given a prior $P(\Theta)$ over the parameters and a set of locations $\mathcal{A} \subseteq \mathcal{V}$ and their values $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, we seek the distribution

$$\widehat{P}(\Theta) = \operatorname*{argmin}_{P' \text{ factorized}} KL(P(\Theta \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) \mid\mid P'(\Theta)).$$

For the multinomial distribution, the solution $\widehat{P}$ minimizing the KL divergence can be obtained by matching the marginals of the exact posterior [Koller and Friedman, 2008]. The following proposition shows that this procedure does not invalidate our stopping criterion.

Proposition 14.9. *It holds that*

$$H(\widehat{P}(\Theta)) \geq H(P(\Theta \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})).$$

Hence, using Theorem 14.4, our variational approach never stops exploring too early.

In order to use this nonstationary model for active learning, we need to condition on observations and compute mutual information efficiently.

Computing conditional distributions. We assume we have a fully factorized distribution $\widehat{P}(\Theta)$, which already incorporates previous observations $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, and we want to incorporate a new observation $\mathcal{X}_s = \mathbf{x}_s$ at location s. We first find the *relevant regions* $\mathcal{V}^{(i_1)}, \ldots, \mathcal{V}^{(i_m)}$. A region is relevant[2] to location s if $\nu_j(s) > 0$. For each joint instantiation of the relevant parameters $\bar{\theta} = (\theta_{i_1}, \ldots, \theta_{i_m})$, we compute the likelihood of the observation $P(\mathcal{X}_s = \mathbf{x}_s \mid \bar{\theta}, \mathbf{x}_{\mathcal{A}'})$, where $\mathbf{x}_{\mathcal{A}'}$ are the previous observations made within the *relevant* regions. Using Bayes' rule,

$$P(\bar{\theta} \mid \mathbf{x}_s, \mathbf{x}_\mathcal{A}) \propto \widehat{P}(\bar{\theta}) P(\mathbf{x}_s \mid \mathbf{x}_\mathcal{A}, \bar{\theta}),$$

we can compute the exact parameter posterior. Remembering all observed data, we can always compute $P(\bar{\theta} \mid \mathbf{x}_s, \mathbf{x}_\mathcal{A})$ using GP regression. Now that we have the exact parameter posterior, we find the KL-minimizing fully factorized approximation to $P(\bar{\theta} \mid \mathbf{x}_s, \mathbf{x}_\mathcal{A})$ by marginalisation.

Computing entropy and mutual information. In order to implement the greedy policy for mutual information π_{GMI} or entropy π_{GH}, we need to be able to compute $H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{A}, \theta)$ for the location s under consideration, and a set of observations $\mathcal{A}$ (or $\mathcal{V} \setminus (\mathcal{A} \cup \{s\})$ for mutual information). We can compute this quantity very similarly to the procedure described above. We first find the regions relevant to s, $\mathcal{V}^{(i_1)}, \ldots, \mathcal{V}^{(i_m)}$, and set $\mathcal{A}' = \mathcal{V}' \cap \mathcal{A}$, where $\mathcal{V}' = \mathcal{V}^{(i_1)} \cup \cdots \cup \mathcal{V}^{(i_m)}$. As above, for every joint

[1] More complex distributions, which still allow efficient inference, such as trees, can be used as well.

[2] We assume here that the ν_i are supported in a small number of regions. If this is not the case, we can use truncation arguments similar to those used in Chapter 6.

instantiation of the relevant parameters $\bar{\theta}$, we compute the conditional entropy on the GP $\mathcal{X}'_{\mathcal{V}}$, which we can do efficiently in closed form given the parameters $\bar{\theta}$. We can then compute

$$H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{A}} = \mathbf{x}_{\mathcal{A}}, \Theta) = \sum_{\bar{\theta}} \widehat{P}(\bar{\theta}) H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{A}} = \mathbf{x}_{\mathcal{A}}, \bar{\theta}).$$

In summary, our active learning strategy for nonstationary GPs is similar to the isotropic case: We explore until Corollary 14.3 proves that the advantage of the sequential strategy is small enough, then switch to exploitation. The difference is that we use a variational approach to leverage the structure of the nonstationary GP as a linear combination of locally supported isotropic GPs.

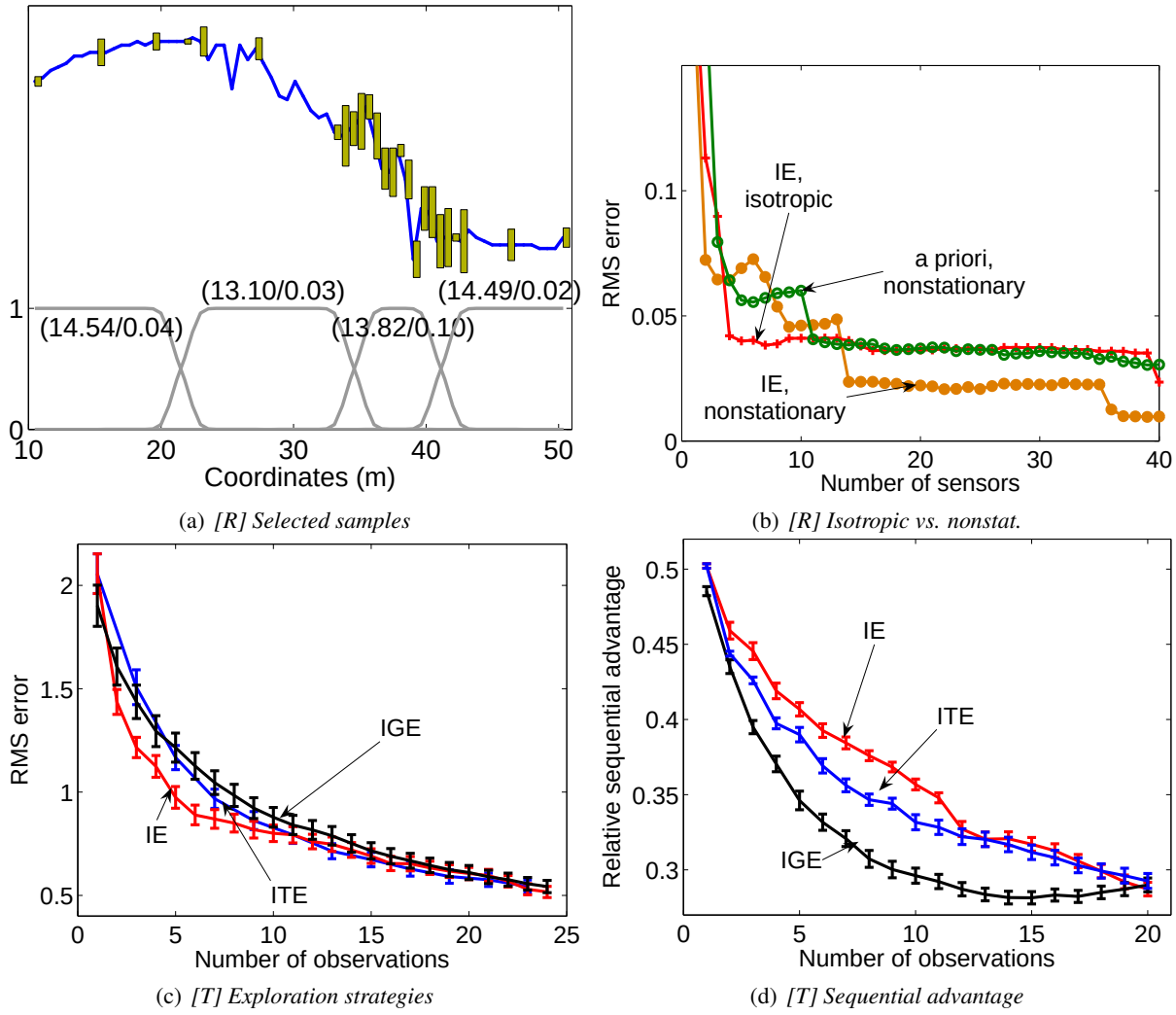
14.6 Experiments

14.6.1 River Monitoring

We first describe results on our river monitoring application. We consider one high-resolution spatial scan of pH measurements from the NIMS sensor deployed just below the confluence of the San Joaquin and the Merced rivers in California (denoted by [R]) [Harmon et al., 2006]. We partition the transect into four regions, with smoothing weights indicated in Figure 14.2(a), and we use 2 bandwidth and 5 noise variance levels. Figure 14.2(a) illustrates the samples chosen by implicit exploration (IE) using the entropy criterion. The bars indicate the sequence of observations, and larger bars correspond to later observations (i.e., based on more knowledge about the model). We can observe that while the initial samples are roughly uniformly distributed, the later samples are mostly chosen in the weakly correlated, high variance turbulent confluence region. In parentheses, we display the estimated bandwidths and noise standard deviations. Figure 14.2(b) presents the results from our algorithms. The sequential algorithm leads to a quicker decrease in Root Mean Squared (RMS) error than the a priori design. Initially, the isotropic model with two parameters provides a better fit than the nonstationary model with 8 parameters, but, after about 15 samples, the situation is inverted, and the nonstationary model drastically outperforms the isotropic model after 28 samples, providing more than 50% lower error.

14.6.2 Temperature Data

We consider temperature data [T] from the sensor network deployment with 54 sensors at Intel Research Berkeley, as introduced in Chapter 6. Our 145 samples consist of measurements taken every hour by the sensors over 5 days. We modeled the data as an isotropic process with unknown variance and an Exponential kernel with unknown bandwidth. We discretized the variance in $\sigma^2 \in \{1^2, 2^2, 3^2, 4^2, 5^2\}$, and the bandwidth in $\{3, 5, 7, 9, 11, 13, 15\}$ meters based on expert knowledge. We compared the performance of the active learning strategies, each using a different exploration strategy. Figure 14.2(c) shows the RMS prediction error, and Figure 14.2(d) presents the potential relative advantage obtained by Theorem 14.3 (our stopping criterion). While IE leads to the best prediction, followed by the independence test exploration (ITE), information gain exploration (IGE) tightens the bound on the sequential advantage the fastest. For example., if we decide to stop exploring once the sequential advantage drops below $\eta = 35\%$, 5 samples suffice for



(a) *[R] Selected samples*

(b) *[R] Isotropic vs. nonstat.*

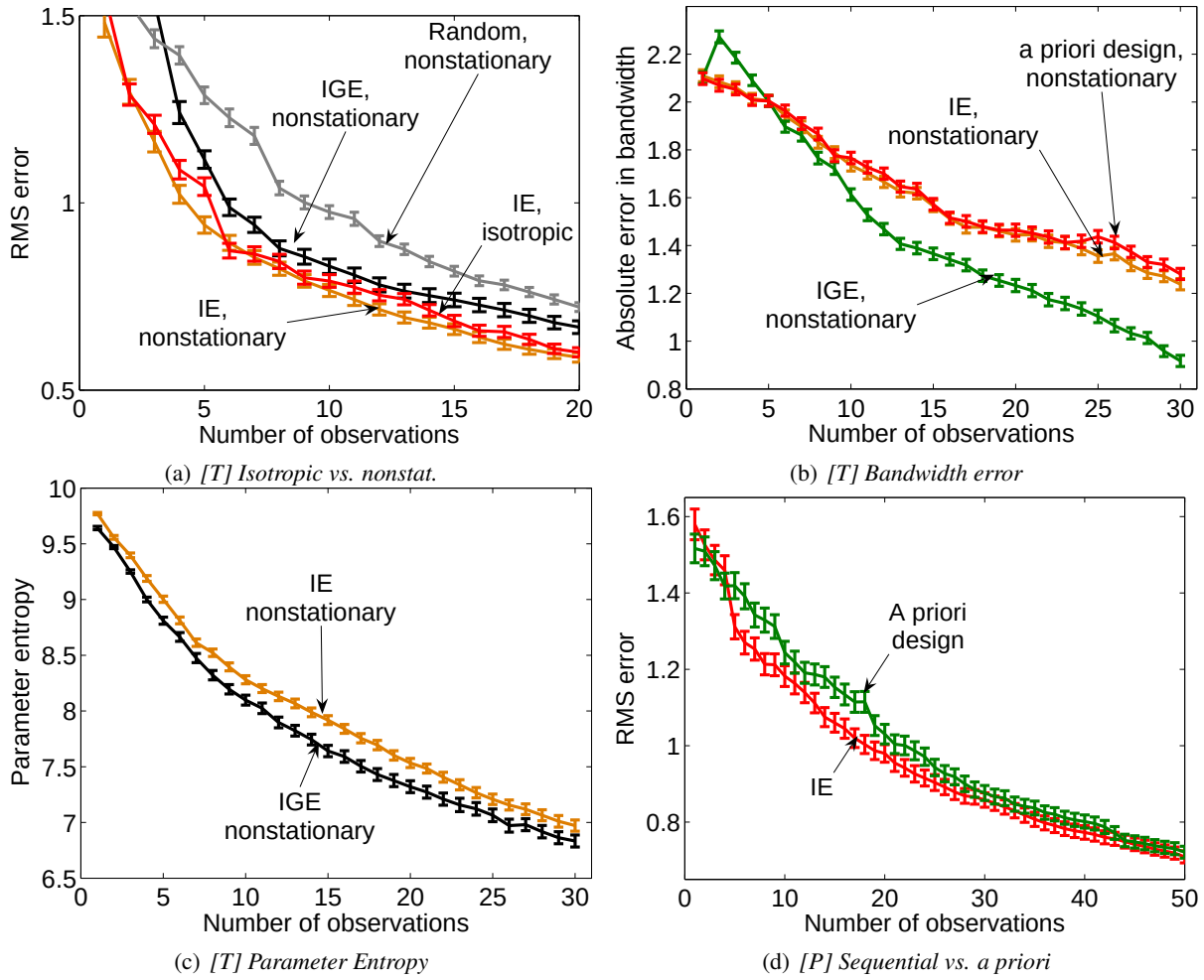
(c) *[T] Exploration strategies*

(d) *[T] Sequential advantage*

Figure 14.2: Results on pH [R] and temperature [T] data. (a) Top: sampling locations chosen by active learning algorithm. Higher bars indicate later (i.e., more informed) choice. Bottom: Smoothing functions used for spatial partitioning. (b) Comparison of prediction error for pH data. Note that the sequential algorithm on the nonstationary model eventually reduces the error incurred by the a priori design and isotropic model by more than 50%. (c) Comparison of exploration strategies, isotropic model. (d) Bounds on the potential advantage of the sequential algorithm using Theorem 14.3 (Stopping criterion). Information gain leads to quickest drop of bound, but worse spatial prediction.

IGE, 8 for ITE and 12 for IE. This analysis (which is also supported by other data sets) indicates that none of the exploration strategies dominates each other, their differences can be well-characterized, and the choice of strategy depends on the needs of each application. Hence, if the goal is to switch to a priori design as quickly as possible, IGE might be the right choice, whereas if we can afford to always perform the logistically more complex sequential design, IE would decrease the predictive RMS error the fastest. ITE performs well w.r.t. both criteria, and has theoretical sample complexity guarantees.

We also modeled the temperature using a nonstationary GP, with the space partitioned into four regions, each modeled as an isotropic GP. We adopted a softmax function with smoothing bandwidth 8 meters to



(a) *[T] Isotropic vs. nonstat.*

(b) *[T] Bandwidth error*

(c) *[T] Parameter Entropy*

(d) *[P] Sequential vs. a priori*

Figure 14.3: Results on temperature [T] and precipitation [P] data. (a) Comparison of isotropic, nonstationary model, using random and sequential selection. Information gain achieves worst prediction, but reduces error in bandwidth (b) and parameter entropy (c) fastest. (d) Sequential design outperforms a priori design on rain data.

spatially average over the local isotropic GPs. The results in Figure 14.3(a) show that the nonstationary model leads to reduced prediction error compared to the isotropic model. All active learning models drastically outperform random selection. Since the parameter uncertainty is still very high after 20 samples, IGE leads to worse prediction accuracy than IE. However, IGE decreases the parameter error Figure 14.3(b) (compared to the estimates when given all observations) and parameter entropy $H(\Theta)$ Figure 14.3(c) the fastest. These results indicate (along with higher log-likelihood), that even though we are estimating its 8 parameters from only up to 20 data points, the nonstationary model provides a better fit to the data.

14.6.3 Precipitation Data

In another experiment, we considered precipitation data [P] from 167 detector stations in the Pacific Northwest. We followed the preprocessing as described in Chapter 6. Figure 14.3(d) shows the RMS error for

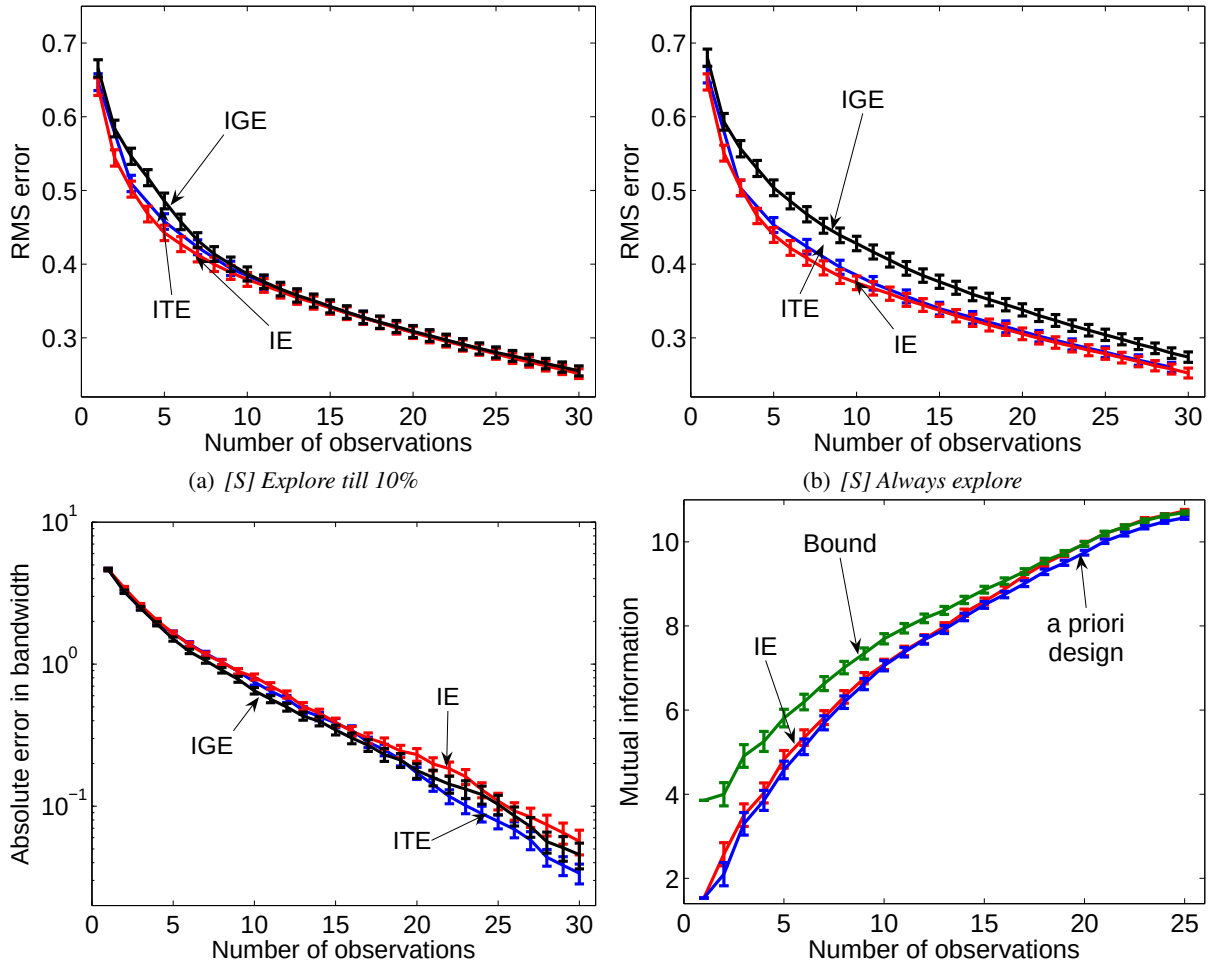
110 samples, spaced roughly three months apart, using an isotropic GP with 5 bandwidth and 3 variance parameter levels. Here, IE, ITE, IGE all outperform the a priori design.

14.6.4 Synthetic data

In order to study our method in more detail, we created a synthetic data set [S] out of 1000 samples from an isotropic GP with unit variance, on a 8×8 grid. We uniformly selected bandwidths for the Exponential kernel from $\{1, 2, 4, 8, 16\}$. For each sample, we ran our exploration-exploitation algorithm with all three exploration strategies (ITE, IGE and IE), as well as the a priori design based on $F_{MI}(\cdot \mid \Theta)$. Figure 14.4(a) shows the average Root Mean Squares (RMS) prediction error with increasing number of observations. Hereby, the exploration stopped after, based on Corollary 14.3, the sequential design could only perform at most 10% better than the a priori design. Information gain exploration (IGE) decreases the RMS error slightly more slowly. In this experiment, the a priori design achieved prediction accuracy only insignificantly worse than the sequential designs. When we never stop exploring (*c.f.*, Figure 14.4(b)), then IGE performs significantly worse. The independence testing (ITE) performs almost as well as the implicit exploration (IE), since it prefers tests with high mutual information. Figure 14.4(c) shows the error in estimating the bandwidth parameter with an increasing number of samples. All three strategies decrease the parameter error exponentially, empirically demonstrating the logarithmic bound. Initially, IGE decreases the parameter error significantly more quickly. Both explicit exploration strategies, IGE and ITE, lead to a lower bandwidth error after all 30 samples are observed. Figure 14.4(d) compares the mutual information achieved by the a priori and active strategies, along with the bound from Corollary 14.3 on the best sequential strategy (without the factor $(1 - 1/e)^{-1}$, since the greedy sets tend to be very close to optimal in practice, *c.f.*, Chapter 6). Experimentally we observed that the contribution by the term $\sum_\theta P(\theta) \max_{|A|=k} F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta)$ to the stopping criterion was negligible compared to the parameter entropy $H(\Theta)$.

14.7 Summary

In this chapter, we presented a nonmyopic analysis for active learning of Gaussian Processes. We proved bounds on how much better a sequential algorithm can perform than an a priori design when optimizing observation locations under unknown parameters. Our bounds show that key potential for improvement is in the parameter entropy, motivating an exploration–exploitation approach to active learning, and provide insight into when to switch between the two phases. Using submodularity of our objective function, we provided bounds on the quality of our exploitation strategy. We proposed several natural exploration strategies for decreasing parameter uncertainty, and proved logarithmic sample complexity results for exploration phase using hypothesis testing. We extended our algorithm to handle nonstationary GP, exploiting local structure in the model. Here, we used a variational approach to address the combinatorial growth of the parameter space. In addition to our theoretical analyses, we evaluated our algorithms on several real-world problems, including data from a real deployment for monitoring the ecological condition of a river. We believe that our results provide significant new insights on the potential of sequential active learning strategies for monitoring spatial phenomena using GPs.



(a) *[S] Explore till 10%*

(b) *[S] Always explore*

(c) *[S] Bandwidth error*

(d) *[S] Mutual information*

Figure 14.4: Results on synthetic data [S]. (a) If we stop exploring when the potential advantage goes below 10%, all 3 exploration strategies achieve approximately the same prediction. (b) If we never stop exploring, IGE leads to worse prediction than ITE. (c) Both IGE and ITE decrease parameter error more quickly than IE. (d) bound on optimal sequential performance quickly becomes tight, indicating that one can stop exploring early.

Chapter 15

Optimal Value of Information

In probabilistic reasoning, where one can choose among several possible but expensive observations, it is often a central issue to plan on which variables to observe in order to be able to make the most effective decisions [Howard and Matheson, 1984, Mookerjee and Mannino, 1997]. In a medical expert system, for example, multiple tests are available, and each test has a different cost [Heckerman et al., 1993, Turney, 1995]. In such systems, it is thus important to decide on a sequence in which tests are performed in order to become most certain about the patient's condition, at a minimum cost. Occasionally, the cost of testing can even exceed the value of information for any possible outcome. In the medical example, the information is acquired not in order to reduce uncertainty per se, but rather to facilitate decision making. Observing the outcome of medical tests only provides value if it is conclusive enough to allow making an informed decision.

In the previous chapters we have seen that many important sensing tasks require the optimization of a submodular set function. Unfortunately, the decision theoretic value of information (as introduced in Chapter 2) does *not* satisfy submodularity.

Proposition 15.1. *Decision-theoretic value of information is not submodular, even in Naive Bayes models.*

Intuitively, value of information can be non-submodular, if we need to make several observations (e.g., medical tests) in order to "convince" ourselves that we need to change our decision.

Even though decision-theoretic value of information is not submodular, one might hope that the greedy algorithm still performs well. In fact, such greedy algorithms have been used by many researchers to select observations in such sequential problems [Dittmer and Jensen, 1997, Kapoor et al., 2007, Scheffer et al., 2001, van der Gaag and Wessels, 1993]. Unfortunately, in general, this heuristic does not provide any performance guarantees. In fact, we can show that this greedy algorithm can perform arbitrarily badly:

Proposition 15.2. *The greedy algorithm for optimizing decision-theoretic value of information can perform arbitrarily badly.*

In this chapter, we will consider the most general problem of *optimally* selecting a priori subsets or sequential policies for objective functions that are not required to be submodular, such as the decision-theoretic value of information.

The following running example motivates our research and is empirically evaluated in Section 15.5. Consider a sequential temperature monitoring task, where wireless temperature sensors are distributed across

a building. The task is to become most certain about the temperature distribution at all times, whilst minimizing energy expenditure, a critically constrained resource [Deshpande et al., 2004]. Such fine-grained building monitoring is required to obtain significant energy savings [Singhvi et al., 2005]. In this example, our value of information goal would be to adaptively determine at which timesteps the sensor should be activated to minimize the uncertainty, or to facilitate building management decisions.

We present efficient algorithms, which guarantee optimal *nonmyopic* value of information in chain graphical models such as Hidden Markov Models (HMMs) [Baum and Petrie, 1966]. We address two settings: *subset selection*, where the optimal subset of observations is obtained in an open-loop fashion, and *conditional plans*, a sequential, closed-loop plan where the observation strategy depends on the actual value of the observed variables (*c.f.*, Figure 15.1). To our knowledge, these are the first optimal and efficient algorithms for these tasks for this class of graphical models. For both settings, we address the *filtering* and the *smoothing* versions: Filtering is important in online decision making, where our decisions can only utilize observations made in the past. Smoothing arises for example in structured classification tasks, where there is no temporal dimension in the data, and hence all observations can be taken into account. We call our approach VOIDP as the algorithms use Dynamic Programming to optimize Value of Information. We evaluate our VOIDP algorithms empirically on three real-world datasets, and also show that they are well-suited for interactive classification of sequential data.

Most problems in graphical models, such as probabilistic inference and finding the most probable explanation, that can be solved efficiently for chain-structured graphs, can also be solved efficiently for polytrees. We prove that the problem of selecting the best k observations for maximizing decision theoretic value of information is $\mathbf{NP}^{\mathbf{PP}}$-complete even for discrete polytree graphical models, giving a complexity theoretic classification of a core artificial intelligence problem. $\mathbf{NP}^{\mathbf{PP}}$-complete problems are believed to be significantly harder than $\mathbf{NP}$-complete or even $\#\mathbf{P}$-complete problems commonly arising in the context of graphical models. We furthermore prove that just evaluating decision-theoretic value of information objective functions is $\#\mathbf{P}$-complete even in the case of Naive Bayes models, a simple special case of polytree graphical models that is frequently used in practice [*c.f.*, Domingos and Pazzani, 1997].

Unfortunately, these hardness results show that, while the problem of optimally scheduling a single sensor can be optimally solved using our algorithms, the problem of scheduling multiple, correlated sensors is wildly intractable. Nevertheless, we show how our algorithms for single sensor scheduling can be used to approximately optimize a multi-sensor schedule. We demonstrate the effectiveness of this approach on a real sensor network testbed for building management.

In summary, we provide the following contributions:

- We present the first optimal algorithms for nonmyopically computing and optimizing value of information on chain graphical models.
- We show that optimizing decision theoretic value of information is $\mathbf{NP}^{\mathbf{PP}}$-hard for discrete polytree graphical models. Just computing decision theoretic value of information is $\#\mathbf{P}$-hard even for Naive Bayes models.
- We present several extensions of our algorithms, e.g., to tree graphical models with few leaves, and to multiple correlated chains (for multi-sensor scheduling).
- We extensively evaluate our algorithms on several real-world problems, including sensor scheduling on a real sensor testbed and active labeling in bioinformatics and Natural Language Processing.

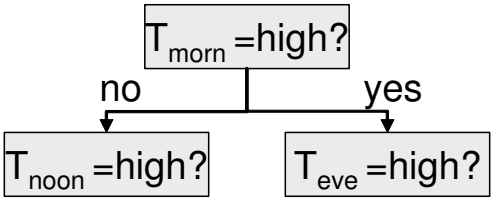


Figure 15.1: Example of a conditional plan.

15.1 Problem statement

As in Chapter 2, we will assume that the state of the world is described by a collection of random variables $\mathcal{X}_\mathcal{V} = (\mathcal{X}_1, \ldots, \mathcal{X}_n)$, where $\mathcal{V}$ is an index set. For example, $\mathcal{V}$ could denote a set of locations, and $\mathcal{X}_i$ models the temperature reading of a sensor placed at location $i \in \mathcal{V}$. For a subset $\mathcal{A} = \{i_1, \ldots, i_k\} \subseteq \mathcal{V}$, we use the notation $\mathcal{X}_\mathcal{A}$ to refer to the random vector $\mathcal{X}_\mathcal{A} = (\mathcal{X}_{i_1}, \ldots, \mathcal{X}_{i_k})$. While some of our algorithms extend to continuous distributions, we we generally assume that the variables $\mathcal{X}_\mathcal{V}$ are discrete.

As discussed in Chapter 2, we take a Bayesian model-based approach, and assume a prior probability distribution $P(\mathcal{X}_\mathcal{V})$ over the outcomes of the variables. Suppose we select a subset of the variables, $\mathcal{X}_\mathcal{A}$ (for $\mathcal{A} \subseteq \mathcal{V}$), and observe $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$. For example, $\mathcal{A}$ is the set of locations where we place sensors, or a set of medical tests we decide to perform. After observing the realization of these variables $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, we can compute the posterior distribution over all variables $P(\mathcal{X}_\mathcal{V} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$. Based on this posterior probability we obtain a reward

$$u(\mathbf{x}_\mathcal{V}, \mathcal{A}) = R(P(\mathcal{X}_\mathcal{V} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})).$$

For example, this reward function could depend on the uncertainty (measured by the entropy) of the distribution $P(\mathcal{X}_\mathcal{V} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$. We will describe several examples in more detail below.

In general, when selecting observation, we will not know ahead of time what observations we will make. Instead, we only have a distribution over the possible observations. Hence, we will be interested in the *expected reward*, where we take the expectation over the possible observations.

When optimizing the selection of variables, we can consider different settings: In *subset selection*, our goal is to pick a subset $\mathcal{A}^* \subseteq \mathcal{V}$ of the variables, maximizing

$$\mathcal{A}^* = \operatorname*{argmax}_\mathcal{A} \sum_{\mathbf{x}_\mathcal{A}} P(\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) R(P(\mathcal{X}_\mathcal{V} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})), \tag{15.1}$$

where we possible impose some constraints on the set $\mathcal{A}$ we are allowed to pick. In the subset selection setting, we commit to the selection of the variables before we get to see their realization.

Instead, we can also *sequentially* select one variable after the other, letting our choice depend on the observations made in the past. In this setting, we would like to find a *conditional plan* π^* that maximizes

$$\pi^* = \operatorname*{argmax}_\pi \sum_{\mathbf{x}_\mathcal{V}} P(\mathbf{x}_\mathcal{V}) R(P(\mathcal{X}_\mathcal{V} \mid \mathcal{X}_{\pi(\mathbf{x}_\mathcal{V})} = \mathbf{x}_{\pi(\mathbf{x}_\mathcal{V})})). \tag{15.2}$$

Hereby, π is a conditional plan (such as the ones considered in Chapter 14), that can select a different set of variables for each possible state of the world $\mathbf{x}_\mathcal{V}$. We use the notation $\pi(\mathbf{x}_\mathcal{V}) \subseteq \mathcal{V}$ to refer to the subset

of variables selected by the conditional plan π in state $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$. We will define the notion of conditional planning more formally in Section 15.3.2.

This general setup of selecting observations goes back in the decision analysis literature to the notion of value of information by Howard [1966] and in the statistical literature to the notion of Bayesian Experimental Design by Lindley [1956]. In this chapter, we refer to the problems (15.1) and (15.2) as the problem of *optimizing value of information*.

In this chapter, we show how the complexity of solving these value of information problems depend on the properties of the probability distribution P. We give the first algorithms for optimally solving value of information for an interesting and challenging class of distributions including Hidden Markov Models. We also present hardness results showing that optimizing value of information is wildly intractable ($\mathbf{NP}^{\mathbf{PP}}$-complete) even for probability distributions for which efficient inference is possible (even for Naive Bayes models and discrete polytrees).

15.1.1 Optimization criteria

In this chapter, we will only make the assumption, that the reward functions R_i is *local* [1], i.e., they are defined on the marginal probability distributions of the variables $\mathcal{X}_i$. This class has the computational advantage that local rewards can be evaluated using probabilistic inference techniques. The total reward will then be the sum of all local rewards. We do not assume that R_i is submodular.

Let $\mathcal{A}$ be a subset of $\mathcal{V}$. Then $P(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$ denotes the marginal distribution of variable $\mathcal{X}_j$ conditioned on observations $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$. For example, in our temperature monitoring application, $\mathcal{X}_j$ models the temperature at location $j \in \mathcal{V}$. The conditional marginal distribution $P(\mathcal{X}_j = x_j \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$ then models the conditional distribution of the temperature at location j after observing the temperature at locations $\mathcal{A} \subseteq \mathcal{V}$.

For classification purposes, it can be more appropriate to consider the max-marginals

$$P^{max}(\mathcal{X}_j = \mathbf{x}_j \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \max_{\mathbf{x}_\mathcal{V}} P(\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}, \mathcal{X}_j = x_j \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}),$$

that is, for $\mathcal{X}_j$ set to value x_j, the probability of the most probable assignment $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$ to all other random variables conditioned on the observations $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$.

The *local reward* R_j is a functional on the probability distribution P or P^{max} over $\mathcal{X}_j$. I.e., R_j takes an entire distribution over the variable $\mathcal{X}_j$ and maps it to a reward value. Typically, the reward functions will be chosen such that "certain" or "peaked" distributions obtain higher reward.

To simplify notation, we write

$$R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) \triangleq R_j(P(X_j \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}))$$

to denote the reward for variable $\mathcal{X}_j$ upon observing $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, and

$$R_j(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A}) \triangleq \sum_{\mathbf{x}_\mathcal{A}} P(\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A})$$

to refer to *expected local rewards*, where the expectation is taken over all assignments $\mathbf{x}_\mathcal{A}$ to the observations $\mathcal{A}$. Important local reward functions include:

[1] Local reward functions are also sometimes called decomposable scoring functions, which are widely used, e.g., in structure learning of graphical models [*c.f.*, Koller and Friedman, 2008].

Residual entropy If we set

$$R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) = -H(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) = \sum_{x_j} P(x_j, \mathbf{x}_\mathcal{A}) \log_2 P(x_j \mid \mathbf{x}_\mathcal{A}),$$

the objective in the optimization problem becomes to minimize the sum of residual entropies. Optimizing this reward function attempts to reduce the uncertainty in predicting the marginals $\mathcal{X}_i$. We choose this reward function in our running example to measure the uncertainty about the temperature distribution.

Joint entropy. Instead of minimizing the sum of residual entropies $\sum_i H(\mathcal{X}_i)$, we can also attempt to minimize the joint entropy of the entire distribution,

$$H(\mathcal{X}_\mathcal{V}) = -\sum_{\mathbf{x}_\mathcal{V}} P(\mathbf{x}_\mathcal{V}) \log_2 P(\mathbf{x}_\mathcal{V}).$$

Note, that the joint entropy depends on the full probability distribution $P(\mathcal{X}_\mathcal{V})$, rather than on the marginals $P(\mathcal{X}_i)$, and hence it is not local. Nevertheless, we can exploit the chain rule for the joint entropy $H(\mathcal{X}_\mathcal{B})$ of a set of random variables $\mathcal{B} = \{1, \ldots, m\}$ [*c.f.*, Cover and Thomas, 1991], i.e.,

$$H(\mathcal{X}_\mathcal{B}) = H(\mathcal{X}_1) + H(\mathcal{X}_2 \mid \mathcal{X}_1) + H(\mathcal{X}_3 \mid \mathcal{X}_1, \mathcal{X}_2) + \cdots + H(\mathcal{X}_m \mid \mathcal{X}_1, \ldots, \mathcal{X}_{m-1}).$$

Hence, if we choose the local reward functions $R_j(cX_j \mid \mathcal{X}_\mathcal{A}) = -H(\mathcal{X}_j \mid \mathcal{X}_1, \ldots, \mathcal{X}_{j-1}, \mathcal{X}_\mathcal{A})$, we can optimize a non-local reward function (the joint entropy) using only local reward functions.

Decision-theoretic value of information. The concept of local reward functions also includes the concept of decision theoretic value of information. The notion of value of information is widely used [*c.f.*, Heckerman et al., 1993, Howard, 1966, Lindley, 1956], and is formalized, e.g., as utility nodes in influence diagrams [Howard and Matheson, 1984], or reward functions in Partially Observable Markov Decision Processes (POMDPs, Smallwood and Sondik [1973]). For each variable $\mathcal{X}_j$, let $\mathcal{D}_j$ be a finite set of decisions. Also, let $d_j : \mathcal{D}_j \times \operatorname{dom} \mathcal{X}_j \to \mathbb{R}$ be a utility function mapping a decision $d \in \mathcal{D}_j$ and an outcome $x \in \operatorname{dom} \mathcal{X}_j$ to a real number. The *maximum expected utility principle* states that actions should be selected as to maximize the expected utility, $EU_j(d \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \sum_{x_j} P(x_j \mid \mathbf{x}_\mathcal{A}) g_j(d, x_j)$. The more certain we are about $\mathcal{X}_j$, the more economically we can choose our action. This idea is captured by the notion of value of information, where we choose our local reward function

$$R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) = \max_d EU_j(d \mid \mathbf{x}_\mathcal{A}).$$

Margin for structured prediction. We can also consider the margin of confidence:

$$R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) = P^{max}(x_j^* \mid \mathbf{x}_\mathcal{A}) - P^{max}(\bar{x}_j \mid \mathbf{x}_\mathcal{A}),$$

where

$$x_j^* = \operatorname*{argmax}_{x_j} P^{max}(x_j \mid \mathbf{x}_\mathcal{A}) \text{ and } \bar{x}_j = \operatorname*{argmax}_{x_j \neq x_j^*} P^{max}(x_j \mid \mathbf{x}_\mathcal{A}),$$

which describes the margin between the most likely outcome and the closest runner up. This reward function is very useful for structured classification purposes, as shown in Section 15.5.

Weighted mean-squared error If the variables are continuous, we might want to minimize the mean squared error in our prediction. We can do this by choosing

$$R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) = -w_j \operatorname{Var}(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}),$$

where

$$\operatorname{Var}(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) = \int P(x_j \mid \mathbf{x}_\mathcal{A}) \left[x_j - \int x'_j P(x'_j \mid \mathbf{x}_\mathcal{A}) dx'_j \right]^2 dx_j$$

is the conditional variance of $\mathcal{X}_j$ given $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$, and w_j is a weight indicating the importance of variable $\mathcal{X}_j$.

Monitoring for critical regions (Hotspot sampling) Suppose we want to use sensors for detecting fire. More generally, we want to detect, for each i, whether $\mathcal{X}_i \in \mathfrak{C}_i$, where $\mathfrak{C}_j \subseteq \operatorname{dom} X_j$ is a "critical region" for variable $\mathcal{X}_j$. Then the local reward function

$$R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) = P(\mathcal{X}_j \in \mathfrak{C}_j \mid \mathbf{x}_\mathcal{A})$$

favors observations $\mathcal{A}$ that maximize the probability of detecting critical regions.

Function optimization (Correlated bandits) Consider a setting where we have a collection of random variables $\mathcal{X}_\mathcal{V}$ taking numerical values in some interval $[-m, m]$, and, after selecting some of the variables, we get the reward $\sum_i x_i$. This setting arises if we want to optimize an unknown (random) function, where evaluating the function is expensive. In this setting, we are encouraged to only evaluate the function where it is likely to obtain high values. We can maximize our expected total reward if we choose the local reward function

$$R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) = \int x_j P(x_j \mid \mathbf{x}_\mathcal{A}) dx_j,$$

i.e., the expectation of variable $\mathcal{X}_j$ given observations $\mathbf{x}_\mathcal{A}$. This setting of optimizing a random function can also be considered a version of the classical k-armed bandit problem with correlated arms. More details about the relationship with bandit problems are given in Section 15.7.

These examples demonstrate the generality of our notion of local reward. Note that most examples apply to continuous distributions just as well as for discrete distributions and vice versa.

15.1.2 Cost of selecting observations

We also want to capture the constraint that observations are expensive. This can mean that each observation $\mathcal{X}_j$ has an associated positive *penalty* C_j that effectively decreases the reward. In our example, we might be interested in trading off accuracy with sensing energy expenditure. Alternatively, it is also possible to define a *budget* B for selecting observations, where each one is associated with an integer *cost* β_j. Here, we want to select observations whose sum cost is within the budget, but these costs do not decrease the reward. In our running example, the sensors could be powered by solar power, and regain a certain amount of energy per day, which allows a certain amount of sensing. Our formulation of the optimization problems allows both for penalties and budgets. To simplify notation we also write $C(\mathcal{A}) = \sum_{j \in \mathcal{A}} C_j$ and $\beta(\mathcal{A}) = \sum_{j \in \mathcal{A}} \beta_j$ to extend C and β to sets.

Instead of fixed penalties and costs per observation, both can also depend on the state of the world. For example, in the medical domain, applying a particular diagnostic test can bear different risks for the health of the patient, depending on the patient's illness. The algorithms we will develop below can be adapted to accommodate such dependencies in a straight-forward manner. We will present details only for the conditional planning algorithm in Section 15.3.2.

15.2 Decomposing rewards

In this section, we will present the key observation that allows us to develop efficient algorithms for nonmyopically optimizing value of information in the class of chain graphical models. The algorithms will be presented in Section 15.3.

The set of random variables $\mathcal{X}_\mathcal{V} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ forms a *chain graphical model* (a chain), if $\mathcal{X}_i$ is conditionally independent of $\mathcal{X}_{\mathcal{V}\setminus\{i-1,i,i+1\}}$ given $\mathcal{X}_{i-1}$ and $\mathcal{X}_{i+1}$. Without loss of generality we can assume that the joint distribution is specified by the prior $P(\mathcal{X}_1)$ of variable $\mathcal{X}_1$ and the conditional probability distributions $P(\mathcal{X}_{i+1} \mid \mathcal{X}_i)$. The time series model for the temperature measured by one sensor in our example can be formulated as a chain graphical model. Note that the transition probabilities $P(\mathcal{X}_{i+1} \mid \mathcal{X}_i)$ are allowed to depend on the index i (i.e., the chain models are allowed to be *nonstationary*). Chain graphical models have been extensively used in machine learning and signal processing.

Consider for example a Hidden Markov Model unrolled for n time steps, i.e., $\mathcal{V}$ can be partitioned into the hidden variables $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ and the emission variables $\{\mathcal{Y}_1, \ldots, \mathcal{Y}_n\}$. In HMMs, the $\mathcal{Y}_i$ are always observed and the variables $\mathcal{X}_i$ form a chain. In many applications, some of which are discussed in Section 15.5, we can observe some of the hidden variables $\mathcal{X}_i$ as well, e.g., by asking an expert, in addition to observing the emission variables. In these cases, the problem of selecting expert labels also belongs to the class of chain graphical models addressed in this chapter, since the variables X_i form a chain conditional on the observed values of the emission variables Y_i. This idea can be generalized to the class of Dynamic Bayesian Networks where the separators between time slices have size one, and only these separators can be selected for observation. This formulation also includes certain conditional random fields [Lafferty et al., 2001] which form chains, conditional on the emission variables (the features).

Chain graphical models originating from time series have additional, specific properties: In a system for online decision making, only observations from the past and present time steps can be taken into account, not observations which will be made in the future. This is generally referred to as the *filtering* problem. In this setting, the notation $P(\mathcal{X}_i \mid \mathcal{X}_\mathcal{A})$ will refer to the distribution of $\mathcal{X}_i$ conditional on observations in $\mathcal{X}_\mathcal{A}$ prior to and including time i. For structured classification problems as discussed in Section 15.5, in general observations made anywhere in the chain must be taken into account. This situation is usually referred to as the *smoothing* problem. We will provide algorithms both for filtering and smoothing.

We will now describe the key insight, which allows for efficient optimization in chains. Consider a set of observations $\mathcal{A} \subseteq \mathcal{V}$. If the j variable is observed, i.e., $j \in \mathcal{A}$, then the local reward is simply $R(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A}) = R(\mathcal{X}_j \mid \mathcal{X}_j)$. Now consider $j \notin \mathcal{A}$, and let $\mathcal{A}_j$ be the subset of $\mathcal{A}$ containing the closest ancestor (and for the smoothing problem also the closest descendant) of $\mathcal{X}_j$ in $\mathcal{X}_\mathcal{A}$. The conditional independence property of the graphical model implies that, given $\mathcal{X}_{\mathcal{A}_j}$, $\mathcal{X}_j$ is independent of the rest of the observed variables, i.e., $P(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A}) = P(\mathcal{X}_j \mid \mathcal{X}_{\mathcal{A}_j})$. Thus, it follows that $R(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A}) = R(\mathcal{X}_j \mid \mathcal{X}_{\mathcal{A}_j})$.

These observations imply that the expected reward of some set of observations decomposes along the chain. For simplicity of notation, we add two independent dummy variables $\mathcal{X}_0$ and $\mathcal{X}_{n+1}$, where $R_0 =$

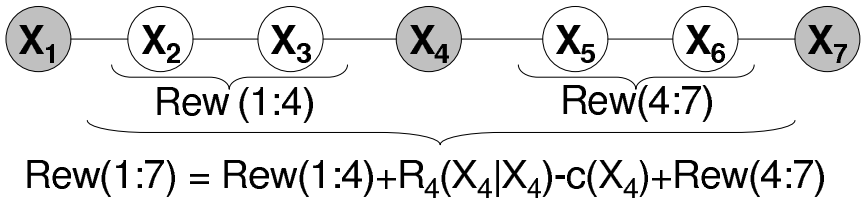


Figure 15.2: Illustration of the decomposing rewards idea. The reward for chain 1:7 when observing variables $\mathcal{X}_1$, $\mathcal{X}_4$ and $\mathcal{X}_7$ decomposes as the sum of chain 1:4 plus the reward for chain 4:7 plus the immediate reward for observing $\mathcal{X}_4$ minus the cost of observing $\mathcal{X}_4$. Hereby for brevity we use the notation $Rew(a:b) = \sum_{j=a}^{b} R_j(\mathcal{X}_j \mid \mathcal{X}_1, \mathcal{X}_4, \mathcal{X}_7)$.

$C_0 = \beta_0 = R_{n+1} = C_{n+1} = \beta_{n+1} = 0$. Let $\mathcal{A} = \{i_0, \ldots, i_{m+1}\}$ where $i_l < i_{l+1}$, $i_0 = 0$ and $i_{m+1} = n+1$. Using this notation, the total reward $R(\mathcal{A}) = \sum_j R_j(X_j \mid \mathcal{X}_\mathcal{A})$ for the smoothing case is given by:

$$\sum_{v=0}^{m} \left(R_{i_v}(\mathcal{X}_{i_v} \mid \mathcal{X}_{i_v}) - C_{i_v} + \sum_{j=i_v+1}^{i_{v+1}-1} R_j(\mathcal{X}_j \mid \mathcal{X}_{i_v}, \mathcal{X}_{i_{v+1}}) \right).$$

In filtering settings, we simply replace $R_j(\mathcal{X}_j \mid \mathcal{X}_{i_v}, \mathcal{X}_{i_{v+1}})$ by $R_j(\mathcal{X}_j \mid \mathcal{X}_{i_v})$. Figure 15.2 illustrates this decomposition.

15.3 Efficient algorithms for optimizing value of information

In this section, we present algorithms for efficiently and nonmyopically optimizing value of information in chain graphical models.

15.3.1 Efficient algorithms for optimal subset selection in chain models

In the *subset selection* problem, we want to find a most informative subset of the variables to observe *in advance*, i.e., before any observations are made. In our running example, we would, before deploying the sensors, identify k time points that are expected to provide the most informative sensor readings according to our model.

First, define the objective function L on subsets of $\mathcal{V}$ by

$$L(\mathcal{A}) = \sum_{j=1}^{n} R_j(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A}) - C(\mathcal{A}). \tag{15.3}$$

The *subset selection* problem is to find the optimal subset

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A} \subseteq \mathcal{V}, \beta(\mathcal{A}) \leq B} L(\mathcal{A})$$

maximizing the sum of expected local rewards minus the penalties, subject to the constraint that the total cost must not exceed the budget B.

We solve this optimization problem using a dynamic programming algorithm, where the chain is broken into sub-chains using the insight from Section 15.2. Consider a sub-chain from variable $\mathcal{X}_a$ to $\mathcal{X}_b$. We define $L^{sm}_{a:b}(k)$ to represent the expected total reward for the sub-chain $\mathcal{X}_a, \ldots, \mathcal{X}_b$, in the smoothing setting where $\mathcal{X}_a$ and $\mathcal{X}_b$ are observed, and with a budget level of k. $L^{flt}_{a:b}(k)$ represents the expected reward in the filtering setting where only $\mathcal{X}_a$ is observed. More formally:

$$L^{flt}_{a:b}(k) = \max_{\substack{\mathcal{A}\subset\{a+1\ldots b-1\}\\ \beta(\mathcal{A})\leq k}} \sum_{j=a+1}^{b-1} R_j(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A}, \mathcal{X}_a) - C(\mathcal{A}),$$

for the filtering version, and

$$L^{sm}_{a:b}(k) = \max_{\substack{\mathcal{A}\subset\{a+1\ldots b-1\}\\ \beta(\mathcal{A})\leq k}} \sum_{j=a+1}^{b-1} R_j(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A}, \mathcal{X}_a, \mathcal{X}_b) - C(\mathcal{A}),$$

for the smoothing version. Note that in both cases, $L_{0:n+1}(B) = \max_{\mathcal{A}:\beta(\mathcal{A})\leq B} L(\mathcal{A})$, as in Equation (15.3), i.e., by computing the values for $L_{a:b}(k)$, we compute the maximum expected total reward for the entire chain.

Algorithm 15.1: VOIDP algorithm for optimal subset selection (for both filtering and smoothing).

Input: Budget B, rewards R_j, costs β_j and penalties C_j
Output: Optimal selection $\mathcal{A}$ of observation times
begin
  for $0 \leq a < b \leq n+1$ **do** compute $L_{a:b}(0)$;
  for $k = 1$ **to** B **do**
    for $0 \leq a < b \leq n+1$ **do**
      $sel(-1) := L_{a:b}(0)$;
      for $j = a+1$ **to** $b-1$ **do** $sel(j) := R_j(\mathcal{X}_j \mid \mathcal{X}_j) - C_j + L_{a:j}(0) + L_{j:b}(k-\beta_j)$;
      $L_{a:b}(k) = \max_{j\in\{a+1,\ldots,b-1,-1\}} sel(j)$;
      $\Lambda_{a:b}(k) = \text{argmax}_{j\in\{a+1,\ldots,b-1,-1\}} sel(j)$;
    end
  end
  $a := 0$; $b := n+1$; $k := B$; $\mathcal{A} := \emptyset$;
  repeat
    $j := \Lambda_{a:b}(k)$;
    if $j \geq 0$ **then** $\mathcal{A} := \mathcal{A} \cup \{j\}$; $k := k - \beta_j$;
  until $j = -1$;
end

We can compute $L^{sm}_{a:b}(k)$ and $L^{flt}_{a:b}(k)$ using dynamic programming. The base case is simply:

$$L^{flt}_{a:b}(0) = \sum_{j=a+1}^{b-1} R_j(\mathcal{X}_j \mid \mathcal{X}_a),$$

for filtering, and

$$L^{sm}_{a:b}(0) = \sum_{j=a+1}^{b-1} R_j(\mathcal{X}_j \mid \mathcal{X}_a, \mathcal{X}_b),$$

for smoothing. The recursion for $L_{a:b}(k)$ has two cases: we can choose not to spend any more of the budget, reaching the base case, or we can break the chain into two sub-chains, selecting the optimal observation $\mathcal{X}_j$, where $a < j < b$. In both filtering and smoothing we have

$$L_{a:b}(k) = \max \left\{ L_{a:b}(0), \max_{j:a<j<b, \beta_j \leq k} \{ R_j(\mathcal{X}_j \mid \mathcal{X}_j) - C_j + L_{a:j}(0) + L_{j:b}(k - \beta_j) \} \right\}.$$

At first, it may seem that this recursion should consider the optimal split of the budget between the two sub-chains. However, since the subset problem is open-loop and the order of the observations is irrelevant, we only need to consider split points where the first sub-chain receives zero budget.

A pseudo code implementation for this dynamic programming approach, which we call VOIDP *for subset selection* is given in Algorithm 15.1. The algorithm fills the dynamic programming tables in two loops, the inner loop ranging over all pairs (a, b), $a < b$, and the outer loop increasing k. Within the inner loop, when computing the best reward for the sub-chain from a to b, it fills out a table sel, where $sel(j)$ is the reward that could be obtained by making an observation at j, and $sel(-1)$ is the reward if no observation is made.

In addition to computing the optimal rewards $L_{a:b}(k)$ that could be achieved for sub-chain $a : b$ and budget k, the algorithm also stores the choices $\Lambda_{a:b}(k)$ that realize this maximum score. Here, $\Lambda_{a:b}(k)$ is the index of the next variable that should be selected for sub-chain $a : b$ with budget k, or -1 if no variable should be selected. In order to recover the optimal subset for budget k, Algorithm 15.1 uses the quantities $\Lambda_{a:b}$ to recover the optimal subset by tracing the maximal values occurring in the dynamic programming equations.

Using an induction proof, we obtain:
Theorem 15.3 (Subset Selection). *The dynamic programming algorithm described above computes the optimal subset with budget B in $(\frac{1}{6}n^3 + \mathcal{O}(n^2))B$ evaluations of expected local rewards.* □

Note that if we do not consider different costs β for each variable, we would simply choose $\beta_j = 1$ for all variables and compute $L_{a:b}(N)$.

Further note that if the variables $\mathcal{X}_i$ are continuous, our algorithm is still applicable when the integrations and inferences necessary for computing the expected rewards can be performed efficiently. This is the case, for example, in a Gaussian linear model (i.e., the variables $\mathcal{X}_i$ are normally distributed) and the local reward functions are the residual entropies or the residual variances for each variable.

15.3.2 Efficient algorithms for optimal conditional planning in chain models

In the *conditional plan* problem, we want to compute an optimal sequential querying policy π: We observe a variable, pay the penalty, and depending on all values observed in the past, select the next query, proceeding as long as our budget suffices. The objective is to find the plan with the highest expected reward, where, for each possible sequence of observations, the budget B is not exceeded. For filtering, we can only select observations in the future, whereas in the smoothing case, the next observation can be anywhere in the chain. In our running example, the filtering algorithm would be most appropriate: The sensors would sequentially follow the conditional plan, deciding on the most informative times to sense based on the previous observations. Figure 15.1 shows an example of such a conditional plan.

From subset selection to conditional planning

Note that in contrast to the subset selection setting that we considered in Section 15.3.1, in conditional planning, the set of variables depend on the state of the world $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$. Hence, for each such state, the conditional plan π could select a different set of variables, $\pi(\mathbf{x}_\mathcal{V}) \subseteq \mathcal{V}$. As an example, consider Figure 15.1, where the set of possible observations is $\mathcal{V} = \{morn, noon, eve\}$, and $\mathcal{X}_\mathcal{V} = \{T_{morn}, T_{noon}, T_{eve}\}$. If the world is in state $\mathbf{x}_\mathcal{V} = (high, low, high)$, then the conditional plan π presented in Figure 15.1 would select $\pi(\mathbf{x}_\mathcal{V}) = \{morn, eve\}$, whereas, if $\mathbf{x}_\mathcal{V} = (low, low, high)$, it would select $\pi(\mathbf{x}_\mathcal{V}) = \{morn, noon\}$. Since the conditional plan is a function of the (random) state of the world, it is a set-valued random variable. In order to optimize Problem (15.2), we define the objective function[2]

$$\mathrm{J}(\pi) = \sum_{\mathbf{x}_\mathcal{V}} P(\mathbf{x}_\mathcal{V}) \sum_{j=1}^{n} \left[R_j(\mathcal{X}_j \mid \mathbf{x}_{\pi(\mathbf{x}_\mathcal{V})}) - C(\pi(\mathbf{x}_\mathcal{V})) \right],$$

i.e., the expected sum of local rewards given the observations made by plan $\pi(\mathbf{x}_\mathcal{V})$ in state $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$ minus the penalties of the selected variables, where the expectation is taken with respect to the distribution $P(\mathcal{X}_\mathcal{V})$. In addition to defining the value of a policy $\mathrm{J}(\pi)$, we also define the cost $\beta(\pi)$ as

$$\beta(\pi) = \max_{\mathbf{x}_\mathcal{V}} \beta(\pi(\mathbf{x}_\mathcal{V})),$$

i.e., as maximum cost $\beta(\mathcal{A})$ (as defined in Section 15.3.1) of any set $\mathcal{A} = \pi(\mathbf{x}_\mathcal{V})$ that could be selected by the policy π, in any state of the world $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$.

Based on this notation, our goal is to find a policy π^* such that

$$\pi^* = \operatorname*{argmax}_{\pi \in \Pi} \mathrm{J}(\pi) \text{ such that } \beta(\pi) \leq B,$$

i.e., a policy that has maximum value, and is guaranteed to never have cost exceeding our budget B. Hereby Π is the class of *sequential* policies (i.e., those, where the observations are chosen sequentially, only based on observations that have been previously made).

It will be useful to introduce the following notation:

$$J(\mathbf{x}_\mathcal{A}; k) = \max_{\pi \in \Pi} \mathrm{J}(\pi \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) \text{ such that } \beta(\pi) \leq k,$$

where

$$\mathrm{J}(\pi \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \sum_{\mathbf{x}_\mathcal{V}} P(\mathbf{x}_\mathcal{V} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) \sum_{j=1}^{n} \left[R_j(\mathcal{X}_j \mid \mathbf{x}_{\pi(\mathbf{x}_\mathcal{V})}) - C(\pi(\mathbf{x}_\mathcal{V})) \right].$$

Hence, $J(\mathbf{x}_\mathcal{A}; k)$ is the best possible reward that can be achieved by any sequential policy with cost at most k, after observing $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$. Using this notation, our goal is to find the optimal plan with reward $J(\emptyset; B)$. Note that the function $J(\mathbf{x}_\mathcal{A}; k)$ is analogous to the concept of a *value function* in Markov Decision Processes [*c.f.*, Bellman, 1957b][3]: Such a value function quantifies the best possible long term

[2]Recall that, in the filtering setting, $R(\mathcal{X}_j \mid \mathbf{x}_{\pi(\mathbf{x}_\mathcal{V})}) \triangleq R(\mathcal{X}_j \mid \mathbf{x}_{\mathcal{A}'})$, where $\mathcal{A}' = \{t \in \pi(\mathbf{x}_\mathcal{V}) \text{ s.t. } t \leq j\}$, i.e., only observations from the past are taken into account.

[3]Note that the state space of the MDP corresponding to our setting is exponentially large, as it consists of all possible joint assignments to the vector $\mathcal{X}_\mathcal{V} = \mathbf{x}_\mathcal{V}$. Hence, existing algorithms for solving MDPs cannot be applied to large value of information problems.

reward achievable in any given state. In our setting, the state is uniquely determined by the observed evidence $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$.

In analogy to finite-horizon Markov Decision Processes, the value function J satisfies the following recursion. The base case considers the exhausted budget:

$$J(\mathbf{x}_\mathcal{A}; 0) = \sum_{j \in \mathcal{V}} R_j(\mathcal{X}_j \mid \mathbf{x}_\mathcal{A}) - C(\mathcal{A}).$$

For the recursion, it holds that

$$J(\mathbf{x}_\mathcal{A}; k) = \max \left\{ J(\mathbf{x}_\mathcal{A}; 0), \max_{j \notin \mathcal{A}} \left\{ \sum_{x_j} P(x_j \mid \mathbf{x}_\mathcal{A}) J(\mathbf{x}_\mathcal{A}, x_j; k - \beta_j) \right\} \right\},$$

i.e., the best one can do in state $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$ with budget k is to either stop selecting variables, or chose the best next variable and act optimally thereupon.

Note that we can easily allow the cost β_j depend on the state x_j of variable $\mathcal{X}_j$. In this case, we would simply replace β_j by $\beta_j(x_j)$, and define $J(\mathcal{X}_\mathcal{A}, r) = -\infty$ whenever $r < 0$. Equivalently, we can let the cost $C(\mathcal{A})$ depend on the state by replacing $C(\mathcal{A})$ by $C(\mathbf{x}_\mathcal{A})$.

Dynamic programming for optimal conditional planning in chains

We propose a dynamic programming algorithm for obtaining the optimal conditional plan that is similar to the subset algorithm presented in Section 15.3.1. Again, we utilize the decomposition of rewards described in Section 15.2. The difference here is that the observation selection and budget allocation now depend on the actual values of the observations. In order to compute the value function $J(\mathbf{x}_\mathcal{A}; k)$ for the entire chain, we will compute the value functions $J_{a:b}(\mathbf{x}_\mathcal{A}; k)$ for sub-chains $\mathcal{X}_a, \ldots, \mathcal{X}_b$.

The base case of our dynamic programming approach deals with the zero budget setting:

$$J^{flt}_{a:b}(x_a; 0) = \sum_{j=a+1}^{b-1} R_j(\mathcal{X}_j \mid \mathcal{X}_a = x_a),$$

for filtering, and

$$J^{sm}_{a:b}(x_a, x_b; 0) = \sum_{j=a+1}^{b-1} R_j(\mathcal{X}_j \mid \mathcal{X}_a = x_a, \mathcal{X}_b = x_b),$$

for smoothing. The recursion defines $J_{a:b}(x_a; k)$ ($J_{a:b}(x_a, x_b; k)$ for smoothing), the expected reward for the problem restricted to the sub-chain $\mathcal{X}_a, \ldots, \mathcal{X}_b$ conditioned on the values of $\mathcal{X}_a = x_a$ (and $\mathcal{X}_b = x_b$ for smoothing), and with budget limited by k. To compute this quantity, we again iterate through possible split points j, such that $a < j < b$. Here we observe a notable difference between the filtering and the smoothing case. For smoothing, we now must consider all possible splits of the budget between the two resulting sub-chains, since an observation at time j might require us to make an additional, earlier observation:

$$J^{sm}_{a:b}(x_a,x_b; k) = \max \Bigg\{ J^{sm}_{a:b}(x_a, x_b; 0), \max_{a<j<b} \Bigg\{ \sum_{x_j} P(\mathcal{X}_j = x_j \mid \mathcal{X}_a = x_a, \mathcal{X}_b = x_b) \Big\{ R_j(\mathcal{X}_j \mid x_j) - C_j(x_j) + \max_{0 \le l \le k - \beta_j(x_j)} \left[J^{sm}_{a:j}(x_a, x_j; l) + J^{sm}_{j:b}(x_j, x_b; k - l - \beta_j(x_j)) \right] \Big\} \Bigg\} \Bigg\}.$$

Looking back in time is not possible in the filtering case, hence the recursion simplifies to

$$J^{flt}_{a:b}(x_a;k) = \max\Big\{ J^{flt}_{a:b}(x_a;0), \max_{a<j<b:\beta_j(x_j)\le k} \Big\{ \sum_{x_j} P(\mathcal{X}_j = x_j \mid \mathcal{X}_a = x_a) \Big\{ R_j(\mathcal{X}_j \mid x_j) - C_j(x_j) + J^{flt}_{a:j}(x_a;0) + J^{flt}_{j:b}(x_j;k-\beta_j(x_j)) \Big\} \Big\} \Big\}.$$

For both J^{flt} and J^{sm}, the optimal reward is obtained by $J_{0:n+1}(\emptyset;B) = J(\emptyset;B) = \mathrm{J}(\pi^*)$. Algorithm 15.2 presents a pseudo code implementation for the smoothing version – the filtering case is a straight-forward modification. We call Algorithm 15.2 the VOIDP *algorithm for conditional planning*. The algorithm will fill the dynamic programming tables using three loops, the inner loop ranging over all assignments x_a, x_b, the middle loop ranging over all pairs (a, b) where $a < b$, and the outer loop covers increasing values of $k \le B$. Within the innermost loop, the algorithm again computes a table sel such that $sel(j)$ is the optimal reward achievable by selecting variable j next. This value is now an expectation over any possible observation that variable $\mathcal{X}_j$ can make. Note that for every possible instantiation $\mathcal{X}_j = x_j$ a different allocation of the remaining budget $k - \beta_j(x_j)$ to the left and right sub-chain ($a : j$ and $j : b$ respectively) can be chosen. The quantity $\sigma(j, x_j)$ tracks this optimal budget allocation.

Algorithm 15.2: VOIDP algorithm for computating an optimal conditional plan (for the smoothing setting).

Input: Budget B, rewards R_j, costs β_j and penalties C_j
Output: Optimal conditional plan ($\pi_{a:b}$, $\sigma_{a:b}$)
begin
  for $0 \le a < b \le n+1, x_a \in \mathrm{dom}\,\mathcal{X}_a, x_b \in \mathrm{dom}\,\mathcal{X}_b$ **do** compute $J^{sm}_{a:b}(x_a, x_b; 0)$;
  for $k = 1$ **to** B **do**
    for $0 \le a < b \le n+1, x_a \in \mathrm{dom}\,\mathcal{X}_a, x_b \in \mathrm{dom}\,\mathcal{X}_b$ **do**
      $sel(-1) := J^{sm}_{a:b}(0)$;
      for $a < j < b$ **do**
        $sel(j) := 0$;
        for $x_j \in \mathrm{dom}\,\mathcal{X}_j$ **do**
          for $0 \le l \le k - \beta_j(x_j)$ **do**
            $bd(l) := \quad J^{sm}_{a:j}(x_a, x_j; l) + J^{sm}_{j:b}(x_j, x_b; k - l - \beta_j(x_j))$;
          end
          $sel(j) := sel(j) + P(x_j \mid x_a, x_b) \cdot [R_j(\mathcal{X}_j \mid x_j) - C_j(x_j) + \max_l bd(j)]$;
          $\sigma(j, x_j) = \mathrm{argmax}_l\, bd(j)$;
        end
      end
      $J^{sm}_{a:b}(k) = \max_{j\in\{a+1,\ldots,b-1,-1\}} sel(j)$;
      $\pi_{a:b}(x_a, x_b; k) = \mathrm{argmax}_{j\in\{a+1,\ldots,b-1,-1\}} sel(j)$;
      for $x_j \in \mathrm{dom}\, X_{\pi_{a:b}(k)}$ **do** $\sigma_{a:b}(x_a, x_b, x_j; k) = \sigma(\pi_{a:b}(k), x_j)$;
    end
  end
end

Algorithm 15.3: Observation selection using conditional planning.

Input: Budget k, observations $\mathcal{X}_a = x_a$, $\mathcal{X}_b = x_b$, σ, π
begin
  $j := \pi_{a:b}(x_a, x_b; k)$;
  if $j \geq 0$ **then**
    Observe $\mathcal{X}_j = x_j$;
    $l := \sigma_{a:b}(x_a, x_b, x_j; k)$;
    Recurse with $k := l$, $\mathcal{X}_a = x_a$ and $\mathcal{X}_j = x_j$ instead of $\mathcal{X}_b = x_b$;
    Recurse with $k := k - l - \beta_j$, $\mathcal{X}_j = x_j$ instead of $\mathcal{X}_a = x_a$, and $\mathcal{X}_b = x_b$;
  end
end

The plan itself is compactly encoded in the quantities $\pi_{a:b}$ and $\sigma_{a:b}$. Hereby, $\pi_{a:b}(x_a, x_b; k)$ determines the next variable to query after observing $\mathcal{X}_a = x_a$ and $\mathcal{X}_b = x_b$, and with remaining budget k. $\sigma_{a:b}(x_a, x_b, x_j; k)$ determines the allocation of the budget after the new observation $\mathcal{X}_j = x_j$ has been made. Considering the exponential number of possible sequences of observations, it is remarkable that the optimal plan can be represented using only polynomial space. Algorithm 15.3 indicates how the computed plan can be executed. The procedure is recursive, requiring the parameters $a := 0$, $x_a := 1$, $b := n + 1$, $x_b := 1$ and $k := B$ for the initial call. In our temperature monitoring example, we could first collect some temperature timeseries as training data, and then learn the chain model from this data. Offline, we would then compute the conditional plan (for the filtering setting), and encode it in the quantities $\pi_{a:b}$ and $\sigma_{a:b}$. We would then deploy the computed plan on the actual sensor node, together with an implementation of Algorithm 15.3. While computation of the optimal plan (Algorithm 15.2) is fairly computationally expensive, the execution of the plan (Algorithm 15.3) is very efficient (selecting the next timestep for observation requires a single lookup in the $\pi_{a:b}$ and $\sigma_{a:b}$ tables) and hence well-suited for deployment on a small, embedded device.

We summarize our analysis in the following Theorem:
Theorem 15.4 (Conditional Planning). *The algorithm for smoothing presented above computes an optimal conditional plan in $d^3 \cdot B^2 \cdot (\frac{1}{6}n^3 + \mathcal{O}(n^2))$ evaluations of local rewards, where d is the maximum domain size of the random variables $X_1, \ldots, X_n$. In the filtering case, the optimal plan can be computed using $d^3 \cdot B \cdot (\frac{1}{6}n^3 + \mathcal{O}(n^2))$ evaluations, or, if no budget is used, in $d^3 \cdot (\frac{1}{6}n^4 + \mathcal{O}(n^3))$ evaluations.* □

The faster computation for the filtering / no-budget case is obtained by observing that we do not require the third maximum computation, which distributes the budget into the sub-chains.

Also, note that contrary to the algorithm for computing optimal subsets in Section 15.3.1, Algorithm 15.2 only requires evaluations of the form $R(\mathcal{X}_j \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$, which can in general be computed d^2 times faster than the expectations $R(X_j \mid \mathcal{X}_\mathcal{A})$. Under this consideration, the subset selection algorithm is in general only a factor $d \cdot B$ faster, even though the conditional planning algorithm has more nested loops.

15.3.3 Efficient algorithms for trees with few leaves

In Sections 15.3.1 and 15.3.2 we have presented dynamic programming-based algorithms that can optimize value of information on chain graphical models. In fact, the key observations of Section 15.2 that local rewards decompose along chains holds not just in chain graphical models, but also in trees.

More formally, a tree graphical model is a joint probability distribution $P(\mathcal{X}_\mathcal{V})$ over a collection of random variables $\mathcal{X}_\mathcal{V}$ if $P(\mathcal{X}_\mathcal{V})$ factors as

$$P(\mathcal{X}_\mathcal{V}) = \frac{1}{Z} \prod_{(i,j)\in\mathcal{E}} \psi_{i,j}(\mathcal{X}_i, \mathcal{X}_j),$$

where $\psi_{i,j}$ is a nonnegative potential function, mapping assignments to x_i and x_j to the nonnegative real numbers, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges that form an undirected tree over the index set $\mathcal{V}$, and Z is a normalization constant enforcing a valid probability distribution.

The dynamic programming algorithms presented in the previous sections can be extended to such tree models in a straightforward manner. Instead of identifying optimal subsets and conditional plans for subchains, the algorithms would then select optimal subsets and plans for sub-trees of increasing size. Note however that the number of sub-trees can grow exponentially in the number of leaves of the tree: A star on n leaves for example has a number of subtrees that is exponential in n. In fact, counting the number of subtrees of an arbitrary tree with n vertices is believed to be intractable ($\#\mathbf{P}$-complete, Goldberg and Jerrum [2000]). However, for trees that contain only a small (constant) number of leaves, the number of subtrees is polynomial, and the optimal subset and conditional plans can be computed in polynomial time.

15.4 Theoretical limits

Many problems that can be solved efficiently for discrete chain graphical models can also be efficiently solved for discrete polytrees[4]. Examples include probabilistic inference and the most probable explanation (MPE).

In Section 15.3.3 however we have seen that the complexity of the dynamic programming algorithms for chains increases dramatically when extended to trees: The complexity increases exponentially in the number of leafs of the tree.

Surprisingly, we prove that for the problem of optimizing value of information, this exponential increase in complexity cannot be avoided, under reasonable complexity theoretic assumptions. Before making this statement more formal, we briefly review the complexity classes used in our results.

15.4.1 Brief review of relevant computational complexity classes

We briefly review the complexity classes used in the following statements by presenting a complete problem for each of the class. For more details see, e.g., the reference by Papadimitriou [1995]. The popular class $\mathbf{NP}$ contains decision problems which have polynomial-time verifiable proofs. A well-known complete problem is $3SAT$ for which the instances are Boolean formulas ϕ in conjunctive normal form containing at most three literals per clause (3CNF form). The complexity class $\#\mathbf{P}$ contains counting problems. A complete problem for the class $\#\mathbf{P}$ is $\#3SAT$ which counts the number of satisfying instances to a 3CNF formula. $\mathbf{PP}$ is a decision version of the class $\#\mathbf{P}$: A complete problem is $MAJSAT$, which decides whether a given 3CNF formula ϕ is satisfied by the majority, i.e., by more than half of all

[4]Polytrees are Bayesian Networks that form trees if the edge directions are dropped.

its possible assignments. If A and B are Turing machine based complexity classes, then A^B is the complexity class derived by allowing the Turing machines deciding instances of A oracle calls to Turing machines in B. We can intuitively think of the problems in class A^B as those that can be solved by a Turing Machine for class A, that has a special command which solves any problem in B. $\mathbf{PP}$ is similar to $\#\mathbf{P}$ in that $\mathbf{P}^{\mathbf{PP}} = \mathbf{P}^{\#}\mathbf{P}$, i.e., if we allow a deterministic polynomial time Turing machine to have access to a counting oracle, we cannot solve more complex problems than if we give it access to a majority oracle. Combining these ideas, the class $\mathbf{NP}^{\mathbf{PP}}$ is the class of problems that can be solved by nondeterministic polynomial time Turing machines that have access to a majority (or a counting) oracle. A complete problem for $\mathbf{NP}^{\mathbf{PP}}$ is $EMAJSAT$ which, given a 3CNF on variables $X_1, \ldots, X_{2n}$, it decides whether there exists an assignment to $X_1, \ldots, X_n$ such that ϕ is satisfied for the majority of assignments to $X_{n+1}, \ldots, X_{2n}$. $\mathbf{NP}^{\mathbf{PP}}$ has been found to be a natural class for modeling AI planning problems [Littman et al., 1998]. As an example, the MAP assignment problem is $\mathbf{NP}^{\mathbf{PP}}$-complete for general graphical models, as shown by Park and Darwiche [2004].

The complexity classes satisfy the following set of inclusions (where the inclusions are assumed, but not known to be strict):

$$\mathbf{P} \subseteq \mathbf{NP} \subseteq \mathbf{PP} \subseteq \mathbf{P}^{\mathbf{PP}} = \mathbf{P}^{\#}\mathbf{P} \subseteq \mathbf{NP}^{\mathbf{PP}}.$$

15.4.2 Complexity of computing and optimizing value of information

In order to solve the optimization problems, we will most likely have to evaluate the objective function, i.e., the expected local rewards. Our first result states that, even if we specialize to decision theoretic value of information objective functions as defined in Section 15.1.1, this problem is intractable even for Naive Bayes models, a special case of discrete polytrees. Naive Bayes models are often used in classification tasks [*c.f.*, Domingos and Pazzani, 1997], where the class variable is predicted from noisy observations (features), that are assumed to be conditionally independent given the class variable. In a sense, Naive Bayes models are the "next simplest" (from the perspective of inference) class of Bayesian networks after chains. Note that Naive Bayes models correspond to the "stars" referred to in Section 15.3.3, that have a number of subtrees that is exponential in the number of variables.

Theorem 15.5 (**Hardness of computation for Naive Bayes models).** *The computation of decision theoretic value of information functions is $\#\mathbf{P}$-complete even for Naive Bayes models. It is also hard to approximate to any factor unless $\mathbf{P} = \mathbf{NP}$.* □

All proofs in this section are stated in the Appendix. We have the immediate corollary that the subset selection problem is $\mathbf{PP}$-hard for Naive Bayes models:

Corollary 15.6 (**Hardness of subset selection for Naive Bayes models).** *The problem of determining, given a Naive Bayes model, constants c and B, cost function β and a set of decision-theoretic value of information objective functions R_i, whether there is a subset of variables $\mathcal{A} \subseteq \mathcal{V}$ such that $L(\mathcal{A}) \geq c$ and $\beta(\mathcal{A}) \leq B$ is $\mathbf{PP}$-hard.* □

In fact, we can show that subset selection for arbitrary discrete polytrees (that are more general than Naive Bayes models, but inference is still tractable) is even $\mathbf{NP}^{\mathbf{PP}}$-complete, a complexity class containing problems that are believed to be significantly harder than $\mathbf{NP}$ or $\#\mathbf{P}$ complete problems. This result provides a complexity theoretic classification of value of information, a core AI problem.

Theorem 15.7 (**Hardness of subset selection computation for polytrees).** *The problem of determining, given a discrete polytree, constants c and B, cost function β and a set of decision-theoretic value of information objective functions R_i, whether there is a subset of variables $\mathcal{A} \subseteq \mathcal{V}$ such that $L(\mathcal{A}) \geq c$ and*

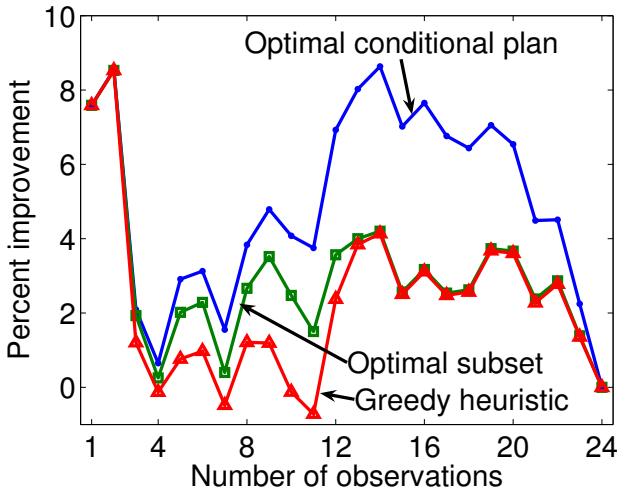


Figure 15.3: Sensor scheduling on temperature data: Improvement over the uniform spacing heuristic.

$\beta(\mathcal{A}) \leq B$ *is* $\mathbf{NP}^{\mathbf{PP}}$*-complete.* □

For our running example, this implies that the generalized problem of optimally selecting k sensors from a network of correlated sensors is most likely computationally intractable without resorting to heuristics. A corollary extends the hardness of subset selection to the hardness of conditional plans.

Corollary 15.8 (**Hardness of conditional planning computation for polytrees).** *Computing conditional plans is* $\mathbf{PP}$*-hard for Naive Bayes models and* $\mathbf{NP}^{\mathbf{PP}}$*-hard for discrete polytrees.* □

15.5 Experiments

In this section, we evaluate our algorithms on several real world data sets. A special focus is on the comparison of the optimal methods with the greedy heuristic and other heuristic methods for selecting observations, and on how the algorithms can be used for interactive structured classification.

15.5.1 Temperature time series

The first data set consists of temperature time series collected from a sensor network deployed at Intel Research Berkeley [Deshpande et al., 2004] as described in our running example. Data was continuously collected for 19 days, linear interpolation was used in case of missing samples. The temperature was measured once every 60 minutes, and it was discretized into 10 bins of 2 degrees Kelvin. To avoid overfitting, we used pseudo counts $\alpha = 0.5$ when learning the model. Using parameter sharing, we learned four sets of transition probabilities: from 12 am - 7am, 7 am - 12 pm, 12 pm - 7 pm and 7 pm - 12 am. Combining the data from three adjacent sensors, we got 53 sample time series.

The goal of this task was to select k out of 24 time points during the day, during which sensor readings are most informative. The experiment was designed to compare the performance of the optimal algorithms, the greedy heuristic, and a uniform spacing heuristic, which distributed the k observations uniformly over the day. Figure 15.3 shows the relative improvement of the optimal algorithms and the greedy heuristic

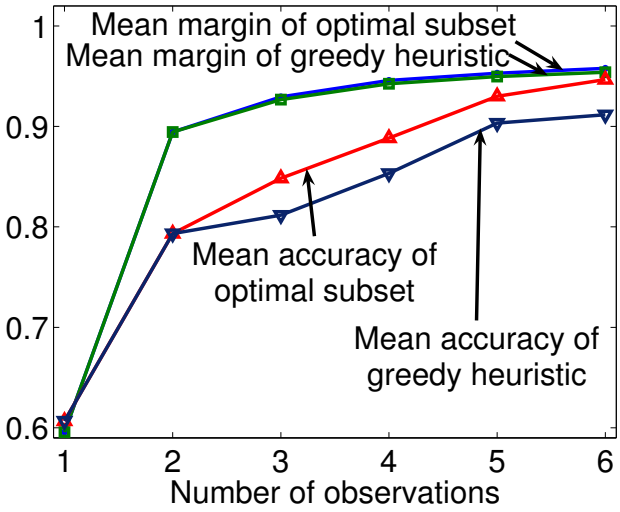


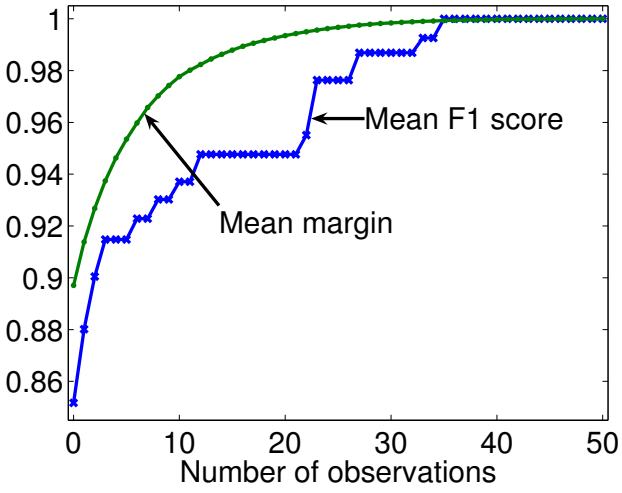
Figure 15.4: CpG island detection data set: Effect of increasing the number of observations on margin and classification accuracy.

over the uniform spacing heuristic. The performance is measured in decrease of expected entropy, with zero observations as the baseline. It can be seen that if k is less than about the half of all possible observations, the optimal algorithms decreased the expected uncertainty by several percent over both heuristics. The improvement gained by the optimal plan over the subset selection algorithms appears to become more drastic if a large number of observations (over half of all possible observations) is allowed. Furthermore, for a large number of observations, the optimal subset and the subset selected by the greedy heuristic were almost identical.

15.5.2 CpG-Island detection

We then studied the bioinformatics problem of finding CpG islands in DNA sequences. CpG islands are regions in the genome with a high concentration of the cytosine-guanine sequence. These areas are believed to be mainly located around the promoters of genes, which are frequently expressed in the cell. In our experiment, we considered the gene loci HS381K22, AF047825 and AL133174, for which the GenBank annotation listed three, two and one CpG islands each. We ran our algorithm on a 50 base window at the beginning and end of each island, using the transition and emission probabilities from Durbin et al. [1999] for our Hidden Markov Model, and we used the sum of margins as reward function.

The goal of this experiment was to locate the beginning and ending of the CpG islands more precisely by asking experts, whether or not certain bases belong to the CpG region or not. Figure 15.4 shows the mean classification accuracy and mean margin scores for an increasing number of observations. The results indicate that, although the expected margin scores are similar for the optimal algorithm and the greedy heuristic, the mean classification performance of the optimal algorithm was still better than the performance of the greedy heuristic.



(a) *Part of Speech Tagging*

Figure 15.5: Part-of-Speech tagging data set: Effect of increasing the number of observations on margin and F1 score.

15.5.3 Part-of-Speech Tagging

In our third experiment, we investigated the structured classification task of part-of-speech (POS) tagging [CoNLL, 2003]. Problem instances are sequences of words (sentences), where each word is part of an entity (e.g., "European Union"), and each entity belongs to one of five categories: Location, Miscellaneous, Organization, Person or Other. Imagine an application, where automatic information extraction is guided by an expert: Our algorithms compute an optimal conditional plan for asking the expert, trying to optimize classification performance while requiring as little expert interaction as possible.

We used a conditional random field for the structured classification task, where each node corresponds to a word, and the joint distribution is described by node potentials and edge potentials. The sum of margins was used as reward function. Measure of classification performance was the F1 score, the geometric mean of precision and recall. The goal of this experiment was to analyze how the addition of expert labels increases the classification performance, and how the indirect, decomposing reward function used in our algorithms corresponds to real world classification performance.

Figure 15.5 shows the increase of the mean expected margin and F1 score for an increasing number of observations, summarized over ten 50 word sequences. It can be seen that the classification performance can be effectively enhanced by optimally incorporating expert labels. Requesting only three out of 50 labels increased the mean F1 score from by more than five percent. The following example illustrates this effect: In one scenario both words of an entity, the sportsman 'P. Simmons', were classified incorrectly – 'P.' as *Other* and 'Simmons' as *Miscellaneous*. The first request of the optimal conditional plan was to label 'Simmons'. Upon labeling this word correctly, the word 'P.' was automatically labeled correctly also, resulting in an F1 score of 100 percent.

15.6 Applying chain algorithms to more general graphical models

In Section 15.3 we have seen algorithms that can be used to schedule a single sensor, assuming the time series of sensor readings (e.g., temperature) form a Markov chain. This is a very natural assumption for sensor networks [Deshpande et al., 2004]. When deploying sensor networks however, multiple sensors need to be scheduled. If the time series for all the sensors were independent, we could use our algorithms to schedule all the sensors independently of each other. However, in practice, the measurements will be correlated across the different sensors – in fact, this dependence is essential to allow generalization of measurements to locations where no sensor has been placed. In the following, we will describe an approach for using our single-sensor scheduling algorithm to coordinate multiple sensors.

More formally, we are interested in monitoring a spatiotemporal phenomenon at a set of locations $\mathcal{S} = \{1, \ldots, m\}$, and time steps $\mathcal{T} = \{1, \ldots, T\}$. With each location–time pair s, t, we associate a random variable $\mathcal{X}_{s,t}$ that describes the state of the phenomenon at that location and time. The random vector $\mathcal{X}_{\mathcal{S},\mathcal{T}}$ fully describes the relevant state of the world and the vector $\mathcal{X}_{\mathcal{S},t}$ describes the state at a particular time step t. As before, we make the Markov assumption, assuming conditional independence of $\mathcal{X}_{\mathcal{S},t}$ from $\mathcal{X}_{\mathcal{S},t'}$ given $\mathcal{X}_{\mathcal{S},t-1}$ for all $t' < t - 1$.

Similarly as in the single-chain case, we consider reward functions $R_{s,t}$ that are associated with each variable $\mathcal{X}_{s,t}$. Our goal is then to select, for each timestep, a set $\mathcal{A}_t \subseteq \mathcal{S}$ of sensors to activate, in order to maximize the sum of expected rewards. Letting $\mathcal{A}_{1:t} = \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_t$, the expected total reward is then given as

$$\sum_{s,t} R_{s,t}(\mathcal{X}_{s,t} \mid \mathcal{X}_{\mathcal{A}_{1:t}})$$

for the filtering setting (i.e., only observations in the past are taken into account for evaluating the rewards), and

$$\sum_{s,t} R_{s,t}(\mathcal{X}_{s,t} \mid \mathcal{X}_{\mathcal{A}_{1:T}})$$

for the smoothing setting (where all observations are taken into account). The generalization to conditional planning is done as described in Section 15.1.

For submodular objective functions, we could use the ESPASS algorithm from Chapter 12 to solve this problem. However, as shown by Proposition 15.1, the decision theoretic value of information does not satisfy submodularity. Furthermore, ESPASS can only be applied in the subset selection setting, not the sequential, adaptive setting. Lastly, it requires the objective functions F_i at all time steps to be the same. Note that in the case of a single sensor ($\ell = 1$), the problem of optimal sensor scheduling can be solved using Algorithm 15.1. Unfortunately, the optimization problem is wildly intractable even for the case of two sensors, $\ell = 2$:

Corollary 15.9 (Hardness of sensor selection for two chains). *Given a model with two dependent chains, constants c and B, a cost function β and a set of decision theoretic value of information functions $R_{s,t}$, it is $\mathbf{NP}^{\mathbf{PP}}$-complete to determine whether there is a subset $\mathcal{A}_{1:T}$ of variables such that $L(\mathcal{A}_{1:T}) \geq c$ and $\beta(\mathcal{A}_{1:T}) \leq B$.*

In the following, we will develop an approximate algorithm that uses our optimal single-chain algorithms and performs well in practice.

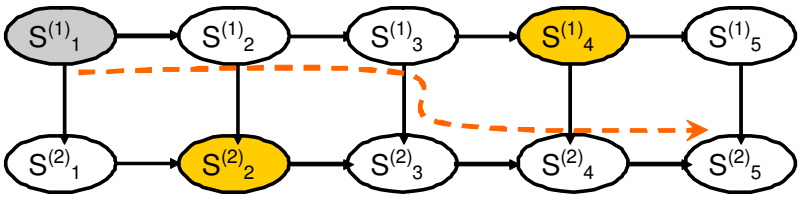


Figure 15.6: Scheduling multiple correlated sensors in dynamic processes.

15.6.1 Approximate sensor scheduling by lower bound maximization

The reason for the sudden increase in complexity in the case of multiple chains is that the decomposition of rewards along sub-chains (as described in Section 15.2) does not extend to the case of multiple sensors, since influence can flow across chains. Figure 15.6 visualizes this problem – there, the distribution for sensor (2) depends on all three observations $S_1^{(1)}$ and $S_4^{(1)}$ from sensor (1) and $S_2^{(2)}$ from sensor (2).

We address this complexity issue using an (approximate) extension of the decomposition approach used for single chains. We will focus on the decision-theoretic value of information objective (as described in Section 15.1.1), but other local reward functions, such as residual entropy, can be used as well.

Considering only recent observations. As a first approximation, we only allow a sensor to take into account the most recent observations. Intuitively, this appears to be a reasonable approximation, especially if the potential scheduling times in $\mathcal{T}$ are reasonably far apart. Formally, when evaluating the local rewards at time t, we replace the set of observations up to time t, $\mathcal{A}_{1:t} \subseteq \mathcal{T}$ by a subset $\mathcal{A}'_{1:t} \subseteq \mathcal{A}_{1:t}$ such that

$$\mathcal{A}'_{1:t} = \big\{(s,t) \in \mathcal{A}_{1:t} : t \geq t' \text{ for all } (s,t') \in \mathcal{A}_{1:t}\big\},$$

i.e, for each sensor s, only the last observation (with largest time index t) is kept. We then approximate $R_{s,t}(\mathcal{X}_{s,t} \mid \mathcal{A}_{1:t})$ by $R_{s,t}(\mathcal{X}_{s,t} \mid \mathcal{A}'_{1:t})$. In Figure 15.6 for example, where $\mathcal{A}_{1:5} = \{(s_1,1),(s_2,2),(s_1,4)\}$, the total expected utility at time t_5 would be computed using only observations $\mathcal{A}'_{1:5} = \{(s_2,2),(s_1,4)\}$, i.e., using time t_4 for sensor one, and time t_2 for sensor two, ignoring influence originating from observation $S_1^{(1)}$ and flowing through the chains as indicated by the dashed arrow. The following proposition proves that this approximation is a lower bound to the true value of information:

Proposition 15.10 (Monotonicity of value of information). *The decision-theoretic value of information $R_{s,t}(\mathcal{A})$ of a set $\mathcal{A}$ of sensors is monotonic in $\mathcal{A}$,*

$$R_{s,t}(\mathcal{A}') \leq R_{s,t}(\mathcal{A})$$

for all $\mathcal{A}' \subseteq \mathcal{A}$.

Proposition 15.10 proves that conditioning only on the most recent observations can only decrease our objective function, hence maximizing this approximate objective implies maximizing a lower bound on the true objective.

A coordinate ascent approach. We propose the following heuristic for maximizing the lower bound on the expected utility. Instead of jointly optimizing over all schedules (timesteps selected for each sensor), the algorithm will repeatedly iterate over all sensors. For all sensors s, it will optimize the selected

observations $\mathcal{A}^s_{1:T}$, holding the schedules for all other sensors fixed. This procedure resembles a coordinate ascent approach, where each "coordinate" ranges over all possible schedules for a fixed sensor s.

When optimizing for sensor s, the algorithm finds a schedule $\mathcal{A}^s_{1:T}$ such that

$$\mathcal{A}^s_{1:T} = \operatorname*{argmax}_{\mathcal{A}_{1:T}} \sum_{s,t} R_{s,t}\Big(\mathcal{X}_{s,t} \mid \mathcal{X}_{\mathcal{A}'_{1:t}} \bigcup_{s' \neq s} \mathcal{X}_{\mathcal{A}'^{s'}_{1:t}}\Big) \text{ such that } \beta(\mathcal{A}^s_{1:T}) \leq B, \tag{15.4}$$

i.e., that maximizes, over all schedules $\mathcal{A}_{1:T}$, the sum of expected rewards for all time steps and sensors, given the schedules $\mathcal{A}^{s'}_{1:T}$ for all non-selected sensors s'.

Solving the single-chain optimization problem. In order to solve the maximization problem (15.4) for the individual sensors, we use the same dynamic programming approach as introduced in Section 15.3. The recursive case $L^{flt}_{a:b}(k)$ for $k > 0$ is exactly the same. However, the base case is computed as

$$L^{flt}_{a:b}(0) = \sum_{j=a+1}^{b-1} \sum_{s} R_{s,j}\Big(\mathcal{X}_{s,j} \mid \mathcal{X}_a \bigcup_{s' \neq s} \mathcal{X}_{\mathcal{A}'^{s'}_{1:j}}\Big),$$

i.e., it takes into account the most recent observation for all non-selected sensors s'.

Several remarks need to be made about the computation of the base case $L^{flt}_{a:b}(0)$. First of all, in a naive implementation, the computation of the expected utility

$$R_{s,j}\Big(\mathcal{X}_{s,j} \mid \mathcal{X}_a \bigcup_{s' \neq s} \mathcal{X}_{\mathcal{A}'^{s'}_{1:j}}\Big)$$

requires time exponential in the number of chains. This is the case since, in order to compute the reward $R_{s,t}$, for each chain, all possible observations $\mathcal{X}_{\mathcal{A}'^{s}_{1:t}} = \mathbf{x}_{\mathcal{A}'^{s}_{1:t}}$ that could be made need to be taken into account. This computation requires computing the expectation over the joint distribution $P(\mathcal{X}_{\mathcal{A}'_{1:t}})$, which is exponential in size. This increase in complexity can be avoided using a sampling approximation: Hoeffding's inequality can be used to derive polynomial bounds on sample complexity for approximating the value of information up to arbitrarily small additive error ε, similarly as done in the approach in Chapter 7 [5]. In practice, a small number of samples appears to provide reasonable performance. Secondly, inference itself becomes intractable with an increasing number of sensors. Approximate inference algorithms such as the algorithm proposed by Boyen and Koller [1998] provide a viable way around this problem.

Analysis. Since all sensors maximize the same global objective $L(\mathcal{A}_{1:T})$, the coordinated ascent approach is guaranteed to monotonically increase the global objective with every iteration (ignoring possible errors due to sampling or approximate inference). Hence it must converge (to a local optimum) after a finite number of steps. The procedure is formalized in Algorithm 15.4.

Although we cannot in general provide performance guarantees for the procedure, we are building on an algorithm that provides an optimal schedule for each sensor in isolation, which should benefit from observations provided by the remaining sensors. Also, note that if the sensors are all independent, Algorithm 15.4 will obtain the optimal solution. Even if the sensors are correlated, the obtained solution

[5] An absolute error of at most ε when evaluating each reward $R_{s,t}$ can accumulate to a total error of at most $|\mathcal{T}||\mathcal{S}|\varepsilon$ for all variables and hence to the error of the optimal schedule.

Algorithm 15.4: Multi-Sensor scheduling.

Input: Budget B
Output: Selection $\mathcal{A}_1, \ldots, \mathcal{A}_\ell$ of observation times for each sensor
begin
  Select $\mathcal{A}_i$, $1 \leq i \leq \ell$ at random;
  repeat
    for $i = 1$ **to** ℓ **do**
      Use Algorithm 15.1 to select observations $\mathcal{A}_i$ for sensor i, but conditioning on current sensor scheduling $\mathcal{A}_j$, $j \neq i$, for remaining sensors;
    end
    Compute improvement δ in total expected utility;
  until δ *small enough* ;
end

will be at least as good as the solution obtained when scheduling all sensors independently of each other. Algorithm 15.4 will always converge, and always compute a lower bound on the expected total utility. Considering the intractability of the general problem even for two chains (*c.f.*, Corollary 15.9), these properties are reassuring. In our experiments, the coordinated sensor scheduling performed very well, as discussed in Section 15.6.2.

15.6.2 Proof of concept study on real deployment

In [Singhvi et al., 2005], we presented an approach for optimizing light control in buildings, with the purpose of satisfying building occupants' preferences about lighting conditions, and simultaneously minimizing energy consumption. In our approach, a wireless sensor network is deployed that monitors the building for environmental conditions (such as the sunlight intensity etc.). The sensors feed their measurements to a building controller that actuates the lighting system (lamps, blinds, etc.) accordingly. At every timestep $t \in \mathcal{T}$, the building controller can choose an action that affects the lighting conditions at all locations $\mathcal{S}$ in the building. Utility functions $U_t(a, \mathbf{x}_{\mathcal{S},t})$ are specified that map the chosen actions and the current lighting levels to a utility value. This utility is chosen to capture both users' preferences about light levels, as well as the energy consumption of the lighting system. Details on the utility functions are described in detail in Singhvi et al. [2005].

We evaluated our multi-sensor scheduling approach in a real building controller testbed, as described in detail by Singhvi et al. [2005]. In our experiments, we used Algorithm 15.4 to schedule three sensors, allowing each sensor to choose a subset out of ten time steps (in one-hour intervals during daytime). We varied the number of timesteps during which each sensor is activated, and computed the total energy consumption and total user utility (as defined by Singhvi et al. [2005]). Figure 15.7 shows the mean user utility and energy savings achieved, for a number of observations varying from no observations to continuous sensing. These results imply that using the predictive model and our active sensing strategy, even a very small number of observations achieves results approximately as good as the results achieved by continuous sensing.

Figure 15.8(a) presents the mean total utility achieved using no observations, one observation or ten observations per sensor each day. It can be seen that even a single observation per sensor increases the total utility close to the level achieved by continuous sensing. Figure 15.8(b) shows the mean energy consump-

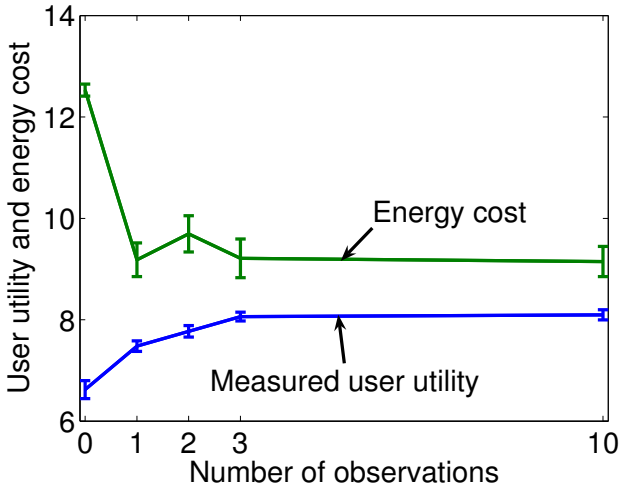
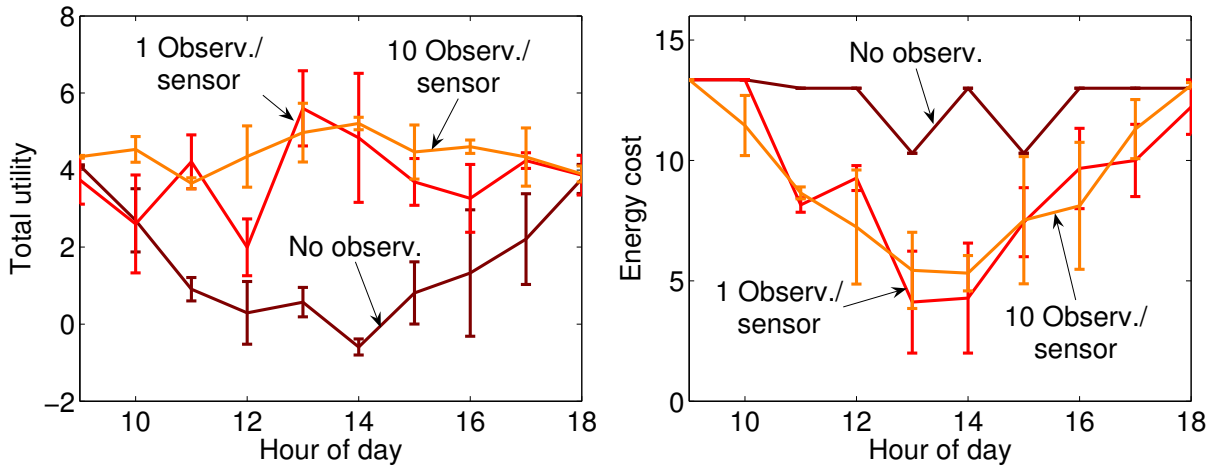


Figure 15.7: Active sensing results: Sensing scheduling evaluation.



(a) *Total utility* (b) *Energy cost*

Figure 15.8: Active sensing results.

tion required for the same experiment. Here, the single sensor observation strategy comes even closer to the power savings achieved for continuous sensing.

Since the sensor network battery lifetime is in general inversely proportional to the amount of power expended for sensing and communication, we conclude that our sensor scheduling strategy promises to lead to drastic increases in sensor network lifetime, deployment permanence and reduced maintenance cost. In our testbed, the network lifetime could be increased by a factor of 3 without significant reduction in user utility and increase in energy cost.

15.7 Related work

In this section, we review related work in a number of different areas.

15.7.1 Experimental design

As discussed in Chapter 3, there is a large related work on Bayesian experimental design. The value of observation problems considered in this chapter can be considered instances of Bayesian experimental design problems, as defined by Lindley and Smith [1972]. Typically, Bayesian Experimental Design is employed for continuous distributions, often the multivariate normal distribution.

Even for multivariate normal distributions, optimal Bayesian Experimental design is $\mathbf{NP}$-hard [Ko et al., 1995]. In some applications of experimental design, the number of experiments to be selected is often large compared to the number of design choices. In these cases, one can find a fractional design (i.e., a non-integral solution defining the proportions by which experiments should be performed), and round the fractional solutions. In the fractional formulation, A-, D-, and E-optimality criteria can be solved exactly using a semi-definite program [Boyd and Vandenberghe, 2004]. There are however no known bounds on the integrality gap, i.e., the loss incurred by this rounding process.

The algorithms presented in Section 15.3.1 can be used to *optimally* solve non-fractional Bayesian Experimental Design problems for chain graphical models, even for continuous distributions, as long as inference in these distributions is tractable (such as normal distributions). This chapter hence provides a new class of combinatorial algorithms for an interesting class of Bayesian experimental design problems.

15.7.2 Value of information in graphical models

Decision-theoretic value of information has been frequently used for principled information gathering [*c.f.*, Heckerman et al., 1993, Howard, 1966, Lindley, 1956], and popularized in decision analysis in the context of influence diagrams [Howard and Matheson, 1984]. In a sense, value of information problems are special cases of Bayesian experimental design problems, where the prior distribution has a particular structure, typically given by a graphical model as considered in this chapter.

Several researchers [Dittmer and Jensen, 1997, Kapoor et al., 2007, Scheffer et al., 2001, van der Gaag and Wessels, 1993] suggested myopic, i.e., greedy approaches for selectively gathering evidence in graphical models, as considered in this chapter. Heckerman *et al.* [1993] propose a method to compute the maximum expected utility for specific sets of observations. While their work considers more general graphical models than this chapter, they provide only large sample guarantees for the evaluation of a given sequence of observations, and use a heuristic without guarantees to select such sequences. Bilgic and Getoor [2007] present a branch and bound approach towards exactly optimizing value of information in more complex probabilistic models. In contrast to the algorithms described in this chapter however, their approach has running time that is worst-case exponential. Munie and Shoham [2008] present algorithms and hardness results for optimizing a special class of value of information objective functions that are motivated by optimal educational testing problems. Radovilsky et al. [2006] recently extended some of our results to obtain approximation algorithms with guarantees in the case of noisy observations (i.e., selecting a subset of the emission variables to observe, rather than selecting among the hidden variables as considered in this chapter).

15.7.3 Bandit problems and exploration / exploitation

As discussed in Chapter 3, an important class of sequential value of information problems is the class of *Bandit problems*, containing, e.g., the classical k-armed bandit problem, as formalized by Robbins [1952].

A celebrated result by Gittins and Jones [1979] shows that for a fixed number of draws, an optimal strategy can be computed in polynomial time, using a dynamic programming based algorithm. While similar in the sense that an optimal sequential strategy can be computed in polynomial time, Gittins algorithm however has different structure from the dynamic programming algorithms presented in this chapter.

Note that using the "function optimization" objective function described in Section 15.1.1, our approach can be used to solve a particular instance of bandit problems, where the arms are not required to be independent, but, in contrary to the classical notion of bandit problems, can not be chosen repeatedly.

15.7.4 Relationship to machine learning

Decision Trees [Quinlan, 1986] popularized the value of information as a criterion for creating conditional plans. Unfortunately, there are no guarantees on the performance of this greedy method.

The subset selection problem as an instance of feature selection is a central issue in machine learning, with a vast amount of literature (see Molina et al. [2002] for a survey). However, we are not aware of any work providing similarly strong performance guarantees than the algorithms considered in this chapter.

The problem of choosing observations also has a strong connection to the field of active learning [*c.f.*, Cohn et al., 1996, Tong and Koller, 2001] in which the learning system designs experiments based on its observations. While sample complexity bounds have been derived for some active learning problems [*c.f.*, Balcan et al., 2006, Dasgupta, 2005], we are not aware of any active learning algorithms that perform provably optimal (even for restricted classes of problem instances).

15.8 Summary

We have described novel efficient algorithms for optimal subset selection and conditional plan computation in chain graphical models (and trees with few leaves), including HMMs. Our empirical evaluation indicates that these algorithms can improve upon commonly used heuristics for decreasing expected uncertainty. Our algorithms can also effectively enhance performance in interactive structured classification tasks.

Unfortunately, the optimization problems become wildly intractable for even a slight generalization of chains. We presented surprising theoretical limits, which indicate that even the class of decision theoretic value of information functions (as widely used, e.g., in influence diagrams and POMDPs) cannot be efficiently computed even in Naive Bayes models. We also identified optimization of value of information as a new class of problems that are intractable ($\mathbf{NP}^{\mathbf{PP}}$-complete) for polytrees.

Our hardness results, along with other recent results for polytree graphical models, the $\mathbf{NP}$-completeness of maximum a posteriori assignment [Park and Darwiche, 2004] and $\mathbf{NP}$-hardness of inference in conditional linear Gaussian models [Lerner and Parr, 2001], suggest the possibility of developing a generalized complexity characterization of problems that are hard in polytree graphical models.

In light of these theoretical limits for computing optimal solutions, it is a natural question to ask whether approximation algorithms with non-trivial performance guarantees can be found, even if the objective functions are not submodular.

Part V

Open Problems and Conclusions

Chapter 16

Open Problems

In this chapter, we propose some open questions which we consider promising directions for future work.

16.1 The simultaneous placement and tasking problem

In many sensor placement problems, we want to simultaneously solve multiple sensing tasks with the deployed sensors. For example, we want to both decide, where to place sensors, and when to poll them, in order to maximize battery lifetime. In such problems, we want to place m disjoint sets of sensors, $\mathcal{A}_1, \dots, \mathcal{A}_m$, which we each assign to some task T_i (e.g., prediction in timeslice i). We assume that we can quantify the performance of placement $\mathcal{A}_i$ in task T_i by some nondecreasing submodular function $F_i(\mathcal{A}_i)$. Our goal is then to maximize

$$\max_{\mathcal{A}} \min_i F_i(\mathcal{A}_i) \text{ subject to } \mathcal{A} = \bigcup_i \mathcal{A}_i, \mathcal{A}_i \cap \mathcal{A}_j = \emptyset, |\mathcal{A}| \leq k.$$

Hence, in order to solve this problem, we must both reason about the set $\mathcal{A}$ of at most k sensors to purchase *and* the partitioning of this set $\mathcal{A}$ into subsets $\mathcal{A}_i$.

The special case where $k = |\mathcal{V}|$ (i.e., we only need to find the optimal partitioning without considering the budget constraint) has been previously considered in the literature [*c.f.*, Ponnuswami and Khot, 2007].

Further note that the special case where all the F_i are equal is exactly the SPASS problem considered in Chapter 12. However, our ESPASS algorithm does not extend to this more general setting, as the reallocation procedure cannot be applied any longer.

The SATURATE algorithm can be applied to the placement and tasking problem. However, its application requires to relax the constraint that each sensor location $s \in \mathcal{V}$ can be selected only once.

16.2 Supermodular cost functions

Previously, we have mainly considered optimization problems with additive (modular) cost functions. In Chapter 11, we developed an algorithm which can optimize a submodular objective function with a particular *submodular* cost function, namely one that corresponds to the cost of connecting a set of nodes $\mathcal{A}$ as cheaply as possible in a graph, whose edges represent communication cost.

However, several practical cost functions are *supermodular*. For example, in the case of optimizing mutual information in GP, the *computational cost* associated with inference is cubic in the number of chosen locations. Hence, if we regard computational cost as the cost function, then the cost of a set $\mathcal{A}$ of observations is given by $C(\mathcal{A}) = |\mathcal{A}|^3$, or more generally as $C(\mathcal{A}) = g(|\mathcal{A}|)$ for some convex nondecreasing function g. We propose to analyze observation selection problems with such a supermodular cost function.

In such problems, the scalarized objective $F(\mathcal{A}) = F(\mathcal{A}) - \lambda C(\mathcal{A})$ is submodular, but not necessarily nondecreasing. We will investigate, to which extent recent developments in optimizing non-monotone submodular functions [Feige et al., 2007] can be applied to observation selection problems with supermodular cost.

16.3 Relationship to probabilistic planning

All the observation selection problems analyzed in this thesis can be modeled as a probabilistic planning problem using the concept of Partially Observable Markov Decision Processes (POMDP). While solving general POMDP problems is extremely hard (some formulations even being undecidable), the algorithms presented in this thesis effectively and near-optimally approximate the solution to an interesting subclass of POMDPs. It would be a very interesting question to investigate, how far combinatorial algorithms of the same spirit as those considered in this thesis can be used to solve more complex classes of POMDPs. Equivalently: Is there any interesting combinatorial structure in certain POMDPs, which can be exploited?

In [Singh et al., 2007], we considered the problem of planning informative paths for observation selection with multiple robots in Gaussian Processes. So far, we did not consider settings where there is *action noise*, i.e., where movement actions have uncertain outcome. Possible applications of such an approach would be active exploration problems such as Active SLAM (Simultaneous Localization and Mapping).

In general, observation selection problems imply the fundamental restriction that the state of the world is ever only (partially) observed, but never affected and changed. An extremely interesting question would be to study whether some of the techniques for observation selection generalize to problems where the state of the world is affected. For example, in the water distribution network monitoring application, one might not just want to place sensors to measure the water quality accurately over the network, but also place some treatment units.

16.4 Transfer learning and predicting informativeness

Another interesting question is, whether observations made in one observation selection problem can be transferred to another problem. For example, if we use the PSPIEL approach described in Chapter 11 in

order to place sensors in a building, we need a pilot deployment, in order to acquire data which can be used to predict the link qualities between arbitrary pairs of locations, as well as the informativeness of a sensor placed at an arbitrary location.

Such a pilot deployment can be quite costly. If multiple sensor networks have to be deployed (e.g., in different floors of a building, or multiple water distribution networks are deployed in different cities) it would be more cost-effective to perform only a single pilot deployment, and reuse the obtained knowledge for each subsequent deployment task. It would be very interesting to develop approaches which would allow such a knowledge transfer. Such an approach could build on transfer learning techniques for Gaussian Processes [Yu et al., 2006], but also try to explicitly predict the informativeness of sensors based, e.g., on features of the environment (e.g., distance to walls, etc.).

This question also requires to consider an exploration–exploitation tradeoff — it might be more cost-effective to perform a larger pilot deployment, the higher cost of which would amortize through the possible reuse of obtained information (estimated parameters etc.) in later deployments.

16.5 Other resource constraints for observation selection

In this thesis, we consider a variety of realistic constraints, modeled as cost functions $C(\mathcal{A})$ on the set of chosen observations. When deploying sensors networks for example, there are a variety of different notions of cost, such as deployment, hardware, communication cost and power consumption.

However, in other applications, very different notions of cost can arise. For example, in an information retrieval or activity recognition context, where observations correspond to labels obtained by a human expert, very different notions of cost arise. For example, the cost of an observation can depend on the state of the world. Consider an application where sensor data is used to recognize the current activity of a person. In such an application, an observation selection strategy might occasionally query the user to label their current activity. However, if the person is very busy, a query can be more expensive than when the person is idle. Such a dependency of the cost on the state of the world can be modeled by extending the definition of the cost function as

$$C(\mathcal{A}) = \int P(\mathbf{x}_\mathcal{V}) c(\mathbf{x}_\mathcal{V}, \mathcal{A}) d\mathbf{x}_\mathcal{V},$$

i.e., by introducing a notion of expected cost (similar to the expected sensing quality). However, when specifying a budget on the expected cost, the actual cost incurred can possibly violate this budget. Depending on the application, such a violation can be not acceptable.

Another interesting class of constraints arises in the context of privacy. Consider an online service, which can choose to acquire personal information about a user in order to improve its performance (e.g., by personalization). In such applications, after sharing a single bit (e.g., the gender of a person), a user is still indistinguishable from many other users. However, the more information is shared, the more privacy is threatened. This loss in indiscriminability can potentially be quantified as another type of cost metric, leading to new classes of optimization problems. In such cases, the cost also depends on the state of the world (i.e., the particular personal attributes of a user) – particular combinations of private information can be more threatening than others.

16.6 Observation selection for symbolic reasoning

In this Thesis, we have mainly considered probabilistic prediction problems, where the unobserved variables are related by some (often smooth) process with the observed variables. In many practical applications, a decision depends on whether some particular logical statements about the world are true or false. Observations can be made in order to establish the validity of some other facts, which are in a deterministic, logical relationship with each other. Prior work has considered probabilistic models for describing such logical / symbolic relationships, such as Markov Logic Networks [Richardson and Domingos, 2006]. In such logical contexts, we expect other problem structure to be present than in the problems we consider in this Thesis. The XOR example presented in Chapter 7 indicates that we generally cannot expect submodularity in such contexts, as the validity of a statement can potentially require the knowledge about a collection of facts which have to be established. Nevertheless, it would be very interesting to consider if other problem structure can be exploited in such problems.

16.7 Online optimization, dynamic phenomena

In many observation selection problems, the monitored phenomenon evolves dynamically over time, and at every point in time one has to decide which observations to acquire. In cases, where the changes are predictable, and, e.g., a probabilistic spatiotemporal model is available, techniques such as those presented by Meliou et al. [2007] can be used for optimizing observation selection.

In some cases however, no such model is available (or is only being estimated and hence changing over time). In such problems, the objective functions quantifying informativeness change over time, and observations have to be selected which are robust against such changes. There are some recent advances indicating that near-optimally optimizing submodular functions is possible in such an online context [Golovin and Streeter, 2008]. Nevertheless, it is not clear how such problems can be solved, e.g., in case more complex constraints (communication / path constraints) are present.

16.8 High dimensional and continuous spaces

In this Thesis, we mainly considered problems where the number of possible observations is large, but still tractable (e.g., the greedy algorithm can iterate over all possible choices of observations to add). In some practical problems, the dimensionality is so high that fine-granular discretization is not feasible. In such problems, other techniques would have to be developed for selecting observations. As we have seen in this Thesis, the analysis of algorithms such as greedy algorithm can be applied even if the algorithm is not executed *exactly* (i.e., the greedy algorithm does not necessarily select the *best* next element, but a *good* next element). One possible approach towards observation selection in high dimensional spaces would be to consider techniques from continuous optimization (such as conjugate gradient etc.). However, observation selection problems are typically non-convex (due to multimodality caused by symmetries in the objective functions). An interesting open question is, in how far techniques for analyzing the *discrete* optimization problems considered in this Thesis can be used to get guarantees for the corresponding continuous optimization problems.

Chapter 17

Conclusions

In this Thesis, we studied the fundamental question

> Which observations should we make, in order to acquire the most useful information as cost-effectively as possible?

Most previous approaches that addressed this question have relied upon myopic heuristics, i.e., approaches which only consider the next best observation to add, without planning ahead for future observation possibilities. Those approaches typically have no performance guarantees. In security critical applications, such as protecting drinking water from contamination (as considered in this Thesis), such myopic approaches perform very poorly. Existing nonmyopic approaches have involved computationally intensive techniques which are very difficult to scale to larger problems.

In this Thesis, we presented a novel class of approaches for sensing and information gathering, using techniques from combinatorial optimization. Our algorithms rely on structural properties, such as submodularity, locality and conditional independence.

17.1 Summary

In this section, we will briefly summarize the key contributions presented in this Thesis.

17.1.1 Submodular sensing problems

We showed that many observation selection objectives satisfy *submodularity*, an intuitive diminishing returns property – adding a sensor to a small deployment helps more than adding it to a large deployment. The example we studied in detail included mutual information for spatial prediction, selecting informative variables in graphical models, and placing sensors for outbreak detection. Our approaches systematically exploit this submodularity property to *efficiently* achieve provably *near-optimal* solutions. For example, for submodular functions, the greedy algorithm, which iteratively adds the observation that increases the sensing quality the most, produces a solution that obtains at least a constant fraction of 63% of the optimal value, which we showed to be optimal for the problem of optimizing information gain. We also exploited lazy evaluation to drastically speed up our algorithms, allowing us, for example, to speed up our solution

to the problem of placing sensors in metropolitan area sized water distribution networks from 30 hours to 1 hour, and improving the efficiency of selecting informative weblogs by a factor of 700. We also used submodularity to compute tight, data-dependent bounds proving that the solutions obtained by our algorithms are typically within 90% of optimal.

17.1.2 Non-greedy algorithms for complex sensing problems

While the greedy algorithm of Nemhauser et al. [1978] applies to the problem of selecting a near-optimal set of k elements, in Part II of this Thesis we saw that this result does not generalize for more complex sensing problems. This insight necessitated the development of non-greedy algorithms to address these problems.

We first studied the problem of *robust sensing*, where the chosen observations need to be robust against sensor failure or uncertainty about model parameters. We showed, that many such problems require the optimization of an adversarially chosen submodular objective function, and we developed an algorithm, SATURATE, which provides best possible guarantees for this setting. We empirically demonstrated that SATURATE performs well when compared to a highly fine-tuned, specialized simulated annealing algorithm.

We also considered the problem of selecting observations subject to complex combinatorial cost functions. For example, when placing wireless sensor networks, the chosen locations need to be not only very informative, but also allow efficient communication. When using robots for making observations, the chosen locations need to lie on a collection of paths. We developed PSPIEL, an efficient algorithm which finds solutions which near-optimally solve this trade-off. Our algorithm exploits *locality*, a structural property present in many sensing problems.

We also considered the problem of deploying wireless sensor networks under battery lifetime constraints: Since every measurement draws power, sensors can typically be activated only a fraction of the time. Hence, the problem of scheduling the sensors becomes of crucial importance. In this Thesis, we presented ESPASS, an efficient algorithm for simultaneously optimizing the placement and scheduling. We showed that this simultaneous approach leads to drastic improvements in network lifetime when compared to the traditional, stage-wise approach.

17.1.3 Sequential sensing problems

A key question in many observation selection problems is how much better a sequential algorithm, which decides on the next observation based on previous observations, can perform when compared to the best fixed set of observations chosen a priori. We present a partial answer to the question for spatial prediction in Gaussian Processes. We develop a theoretical bound quantifying the gap between the best sequential and a priori selections, and use it to develop an exploration–exploitation approach for active learning in Gaussian Processes.

Lastly, we look beyond submodular observation selection problems, and consider the problem of selecting *optimal* observations in graphical models. We show that in chain-graphical (Markovian) models, it is possible to efficiently find the optimal sequential policy. However, even for slightly larger classes of graphical models, we prove strong hardness results.

17.1.4 Applications

In addition to providing algorithms and theoretical analyses, we present extensive empirical evaluation of our approaches on several real-world problems.

A major focus of this Thesis were spatial monitoring problems. We studied the problem of deploying wireless sensor networks for building and traffic monitoring, as well as environmental monitoring using mobile robots.

We also used our algorithm to place sensors on a chair for posture recognition. In this study, we achieve a 30-fold reduction in hardware cost without major loss of prediction accuracy.

We participated in the international *Battle of the Water Sensor Networks* challenge where the goal was to optimize sensor placements for securing drinking water distribution networks against contamination. In this challenge, our algorithms obtained the top score.

We also considered problems of information gathering on the web. We used our algorithms to select influential, informative weblogs, that detect the largest information cascades as early as possible. The results of this research created excitement in the blogosphere, resulting in 30,000 visits to our project website `http://www.blogcascades.org`, as well as coverage in popular media.

17.1.5 Summary of key contributions

In the technical Parts II, III and IV of this Thesis, we presented the following contributions.

1. We proved that several important sensing quality functions satisfy *submodulariy*. These include
   - mutual information for spatial prediction in Gaussian Processes [Krause et al., 2008b],
   - information gain with respect to a set of target variables in graphical models and [Krause and Guestrin, 2005b]
   - outbreak detection in networks [Krause et al., 2008a, Leskovec et al., 2007b].
2. We showed how this submodularity can be exploited [Krause et al., 2008b] to
   - obtain provably near-optimal solutions for selecting the k most informative sensing locations, by invoking powerful existing results about optimizing submodular functions,
   - speed up algorithms using lazy evaluations and
   - compute novel tight, data dependent-bounds.
3. We considered complex optimization tasks arising in sensing problems. In all these problems, existing greedy approaches perform arbitrarily badly. We developed novel, efficient approximation algorithms:
   - SATURATE for solving robust sensing problems [Krause et al., 2009],
   - PSPIEL for optimizing sensor placements under complex cost functions and [Krause et al., 2006a]
   - ESPASS for simultaneously optimizing sensor placements and schedules.
4. We developed novel nonmyopic algorithms for sequential sensing problems, the

- EEGP algorithm for active learning for spatial prediction in Gaussian Processes [Krause and Guestrin, 2007] and
- VOIDP algorithm for optimal conditional planning in chain graphical models [Krause and Guestrin, 2005a].

5. We proved hardness of sensing problems. Under reasonable complexity theoretic assumptions,
   - optimizing information gain is hard to approximate within a factor of $(1 - 1/e)$ even for Naive Bayes models. This bound matches the performance of the greedy algorithm for this problem [Krause and Guestrin, 2005b],
   - optimizing robust sensing cannot be approximated by any factor unless $\mathbf{P} = \mathbf{NP}$. Our algorithm, SATURATE, provides the best possible bicriterion guarantees (allowing relaxation of certain constraints) for this problem [Krause et al., 2009],
   - exact optimization of value of information is $\#\mathbf{P}$-hard for Naive Bayes models and $\mathbf{NP}^{\mathbf{PP}}$-hard for discrete polytrees [Krause and Guestrin, 2005a].
6. We demonstrated the performance of our algorithms on several real-world case studies, including
   - sensor placement for spatial prediction in Gaussian Processes, demonstrated on real-world sensing data sets and a proof-of-concept case study of a deployed sensor network [Krause et al., 2006a, 2008b],
   - sensor placement for posture recognition on a sensing chair, where our algorithms obtain 30-fold reduction in hardware cost without significant reduction in prediction accuracy [Mutlu et al., 2007],
   - sensor placement for outbreak detection in a metropolitan-area drinking water distribution network, where our algorithms obtained a top score in an international competition [Krause et al., 2008a],
   - sensor scheduling for improving the lifetime of wireless traffic sensors, and for light sensors in a building management prototype testbed [Singhvi et al., 2005],
   - selecting informative weblogs to read on the Internet, showing that our algorithms scale to a blog data set with 45,000 blogs and millions of postings [Leskovec et al., 2007b].

The references indicate the publications that this thesis and its contributions are based upon.

17.2 Further research plans

My long-term goal is to continue to push at the frontier of our knowledge about ideal sensing and action, especially in situations where there is an opportunity to collect immense amounts of uncertain data. I plan to use these insights in order to enable new classes of systems and services for science and society.

Massive volumes of media are now generated from corporate and public sources every second. In addition, more and more *low level sensor* devices are becoming available and accessible. Examples include GPS devices in cars, cameras, microphones and accelerometers in mobile phones, personal weather stations (`http://www.wunderground.com/`), etc. Numerous applications could leverage the integration of these different forms of data. We already performed an investigation highlighting the potential benefits of such applications [Krause et al., 2008c].

I will be pursuing a set of interrelated questions: How can we build systems and services which optimally harness these massive amounts of highly uncertain, heterogenous, seemingly disparate data? How can we determine, which data we need to access in order to solve a given task at hand? How can we make use of a wealth of personal information, without violating users' privacy? Can we obtain principled, theoretically well-founded answers to these problems? Answering these questions requires solving difficult challenges, bringing together robust probabilistic modeling and reasoning, decision theory, machine learning, sensor networks, and other systems areas. In order to make progress on this long-term vision and these challenges, we need to identify intermediate steps and directions along the way, some of which I outline in the following.

17.2.1 Identifying structure in sensing and information acquisition.

When combining and interpreting data from heterogenous sources, we have to integrate more complex data types and dependencies than those arising from spatial monitoring problems: Information must be combined from such disparate sources as noisy sensor data, semi-structured text documents and symbolic knowledge bases. Submodular structure might not be present in such problems. Thus, we may need to seek out and leverage other structure such as locality, logical relationships, and specific patterns of conditional or context-specific independence. I believe there are rich opportunities in studying the presence and uses of different forms of structure for solving important and complex information-acquisition tasks.

17.2.2 New sensing architectures and applications.

When reasoning about data from diverse, non-centrally owned sensors, we need to develop appropriate abstractions for reasoning about low level sensor data, as well as providing facilities for negotiating access to shared sensor data. We also need to develop and prototype applications which allow us to concretely study our information acquisition questions and obtain results for quantitative evaluation. I am interested in both sensing- and control-centric applications (such as applications of sensor-actuator networks in civil and environmental engineering applications, supported by higher-order, symbolic reasoning about the task to be solved) as well as in new classes of online services.

17.2.3 Privacy and attention as fundamental constraints.

Dealing with information acquisition and presentation involving personal (sensor and other) data requires solving challenging optimization problems. For example, when presenting acquired information (such as lists of informative weblogs as we considered in previous work) to users, only a short attention span is available, and we need to maximize the amount of relevant information conveyed. Furthermore, as we consider accessing personal data such as sensor data from body-worn sensors like cellphones, demographic data or traces of interactions with online services, privacy concerns become increasingly imminent. Hence, we would like to minimize the amount of personal information acquired about users while maximizing the utility of the provided service. We have performed investigations using techniques from observation selection to address the utility-privacy tradeoff in personalized online services, with very promising results [Krause and Horvitz, 2008].

I believe that these directions pose great opportunities for scholarly study as well as for creating methods and artifacts that will have tremendous real-world influence with multiple benefits to our society. Even

though these goals are challenging, I am passionate about innovating as a leader on a long-term research program focused on finding new ways of coping with ever-increasing amounts of data, and enabling effective sensing, decision making, and decision support.

Appendix A

Review: Learning Gaussian Processes

In this section, we give a brief overview of learning parameters of a Gaussian Process prior.

A.1 Learning stationary GPs

Often, for sake of simplicity and model complexity, the kernel function $\mathcal{K}(u, v)$ is chosen from a parametric family, selected in terms of parameters θ. A requirement for a valid kernel function is that it is always positive-definite. For example, the square-exponential kernel

$$\mathcal{K}(u, v) = \sigma^2 \exp\left(\frac{||u - v||^2}{h^2}\right)$$

has two parameters, $\theta = \{\sigma, h\}$. Other popular choices of parametric kernel functions are the exponential kernel, and the Matérn [Abrahamsen, 1997] class of covariance functions, which allow to explicitly control the differentiability of the sample functions.

Learning a Gaussian process refers to the estimation of θ from training samples, each consisting of the value of the field at an index set. Given independent training samples, learning a Gaussian process in either a maximum likelihood or Bayesian framework is possible [Rasmussen and Williams, 2006, ch. 5]. In the geostatistics literature, it is common to learn the parameters by least squares fit of the empirical variogram $2\gamma(\mathbf{h}) = \mathbb{E}[(\mathcal{X}_{\mathbf{s}+\mathbf{h}} - \mathcal{X}_{\mathbf{s}})^2])$ [*c.f.*, Cressie, 1991]. Another possibility, if multiple independent samples of the process are available, is to use cross-validation.

Multiple training samples are often noisy replicates of the field over time. The training data are not independent, and the log-likelihood does not decompose over the data. Sometimes, only a single sample of a process is available, which can be observed at different locations. However, if the time between training samples is sufficiently large, and one assumes temporal stationarity (i.e., covariance between sensing locations does not vary with time), then treating the data as independent may suffice.

A.2 Overview of nonstationarity

A Gaussian process is (weakly) stationary if it has constant mean, finite variance, and kernel function $\mathcal{K}(u, v)$ that depends only on the difference vector $u - v$. If we further assume that the kernel function only depends on the norm of the difference vector, then the process is isotropic.

In practice, stationarity is frequently violated, even once a mean trend has been subtracted from the data. Experiments in Section 6.6.2 illustrate the need for nonstationary models. Unlike the stationary case, there is no consensus on how to represent and learn nonstationary models.

Capturing complex nonstationary behavior as exhibited by the phenomena we are interested in (*c.f.*, Figure 6.2(a) and Figure 6.2(b)) is a difficult problem. The simplest nonstationary model for finite GPs (i.e., multivariate normal distributions) is the empirical covariance

$$S = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{x})(x_i - \bar{x})^T \in \mathbb{R}^{p \times p}$$

which requires that $n > p$ for S to be non-singular. Furthermore the efficiency of this estimator is poor in high dimensions. Shrinking the eigenvalues of S [Ledoit and Wolf, 2004, Stein, 1975] or shrinking the estimator towards a simpler full-rank matrix [Daniels and Kass, 2001] addresses these problems. In low dimensions, this problem can be treated as an instance of finding a positive semidefinite Toeplitz matrix that is closest to the empirical covariance in Frobenius norm [Higham, 2002]. In the sensor placement problem however, the covariance of at most a small set of initial locations is known, from observing a pilot deployment. It is crucial to be able to predict covariances for pairs of locations, at which no sensor has been placed. In this setting, several approaches have been proposed, mostly based on *deformation*, *convolution* or *partitioning*.

An intuitively appealing approach for modeling nonstationarity is spatial *deformation*, which is similar in spirit to the nonlinear feature maps used, e.g., in Support Vector Machines [Vapnik, 1995]. Spatial *deformation* maps nonstationary data, through a nonlinear function onto another space which is stationary. The main difficulty is determining an appropriate mapping. The pioneering approach by Sampson and Guttorp [1992] involves mapping the data onto a two dimensional space using nonmetric multidimensional scaling, and then warping the rest of the space using thin-plate splines. In a Bayesian approach, Snelson et al. [2003] suggest to model the mapping itself as a Gaussian process and learn it simultaneously with the process on the data. In our experiments with the data sets discussed in Section 6.6.1, we were not able to find appropriate good transformations of the process.

Nonstationary GPs can be defined as the *convolution* of a positive-definite function $f(s)$ at each point in the space with a white noise process. If the bandwidth of a Gaussian basis f is allowed to vary with its position, this yields the model of Gibbs [1997], and with fewer restrictions, [Higdon et al., 1998]. These results were generalized to arbitrary isotropic f by Paciorek [2003] and subsequently by Stein [2005]. While this approach is theoretically very elegant, as it for example allows to vary the differentiability of the process with space, the difficulties are to determine the spatially varying functions $f(s)$. Paciorek [2003] suggests to equip these functions with a GP prior.

In the *partitioning* approach, the space is partitioned into a disjoint collection of cells, each of which is modeled by a stationary Gaussian process [Gramacy, 2005, Kim et al., 2005]. The method by Gramacy [2005] recursively partitions the space, and ties the parameters of the local stationary processes in a hierarchical model. The method of Kim et al. [2005] simultaneously infers a Voronoi partition and the

models in each region. More generally, nonstationary behavior can be modeled using mixtures of Gaussian processes, in which each component has a different kernel function [Tresp, 2000]. A related approach by Rasmussen and Ghahramani [2002] does not require a fixed number of mixture components. These models however are not GPs anymore, and the predictive distributions (6.1) and (6.2) can no longer be computed in closed form.

A.3 Learning nonstationary kernels by extrapolating estimated covariances

In this section we review the method we used in Chapters 6 and 11 for learning nonstationary GPs [Nott and Dunsmuir, 2002]. For clarity, we limit the presentation to zero mean processes. The approach of Nott and Dunsmuir [2002] assumes that data from an initial deployment is available. More specifically, the assumption is that an estimate of the empirical covariance $\Sigma_{\mathcal{A}\mathcal{A}}$ at a set of observed locations is available, and that the process can be locally described by a collection of isotropic processes, associated with a set of reference points.

More formally, define a set of reference points $\mathcal{A} = \{\mathbf{z}_1, \mathbf{z}_2, \ldots, \mathbf{z}_m\}$, $m = |\mathcal{A}|$, and associate a zero mean Gaussian Process $\mathcal{W}_i$ with each of these reference points, with isotropic covariance $R_i(\mathbf{s}, \mathbf{t}) = \mathcal{K}_i(|\mathbf{s} - \mathbf{t}|_2)$. Let $C_i = \mathbb{E}[\mathcal{W}_i(\mathbf{z}_j)\mathcal{W}_i(\mathbf{z}_k)]_{j,k}$ be the covariance at the initial deployment $\mathcal{A}$, and let $c_i(\mathbf{s}) = (R_i(\mathbf{s}, \mathbf{z}_1), \ldots, R_i(\mathbf{s}, \mathbf{z}_m))$ the cross-covariances of the (arbitrary) location $\mathbf{s}$ and the initial deployment. Then the posterior GP $\mathcal{W}_i \mid [\mathcal{W}_i(\mathcal{A}) = \mathbf{a}]$ has the form

$$\mathcal{W}_i(\mathbf{s}) \mid [W_i(\mathcal{A}) = \mathbf{a}] = c_i(\mathbf{s})C_i^{-1}w + \delta_i(\mathbf{s}),$$

where δ_i, the residual variation, is a nonstationary, zero mean GP with covariance

$$R_{\delta_i}(\mathbf{s}, \mathbf{t}) = \mathcal{K}_i(|\mathbf{s} - \mathbf{t}|_2) - c_i(\mathbf{s})C_i^{-1}c_i(\mathbf{t}).$$

If the kernel functions R_i have been estimated from the sensors of the initial deployment which are close to $\mathbf{z}_i$, the hope is that the conditional process $W_i \mid [W_i(\mathcal{A}) = \mathbf{a}]$ correctly describes the behavior of the global GP, locally around the location $\mathbf{z}_i$. The idea now is to define the global GP as a weighted sum of the local conditional GPs, "synchronized" by their measurements at $\mathcal{A}$. More formally, define a zero-mean, multivariate normal distribution W^* with covariance $\Sigma_{\mathcal{A}\mathcal{A}}$, and let

$$Z_i(\mathbf{s}) = c_i(\mathbf{s})C_i^{-1}W^*.$$

By construction, W^* is uncorrelated with each δ_i. We define the global GP as a weighted combination of the Z_i and the δ_i, namely

$$Z(\mathbf{s}) = \sum_i \pi_i(\mathbf{s})Z_i(\mathbf{s}) + \pi_i(\mathbf{s})^{\frac{1}{2}}\delta_i(\mathbf{s}).$$

Hereby, $\pi_i(\mathbf{s})$ is a mixture weight of process W_i at location $\mathbf{s}$, and for all $\mathbf{s}$, $\sum_i \pi_i(\mathbf{s}) = 1$. Nott and Dunsmuir [2002] show that the covariance of Z takes the following form:

$$R_Z(\mathbf{s}, \mathbf{t}) = \sum_{i,j} \pi_i(\mathbf{s})\pi_j(\mathbf{t})c_i(\mathbf{s})C_i^{-1}\Sigma_{\mathcal{A}\mathcal{A}}C_j^{-1}c_j(\mathbf{t}) + \sum_i \pi_i(\mathbf{s})^{\frac{1}{2}}\pi_i(\mathbf{t})^{\frac{1}{2}}R_{\delta_i}(\mathbf{s}, \mathbf{t}).$$

As explained by Nott and Dunsmuir [2002], the reason for the particular choice of Z is the following property: Conditional on observing $\mathcal{A} = \mathbf{a}$, for the predictive distributions of the process it holds that

$$\mu_{\mathbf{s}|\mathcal{A}=\mathbf{a}} = \sum_i \pi_i(\mathbf{s})c_i^T C_i^{-1}\mathbf{a},$$

and

$$\sigma^2_{\mathbf{s}|\mathcal{A}=\mathbf{a}} = \sum_i \pi_i(\mathbf{s}) R_{\delta_i}(\mathbf{s},\mathbf{s}) = \sum_i \pi_i(\mathbf{s})[\mathcal{K}_i(0) - c_i(\mathbf{s})^T C_i^{-1} c_i(\mathbf{s})],$$

hence both the predictive mean and variance of the process Z are spatial averages over the means and variances of the predictive distributions of the local processes Z_i. The process Z observes the covariance $\Sigma_{\mathcal{A}\mathcal{A}}$ at the initial deployment, and locally behaves similarly to the local processes Z_i.

The weights π_i are chosen similar fashion to interpolation weights in kernel regression, and the local processes Z_i are estimated using least squares fit of the empirical variogram, or by maximizing the marginal likelihood of the data, only taking into account the sensors from the initial deployment which are closest by. For more details on estimation please refer to Nott and Dunsmuir [2002].

Appendix B

Proofs

B.1 Proofs from Chapter 5

Proof of Theorem 5.4. The following proof is an extension of the proof by Nemhauser et al. [1978], using some simplifications by Jon Kleinberg.

Let $s_1, \ldots, s_k$ be the locations selected by the greedy algorithm. Let $\mathcal{A}_i = \{s_1, \ldots, s_i\}$, $\mathcal{A}^*$ be the optimal solution, and $\delta_i = F(\mathcal{A}_i) - F(\mathcal{A}_{i-1})$. By monotonicity, we have, for all $1 \leq i \leq k$,

$$F(\mathcal{A}_i \cup \mathcal{A}^*) \geq F(\mathcal{A}^*).$$

By (α, ε)-approximate evaluation of the greedy rule, We also have, for $0 \leq i < k$,

$$F(\mathcal{A}_i \cup \mathcal{A}^*) \leq F(\mathcal{A}_i) + (\alpha k)\delta_{i+1} + k\varepsilon = \sum_{j=1}^{i} \delta_j + (\alpha k)\delta_{i+1} + k\varepsilon.$$

Hence we have the following sequence of inequalities:

$$\begin{aligned} F(\mathcal{A}^*) - k\varepsilon &\leq (\alpha k)\delta_1 \\ F(\mathcal{A}^*) - k\varepsilon &\leq \delta_1 + (\alpha k)\delta_2 \\ &\vdots \\ F(\mathcal{A}^*) - k\varepsilon &\leq \sum_{j=1}^{k-1} \delta_j + (\alpha k)\delta_k \end{aligned}$$

Now we multiply both sides of the i-th inequality by $\left(1 - \frac{1}{\alpha k}\right)^{k-i}$, and add all inequalities up. After cancelation, we get

$$\left(\sum_{i=0}^{k-1} \left(1 - 1/(\alpha k)\right)^i\right) \left(F(\mathcal{A}^*) - k\varepsilon\right) \leq (\alpha k) \sum_{i=1}^{k} \delta_i = (\alpha k) F(\mathcal{A}_k).$$

Hence, as claimed, with $\mathcal{A}_G = \mathcal{A}_k$ (i.e., $\mathcal{A}_G$ is the k-element greedy solution)

$$F(\mathcal{A}_G) \geq \left(1 - \left(1 - 1/(\alpha k)\right)^k\right) \left(F(\mathcal{A}^*) - k\varepsilon\right) \geq \left(1 - \frac{1}{e^{1/\alpha}}\right)\left(F(\mathcal{A}^*) - k\varepsilon\right).$$

□

To prove Theorem 5.6, we need two lemmas:

Lemma B.1 (generalized from Khuller et al. [1999]). *For $i = 1, \ldots, l+1$, it holds that*

$$F(\mathcal{G}_i) - F(\mathcal{G}_{i-1}) \geq \frac{c(X_i)}{\alpha L}(F(OPT) - F(\mathcal{G}_{i-1})) - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right)$$

Proof. Using nondecreasingness of F, we have

$$F(OPT) - F(\mathcal{G}_{i-1}) \leq F(OPT \cup \mathcal{G}_{i-1}) - F(\mathcal{G}_{i-1}) = F(OPT \setminus \mathcal{G}_{i-1} \cup \mathcal{G}_{i-1}) - F(\mathcal{G}_{i-1})$$

Assume $OPT \setminus \mathcal{G}_{i-1} = \{Y_1, \ldots, Y_m\}$, and let for $j = 1, \ldots, m$

$$Z_j = F(\mathcal{G}_{i-1} \cup \{Y_1, \ldots, Y_j\}) - F(\mathcal{G}_{i-1} \cup \{Y_1, \ldots, Y_{j-1}\}).$$

Then $F(OPT) - F(\mathcal{G}_{i-1}) \leq \sum_{j=1}^{m} Z_j$.

Now notice that

$$\frac{Z_j - \varepsilon}{\alpha c(Y_j)} \leq \frac{F(\mathcal{G}_{i-1} \cup Y_j) - F(\mathcal{G}_{i-1}) - \varepsilon}{\alpha c(Y_j)} \leq \frac{F(\mathcal{G}_i) - F(\mathcal{G}_{i-1}) + \varepsilon}{c(X_i)}$$

using submodularity in the first and the greedy rule in the second inequality. Since $\sum_{j=1}^{m} c(Y_j) \leq L$ it holds that

$$F(OPT) - F(\mathcal{G}_{i-1}) = \sum_{j=1}^{m} Z_j \leq \alpha L \frac{F(\mathcal{G}_i) - F(\mathcal{G}_{i-1}) + \varepsilon}{c(X_i)} + m\varepsilon$$

□

Lemma B.2 (adapted from Khuller et al. [1999]). *For $i = 1, \ldots, l+1$ it holds that*

$$F(\mathcal{G}_i) \geq \left[1 - \prod_{k=1}^{i}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right] F(OPT) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon.$$

Proof. Let $i = 1$ for sake of induction. We need to prove that $F(\mathcal{G}_1) \geq \frac{c(X_1)}{\alpha L} F(OPT) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon$. This follows from Lemma B.1 and since

$$\frac{\alpha L}{c(X_i)} + w \geq 1 + \frac{wc(X_i)}{\alpha L}.$$

Now let $i > 1$. We have

$$\begin{aligned}
F(\mathcal{G}_i) &= F(\mathcal{G}_{i-1}) + [F(\mathcal{G}_i) - F(\mathcal{G}_{i-1})] \\
&\geq F(\mathcal{G}_{i-1}) + \frac{c(X_i)}{\alpha L}[F(OPT) - F(\mathcal{G}_{i-1})] - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right) \\
&= \left(1 - \frac{c(X_i)}{\alpha L}\right) F(\mathcal{G}_{i-1}) + \frac{c(X_i)}{\alpha L} F(OPT) - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right) \\
&\geq \left(1 - \frac{c(X_i)}{\alpha L}\right)\left[\left(1 - \prod_{k=1}^{i-1}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right) F(OPT) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon\right] \\
&\quad + \frac{c(X_i)}{\alpha L} F(OPT) - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right) \\
&= \left(1 - \prod_{k=1}^{i}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right) F(OPT) - \varepsilon\left(1 + \frac{wc(X_i)}{\alpha L}\right) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon\left(1 - \frac{c(X_i)}{\alpha L}\right) \\
&= \left(1 - \prod_{k=1}^{i}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right) F(OPT) - \left(\frac{\alpha L}{c(X_i)} + w\right)\varepsilon
\end{aligned}$$

using Lemma B.1 in the first and the induction hypothesis in the second inequality. □

From now on let $\beta = \frac{\alpha L}{c_{min}} + w$.

Proof of Theorem 5.6. Observe that for $a_1, \ldots, a_n \in \mathbb{R}^+$ such that $\sum a_i = A$, the function $(1 - \prod_{i=1}^{n}(1 - \frac{a_i}{\alpha A}))$ achieves its minimum at $a_1 = \cdots = a_n = \frac{A}{n}$. In order to show this, let $G(a_1, \ldots, a_{m-1}) = \log \frac{A - \sum_{j=1}^{m-1} a_j}{A} + \sum_i \log(\alpha - \frac{a_i}{A})$. It holds that

$$\frac{\partial G}{\partial a_i} = \frac{-1/A}{\alpha - a_i/A} + \frac{1/A}{\alpha - (A - \sum_{j=1}^{m-1} a_j)/A}.$$

This derivative is 0 iff $A - \sum_{j=1}^{m-1} a_j = a_i$ for all $1 \leq i < m$. Hence, for $1 \leq i \leq m$ it must hold that $a_i = A/m$. We hence have

$$\begin{aligned}
F(\mathcal{G}_{l+1}) &\geq \left[1 - \prod_{k=1}^{l+1}\left(1 - \frac{c(X_k)}{\alpha L}\right)\right] F(OPT) - \beta\varepsilon \\
&\geq \left[1 - \prod_{k=1}^{i}\left(1 - \frac{c(X_k)}{\alpha c(\mathcal{G}_{l+1})}\right)\right] F(OPT) - \beta\varepsilon \\
&\geq \left[1 - \left(1 - \frac{1}{\alpha(l+1)}\right)^{l+1}\right] F(OPT) - \beta\varepsilon \\
&\geq \left(1 - \frac{1}{e^{1/\alpha}}\right) F(OPT) - \beta\varepsilon
\end{aligned}$$

where the first inequality follows from Lemma B.2 and the second inequality follows from the fact that $c(\mathcal{G}_{l+1}) > L$, since it violates the budget.

Furthermore note, that the violating increase $F(\mathcal{G}_{l+1}) - F(\mathcal{G}_l)$ is bounded by $\alpha F(\{X^*\})$ for $X^* = \operatorname{argmax}_{X \in \mathcal{W}} F(\{X\})$, i.e. the second candidate solution considered by the modified greedy algorithm. Hence

$$F(\mathcal{G}_l) + F(\{X^*\}) \geq F(\mathcal{G}_{l+1})/\alpha \geq \frac{1}{\alpha}(1 - 1/e^{1/\alpha})F(OPT) - \beta\varepsilon$$

and at least one of the values $F(\{X^*\})$ or $F(\mathcal{G}_l)$ must be greater than or equal to $\frac{1}{2\alpha}((1-1/e^{1/\alpha})F(OPT) - \beta\varepsilon)$. □

Proof of Theorem 5.10. For all nodes $s \in \mathcal{V} \setminus \mathcal{A}$, let $\delta_s = F(\mathcal{A} \cup \{s\}) - F(\mathcal{A})$. Let us assume the costs $c(s)$ and B are rational . Without loss of generality, we can multiply costs and budget by their least common multiple, and hence we are left with integral costs and budget. Let us replicate all elements according to their cost, and assign weights to them according to their benefit cost ratio, i.e., for all replica s' of element s, set weight $\delta'_{s'} = \delta_s / c(s)$. Also, let $\mathcal{A}'$ be the set of all replicas corresponding to the nodes in $\mathcal{A}$. Let $\mathcal{B}'$ be the replicas of all elements in the optimal solution $\mathcal{B}$. Since F is nondecreasing, $F(\mathcal{A}' \cup \mathcal{B}') \geq F(\mathcal{B}') = OPT$. Due to submodularity,

$$F(\mathcal{A}' \cup \mathcal{B}') \leq F(\mathcal{A}') + \sum_{s' \in \mathcal{B}'} \delta'_{s'}.$$

Furthermore,

$$\sum_{s' \in \mathcal{B}'} \delta'_{s'} \leq \max_{\mathcal{C}' : |\mathcal{C}'| \leq B} \sum_{s' \in \mathcal{C}'} \delta'_{s'}.$$

Now we have a unit-cost modular optimization problem: We want to pick the best set $\mathcal{C}'$ of B elements, maximizing the sum of their weights $\delta'_{s'}$. The ordinary unit cost greedy algorithm solves this optimally. More specifically, we can sort the new weights $\delta'_{s'}$ in decreasing order (in case of ties we keep the replicas of the elements in contiguous blocks), and pick the B first elements. Hence, the greedy algorithm on the replicated unit cost problem will have to integrally pick the first $k-1$ original elements, and will fractionally pick the last $(k-th)$ element, selecting $M = (B - \sum_{i=1}^{k-1} c(s_j))$ elements. Summing up the weights of the unit cost elements will give us $\sum_i = 1^{k-1}\delta_{s_i} + \lambda * \delta_{s_k}$, where $\lambda = M/c(s_k)$. □

B.2 Proofs from Chapter 6

Proof of Theorem 6.2. Our reduction builds on the proof by Ko et al. [1995], who show that for any graph G, there exists a polynomially related, symmetric positive-definite matrix Σ such that Σ has a subdeterminant (of a submatrix resulting from the selection of k rows and columns $i_1, \ldots, i_k$) greater than some M if G has a clique of size at least k, and Σ does not have a subdeterminant greater than $M - \varepsilon$ for some (polynomially-large) $\varepsilon > 0$ if G does not have such a clique. Let G be a graph, and let Σ be the matrix constructed in Ko et al. [1995]. We will consider Σ as the covariance matrix of a multivariate Gaussian distribution with variables $\mathcal{X}_\mathcal{U} = \{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$. Introduce additional variables $\mathcal{X}_\mathcal{S} = \{y_1, \ldots, y_n\}$ such that $y_i \mid X_i = x \sim \mathcal{N}(x, \sigma^2)$. Note that a subset $\mathcal{A} \subseteq \mathcal{S}$, $|\mathcal{A}| = k$, has maximum entropy of all such subsets if and only if the parents $\Gamma_\mathcal{A} \subset \mathcal{U}$ of $\mathcal{A}$ have maximum entropy among all such subsets of $\mathcal{U}$. Now note that $I(\mathcal{X}_\mathcal{A}; \mathcal{X}_{(\mathcal{U} \cup \mathcal{S}) \setminus \mathcal{A}}) = H(\mathcal{X}_\mathcal{A}) - H(\mathcal{X}_\mathcal{A} \mid \mathcal{X}_{(\mathcal{U} \cup \mathcal{S}) \setminus \mathcal{A}}) = H(\mathcal{X}_\mathcal{A}) - H(\mathcal{X}_\mathcal{A} \mid \mathcal{X}_\mathcal{U})$, because y_i and y_j are conditionally independent given $\mathcal{X}_\mathcal{U}$. Furthermore, again because of independence, $H(\mathcal{X}_\mathcal{A} \mid \mathcal{X}_\mathcal{U})$ is a constant only depending on the cardinality of $\mathcal{A}$. Assume we could decide efficiently whether there is a subset $\mathcal{A} \subset \mathcal{S}$ such that $I(\mathcal{X}_\mathcal{A}; \mathcal{X}_{\mathcal{V} \setminus \mathcal{A}}) \geq M'$. If we choose σ^2 small enough, then this would allow us to decide whether G has a clique of size k, utilizing the gap ε. □

Proof of Lemma 6.4. Define $\hat{\mathcal{K}}(x,y) = \mathcal{K}(x,y)$ for $x \neq y$ and $\hat{\mathcal{K}}(x,x) = \mathcal{K}(x,x) + \sigma^2$ to include the sensor noise σ^2. Since $\mathcal{C}$ is compact and $\mathcal{K}$ continuous, $\mathcal{K}$ is uniformly continuous over $\mathcal{C}$. Hence, for any ε_1, there exists a δ_1 such that for all $x, x', y, y', \|x - x'\|_2 \leq \delta_1, \|y - y'\|_2 \leq \delta_1$ it holds that $|\mathcal{K}(x,y) - \mathcal{K}(x',y')| \leq \varepsilon_1$. Assume $\mathcal{C}_1 \subset \mathcal{C}$ is a finite mesh grid with mesh width $2\delta_1$. We allow sensor placement only on grid $\mathcal{C}_1$. Let $\mathcal{C}_2 \subset \mathcal{C}$ be a mesh grid of mesh width $2\delta_1$, which is derived by translating $\mathcal{C}_1$ by δ_1 in Euclidean norm, and let G_1, G_2 denote the restriction of the GP G to $\mathcal{C}_1, \mathcal{C}_2$. We assume $\mathcal{C}_1, \mathcal{C}_2$ cover $\mathcal{C}$ in the sense of compactness. We use the notation $\tilde{\cdot}$ to refer to the translated version in G_2 of the random variable $\cdot$ in G_1. $\hat{\mathcal{K}}$ is a symmetric strictly positive definite covariance function and $|\hat{\mathcal{K}}(X,y) - \hat{\mathcal{K}}(\tilde{X},\tilde{y})| \leq \varepsilon_1$ for all $X, y \in G_1$. Moreover, since $\mathcal{K}$ is positive semidefinite, the smallest eigenvalue of any covariance matrix derived from $\hat{\mathcal{K}}$ is at least σ^2.

Let $\mathcal{A}$ be a subset of $\mathcal{C}_1$ and $X \in \mathcal{C}_1 \setminus \mathcal{A}$. Using (6.7), we first consider the conditional variance $\sigma^2_{X|\mathcal{A}}$. By definition, $\|y - \tilde{y}\|_2 \leq \delta_1$, and hence $|\hat{\mathcal{K}}(X,y) - \hat{\mathcal{K}}(X,\tilde{y})| \leq \varepsilon_1$ for all $y \in \mathcal{A}$. Hence we know that $\|\Sigma_{\mathcal{A}\mathcal{A}} - \Sigma_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 \leq \|\Sigma_{\mathcal{A}\mathcal{A}} - \Sigma_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_F \leq k^2\varepsilon_1$. We furthermore note that $\|\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\|_2 = \lambda^{max}(\Sigma^{-1}_{\mathcal{A}\mathcal{A}}) = \lambda^{min}(\Sigma_{\mathcal{A}\mathcal{A}})^{-1} \leq \sigma^{-2}$, and hence

$$\begin{aligned}\|\Sigma^{-1}_{\mathcal{A}\mathcal{A}} - \Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 &= \|\Sigma^{-1}_{\mathcal{A}\mathcal{A}}(\Sigma_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}} - \Sigma_{\mathcal{A}\mathcal{A}})\Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 \\ &\leq \|\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\|_2\|\Sigma_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}} - \Sigma_{\mathcal{A}\mathcal{A}}\|_2\|\Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2 \leq \sigma^{-4}k^2\varepsilon_1.\end{aligned}$$

We derive $\|\Sigma_{X\tilde{\mathcal{A}}} - \Sigma_{X\mathcal{A}}\|_2 \leq \|\varepsilon_1 \mathbf{1}^T\|_2 = \varepsilon_1\sqrt{k}$, hence

$$\begin{aligned}|\sigma^2_{X|A} - \sigma^2_{X|\tilde{\mathcal{A}}}| &= |\Sigma_{X\mathcal{A}}\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\Sigma_{\mathcal{A}X} - \Sigma_{X\tilde{\mathcal{A}}}\Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\Sigma_{\tilde{\mathcal{A}}X}|, \\ &\leq\ 2\|\Sigma_{X\mathcal{A}} - \Sigma_{X\tilde{\mathcal{A}}}\|_2\|\Sigma^{-1}_{\mathcal{A}\mathcal{A}}\|_2\|\Sigma_{X\mathcal{A}}\|_2 + \|\Sigma^{-1}_{\mathcal{A}\mathcal{A}} - \Sigma^{-1}_{\tilde{\mathcal{A}}\tilde{\mathcal{A}}}\|_2\|\Sigma_{X\mathcal{A}}\|_2^2 + \mathcal{O}(\varepsilon_1^2), \\ &\leq 2\varepsilon_1\sqrt{k}\sigma^{-2}M\sqrt{k} + \sigma^{-4}k^2\varepsilon_1 M^2 k + \mathcal{O}(\varepsilon_1^2), \\ &\leq \varepsilon_1 k\sigma^{-2}M\left(2 + \sigma^{-2}k^2M\right) + \mathcal{O}(\varepsilon_1^2),\end{aligned}$$

where $M = \max_{x \in C} \mathcal{K}(x,x)$. We choose δ such that the above difference is bounded by $\sigma^2\varepsilon$. We note that (assuming w.l.o.g. $H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{A}}) \geq H(\mathcal{X}_s \mid \mathcal{X}_{\tilde{\mathcal{A}}})$)

$$H(\mathcal{X}_s|\mathcal{X}_{\mathcal{A}}) - H(\mathcal{X}_s|\mathcal{X}_{\tilde{\mathcal{A}}}) = \frac{1}{2}\log\frac{\sigma^2_{\mathcal{X}_s|\mathcal{X}_{\mathcal{A}}}}{\sigma^2_{\mathcal{X}_s|\mathcal{X}_{\tilde{\mathcal{A}}}}} \leq \frac{\log(1+\varepsilon)}{2} \leq \frac{\varepsilon}{2}.$$

which concludes the argument. □

Proof of Corollary 6.5. The higher order terms $\mathcal{O}(\varepsilon_1^2)$ can be worked out as $k\sigma^{-2}\varepsilon^2(1 + Mk^2\sigma^{-2} + \varepsilon k^2\sigma^{-2})$. Assuming that $\varepsilon < \min(M, 1)$, this is bounded by $3k^3M\sigma^{-4}\varepsilon$. Using the Lipschitz assumption, we can directly compute δ_1 from ε_1 in the above proof, by letting $\delta = \varepsilon_1/L$. Let $R = k\sigma^{-2}M\left(2 + \sigma^{-2}k^2M\right) + 3k^3M\sigma^{-4}$. We want to choose δ such that $\varepsilon_1 R \leq \sigma^2\varepsilon$. Hence if we choose $\delta \leq \frac{\sigma^2\varepsilon}{LR}$, then $|H(X|\mathcal{A}) - H(X|\tilde{\mathcal{A}})| \leq \varepsilon$ uniformly as required. Note that in order to apply the result from Nemhauser et al. [1978], the approximate nondecreasingness has to be guaranteed for subsets of size $2k$, which results in the stated bound. □

Proof of Theorem 6.6. The following proof is an extension of the proof by Nemhauser et al. [1978], using some simplifications by Jon Kleinberg.

Let $s_1, \ldots, s_k$ be the locations selected by the greedy algorithm. Let $\mathcal{A}_i = \{s_1, \ldots, s_i\}$, $\mathcal{A}^*$ be the optimal solution, and $\delta_i = F_{MI}(\mathcal{A}_i) - F_{MI}(\mathcal{A}_{i-1})$. By $Lemma$ 6.4, we have, for all $1 \leq i \leq k$,

$$F_{MI}(\mathcal{A}_i \cup \mathcal{A}^*) \geq F_{MI}(\mathcal{A}^*) - k\varepsilon.$$

We also have, for $0 \leq i < k$,

$$F_{MI}(\mathcal{A}_i \cup \mathcal{A}^*) \leq F_{MI}(\mathcal{A}_i) + k\delta_{i+1} = \sum_{j=1}^{i} \delta_j + k\delta_{i+1}.$$

Hence we have the following sequence of inequalities:

$$\begin{aligned}
F_{MI}(\mathcal{A}^*) - k\varepsilon &\leq k\delta_1 \\
F_{MI}(\mathcal{A}^*) - k\varepsilon &\leq \delta_1 + k\delta_2 \\
&\vdots \\
F_{MI}(\mathcal{A}^*) - k\varepsilon &\leq \sum_{j=1}^{k-1} \delta_j + k\delta_k
\end{aligned}$$

Now we multiply both sides of the i-th inequality by $\left(1 - \frac{1}{k}\right)^{k-i}$, and add all inequalities up. After cancelation, we get

$$\left(\sum_{i=0}^{k-1} (1 - 1/k)^i\right) (F_{MI}(\mathcal{A}^*) - k\varepsilon) \leq k \sum_{i=1}^{k} \delta_i = kF_{MI}(\mathcal{A}_k).$$

Hence, as claimed, with $\mathcal{A}_G = \mathcal{A}_k$ (i.e., $\mathcal{A}_G$ is the k-element greedy solution)

$$F_{MI}(\mathcal{A}_G) \geq \left(1 - (1 - 1/k)^k\right) (F_{MI}(\mathcal{A}^*) - k\varepsilon) \geq (1 - 1/e)(F_{MI}(\mathcal{A}^*) - k\varepsilon).$$

□

B.3 Proofs from Chapter 7

Proof of Proposition 7.2. Let $\mathcal{C} = \mathcal{A} \cup \mathcal{B}$ such that $\mathcal{A} \subseteq \mathcal{S}$ and $\mathcal{B} \subseteq \mathcal{U}$. Then

$$\begin{aligned}
&H(\mathcal{X}_{\mathcal{U}\setminus\mathcal{C}} \mid \mathcal{X}_{\mathcal{C}}) = H(\mathcal{X}_{\mathcal{U}\setminus\mathcal{B}} \mid \mathcal{X}_{\mathcal{A}\cup\mathcal{B}}) = H(\mathcal{X}_{\mathcal{U}\cup\mathcal{A}}) - H(\mathcal{X}_{\mathcal{A}\cup\mathcal{B}}) \\
&= H(\mathcal{X}_{\mathcal{A}} \mid \mathcal{X}_{\mathcal{U}}) + H(\mathcal{X}_{\mathcal{U}}) - H(\mathcal{X}_{\mathcal{C}}) = H(\mathcal{X}_{\mathcal{U}}) - H(\mathcal{X}_{\mathcal{C}}) + \sum_{y \in \mathcal{C}\cap\mathcal{S}} H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{U}})
\end{aligned}$$

using the chain rule for the joint entropy. $H(\mathcal{X}_{\mathcal{U}})$ is constant, $H(\mathcal{X}_{\mathcal{C}})$ is submodular and $\sum_{y\in\mathcal{C}\cap\mathcal{S}} H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{U}})$ is linear in $\mathcal{C}$ on $\mathcal{U} \cup \mathcal{S}$ and hence on $\mathcal{W}$.

To show that F is nondecreasing we consider $\Delta = F(\mathcal{C} \cup \{s\}) - F(\mathcal{C}) = H(\mathcal{X}_{\mathcal{C}\cup\{s\}}) - H(\mathcal{X}_{\mathcal{C}}) + \sum_{y\in(\mathcal{C}\cup\{s\})\cap\mathcal{S}} H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{U}}) - \sum_{y\in\mathcal{C}\cap\mathcal{S}} H(\mathcal{X}_y \mid \mathcal{X}_{\mathcal{U}})$. If $s \in \mathcal{U}$, then $\Delta = H(\mathcal{X}_{\mathcal{C}\cup\{s\}}) - H(\mathcal{X}_{\mathcal{C}}) = H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{C}}) \geq 0$. If $s \in \mathcal{S}$, then $\Delta = H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{C}}) - H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{U}}) \geq H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{C}\cup\mathcal{U}}) - H(\mathcal{X}_s \mid \mathcal{X}_{\mathcal{U}}) = 0$ using conditional independence and the fact that conditioning never increases the entropy. □

Proof of Proposition 7.5. We compute the marginal increase $\Delta = F(\mathcal{C} \cup X) - F(\mathcal{C})$ as in the proof of Proposition 7.2. If $s \in \mathcal{U}$, then $\Delta = H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{C})$ and if $s \in \mathcal{S}$, then $\Delta = H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{C}) - H(\mathcal{X}_s \mid \mathcal{X}_\mathcal{U})$. □

Proof of Theorem 7.7. We will reduce MAX-COVER to maximizing the information gain. MAX-COVER is the problem of finding a collection of L sets such their union contains the maximum number of elements. In [Feige, 1998], it is shown that MAX-COVER cannot be approximated by a factor better than $(1 - 1/e)$ unless $\mathbf{P} = \mathbf{NP}$. Our reduction generates a Turing machine, with a polynomial runtime guarantee, which computes information gains for variable selections.

Let $\mathcal{W} = \{s_1, \ldots, s_n\}$ be a set, $\mathcal{V} = \{1, \ldots, m\}$ and $\mathcal{C}_1, \ldots, \mathcal{C}_m$ be a collection of subsets of $\mathcal{W}$, defining an instance of MAX-COVER . Associate with each element s of $\mathcal{W}$ an independent binary random variable $\mathcal{X}_s$ with $\Pr(\mathcal{X}_s = 1) = 0.5$. For each $i \in \mathcal{V}$ let $\mathcal{Y}_i$ be a random vector $\mathcal{Y}_i = (\mathcal{X}_{i_1}, \ldots, \mathcal{X}_{i_l})$ where $\mathcal{C}_i = \{s_{i_1}, \ldots, s_{i_l}\}$, and let $\mathcal{S} = \{\mathcal{Y}_1, \ldots, \mathcal{Y}_m\}$. Considered stochastically, the $\mathcal{Y}_1, \ldots, \mathcal{Y}_m$ are conditionally independent given all $\mathcal{X}_i$. Furthermore, for $\mathcal{B} \subseteq \mathcal{W}, \mathcal{A} \subseteq \mathcal{V}$ it holds that $H(\mathcal{X}_\mathcal{B} \mid \mathcal{Y}_\mathcal{A}) = \sum_{s \in \mathcal{B}} H(\mathcal{X}_s \mid \mathcal{Y}_\mathcal{A})$, and $H(\mathcal{X}_s \mid \mathcal{Y}_\mathcal{A}) = 0$ iff $s \in \mathcal{A}$, 1 otherwise. Hence if $I \subseteq \{1, \ldots, m\}$, then $H(\mathcal{X}_\mathcal{W}) - H(\mathcal{X}_\mathcal{W} \mid \bigcup_{i \in I} \mathcal{Y}_i) = |\bigcup_{i \in I} \mathcal{C}_i|$. Thus if we can approximate the maximum of $I(\mathcal{X}_\mathcal{U}; \mathcal{Y}_\mathcal{A})$ over all L-element subsets $\mathcal{A} \subseteq \mathcal{S}$ by a constant factor of $(1 - 1/e + \varepsilon)$ for some $\varepsilon > 0$, then we can approximate the solution to MAX-COVER by the same factor. Note that the entropies computed in our reduction are all integer-valued, and it is possible to efficiently construct a polynomial time Turing machine computing $I(\mathcal{X}_\mathcal{U}; \mathcal{Y}_\mathcal{A})$, with a runtime guarantee. □

Proof of Lemma 7.8. We will approximate the marginal increase $\Delta = F(\mathcal{A} \cup X) - F(\mathcal{A})$ by sampling. Hence we need to compute the number N of samples we need to guarantee that the sample mean Δ_N does not differ from Δ by more than ε / L with a confidence of at least $1 - \frac{\delta}{Ln}$. First we note that $0 \leq \Delta \leq H(X) \leq \log |\operatorname{dom}(X)|$. Hoeffding's inequality [Hoeffding, 1963] states that

$$\Pr[|\Delta_N - \Delta| \geq \varepsilon / L] \leq 2 \exp(-2N(\frac{\varepsilon}{L \log |\operatorname{dom}(X)|})^2).$$

This quantity is bounded by $\delta / (Ln)$ if $N \geq \frac{1}{2}(\frac{L \log |\operatorname{dom}(X)|}{\varepsilon})^2 \log \frac{2Ln}{\delta}$. □

Proof of Theorem 7.9. Let $\varepsilon, \delta > 0$. We will approximate F by sampling. Let $n = |S|$. In each step of the greedy algorithm, $2n$ sample approximations have to be made. If we run L steps, we have to guarantee a confidence of $1 - \frac{\delta}{2Ln}$ for each sample approximation to guarantee an overall confidence of $1 - \delta$ by the union bound. The result follows from Lemma 7.8 and the remark on approximate maximization of submodular functions. □

B.4 Proofs from Chapter 8

Proof of Theorem 8.1. Our proof is similar to the analysis of [Nemhauser et al., 1978]. Fix scenario i. We first show that the function $F_i(\mathcal{A}) = \pi_i(\infty) - \pi_i(T(\mathcal{A}, i))$ is submodular. Consider $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$. Let $s \in \mathcal{V} \setminus \mathcal{B}$. We have three cases. (i) $T(s, i) \geq T(\mathcal{A}, i)$. Then $T(\mathcal{A} \cup \{s\}) = T(A)$ and $T(\mathcal{B} \cup \{s\}) = T(\mathcal{B})$ and hence $F_i(\mathcal{A} \cup \{s\}) - F_i(\mathcal{A}) = 0 = F_i(\mathcal{B} \cup \{s\}) - F_i(\mathcal{B})$. (ii) $T(\mathcal{B}, i) \leq T(s, i) < T(\mathcal{A}, i)$. In this case, $F_i(\mathcal{A} \cup \{s\}) - F_i(\mathcal{A}) \geq 0 = F_i(\mathcal{B} \cup \{s\}) - F_i(\mathcal{B})$. Finally, (iii), $T(s, i) < T(\mathcal{B}, i)$. In this case, $F_i(\mathcal{A} \cup \{s\}) - F_i(\mathcal{A}) = [\pi_i(\infty) - \pi_i(T(s, i))] - F_i(\mathcal{A}) \geq [\pi_i(\infty) - \pi_i(T(s, i))] - F_i(\mathcal{B}) = F_i(\mathcal{B} \cup \{s\}) - F_i(\mathcal{B})$, where the inequality is due to the nondecreasingness of $F_i(\cdot)$. Hence, for each scenario

i, the function F_i is submodular. Now, $F(\mathcal{A}) = \sum_i P(i)F_i(\mathcal{A})$ is a nonnegative linear combination of submodular functions, and hence submodular too. □

Proof of Theorem 8.2. Let P be a distribution over directed graphs $\mathcal{G}_1 = (\mathcal{V}, \mathcal{E}_1), \ldots, \mathcal{G}_N(\mathcal{V}, \mathcal{E}_N)$ on a fixed set of vertices $\mathcal{V}$, defined according to the Triggering Model. For each i, let $\mathcal{G}'_i$ be the graph obtained from $\mathcal{G}'_i$ by reverting the arcs $\mathcal{E}_i$. Then, the penalty reduction $F_i(\mathcal{A})$ by the set of nodes $\mathcal{A}$ using the population affected score (PA) corresponds exactly to the number of nodes influenced by set $\mathcal{A}$ under the Triggering Model. Hence, also the expected penalty reduction $F(\mathcal{A}) = \sum_i P(i)F_i(\mathcal{A})$ is exactly equal to the influence function $\sigma(\mathcal{A})$ of Kempe et al. [2003]. □

Proof of Proposition 8.3. By reduction from set cover. Let a set $\mathcal{S}$ be given, along with a collection of subsets $\mathcal{B}_1, \ldots, \mathcal{B}_m \subseteq \mathcal{S}$. It is **NP**-hard to decide whether, for a constant k, there exists a collection of subsets $\mathcal{B}_{i_1}, \ldots, \mathcal{B}_{i_k}$ such that their union covers $\mathcal{S}$ [Garey and Johnson, 2003]. We can turn such an instance into a sensor placement problem, where we have a contamination scenario for each element $i \in \mathcal{S}$, and a sensor corresponding to each subset $\mathcal{B}_i$. For a constant k, the penalty minimization problem is the problem of selecting a set of sensors which minimize the number of undetected scenarios. Hence, there exists a set cover of size k if and only if there exists a sensor placement with expected penalty of 0. If we had a sensor placement algorithm which, for a constant k, would be guaranteed to obtain a solution whose expected penalty is some multiple $\alpha(n)$ of the minimum penalty, where $\alpha(n)$ only depends on the problem size n, then we could use this algorithm to decide the set cover problem, implying that $\mathbf{P} = \mathbf{NP}$.

□

B.5 Proofs from Chapter 10

Proof of Theorem 10.1. Consider a hitting set instance with m subsets $\mathcal{S}_i \subseteq \mathcal{V}$ on a ground set $\mathcal{V}$. Our task is to select a set $\mathcal{A} \subseteq \mathcal{V}$ with which intersects all sets $\mathcal{S}_i$, and such that $|\mathcal{A}| = k$ is as small as possible. For each set $\mathcal{S}_i$, define a function F_i such that $F_i(\mathcal{A}) = 1$ if $\mathcal{A}$ intersects $\mathcal{S}_i$, and 0 otherwise. It can be seen that F_i is clearly nondecreasing. F_i is also submodular, since for $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $x \in \mathcal{V} \setminus \mathcal{B}$, if $F_i(\mathcal{B}) = 0$ and $F_i(\mathcal{B} \cup \{x\}) = 1$, then it $x \in \mathcal{S}_i$, hence $F_i(\mathcal{A} \cup \{x\}) = 1$ and $F_i(\mathcal{A}) = 0$. Now assume the optimal hitting set $\mathcal{A}^*$ is of size k. Hence $\min_i F_i(\mathcal{A}^*) = 1$. If there were an algorithm for solving Problem (10.2) with approximation guarantee $\gamma(n)$ it would select a set $\mathcal{A}'$ of size $|\mathcal{A}'| \leq k$ with $\min_i F_i(\mathcal{A}') \geq \gamma(n) \min_i F_i(\mathcal{A}^*) = \gamma(n) > 0$. But $\min_i F_i(\mathcal{A}') > 0$ implies $\min_i F_i(\mathcal{A}') = 1$, hence $\mathcal{A}'$ would be a hitting set. Hence, this approximation algorithm would be able to decide, whether there exists a hitting set of size k, contradicting the **NP**-hardness of the hitting set problem [Feige, 1998]. □

Proof of Lemma 10.2. Wolsey [1982a] proves that, given a nondecreasing submodular function F on a ground set $\mathcal{V}$, it holds that that greedy algorithm (GPC), applied to the optimization problem

$$\min_{\mathcal{A}} |\mathcal{A}| \text{ such that } F(\mathcal{A}) = F(\mathcal{V})$$

returns a solution $\mathcal{A}'$ such that $|\mathcal{A}'| \leq |\mathcal{A}^*|(1 + \log \max_{s \in \mathcal{V}} F(\{s\}))$, where $\mathcal{A}^*$ is an optimal solution. We apply Wolsey's result to the nondecreasing submodular function $\overline{F}_c$. In order to use GPC in the inner loop of the binary search over c, we need to make sure that the approximation guarantee for the greedy

algorithm is independent of c. This can be achieved by choosing

$$\alpha = 1 + \log\left(\max_{s\in\mathcal{V}} \sum_i F_i(\{s\})\right) \geq 1 + \log\left(m \max_{s\in\mathcal{V}} \overline{F}_c(\{s\})\right),$$

i.e., the choice of α stated in Lemma 10.2 is independent of the truncation threshold c. □

Proof of Theorem 10.3. Lemma 10.2 proves that during each of the iterations of the saturation algorithm it holds that $\min_i F_i(\mathcal{A}^*) \leq c_{\max}$, where $\mathcal{A}^*$ is an optimal solution. Furthermore, it holds that $\min_i F_i(\mathcal{A}_{best}) \geq c_{\min}$, and $|\mathcal{A}_{best}| \leq \alpha k$. Since the F_i are integral, if $c_{\max} - c_{\min} < \frac{1}{m}$ then it must hold that $\min_i F_i(\mathcal{A}_{best}) \geq \min_i F_i(\mathcal{A}^*)$ as claimed by Theorem 10.3.

For the running time, since at the first iteration, $c_{\max} - c_{\min} \leq \min_i F_i(\mathcal{V})$, and $c_{\max} - c_{\min}$ is halved during each iteration, it follows that after $1 + \lceil \log_2 m \min_i F_i(\mathcal{V}) \rceil$ iterations, $c_{\max} - c_{\min} < \frac{1}{m}$, at which point the algorithm terminates. During each iteration, Algorithm 10.1 is invoked once, which requires $\mathcal{O}(|\mathcal{V}|^2 m)$ function evaluations. □

Proof of Theorem 10.4. We use the same hitting set construction as in Theorem 10.1. If there were an algorithm for selecting a set $\mathcal{A}'$ of size $|\mathcal{A}'| \leq \beta k$ with $\min_i F_i(\mathcal{A}') = 1$, and $\beta \leq (1-\varepsilon)\alpha$, for some fixed $\varepsilon > 0$, then we would have an approximation algorithm for hitting set with guarantee $(1-\varepsilon)\log m$ which would imply $\mathbf{NP} \subseteq \mathrm{DTIME}(n^{\log\log n})$ [Feige, 1998]. □

Theorem 10.6. The proof is analogous to the proof of Theorem 10.3. The approximation guarantee α is established by noticing that the greedy algorithm is applied to the modified (integral) objective

$$\overline{F}_{c_{wc},c_{ac}}(\mathcal{A}) = \sum_i \min\{F_i(\mathcal{A}), c_{wc}\} + \min\left\{\sum_i F_i(\mathcal{A}), m c_{ac}\right\}.$$

The guarantee α is obtained from the analysis of the greedy submodular coverage algorithm of Wolsey [1982a], similar to Lemma 10.2. Approximate Pareto-optimality follows directly from Pareto-optimality of any solution to (10.11). □

B.6 Proofs from Chapter 11

Proof of Lemma 11.2. Given a collection of weights $\mathcal{P} = \{p_S : S \subseteq \mathcal{B}\}$, we write $E(\mathcal{P}) = \sum_{S\subseteq\mathcal{B}} p_S \cdot F(S)$. Note that $\mathbb{E}[F(A)] = E(\mathcal{P}_0)$ for $\mathcal{P}_0 = \{\Pr[A = S] : S \subseteq \mathcal{B}\}$.

Starting with the set of weights $\mathcal{P}_0$, we iteratively apply the following "uncrossing" procedure. As long as there is a pair of sets $S, T \subseteq \mathcal{B}$ such that neither of S or T is contained in the other, and p_S, $p_T > 0$, we subtract $x = \min(p_S, p_T)$ from both p_S and p_T, and we add x to both $p_{S\cap T}$ and $p_{S\cup T}$. Note the following properties of this procedure.

(i) The quantity $\sum_{S\subseteq\mathcal{B}} p_S$ remains constant over all iterations.

(ii) For each element $X \in \mathcal{B}$, the quantity $\sum_{S\subseteq\mathcal{B}:X\in S} p_S$ remains constant over all iterations,

(iii) The quantity $\sum_{S\subseteq\mathcal{B}} p_S |S|^2$ strictly increases every iteration.

(iv) By the submodularity of F, the quantity $E(\mathcal{P})$ is non-increasing over the iterations.

By (i) and (iii), this sequence of iterations, starting from $\mathcal{P}_0$, must terminate at a set of weights $\mathcal{P}^*$. At termination, the sets S on which $p_S > 0$ must be totally ordered with respect to inclusion, and by (ii) it follows that $p_{\mathcal{B}} \geq \rho$. Finally, by (iv), we have

$$\mathbb{E}[F(\mathcal{A})] = E(\mathcal{P}_0) \geq E(\mathcal{P}^*) \geq \rho F(\mathcal{B}), \tag{B.1}$$

as required. □

In order to prove Theorems 11.1 and 11.5, let us consider the subset $\mathcal{A}^*$ spanned by the optimal tree (or path), and let $\overline{\mathcal{A}^*} \subseteq \mathcal{A}^*$ denote its r-padded nodes with respect to a random partition drawn from the padded decomposition. (Recall that each node is r-padded with probability at least ρ.) Now Lemma 11.2 implies that $F(\overline{\mathcal{A}^*})$, the expected value of the nodes in A that are r-padded, is at least $\rho F(\mathcal{A}^*)$. The algorithm is based on the idea of trying to build a tree (a path) that recoups a reasonable fraction of this "padded value".

The following lemma will be useful in converting subtrees and paths of $\mathcal{G}'$ back to solutions of our original problem.

Proposition B.3. *Given any subtree $\mathcal{T}'$ or path $\mathcal{P}'$ of $\mathcal{G}'$ spanning nodes $\mathcal{A}'$ with weight W containing at least one cluster center, it is possible to find a subtree $\mathcal{T} \subseteq \mathcal{G}$ resp. path $\mathcal{P} \subseteq \mathcal{G}$ spanning the same vertices $\mathcal{A}'$, with a total length no more than $\ell(\mathcal{T}')$ resp. $\ell(\mathcal{P}')$, and with $F(\mathcal{A}') \geq \gamma W$.*

Proof. Each edge of $\mathcal{G}'$ (and hence of $\mathcal{T}'$) corresponds to some shortest path in $\mathcal{G}$, and we can add all these paths together to form a connected subgraph. Let $\mathcal{T}$ be any spanning tree of this subgraph; clearly, its length is no more than $\ell(\mathcal{T}')$. If $V_i \subseteq P_i$ is the subpath of P_i contained in $\mathcal{T}'$, then the weight of these vertices $V(P_i')$ is exactly the total submodular value $F(V(P_i'))$, just by the definition the weights. Furthermore, since each pair of distinct paths are at distance at least r from each other, the locality property assures that the value of their union is at least γW. For paths, just observe that expanding edges of $\mathcal{P}'$ in $\mathcal{G}$ results in another path $\mathcal{P}$. □

Proposition B.4. *If the graph $\mathcal{G}$ contains a subtree $\mathcal{T}^*$ spanning nodes $\mathcal{A}^*$, of length $\mathrm{C}(\mathcal{T}^*) = \ell^*$ and value $F(\mathcal{A}^*)$, then there is a subtree $\mathcal{T}'$ of the graph $\mathcal{G}'$ that has length at most*

$$\ell^* \times (\alpha(r+2)+2) \tag{B.2}$$

and whose expected weight is at least

$$F(\mathcal{A}^*) \times (1 - e^{-1}) \times \rho \tag{B.3}$$

Proof. Let a cluster $\mathcal{C}_i$ be called *occupied* if $\overline{\mathcal{A}^*} \cap \mathcal{C}_i \neq \emptyset$; w.l.o.g., let the $s+1$ clusters $\mathcal{C}_0, \mathcal{C}_2, \ldots, \mathcal{C}_s$ be occupied. Let $z_0, \ldots, z_s$ be the cluster centers (i.e., the first nodes picked by the greedy algorithm). We start building $\mathcal{T}'$ by adding a spanning tree on the centers of the clusters that are occupied.

The Cost. Let us bound the length of this center-spanning tree. Since $\mathcal{A}^*$ contains a point (say a_i) from each $\overline{\mathcal{C}_i}$, the padding condition ensures that the r-balls $B_r(a_i)$ must be disjoint, and hence the length of $\mathcal{T}^*$ is at least rs. Now, to attach a_i to z_i, we can add paths of length at most αr to $\mathcal{T}^*$; thus causing the resulting tree to have length $\ell^* + \alpha rs \leq (\alpha+1)\ell^*$. Since this is a Steiner tree on the centers, we can get

a spanning tree of at most twice the cost; hence the cost of the edges connecting the spanning centers is at most

$$2(\alpha+1)\,\ell^*. \tag{B.4}$$

Now consider an occupied cluster $\mathcal{C}_i$, and let $|\overline{\mathcal{A}^* \cap \mathcal{C}_i}| = n_i$ be the number of padded nodes in $\mathcal{C}_i$. We now add to $\mathcal{T}'$ the subpath of P_i containing first n_i nodes $\{Z_i = G_{i,1}, G_{i,2}, \ldots, G_{i,n_i}\}$. Note that the length of edges added for cluster $\mathcal{C}_i$ is at most $\alpha r n_i$; summing over all occupied clusters gives a total length of $\alpha r \sum_i n_i \leq \alpha r |\mathcal{A}^*| \leq \alpha r \ell^*$, since each edge in $\mathcal{T}^*$ has at least unit length. Adding this to (B.4) proves the claim on the length of $\mathcal{T}'$.

The Weight. Finally, let us calculate the weight of the tree $\mathcal{T}'$: by the properties of the greedy algorithm used in the construction of $\mathcal{G}'$, the *weight* of the set S_{in_i} added in cluster $\mathcal{C}_i$ is at least

$$(1-e^{-1})F(\overline{\mathcal{A}^* \cap \mathcal{C}_i}) \tag{B.5}$$

Summing this over occupied clusters, we get that the total weight is at least $(1-e^{-1})F(\overline{\mathcal{A}^*})$, whose expected value is at least $(1-e^{-1})\rho F(\mathcal{A}^*)$. □

Proposition B.5. *If the graph $\mathcal{G}$ contains an $s-t$ path $\mathcal{P}^*$ spanning nodes $\mathcal{A}^*$, of length $\mathrm{C}(\mathcal{P}^*) = \ell^*$ and value $F(\mathcal{A}^*)$, then there is an $s-t$ path $\mathcal{P}'$ of the graph $\mathcal{G}'$ that has length at most*

$$2 \times \ell^* \times (\alpha(r+2)+6) \tag{B.6}$$

and whose expected weight is at least

$$F(\mathcal{A}^*) \times (1-e^{-1}) \times \rho \tag{B.7}$$

Proof. This result is an immediate corollary from Proposition B.4, noting that the vertices s and t need to be connected to the center spanning tree by paths at most $2\ell^*$, and that the path $\mathcal{P}^*$ is a tree, and hence there exists a subtree $\mathcal{T}'$ in $\mathcal{G}'$ of length at most $\ell^* \times (\alpha(r+2)+4)$. This tree is readily converted into a path (e.g., by traversal) of length at most twice the cost of the tree, i.e., of at most $2 \times \ell^* \times (\alpha(r+2)+6)$. □

Combining these results, we now prove a slightly more detailed statement of Theorems 11.1 and 11.5:

Theorem B.6. ***Trees:*** *For the covering problem* (11.1)*,* pSPIEL *will find a solution $\mathcal{T}$, with cost at most*

$$\kappa_{Quota}\,\ell^*\,(\alpha(r+2)+2) \tag{B.8}$$

and whose expected weight is at least

$$(1-e^{-1})\,\gamma\rho F(\mathcal{A}^*), \tag{B.9}$$

where ℓ^* is the weight of the optimum tree $\mathcal{A}^*$. For the maximization problem (11.2)*,* pSPIEL *will find a solution $\mathcal{T}$ with cost at most*

$$\ell^*\,(\alpha(r+2)+2) \tag{B.10}$$

and whose expected weight is at least

$$\kappa^{-1}_{Budget}(1-e^{-1})\,\gamma\rho F(\mathcal{A}^*), \tag{B.11}$$

where κ_{Quota} and κ_{Budget} denote the approximation guarantees for approximately solving Quota- and Budget-MST problems (currently, $\kappa_{Quota} = 2$ and $\kappa_{Budget} = 3 + \varepsilon$, for $\varepsilon > 0$, are the best known such guarantees [Garg, 2005, Johnson et al., 2000]).

Paths: *For the path planning problem* (11.5)*,* PSPIEL *will find a solution $\mathcal{P}$, with cost at most*

$$\kappa_{Orient}\, 2\ell^* \left(\alpha(r+2)+6\right) \tag{B.12}$$

and whose expected weight is at least

$$(1-e^{-1})\,\gamma\rho F(\mathcal{A}^*), \tag{B.13}$$

where ℓ^* is the weight of the optimum tree $\mathcal{A}^*$. Hereby, κ_{Orient} is the approximation guarantee for approximately solving the Orienteering problem (currently, $\kappa_{Orient} = 3 + \varepsilon$ is the best known such guarantee [Chekuri et al., 2008]).

Proof. Proposition B.4 proves the existence of a tree $\mathcal{T}'$ in the graph $\mathcal{G}'$, for which both cost and weight are close to the optimal tree $\mathcal{T}$ in $\mathcal{G}$. The construction in the proof also guarantees that the tree $\mathcal{T}'$ contains at least one cluster center $G_{i,1}$ for some i (or is empty, in which case $\mathcal{T}$ is empty). Proposition B.3 handles the transfer of the solution to the original graph $\mathcal{G}$. Hence, in order to solve the covering problem (11.1) or optimization problem (11.2) in $\mathcal{G}$, we need to solve the respective covering and maximization problem in the modular approximation graph $\mathcal{G}'$, rooted in one of the cluster centers. Any κ_{Quota} approximate algorithm to the Quota-MST problem can be used for the covering problem, using a quota of $Q = (1 - e^{-1})\rho\, F(\mathcal{A}^*)$. While for the unrooted version of the Budget-MST problem, there is a constant factor $\kappa_{Budget} = 3+\varepsilon$ approximation algorithm, the best known approximation for rooted Budget-MST is $4+\varepsilon$ by Chekuri et al. [2008]. We can however exploit the structure of the MAG to still get an approximation guarantee and prove Theorem 11.1 for an improved guarantee of $3 + \varepsilon$. We simply need to prune all nodes in $\mathcal{G}'$ which are further than $B = \ell^* \left(\alpha(r+2)+2\right)$ away from the core of $\mathcal{G}'$, and then run the unrooted approximation algorithm by Johnson et al. [2000] on $\mathcal{G}'$. If this algorithm, started with budget $B = \ell^* \left(\alpha(r+2)+2\right)$ selects nodes from sub-chain i, not including center $G_{i,1}$, we instead select the entire i-th chain. By construction, this procedure is guaranteed not to violate the budget, and the submodular function value can only increase.

The result for paths follows analogously, using Proposition B.5 instead of Proposition B.4, and applying the approximation algorithm for $s - t$ orienteering to the modular approximation graph. □

B.7 Proofs from Chapter 12

Proof of Theorem 12.1. Define a new ground set $\mathcal{V}' = \mathcal{V} \times \{1, \ldots, k\}$, and a new function

$$F'(\mathcal{A}') = \sum_{t=1}^{k} F(\{s : (s,t) \in \mathcal{A}'\}).$$

F' is nondecreasing and submodular. Let

$$\mathcal{I} = \{\mathcal{A}' \subseteq \mathcal{V}' : |\mathcal{A}'| \leq m \wedge \left(\nexists s, i \neq j : (s,i) \in \mathcal{A}' \wedge (s,j) \in \mathcal{A}'\right)\},$$

i.e., $\mathcal{I}$ is the collection of subsets $\mathcal{A}' \subseteq \mathcal{V}'$ that do not contain two pairs (s,i) and (s,j) of elements for which $i \neq j$. It can be shown that $\mathcal{I}$ form independent sets of a matroid [*c.f.*, Fisher et al., 1978]. Note that there is a one-to-one correspondence between sets $\mathcal{A}'$ and feasible solutions $\mathcal{A}_1, \ldots, \mathcal{A}_k$ to the SPASS Problem (12.1), and furthermore, the corresponding solutions have the same value. Hence, the SPASS problem is equivalent to solving

$$\mathcal{A}^* = \operatorname*{argmax}_{\mathcal{A}' \in \mathcal{I}} F'(\mathcal{A}').$$

As Fisher et al. [1978] proved, the greedy algorithm GAPS is guaranteed to obtain a solution that has at least 1/2 of the optimal value. □

Proof of Lemma 12.3. Consider an optimal allocation $\mathcal{T}_1, \ldots, \mathcal{T}_m$. Let $\mathcal{B}_{opt}$ be the set $\mathcal{B}_{opt} = \{i : \mathcal{T}_i \text{ contains a big element}\}$. Throw away all buckets (and elements) $\mathcal{B}_{opt}$. Now, in order to achieve score c, the optimal solution has to fill $m - |\mathcal{B}_{opt}|$ buckets with small elements (even from a reduced set of small elements, those not thrown away) and still achieve score c on each of those buckets. This solution is in fact an optimal solution achieving score c on the new problem instance (since we will use at least as many big elements and throw away at least as many buckets). □

Proof of Lemma 12.4. Suppose $\mathcal{A}_i$ is a bucket for which $F_c(\mathcal{A}_i) \geq 3\beta c$. Now, $\mathcal{A}_i = \{a_1, \ldots, a_l\}$, and $F_c(\{a_i\}) \leq \beta c$. Choose ℓ such that $F_c(\{a_1, \ldots, a_{\ell-1}\}) < \beta c$ and $F_c(\{a_1, \ldots, a_\ell\}) \geq \beta c$. Let $\Delta = \{a_1, \ldots, a_\ell\}$.

Due to nondecreasingness, $F_c(\mathcal{A}_j \cup \Delta) \geq F_c(\Delta) \geq \beta c$. It remains to show that $F_c(\mathcal{A}_i \setminus \Delta) \geq F_c(\mathcal{A}_i) - 2\beta c$. Suppose that $F_c(\mathcal{A}_i \setminus \Delta) < F_c(\mathcal{A}_i) - 2\beta c$. Let $\mathcal{B} = \mathcal{A}_i \setminus \Delta$. Then

$$F_c(\mathcal{B} \cup \Delta) - F_c(\mathcal{B}) > 2\beta c.$$

But

$$F_c(\Delta) \leq F_c(\{a_1, \ldots, a_{\ell-1}\}) + F_c(\{a_\ell\}) < 2\beta c,$$

due to submodularity of F_c, and the fact that a_ℓ is a small element. Hence

$$F_c(\mathcal{B} \cup \Delta) - F_c(\mathcal{B}) > F_c(\Delta) - F_c(\emptyset),$$

i.e., adding Δ to $\mathcal{B}$ helps more than adding Δ to the empty set, contradicting submodularity of F_c. □

Proof of Lemma 12.5. To simplify notation, w.l.o.g. let us assume that $c = 1$. Since the optimal balanced performance for $F_c = F_1$ is 1, the optimal average-case performance for F_1 is 1 as well. The GAPS algorithm obtains an allocation $\mathcal{A}$ that is a fraction α of optimal. Hence, it holds that $\sum_i F_1(\mathcal{A}_i) \geq \alpha k$. We call $\sum_i F_1(\mathcal{A}_i)$ the "mass" of the allocation $\mathcal{A}$.

How many unsatisfied buckets can there maximally be? Let γ denote the fraction of unsatisfied buckets. We know that

$$k\gamma\beta + k(1-\gamma) \geq \alpha k,$$

since the maximal γ is achieved if all the satisfied buckets are completely full (containing mass $k(1-\gamma)$), and the unsatisfied buckets are as full as possible without being satisfied (hence containing mass less than $k\gamma\beta$). Hence it follows that

$$\gamma \leq \frac{1-\alpha}{1-\beta}.$$

Now consider the mass R distributed over the satisfied buckets. We know that

$$R \geq \alpha k - \gamma k \beta \geq k \frac{\alpha(1-\beta) - \beta(1-\alpha)}{1-\beta},$$

and the worst case is assumed under equality.

The first reallocation move is possible if

$$\frac{R}{k(1-\gamma)} \geq 3\beta,$$

since, if the average remaining mass over all $(1-\gamma)k$ satisfied buckets is 3β, then there must be at least one bucket to which the move can be applied. Since each reallocation move reduces the mass R by at most 2β (as proved by Lemma 12.4), and since we need γk moves to fill all unsatisfied buckets, it suffices to require that

$$\begin{aligned}
&\frac{R - 2\gamma k\beta}{k(1-\gamma)} \geq 3\beta \\
\Leftrightarrow &R - 2\gamma k\beta \geq 3\beta k - 3\beta\gamma k \\
\Leftrightarrow &R \geq 3\beta k - \beta\gamma k
\end{aligned}$$

Hence, a sufficient condition for β such that enough moves can be performed to fill all unsatisfied buckets is

$$\begin{aligned}
&\frac{\alpha(1-\beta) - \beta(1-\alpha)}{1-\beta} \geq 3\beta - \beta\frac{1-\alpha}{1-\beta} \\
\Leftarrow &3\beta^2 + (-3-\alpha)\beta + \alpha \geq 0 \\
\Leftarrow &\beta \leq \alpha/3,
\end{aligned}$$

by solving the quadratic equation for β and ignoring the infeasible solutions $\beta \geq 1$. Now, since β is going to be our approximation factor, we want to maximize β subject to the above constraint, and hence choose $\beta = \alpha/3$. □

Proof of Theorem 12.2. The proof immediately follows from the analysis in Section 12.3.2. For the running time, notice that, in each binary search iteration, the greedy algorithm requires at most kmn function evaluations, and the reallocation step requires at most $k^2 m \leq kmn$ evaluations. The binary search terminates after $\mathcal{O}(1 + \log_2 F(\mathcal{V}))$ iterations, assuming integrality of F. □

Proof of Theorem 12.6. Let $\mathcal{A}_i$ be the candidate solution for time slot i, and $\mathcal{B}_i = \{b_1, \ldots, b_{n_i}\}$ an optimal solution for time slot i. Due to nondecreasingness and submodularity, it holds that

$$F(\mathcal{A}_i) \leq F(\mathcal{A}_i \cup \mathcal{B}_i) \leq F(\mathcal{A}_i) + \sum_{j=1}^{n_i} \delta_{i,b_j}.$$

Hence, the optimal value of Problem (12.2) is upper bounded by the optimal solution to the following integer program:

$$\begin{aligned}
&\max_{\lambda_{i,s}, c} c \text{ s.t.} \\
&\qquad c \leq F(\mathcal{A}_i) + \sum_s \lambda_{i,s}\delta_{i,s} \text{ for all } i \\
&\sum_i \lambda_{i,s} \leq 1 \text{ for all } s \text{ and } \sum_{i,s} \lambda_{i,s} \leq m \text{ and } \lambda_{i,s} \in \{0,1\},
\end{aligned}$$

since any integer solution $\lambda_{i,s}$ corresponds to a possible feasible partition $\mathcal{B}'_1, \ldots, \mathcal{B}'_k$. The linear program in Theorem 12.6 is the linear programming relaxation to the above integer program. □

Algorithm B.1: The greedy average-case placement and scheduling (MCGAPS) algorithm.

```
Algorithm MCGAPS (F, B, V, k, m, c, λ)
A_t ← ∅ for all t; A ← ∅;
for i = 1 to m do
    foreach s ∈ V \ A, 1 ≤ t ≤ k do
        δ_{t,s} ← λF_c(A_t ∪ {s}) + (1 − λ)F(A ∪ B ∪ {s});
    end
    (t*, s*) ← argmax_{t,s} δ_{t,s};
    A_{t*} ← A_{t*} ∪ {s*}; A ← A ∪ {s*};
end
```

Proof Sketch of Theorem 12.7. We will modify ESPASS in the following way. The modified algorithm, MCSPASS (see the pseudo code in Algorithm B.2), will "guess" (binary search for) the value c^* attained by the optimal solution $\mathcal{A}^*$. It will then guess (search for) the number ℓ of large elements used in the optimal solution, where we redefine "large" as $F(\{s\}) \geq \frac{c^*}{8}$. For such a guess, MCSPASS will first greedily select the ℓ large elements (according to F), giving a set $\mathcal{A}_G$. It will then set these large elements aside, and continue on the small elements. Define the function

$$G(\mathcal{A}_1, \ldots, \mathcal{A}_k) = (1-\lambda)(F(\mathcal{A}_G \cup \mathcal{A}) - F(\mathcal{A}_G)) + \frac{\lambda}{m-\ell} \sum_i \min\{F(\mathcal{A}_i), c\}$$

where $\mathcal{A} = \mathcal{A}_1 \cup \cdots \cup \mathcal{A}_k$. MCSPASS will greedily maximize G on the partition matroid (similarly to using GAPS). Suppose c^* is the optimal value $c^* = \widehat{F}_\lambda(\mathcal{A}^*)$. Then the greedy procedure will find a solution $\mathcal{A}'$ such that $G(\mathcal{A}') + (1-\lambda)F(\mathcal{A}_G) \geq \frac{1}{2}c^*$ (since greedy selection of big elements followed by greedy selection of small elements amounts to the "local" greedy optimization over a partition matroid as analyzed by Fisher et al. [1978]).

Now, at least one of $(1-\lambda)F(\mathcal{A}_G \cup \mathcal{A}') \geq c^*/8$, or $\frac{\lambda}{m-\ell}\sum_i F(\mathcal{A}'_i) \geq \frac{3c^*}{8}$, since

$$G(\mathcal{A}') + (1-\lambda)F(\mathcal{A}_G) = (1-\lambda)F(\mathcal{A}_G \cup \mathcal{A}') + \lambda \frac{1}{l-m} \sum_i F(\mathcal{A}'_i).$$

In former case we do not need to reallocate and set $\mathcal{A}_R = \mathcal{A}'$. In latter case, we use the reallocation procedure, and arrive at a solution $\mathcal{A}_R$ where all buckets are satisfied (since $\mathcal{A}'$ contains only small elements), i.e., $\min_i F(\mathcal{A}_{R,i}) \geq \frac{3c^*}{3 \cdot 8} = c^*/8$.

Now, $\mathcal{A}_R \cup \mathcal{A}_G$ is a feasible solution to the multicriterion SPASS problem, with

$$\begin{aligned}\widehat{F}_\lambda(\mathcal{A}_R \cup \mathcal{A}_G) &= (1-\lambda)F(\mathcal{A}_G) + G(\mathcal{A}_R) \\ &= (1-\lambda)F(\mathcal{A}_G \cup \mathcal{A}_R) + \lambda \min_i F(\mathcal{A}_{R,i}) \geq c^*/8.\end{aligned}$$

□

Algorithm B.2: The MCSPASS algorithm for simultaneously optimizing scheduled and high-density performance.

Algorithm MCSPASS (F, $\mathcal{V}$, k, m, ε, λ)
$c_{\min} \leftarrow 0$; $c_{\max} \leftarrow F(\mathcal{V})$; $\beta \leftarrow 1/8$;
while $c_{\max} - c_{\min} \geq \varepsilon$ **do**
  for $\ell = 0$ **to** k **do**
    $c \leftarrow (c_{\max} + c_{\min})/2$;
1   $\mathcal{B} \leftarrow \{s \in \mathcal{V} : F_c(\{s\}) \geq \beta c\}$;
    if $|\mathcal{B}| < \ell$ **then break**;
    $\mathcal{A}_{big} \leftarrow \emptyset$;
    for $i = 1$ **to** ℓ **do**
      $\mathcal{A}_{big} \leftarrow \operatorname{argmax}_{s \in \mathcal{B} \setminus \mathcal{A}_{big}} F(\mathcal{A}_{big} \cup \{s\})$;
      $\mathcal{A}_{k-i+1} \leftarrow \{s\}$;
    end
    $k' \leftarrow k - \ell$;
    if $k' = 0$ **then** $\mathcal{A}_{best,\ell} \leftarrow (\mathcal{A}_1, \ldots, \mathcal{A}_k)$;
      continue;
    $\mathcal{V}' \leftarrow \mathcal{V} \setminus \mathcal{A}_{big}$; $m' \leftarrow m - \ell$;
2   $\mathcal{A}_{1:k'} \leftarrow$MCGAPS $(F, \mathcal{A}_{big}, \mathcal{V}', k', m', c, \lambda)$;
3   **if** $\sum_t F(\mathcal{A}_t) < k'c/2$ **then** $c_{\max} \leftarrow c$; **continue**;
    else
4   **while** $\exists i, j \leq k'$: $F_c(\mathcal{A}_j) \leq \beta c$, $F_c(\mathcal{A}_i) \geq 3\beta c$ **do**
        foreach $s \in \mathcal{A}_i$ **do**
          $\mathcal{A}_j \leftarrow \mathcal{A}_j \cup \{s\}$; $\mathcal{A}_i \leftarrow \mathcal{A}_i \setminus \{s\}$;
          if $F_c(\mathcal{A}_j) \geq \beta c$ **then break**;
        end
      end
      $\mathcal{A}_{best,\ell} \leftarrow (\mathcal{A}_1, \ldots, \mathcal{A}_k)$;
    end
  end
  $\ell^* \leftarrow \operatorname{argmax}_\ell \widehat{F}_\lambda(\mathcal{A}_{best,\ell})$;
  $c_{\min} \leftarrow c$; $\mathcal{A}_{best} \leftarrow \mathcal{A}_{best,\ell^*}$;
end

B.8 Proofs from Chapter 14

Lemma B.7. *Let π be a policy. Then*

$$H(\mathcal{X}_\pi) \leq H(\mathcal{X}_\pi \mid \Theta) + H(\Theta),$$

$$F_{MI}(\mathcal{X}_\pi) \leq F_{MI}(\mathcal{X}_\pi \mid \Theta) + H(\Theta).$$

Proof. From definition, $F_{MI}(\mathcal{X}_\pi) = H(\mathcal{X}_{\mathcal{V}\setminus\pi}) - H(\mathcal{X}_{\mathcal{V}\setminus\pi} \mid \mathcal{X}_\pi)$. Using the chain rule, we find $F_{MI}(\mathcal{X}_\pi) = H(\mathcal{X}_\pi) - H(\mathcal{X}_\pi \mid \mathcal{X}_{\mathcal{V}\setminus\pi})$. Consider

$$\begin{aligned} H(\mathcal{X}_\pi, \Theta) &= -\sum_\theta \int p(\mathbf{x}_\mathcal{V}, \theta) \log p(\mathbf{x}_\pi, \theta) d\mathbf{x}_\mathcal{V} \\ &= -\sum_\theta \int p(\mathbf{x}_\mathcal{V}, \theta) [\log p(\mathbf{x}_\pi) + \log p(\theta \mid \mathbf{x}_\pi)] d\mathbf{x}_\mathcal{V} \\ &= H(\mathcal{X}_\pi) + \int p(\mathbf{x}_\mathcal{V}) H(\Theta \mid \mathbf{x}_\pi) d\mathbf{x}_\mathcal{V} \\ &\geq H(\mathcal{X}_\pi). \end{aligned}$$

Also,

$$\begin{aligned} H(\mathcal{X}_\pi, \Theta) &= -\sum_\theta \int p(\mathbf{x}_\mathcal{V}, \theta) [\log p(\mathbf{x}_\pi \mid \theta) + \log p(\theta)] d\mathbf{x}_\mathcal{V} \\ &= H(\Theta) - \sum_\theta \int p(\mathbf{x}_\mathcal{V}) \log p(\mathbf{x}_\pi \mid \theta) d\mathbf{x}_\mathcal{V} \\ &= H(\Theta) + H(\mathcal{X}_\pi \mid \Theta). \end{aligned}$$

Similarly, $H(\mathcal{X}_{\bar{\pi}}) \leq H(\mathcal{X}_{\bar{\pi}} \mid \Theta) + H(\Theta)$ (just replace π by $\bar{\pi}$ in the above proof).

Now consider

$$
\begin{aligned}
H(\mathcal{X}_{\bar{\pi}} \mid \mathcal{X}_{\pi}) &= -\int p(\mathbf{x}_{\mathcal{V}}) \log p(\mathbf{x}_{\bar{\pi}} \mid \mathbf{x}_{\pi}) d\mathbf{x}_{\mathcal{V}} \\
&= -\int p(\mathbf{x}_{\mathcal{V}}) \sum_{|A|=k} [\pi(\mathbf{x}_{\mathcal{V}}) = A] \log p(\mathbf{x}_{\bar{A}} \mid \mathbf{x}_{A}) d\mathbf{x}_{\mathcal{V}} \\
&= -\sum_{A} \int p(\mathbf{x}_{A})[\pi(\mathbf{x}_{A}) = A] \int p(\mathbf{x}_{\bar{A}} \mid \mathbf{x}_{A}) \log p(\mathbf{x}_{\bar{A}} \mid \mathbf{x}_{A}) d\mathbf{x}_{\mathcal{V}} \\
&= \sum_{A} \int p(\mathbf{x}_{A})[\pi(\mathbf{x}_{A}) = A] H(\mathcal{X}_{\bar{A}} \mid \mathbf{x}_{A}) d\mathbf{x}_{A} \\
&\geq \sum_{A} \int p(\mathbf{x}_{A})[\pi(\mathbf{x}_{A}) = A] H(\mathcal{X}_{\bar{A}} \mid \mathbf{x}_{A}, \Theta) d\mathbf{x}_{A} \\
&= \sum_{A} \int p(\mathbf{x}_{A})[\pi = A] \sum_{\theta} P(\theta \mid \mathbf{x}_{A}) H(\mathcal{X}_{\bar{A}} \mid \mathbf{x}_{A}, \theta) d\mathbf{x}_{A} \\
&= -\sum_{A,\theta} \int p(\mathbf{x}_{A}, \theta)[\pi = A] p(\mathbf{x}_{\bar{A}} \mid \mathbf{x}_{A}, \theta) \log p(\mathbf{x}_{\bar{A}} \mid \mathbf{x}_{A}, \theta) d\mathbf{x}_{\mathcal{V}} \\
&= -\sum_{A,\theta} \int p(\mathbf{x}_{\mathcal{V}}, \theta)[\pi = A] \log p(\mathbf{x}_{\bar{A}} \mid \mathbf{x}_{A}, \theta) d\mathbf{x}_{\mathcal{V}} \\
&= -\sum_{\theta} p(\theta) \int p(\mathbf{x}_{\mathcal{V}} \mid \theta) \log p(\mathbf{x}_{\bar{\pi}} \mid \mathbf{x}_{\pi}, \theta) d\mathbf{x}_{\mathcal{V}} \\
&= \sum_{\theta} P(\theta) H(\mathcal{X}_{\bar{\pi}} \mid \mathcal{X}_{\pi}, \theta) \\
&= H(\mathcal{X}_{\bar{\pi}} \mid \mathcal{X}_{\pi}, \Theta)
\end{aligned}
$$

Hence,

$$
\begin{aligned}
F_{MI}(\mathcal{X}_{\pi}) &= H(\mathcal{X}_{\bar{\pi}}) - H(\mathcal{X}_{\bar{\pi}} \mid \mathcal{X}_{\pi}) \\
&\leq H(\Theta) + H(\mathcal{X}_{\bar{\pi}} \mid \Theta) - H(\mathcal{X}_{\bar{\pi}} \mid \mathcal{X}_{\pi}, \Theta) \\
&= F_{MI}(\mathcal{X}_{\pi} \mid \Theta) + H(\Theta).
\end{aligned}
$$

□

Proof of Theorem 14.1. Using Lemma B.7, it suffices to show that

$$
\max_{|\pi|=k} H(\mathcal{X}_{\pi} \mid \Theta) \leq \sum_{\theta} P(\theta) \max_{|A|=k} H(\mathcal{X}_{A} \mid \theta),
$$

and

$$
\max_{|\pi|=k} F_{MI}(\mathcal{X}_{\pi} \mid \Theta) = \sum_{\theta} P(\theta) \max_{|A|=k} F_{MI}(\mathcal{X}_{A} \mid \theta).
$$

This follows from the fact that

$$H(\mathcal{X}_\pi \mid \Theta) = \sum_\theta P(\theta) H(\mathcal{X}_\pi \mid \theta) \leq \sum_\theta P(\theta) \max_{|\mathcal{A}|=k} H(\mathcal{X}_\mathcal{A} \mid \theta).$$

Similarly,

$$F_{MI}(\mathcal{X}_\pi \mid \Theta) = \sum_\theta P(\theta) F_{MI}(\mathcal{X}_\pi \mid \theta) \leq \sum_\theta P(\theta) \max_{|\mathcal{A}|=k} F_{MI}(\mathcal{X}_\mathcal{A} \mid \theta).$$

□

Proof of Corollary 14.3. We first observe that submodularity is closed under taking expectations, and $F_{MI}(\mathcal{X}_{\mathcal{A}\cup\{y\}} \mid \Theta) - F_{MI}(\mathcal{X}_\mathcal{A} \mid \Theta) \geq \sum P(\theta)[-\varepsilon] = -\varepsilon$ shows that $F_{MI}(\cdot \mid \Theta)$ is ε-nondecreasing. The result follows from combining the statements of Theorem 14.2 and Theorem 14.1. □

Proof of Proposition 14.8. Consider the sequence $H_i = H(\Theta \mid \mathcal{X}_{\pi_{1:i}})$. The information never hurts principle [Cover and Thomas, 1991] shows that $\mathbb{E}[H_{i+1} \mid \mathcal{X}_{\pi_{1:i}}] \leq H_i$ with probability 1 over the observations $\mathcal{X}_{\pi_{1:i}}$. Hence $(H_i)_i$ is a super-martingale, which proves that $H(\Theta \mid \mathcal{X}_{\pi_{1:i}}) = \mathbb{E}[H_i] \leq \mathbb{E}[H_0] = H(\Theta)$. □

Lemma B.8. *Let $\hat{\Theta}$ be the parameter identified by binary search procedure. Let $M = \lceil \log m \rceil$. If $P(T_i \neq I_i \mid \theta) \leq \alpha$, then for all θ, $P(\hat{\Theta} = \theta \mid \Theta = \theta) \geq 1 - M\alpha$.*

Proof of Lemma B.8.

$$\begin{aligned} P(\hat{\Theta} = \theta \mid \Theta = \theta) &= P(T^{(1)} = I^{(1)} \wedge \cdots \wedge T^{(M)} = I^{(M)} \mid \theta) \\ &= 1 - P(T^{(1)} \neq I^{(1)} \vee \cdots \vee T^{(M)} \neq I^{(M)} \mid \theta) \\ &\geq 1 - \sum_{i=1}^{M} P(T^{(i)} \neq I^{(i)} \mid \theta) \geq 1 - M\alpha, \end{aligned}$$

□

Proof of Theorem 14.5. Let C be a statistic of the T_i and I_i, such that $C = 1$ if $T_i = I_i$ for all i. From Lemma B.8, we have that $P(C = 1 \mid \theta) \geq 1 - \alpha/\lceil \log_2 m \rceil$. Hence also unconditionally $P(C = 1) \geq 1 - \alpha/\lceil \log_2 m \rceil$. From Corollary 14.3 we have that if we know the correct parameter, $F_{MI}(\mathcal{X}_{\mathcal{A}_G \cup \pi_H} \mid \Theta)] \geq (1 - 1/e) \max_{|\pi|=k} F_{MI}(\mathcal{X}_\pi) - k\varepsilon$. This event happens with probability $P(C = 1)$. If $C = 0$, we have identified the wrong parameter. In this event, we have that $F_{MI}(\mathcal{X}_{\mathcal{A}_G \cup \pi_H} \mid \Theta)] \geq (1 - 1/e) \sum_\theta P(\theta) F_{MI}(\mathcal{X}_{\mathcal{A}^\theta} \mid \theta) - k\varepsilon + H(\Theta \mid \mathcal{X}_{\pi_H})$. But $H(\Theta \mid \mathcal{X}_{\pi_H}) \leq \lceil \log_2 m \rceil$. Hence

$$\begin{aligned} \mathbb{E}_T[F_{MI}(\mathcal{X}_{\mathcal{A}_G \cup \pi_H} \mid \Theta)] &\geq (1 - 1/e) \max_{|\pi|=k} F_{MI}(\mathcal{X}_\pi) - k\varepsilon - \\ &- \left(0 \cdot P(C = 1) + \lceil \log_2 m \rceil P(C = 0) (\max_\theta |F_{MI}(\mathcal{X}_{\mathcal{A}_G \cup \pi_H} \mid \Theta) - F_{MI}(\mathcal{X}_{\mathcal{A}^\theta} \mid \theta)|) \right). \end{aligned}$$

□

Proof of Theorem 14.6. This asymptotic bound follows directly from confidence intervals obtained after applying Fisher's transform to the sample correlation coefficient. □

Lemma B.9. *Let $\mathcal{X}$ be an isotropic GP with variance σ^2, $\mathcal{X}_s, \mathcal{X}_t$ and $\mathcal{X}_\mathcal{A}$ be such that $\operatorname{Cor}(\mathcal{X}_s, \mathcal{X}_z) \leq \varepsilon$ and $\operatorname{Cor}(\mathcal{X}_t, \mathcal{X}_z) \leq \varepsilon$ for all $z \in \mathcal{A}$, and assume every location has independent measurement noise (nugget) σ_n^2. Then $|\operatorname{Cor}(\mathcal{X}_s, \mathcal{X}_t) - \operatorname{Cor}(\mathcal{X}_s, \mathcal{X}_t \mid \mathcal{X}_\mathcal{A})| \leq 2|\mathcal{A}|\varepsilon^2 \frac{\sigma^2}{\sigma_n^2}$.*

Proof. By induction on $|\mathcal{A}|$. First divide by σ, such that the process has unit variance. This does not change the correlation. Let $\mathcal{X}_z \in \mathcal{A}$. Then

$$\Sigma_{xy,xy|Z} = \Sigma_{xy,xy} - \Sigma_{xy,z}\sigma_{z,z}^{-1}\Sigma_{z,xy}.$$

Hence

$$||\Sigma_{xy,xy|Z} - \Sigma_{xy,xy}||_\infty = ||\Sigma_{xy,z}\sigma_{z,z}^{-1}\Sigma_{z,xy}||_\infty \leq 2\varepsilon^2 \frac{\sigma^2}{\sigma_n^2}.$$

The induction step uses the independence of the measurement noise, such that $\sigma^2_{z|\mathcal{A}'} \geq \sigma_n^2$. □

Proof of Corollary 14.7. This is a direct consequence of Theorem 14.6 and Lemma B.9, by observing that by ignoring n ε-correlated samples, the correlation does not change by more than $2n\varepsilon^2 \frac{\sigma^2}{\sigma_n^2}$. Since the correlation for θ_i cannot increase by and the correlation for θ_{i+1} cannot decrease by more than that amount, the gap in correlation decreases by at most $4n\varepsilon^2 \frac{\sigma^2}{\sigma_n^2}$. Requiring this to be less than ξ and solving for ε proves the claim. □

Proof of Theorem 14.4. Let $M = \max_A \max_{\theta_1,\theta_2} \frac{F_{MI}(A|\theta_1)}{F_{MI}(A|\theta_2)} < \infty$. Also let $K = \max_\theta F_{MI}(A_\theta \mid \theta)$, where $A_\theta = \operatorname{argmax}_A F_{MI}(A \mid \theta)$. Let $A^* = \operatorname{argmax}_A F_{MI}(A \mid \Theta)$. Let θ' be the most likely parameter, and assume $H(\Theta) < 1$, which implies $\delta < \frac{1}{2}$.

Let us show the lower bound first. Clearly, $F_{MI}(\mathcal{X}_{\pi^*}) \geq \max_\mathcal{A} F_{MI}(\mathcal{X}_\mathcal{A})$. Now, $F_{MI}(A) = H(A) - H(A \mid V \setminus A) \geq H(A \mid \Theta) - H(A, \Theta \mid V \setminus A)$, since Θ is discrete. But $H(A, \Theta \mid V \setminus A) = H(A \mid V \setminus A, \Theta) + H(\Theta \mid V \setminus A) \leq H(A \mid V \setminus A, \Theta) + H(\Theta)$, hence the lower bound follows. Note that if $\mathcal{A}$ is "small" compared to $\mathcal{V} \setminus \mathcal{A}$, then we can expect $H(\Theta \mid V \setminus A) \approx 0$.

$$\begin{aligned} F_{MI}(A^* \mid \Theta) &= (1-\delta)F_{MI}(A^* \mid \theta') + \delta F_{MI}(A^* \mid \neg\theta') \\ &\leq (1-\delta)F_{MI}(A^* \mid \theta') + M\delta F_{MI}(A^* \mid \theta') \\ &= (1 + (M-1)\delta)F_{MI}(A^* \mid \theta'), \end{aligned}$$

where

$$F_{MI}(A^* \mid \neg\theta') = \frac{1}{\delta} \sum_{\Theta \neq \theta'} P(\theta) F_{MI}(A^* \mid \theta).$$

We have that $F_{MI}(A^* \mid \Theta) \geq F_{MI}(A_{\theta'} \mid \Theta)$. Hence

$$\begin{aligned} F_{MI}(A^* \mid \theta') &\geq \frac{1}{1 + (M-1)\delta} F_{MI}(A_{\theta'} \mid \Theta) \\ &\geq \frac{1-\delta}{1 + (M-1)\delta} F_{MI}(A_{\theta'} \mid \theta'), \end{aligned}$$

where we use that $F_{MI}(A_{\theta'} \mid \neg\theta') \geq 0$.

Now define the loss

$$L(A) = \sum_{\theta} P(\theta) \left[\max_{A'} F_{MI}(A' \mid \theta) - F_{MI}(A \mid \theta) \right].$$

This loss is minimized by $A^* = \operatorname{argmax}_A F_{MI}(A \mid \Theta)$. Now,

$$\begin{aligned}
L(A^*) \leq & (1-\delta) \left[F_{MI}(A_{\theta'} \mid \theta') - F_{MI}(A^* \mid \theta') \right] \\
& + \delta(K - F_{MI}(A^* \mid \neg\theta')) \\
\leq & (1-\delta) \left[F_{MI}(A_{\theta'} \mid \theta') - \frac{1-\delta}{1+(M-1)\delta} F_{MI}(A_{\theta'} \mid \theta') \right] \\
& + \delta(K - F_{MI}(A^* \mid \neg\theta')) \\
= & \frac{(1-\delta)(M-1)\delta}{1+(M-1)\delta} F_{MI}(A_{\theta'} \mid \theta') + \delta(K - F_{MI}(A^* \mid \neg\theta')) \\
= & \delta \left[\frac{(1-\delta)(M-1)}{1+(M-1)\delta} F_{MI}(A_{\theta'} \mid \theta') + K - F_{MI}(A^* \mid \neg\theta') \right] \\
\leq & \delta \left[(M-1) F_{MI}(A_{\theta'} \mid \theta') + K \right] \\
\leq & \delta M K,
\end{aligned}$$

where we use that $F_{MI}(A^* \mid \neg\theta') \geq 0$. Hence $L(A^*)$ is $O(\delta)$. If $\delta < \frac{1}{2}$, then $\delta \leq \frac{H(\Theta)}{-\log_2 H(\Theta)}$.

We know

$$F_{MI}(A^* \mid \Theta) - H(\Theta) \leq F_{MI}(X_\pi) \leq F_{MI}(A^* \mid \Theta) + H(\Theta) + L(A^*).$$

If we approximate $F_{MI}(\mathcal{X}_{\pi^*})$ by $F_{MI}(\mathcal{X}_{\mathcal{A}^*}|\Theta)$, the absolute error is hence bounded by

$$H(\Theta) \left(1 + \frac{MK}{\log_2 \frac{1}{H(\Theta)}} \right).$$

□

Proof of Proposition 14.9. This proposition follows from the fact that the fully factorized distribution is the maximum entropy distribution with a specified set of marginal distributions. □

B.9 Proofs from Chapter 15

Proof of Proposition 15.1. Take the setup from the proof of Proposition 15.1, and add another Bernoulli random variable $\mathcal{Z}$ with $P(\mathcal{Z} = 1) = .5$ that is independent of $\mathcal{Y}$. Add two actions b_1 and b_{-1} for classifying the value of $\mathcal{Z}$, and a utility function that gives value ε if the correct action is chosen. Now let the set $\mathcal{V}$ of variables contain $\{\mathcal{X}_1, \mathcal{X}_2, \mathcal{Z}\}$, and let us allow ourselves to pick $k = 2$ elements of $\mathcal{V}$, and let $L(\mathcal{A})$ be the value of information for picking variables $\mathcal{A}$. Now, as shown in the proof of Proposition 15.1, $L(\{\mathcal{X}_1\}) = L(\{\mathcal{X}_2\}) = 0$, whereas $L(\{\mathcal{Z}\}) = \varepsilon$. Hence the greedy algorithm will choose to observe $\mathcal{Z}$ first. However, it can then only add either $\mathcal{X}_1$ or $\mathcal{X}_2$. However, since $\mathcal{Y}$ and $\mathcal{Z}$ are independent, $L(\{\mathcal{X}_i, \mathcal{Z}\}) = \varepsilon$, but $L(\{\mathcal{X}_1, \mathcal{X}_2\}) = \frac{6}{16}$. Hence, as $\varepsilon \to 0$, the greedy algorithm performs arbitrarily badly. □

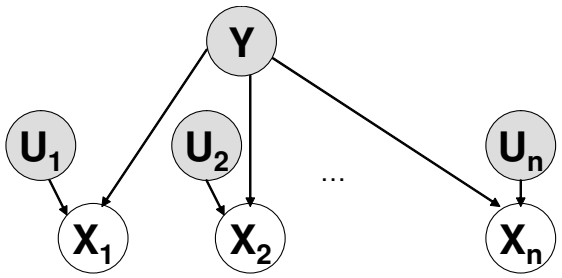


Figure B.1: Graphical model used in the proof of Theorem 15.5.

Proof of Proposition 15.2. Consider the following binary classification problem with assymetric cost. We have one Bernoulli random variable $\mathcal{Y}$ (the class label) with $P(\mathcal{Y} = 1) = 0.5$ and $P(\mathcal{Y} = -1) = 0.5$. We also have two noisy observations $\mathcal{X}_1, \mathcal{X}_2$, which are conditionally independent given $\mathcal{Y}$. Let $P(\mathcal{X}_i = \mathcal{Y}) = 3/4$ (i.e., the observations agree with the class label with probability 3/4, and disagree with probability 1/4. We have three actions, a_1 (classifying $\mathcal{Y}$ as 1), a_{-1} (classifying $\mathcal{Y}$ as -1) and a_0 (not assigning any label). We define our utility function F such that we gain utility 1 if we assign the label correctly ($F(a_1, 1) = F(a_{-1}, -1) = 1$), -3 is we misassign the label ($F(a_1, -1) = F(a_{-1}, 1) = -3$), and 0 if we choose a_0, i.e., not assign any label. Now, we can verify that $L(\emptyset) = L(\{\mathcal{X}_1\}) = L(\{\mathcal{X}_2\}) = 0$, but $L(\{\mathcal{X}_1, \mathcal{X}_2\}) = \left(\frac{3}{4}\right)^2 - 3\left(\frac{1}{4}\right)^2 = \frac{6}{16} > 0$. Hence, adding $\mathcal{X}_2$ to $\mathcal{X}_1$ increases the utility more than adding $\mathcal{X}_2$ to the empty set, contradicting submodularity. □

Proof of Theorem 15.5. Membership in $\#\mathbf{P}$ for arbitrary discrete polytrees is straightforward since inference in such models is in $\mathbf{P}$. Let ϕ be an instance of $\#3SAT$, where we have to count the number of assignments to $X_1, \ldots, X_n$ satisfying ϕ. Let $C = \{C_1, \ldots, C_m\}$ be the set of clauses. Now create a Bayesian network with $2n + 1$ variables, $\mathcal{X}_1, \ldots, \mathcal{X}_n, \mathcal{U}_1, \ldots, \mathcal{U}_n$ and $\mathcal{Y}$, where the $\mathcal{X}_i$ are conditionally independent given $\mathcal{Y}$. Let $\mathcal{Y}$ be uniformly distributed over the values $\{-n, -(n-1), \ldots, -1, 1, \ldots, m-1, m\}$, and each $\mathcal{U}_i$ have Bernoulli prior with $p = 0.5$. Let the observed variables $\mathcal{X}_i$ have CPTs defined the following way:

$$\mathcal{X}_i \mid [\mathcal{Y} = +j, \mathcal{U}_i = u] \sim \begin{cases} 1, & \text{if } X_i = u \text{ satisfies clause } C_j; \\ 0, & \text{otherwise.} \end{cases}$$

$$\mathcal{X}_i \mid [\mathcal{Y} = -j, \mathcal{U}_i = u] \sim \begin{cases} 0, & \text{if } i = j; \\ u, & \text{otherwise.} \end{cases}$$

In this model, which is presented in Figure B.1, it holds that $\mathcal{X}_1 = \mathcal{X}_2 = \cdots = \mathcal{X}_n = 1$ iff $\mathcal{U}_1, \ldots, \mathcal{U}_n$ encode a satisfying assignment of ϕ, and $\mathcal{Y} > 0$. Hence, if we observe $\mathcal{X}_1 = \mathcal{X}_2 = \cdots = \mathcal{X}_n = 1$, we know that $\mathcal{Y} > 0$ with certainty. Furthermore, if at least one $\mathcal{X}_i = 0$, we know that $P(\mathcal{Y} > 0 \mid \mathcal{X} = \mathbf{x}) < 1$. Let all nodes have zero reward, except for $\mathcal{Y}$, which is assigned a reward function with the following properties (we will show below how we can model such a local reward function using the decision-theoretic value of information):

$$R(\mathcal{Y} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \begin{cases} \frac{(n+m)2^n}{m}, & \text{if } P(\mathcal{Y} > 0 \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = 1; \\ 0, & \text{otherwise.} \end{cases}$$

By the above argument, the expected reward

$$R(\mathcal{Y} \mid \mathcal{X}_1, \ldots, \mathcal{X}_n) = \sum_{\mathbf{u},y,\mathbf{x}} P(\mathcal{Y} = y)P(\mathcal{U} = \mathbf{u})P(\mathbf{x}|\mathbf{u})R(\mathcal{Y} \mid \mathcal{X} = \mathbf{x})$$
$$= \sum_{\mathbf{u} \text{ sat } \phi} P(\mathcal{Y} > 0)P(\mathbf{u})\frac{(n+m)2^n}{m} = \sum_{u \text{ sat } \phi} 1$$

is exactly the number of satisfying assignments to ϕ. Note that the model defined above is not yet a Naive Bayes model. However, it can easily be turned into one by marginalizing out $\mathcal{U}$.

We will now show how we can realize a reward function with the above properties in the maximum expected utility sense. Let $D = \{d_1, d_2\}$ be a set of two decisions. Define a utility function with the property:

$$g(d, y) = \begin{cases} \frac{(n+m)2^n}{m}, & \text{if } d = d_1 \text{ and } y > 0; \\ \frac{(n+m)2^{2n+1}}{n}, & \text{if } d = d_1 \text{ and } y < 0; \\ 0, & \text{otherwise.} \end{cases}$$

The reward $R(\mathcal{Y} \mid \mathcal{X}_\mathcal{A})$ is then given as the decision-theoretic value of information:

$$R(\mathcal{Y} \mid \mathcal{X}_\mathcal{A}) = \sum_{\mathbf{x}_\mathcal{A}} P(\mathbf{x}_\mathcal{A}) \max_d \sum_y P(y \mid \mathbf{x}_\mathcal{A})g(d, y).$$

The utility function g is based on the following consideration. Upon observing a particular instantiation of the variables $\mathcal{X}_1, \ldots, \mathcal{X}_n$ we make a decision d about variable $\mathcal{Y}$. Our goal is to achieve that the number of times action d_1 is chosen exactly corresponds to the number of satisfying assignments to ϕ. This is accomplished in the following way. If all $\mathcal{X}_i$ are 1, then we know that the $\mathcal{U}_i$ had encoded a satisfying assignment, and $\mathcal{Y} > 0$ with probability 1. In this case, action d_1 is chosen. Now we need to make sure that whenever at least one $\mathcal{X}_i = 0$ (which indicates either that $\mathcal{Y} < 0$ or $\mathcal{U}$ is not a satisfying assignment) decision d_2 is chosen. Now, if at least one $\mathcal{X}_i = 0$, then either $\mathcal{Y} = j > 0$ and clause j was not satisfied, or $\mathcal{Y} < 0$. The utilities are designed such that unless $P(\mathcal{Y} > 0 \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) \geq 1 - \frac{n2^{-n}}{2m}$, the action d_2 gives the higher expected reward of 0. Hereby, $\frac{n2^{-n}}{2m}$ is a lower bound on the probability of "misclassification" $P(\mathcal{Y} < 0 \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$.

Note that the above construction immediately proves the hardness of approximation: Suppose there were a polynomial time algorithm which computes an approximation $\hat{R}$ that is within any factor $\alpha > 1$ (which can depend on the problem instance) of $R = R(\mathcal{Y} \mid \mathcal{X}_1, \ldots, \mathcal{X}_n)$. Then $\hat{R} > 0$ implies that $R > 0$, and $\hat{R} = 0$ implies that $R = 0$. Hence, the approximation $\hat{R}$ can be used to decide whether ϕ is satisfiable or not, implying that $\mathbf{P} = \mathbf{NP}$. □

Proof of Corollary 15.6. Let ϕ be a 3CNF formula. We convert it into a Naive Bayes model over variables $\mathcal{X}_1, \ldots, \mathcal{X}_n$ and $\mathcal{Y}$ as in the construction of Theorem 15.5. The function $L(\mathcal{V})$ where $\mathcal{V} = \{1, \ldots, n\}$ is the set of all variables $\mathcal{X}_i$ counts the number of satisfying assignments to ϕ. Note that the function $L(\mathcal{A})$ for $\mathcal{A} \subseteq \mathcal{V} = \{1, \ldots, n\}$ is monotonic, i.e., $L(\mathcal{A}) \leq L(\mathcal{V})$ for all $\mathcal{A} \subseteq \mathcal{V}$, as shown in Proposition 15.10. Hence the majority of assignments satisfies ϕ if and only if $L(\mathcal{V}) > 2^{n-1}$. □

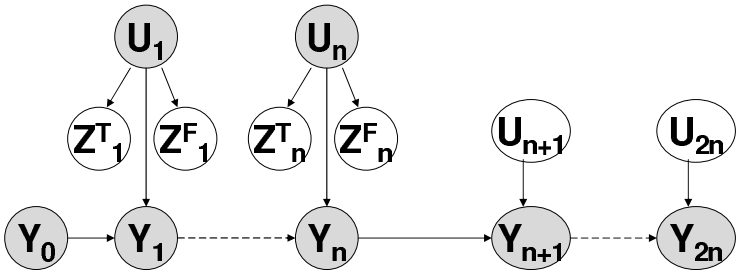


Figure B.2: Graphical model used in proof of Theorem 15.7.

Proof of Theorem 15.7. Membership follows from the fact that inference in polytrees is in $\mathbf{P}$ for discrete polytrees: A nondeterministic Turing machine with $\#\mathbf{P}$ oracle can first "guess" the selection of variables, then compute the value of information using Theorem 15.5 (since such computation is $\#\mathbf{P}$-complete for arbitrary discrete polytrees), and compare against constant c.

To show hardness, let ϕ be an instance of $EMAJSAT$, where we have to find an instantiation of $X_1, \ldots, X_n$ such that $\phi(X_1, \ldots, X_{2n})$ is true for the majority of assignments to $X_{n+1}, \ldots, X_{2n}$. Let $C = \{C_1, \ldots, C_m\}$ be the set of 3CNF clauses. Create the Bayesian network shown in Figure B.2, with nodes $\mathcal{U}_i$, each having a uniform Bernoulli prior. Add bivariate variables $\mathcal{Y}_i = (sel_i, par_i)$, $0 \leq i \leq 2n$, where sel_i takes values in $\{0, \ldots, m\}$ and par_i is a parity bit. The CPTs for $\mathcal{Y}_i$ are defined as: sel_0 uniformly varies over $\{1, \ldots, m\}$, $par_0 = 0$, and for $\mathcal{Y}_1, \ldots, \mathcal{Y}_{2n}$:

$$sel_i \mid [sel_{i-1} = j, \mathcal{U}_i = u_i] \sim \begin{cases} 0, & \text{if } j = 0, \text{ or } u_i \text{ satisfies } C_j; \\ j, & \text{otherwise;} \end{cases}$$
$$par_i \mid [par_{i-1} = b_{i-1}, \mathcal{U}_i] \sim b_{i-1} \oplus \mathcal{U}_i,$$

where $\oplus$ denotes the parity (XOR) operator.

We now add variables $\mathcal{Z}_i^T$ and $\mathcal{Z}_i^F$ for $1 \leq i \leq n$ and let

$$\mathcal{Z}_i^T \mid [\mathcal{U}_i = u_i] \sim \begin{cases} \text{Uniform}(\{0, 1\}), & \text{if } u_i = 1; \\ 0, & \text{otherwise;} \end{cases}$$

where Uniform denotes the uniform distribution. Similarly, let

$$\mathcal{Z}_i^F \mid [\mathcal{U}_i = u_i] \sim \begin{cases} \text{Uniform}(\{0, 1\}), & \text{if } u_i = 0; \\ 0, & \text{otherwise.} \end{cases}$$

Intuitively, $\mathcal{Z}_i^T = 1$ guarantees us that $\mathcal{U}_i = 1$, whereas $\mathcal{Z}_i^T = 0$ leaves us uncertain about $\mathcal{U}_i$. The case of $\mathcal{Z}_i^F$ is symmetric.

We use the subset selection algorithm to choose the $\mathcal{Z}_i$s that encode the solution to $EMAJSAT$. If $\mathcal{Z}_i^T$ is chosen, it will indicate that X_i should set to true, similarly $\mathcal{Z}_i^F$ indicates a false assignment to X_i. The parity function is going to be used to ensure that exactly one of $\{\mathcal{Z}_i^T, \mathcal{Z}_i^F\}$ is observed for each i.

We first assign penalties ∞ to all nodes except $\mathcal{Z}_i^T, \mathcal{Z}_i^F$ for $1 \leq i \leq n$, and $\mathcal{U}_j$ for $n+1 \leq j \leq 2n$, which are assigned zero penalty. Let all nodes have zero reward, except for $\mathcal{Y}_{2n}$, which is assigned the following reward:

$$R(\mathcal{Y}_{2n} \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = \begin{cases} 4^n, & \text{if } P(sel_{2n} = 0 \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = 1 \text{ and} \\ & [P(par_{2n} = 1 \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = 1 \text{ or } \; P(par_{2n} = 0 \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) = 1]; \\ 0, & \text{otherwise.} \end{cases}$$

Note that $sel_{2n} = 0$ with probability 1 iff $\mathcal{U}_1, \ldots, \mathcal{U}_{2n}$ encode a satisfying assignment of ϕ. Furthermore, we get positive reward only if we are both certain that $sel_{2n} = 0$, i.e., the chosen observation set must contain a proof that ϕ is satisfied, and we are certain about par_{2n}. The parity certainty will only occur if we are certain about the assignment $\mathcal{U}_1, \ldots, \mathcal{U}_{2n}$. It is only possible to infer the value of each $\mathcal{U}_i$ with certainty by observing one of $\mathcal{U}_i, \mathcal{Z}_i^T$ or $\mathcal{Z}_i^F$. Since, for $i = 1, \ldots, n$, the cost of observing $\mathcal{U}_i$ is ∞, to receive any reward we must observe at least one of $\mathcal{Z}_i^T$ or $\mathcal{Z}_i^F$. Assume that we compute the optimal subset $\hat{\mathcal{O}}$ for budget $2n$, then we can only receive positive reward by observing exactly one of $\mathcal{Z}_i^T$ or $\mathcal{Z}_i^F$.

We interpret the selection of $\mathcal{Z}_i^T$ and $\mathcal{Z}_i^F$ as an assignment to the first n variables of $EMAJSAT$. Let $\hat{R} = R(\mathcal{Y}_{2n} \mid \hat{\mathcal{O}})$. We claim that $\phi \in EMAJSAT$ if and only if $\hat{R} > 0.5$. First let $\phi \in EMAJSAT$, with assignment $x_1, \ldots, x_n$ to the first n variables. Now add $\mathcal{U}_{n+1}, \ldots, \mathcal{U}_{2n}$ to $\mathcal{O}$ and add $\mathcal{Z}_i^T$ to $\mathcal{O}$ iff $x_i = 1$ and $\mathcal{Z}_i^F$ to $\mathcal{O}$ iff $x_i = 0$. This selection guarantees $\hat{R} > 0.5$.

Now assume $\hat{R} > 0.5$. We call an assignment to $\mathcal{U}_1, \ldots, \mathcal{U}_{2n}$ *consistent* if for any $1 \leq i \leq n$, if $\mathcal{Z}_i^T \in \hat{\mathcal{O}}$, then $\mathcal{U}_i = 1$ and if $\mathcal{Z}_i^F \in \hat{\mathcal{O}}$ then $\mathcal{U}_i = 0$. For any consistent assignment, the chance that the observations $\mathcal{Z}_i$ prove the consistency is 2^{-n}. Hence $\hat{R} > 0.5$ implies that the majority of all provably consistent assignments satisfy ϕ and hence $\phi \in EMAJSAT$. This proves that subset selection is $\mathbf{NP^{PP}}$ complete.

Note that we can realize the local reward function R in the sense of maximum expected utility similarly as described in the Proof of Theorem 15.5. □

Proof of Corollary 15.8. The constructions in the proof of Theorem 15.6 and Theorem 15.7 also prove that computing conditional plans is $\mathbf{PP}$-hard and $\mathbf{NP^{PP}}$-hard respectively, since, in these instances, any plan with positive reward must observe variables corresponding to valid instantiations (i.e., all $\mathcal{X}_1, \ldots, \mathcal{X}_n$ in Corollary 15.6, and all $\mathcal{U}_{n+1}, \ldots, \mathcal{U}_{2n}$ and one each of the $\mathcal{Z}_1, \ldots, \mathcal{Z}_n$ to satisfy the parity condition in Theorem 15.7). In these cases, the order of selection is irrelevant, and, hence, the conditional plan effectively performs subset selection. □

Proof of Corollary 15.9. The proof follows from the observation that polytree construction from the proof of Theorem 15.7 can be arranged into two dependent chains. For this transformation, we revert the arc between $\mathcal{Z}_i^T$ and $\mathcal{U}_i$ by applying Bayes' rule. To make sure there are the same number of nodes for each sensor in each timeslice, we triple the variables $\mathcal{Y}_i$, calling the copies $\mathcal{Y}_i'$ and $\mathcal{Y}_i''$. The conditional probability tables are given as equality constraints, $\mathcal{Y}_i' = \mathcal{Y}_i$ and $\mathcal{Y}_i'' = \mathcal{Y}_i'$. After this transformation, the variables associated with timesteps $3i - 2$ (for $i \geq 1$) are given by the sets $\{\mathcal{Y}_{i-1}'', \mathcal{Z}_i^T\}$. timesteps $3i - 1$ are associated with the sets $\{\mathcal{U}_i, \mathcal{Y}_i\}$, and timesteps $3i$ are associated with $\{\mathcal{Z}_i^F, \mathcal{Y}_i'\}$. □

Proof of Proposition 15.10. This bound follows from the fact that maximization over $\mathbf{a}$ is convex, and an application of Jensen's inequality. Using an induction argument, we simply need to show that $L(\mathcal{A}) \geq L(\emptyset)$.

$$
\begin{aligned}
L(\mathcal{A}) &= \sum_{\mathbf{x}_\mathcal{A}} P(\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) \left(\sum_{t \in \mathcal{V}} \max_a EU(a, t, \mathbf{x} \mid \mathcal{X}_{\mathcal{A}^{(1:t)}} = \mathbf{x}_{\mathcal{A}^{(1:t)}}) \right) \\
&\geq \sum_{t \in \mathcal{V}} \max_a \left(\sum_{\mathbf{x}_\mathcal{A}} P(\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}) EU(a, t, \mathbf{x} \mid \mathcal{X}_{\mathcal{A}^{(1:t)}} = \mathbf{x}_{\mathcal{A}^{(1:t)}}) \right) \\
&= \sum_{t \in \mathcal{V}} \max_a EU(\mathbf{a}, t, \mathbf{x}) = L(\emptyset)
\end{aligned}
$$

where

$$EU(a, t, \mathbf{x} \mid \mathcal{X}_{\mathcal{A}^{(1:t)}} = \mathbf{x}_{\mathcal{A}^{(1:t)}}) = \sum_{x_t} P(x_t \mid \mathcal{X}_{\mathcal{A}^{(1:t)}} = \mathbf{x}_{\mathcal{A}^{(1:t)}}) F_t(a, x_t)$$

is the expected utility of action a at time t after observing $\mathcal{X}_{\mathcal{A}^{(1:t)}} = \mathbf{x}_{\mathcal{A}^{(1:t)}}$. □

Bibliography

P. Abrahamsen. A review of Gaussian random fields and correlation functions. Technical Report 917, Norwegian Computing Center, Oslo, 1997. A.1

Z. Abrams, A. Goel, and S. Plotkin. Set k-cover algorithms for energy efficient monitoring in wireless sensor networks. In *IPSN*, 2004. 1.1.1, 3.2.1, 5.1.1, 12, 12, 1, 12.2.1, 12.2.2, 12.3, 12.5.5, 12.6.2, 12.7

AHCPR. Pressure ulcers in adults: Prediction and prevention, clinical practice guideline. *Agency for Health Care Policy and Research Publication no. 92-0047*, 3, 1992. 7.7

Barbara M. Anthony, Vineet Goyal, Anupam Gupta, and Viswanath Nagarajan. A plant location guide for the unsure. In *Submitted to SODA*, 2008. 10.10.3

Aaron Archer. Inapproximability of the asymmetric facility location and k-median problems. unpublished manuscript, available from the authors web-page, 2000. 8.4.2

Arash Asadpour and Amin Saberi. An approximation algorithm for max-min fair allocation of indivisible goods. In *STOC*, pages 114–121, New York, NY, USA, 2007. ACM Press. ISBN 978-1-59593-631-8. doi: http://doi.acm.org/10.1145/1250790.1250808. 10.10.4, 12.6.3

A. C. Atkinson. Recent developments in the methods of optimum and related experimental designs. *International Statistical Review / Revue Internationale de Statistique*, 56(2):99–115, Aug. 1988. 3.1.2, 3.1.2

A. C. Atkinson. The usefulness of optimum experimental designs. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):59–76, 1996. 3.1.2

Igor Averbakh. On the complexity of a class of combinatorial optimization problems with uncertainty. *Mathematical Programming*, 90:263–272, 2001. 10.10.1

Baruch Awerbuch, Yossi Azar, Avrim Blum, and Santosh Vempala. New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen. *SIAM J. Computing*, 28:254–262, 1999. 11.5.3

S. Axelrod, S. Fine, R. Gilad-Bachrach, R. Mendelson, and N. Tishby. The information of observations and application for active learning with uncertainty. Technical report, Jerusalem: Leibniz Center, Hebrew University, 2001. 3.3.2, 10.10.5

X. Bai, S. Kumar, Z. Yun, D. Xuan, and T. H. Lai. Deploying wireless sensors to achieve both coverage and connectivity. In *ACM International Symposium on Mobile Ad Hoc Networking and Computing*, Florence, Italy, 2006. 1.1.1, 3.1.1, 5.1.1, 6.5.2, 11.8.3, 12.6.1

N. Bailey. *The Mathematical Theory of Infectious Diseases and its Applications*. Griffin, London, 1975. 8.4.2

N. Balcan, A. Beygelzimer, and J. Langford. Agnostic active learning. In *ICML*, 2006. 3.3.2, 10.10.5, 15.7.4

J. Bar-Ilan, G. Kortsarz, and D. Peleg. Generalized submodular cover problems and applications. *Theoretical Computer Science*, 250(1-2):179–200, January 2001. 10.11

G. I. Bardsley. The dundee seating programme. *Physiotheraphy*, 70(2):59–63, 1984. 7.7.3

L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Stat*, 37:1554–1563, 1966. 15

Valentina Bayer-Zubek. Learning diagnostic policies from examples by systematic search. In *UAI*, 2004. 2.2.1, 3.2.3, 7

R. Bellman. *Dynamic Programming*. Princeton Univ. Press, 1957a. 7.7.1

R. Bellman. A markovian decision process. *Journal of Mathematics and Mechanics*, 6, 1957b. 15.3.2

J. F. Benders. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik*, 4:238–252, 1962. 10.9.2

J. Berger. *Robustness of Bayesian Analyses*, chapter The robust Bayesian viewpoint, page 63?144. North-Holland, 1984. 10.10.2

Tanya Y. Berger-Wolf, William E. Hart, and Jared Saia. Discrete sensor placement problems in distribution networks. *Journal of Mathematical and Computer Modelling*, 42(13):1385–1396, December 2005. 10.10.3

J. M. Bernardo. Expected information as expected utility. *Annals of Statistics*, 7(3):686–690, May 1979. 3.1.2

J. Berry, W. Hart, and et.al. A facility location approach to sensor placement optimization. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006a. 8.2.4

J. Berry, W. E. Hart, C. E. Phillips, J. G. Uber, and J. Watson. Sensor placement in municipal water networks with temporal integer programming models. *J. Water Resources Planning and Management*, 2006b. 8.4.2

Dimitris Bertsimas and Melvyn Sim. Robust discrete optimization and network flows. *Mathematical Programming*, 98:49–71, 2003. 10.10.1

S. Bikhchandani, D. Hirshleifer, and I. Welch. A theory of fads, fashion, custom, and cultural change as informational cascades. *J. of Polit. Econ.*, 100(5):992–1026, October 1992. 8

Mustafa Bilgic and Lise Getoor. Voila: Efficient feature-value acquisition for classification. In *Twenty-Second Conference on Artificial Intelligence (AAAI)*, 2007. 15.7.2

Avrim Blum, Shuchi Chawla, David R. Karger, Terran Lane, Adam Meyerson, and Maria Minkoff. Approximation algorithms for orienteering and discounted-reward tsp. In *FOCS*, page 46, 2003. 10.6.4

Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004. 2.5.1, 2.5.1, 2.5.1, 3.1.2, 3.2.2, 6.6.5, 8.1.4, 10.5.3, 10.6.5, 10.7.2, 15.7.1

Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Uncertainty in Artificial Intelligence (UAI)*, 1998. 15.6.1

G. H. Brandeis, W. L. Ooi, M. Hossain, J. N. Morris, and L. A. Lipsitz. A longitudinal study of risk factors associated with the formation of pressure ulcers in nursing homes. *J. Am. Geriatr. Soc.*, 42(4), 1994. 7.7

D. M. Brienza, P.E. Karg, and C.E. Brubaker. Seat cushion design for elderly wheelchair users based onminimization of soft tissue deformation using stiffness and pressuremeasurements. *IEEE Transactions on Rehabilitation Engineering*, 4(4):320–327, December 1996. 7.7.3

T.M. Burgess, R. Webster, and A.B. McBratney. Optimal interpolation and isarithmic mapping of soil properties. iv. sampling strategy. *Journal of Soil Science*, 32:643–659, 1981. 10.10.2

G. Calinescu and A. Zelikovsky. The polymatroid steiner tree problems. *Journal of Combinatorial Optimization*, 3:281–294, 2005. 11.8.4

Emmanuel Candès and Michael Wakin. An introduction to compressive sampling. *IEEE Signal Processing Magazine*, 25(2):21 – 30, March 2008. 3.2.2

Emmanuel Candès, Justin Romberg, and Terence Tao. Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information. *IEEE Trans. on Information Theory*, 52(2): 489 – 509, February 2006. 3.2.2

J. F. Canny. A computational approach to edge detection. *IEEE Trans. Patt. Analysis and Machine Intelligence*, 8(6):679–698, 1986. 1

R. D. Carr, H. J. Greenberg, W. E. Hart, G. Konjevod, E. Lauer, H. Lin, T. Morrison, and C. A. Phillips. Robust optimization of contaminant sensor placement for community water systems. *Mathematical Programming Series B*, 107:337–356, 2006. 10.10.3

W. F. Caselton and T. Hussain. Hydrologic networks: Information transmission. *Journal of Water Resources Planning and Management*, WR2:503–520, 1980. 3.1.2

W. F. Caselton and J. V. Zidek. Optimal monitoring network designs. *Statistics and Probability Letters*, 2 (4):223–227, 1984. 2.3.1, 3.1.2, 3.1.2, 3.1.2, 3.2.1, 6, 6.2.2, 6.5.1, 11.8.1, 14.2.2

W. F. Caselton, L. Kan, and J. V. Zidek. *Statistics in the Environmental and Earth Sciences*, chapter Quality data networks that minimize entropy, pages 10–38. Halsted Press, 1992. 3.1.2, 3.1.2, 6.2.2, 6.5.4, 6.7

R. Castro, R. Willett, and R. Nowak. Faster rates in regression via active learning. In *NIPS*, 2005. 3.3.2

Alberto Cerpa, Jennifer L. Wong, Louane Kuang, Miodrag Potkonjak, and Deborah Estrin. Statistical model of lossy links in wireless sensor networks. In *IPSN*, 2005. 11, 11.8.3

K. Chaloner and I. Verdinelli. Bayesian experimental design: A review. *Statistical Science*, 10(3):273–304, Aug. 1995. ISSN 08834237. 2.3.1, 2.3.2, 3.1.2, 10.5.3, 10.10.2, 11.8.1, 12.1.1, 12.6.1

Moses Charikar and Sudipto Guha. Improved combinatorial algorithms for facility location problems. *SIAM Journal on Computing*, 34(4):803–824, 2005. 8.4.2

Chandra Chekuri and Martin Pal. A recursive greedy algorithm for walks in directed graphs. In *FOCS*, pages 245–253, 2005. 11.7, 11.8.4

Chandra Chekuri, Nitish Korula, and Martin Pal. Improved algorithms for orienteering and related problems. In *SODA*, 2008. 11.7, B.6

S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to nonlinear system identification. *Intl. J. Contr.*, 50(5):1873–1896, 1989. 3.2.2

J. Chuzhoy, S. Guha, E. Halperin, S. Khanna, G. Kortsarz, R. Krauthgamer, and J. Naor. Asymmetric k-center is $\log^* n$-hard to approximate. *Journal of the ACM*, 52(4):538–551, 2005. 10.10.3

D. Cohen. An objective measure of seat comfort. *Journal of Aviation, Space, and Environmental Medicine*, 69(4):410–414, April 1998. 7.7.3

R. Cohen, S. Havlin, and D. ben Avraham. Efficient immunization strategies for computer networks and populations. *Physical Review Letters*, 91:247901, 2003. 8.4.2

D. A. Cohn. Neural network exploration using optimal experiment design. In Jack D. Cowan, Gerald

Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 679–686. Morgan Kaufmann Publishers, Inc., 1994. 3.3.2, 10.10.5

David A. Cohn, Zoubin Gharamani, and Michael I. Jordan. Active learning with statistical models. *J AI Research*, 4:129–145, 1996. 15.7.4

CoNLL. Conference on computational natural language learning shared task. http://cnts.uia.ac.be/conll2003/ner/, 2003. 15.5.3

R. D. Cook and C. J. Nachtsheim. A comparison of algorithms for constructing exact D-optimal designs. *Technometrics*, 22(3):315–324, Aug. 1980. ISSN 00401706. 3.2.1

R. Cooper, J. Gonzales, B. Lawrence, A. Renschler, M. Boninger, and D. VanSickle. Performance of selected lightweight wheelchairs on ansi/resna tests. *Archives of Medicine and Rehabilitations*, 78(10): 1138–1144, 1997. 7.7.3

T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley Interscience, 1991. 2.3.1, 6.3.2, 7.1, 7.1, 7.2.1, 14.4.2, 15.1.1, B.8

N. A. C. Cressie. *Statistics for Spatial Data*. Wiley, 1991. 6, 6.1.2, 6.2.1, 6.6.2, 11.8.1, 12.1.1, 12.6.1, 14, 14.1, A.1

L. Csato, E. Fokue, M. Opper, B. Schottky, and O. Winther. Efficient approaches to gaussian process classification. In *NIPS*, 2000. 11.2

C. Currin, T. Mitchell, M. Morris, and D. Ylvisaker. Bayesian prediction of deterministic functions, with applications to the design and analysis of computer experiments. *Journal of the American Statistical Association*, 86(416):953–963, 1991. 3.1.2, 6.2.1

M. J. Daniels and R. E. Kass. Shrinkage estimators for covariance matrices. *Biometrics*, 57:1173–1184, December 2001. A.2

A. Das and D. Kempe. Algorithms for subset selection in linear regression. In *Symposium on the Theory of Computing*, 2008. 10.1.1, 10.5.1, 10.5.3, 10.7.1, 10.7.2, 12.1.1

S. Dasgupta. Coarse sample complexity bounds for active learning. In *NIPS*, 2005. 3.3.2, 15.7.4

S. Dasgupta, D.J. Hsu, and C. Monteleoni. A general agnostic active learning algorithm. In *NIPS*, 2007. 3.3.2

Sanjoy Dasgupta. An analysis of a greedy active learning strategy. In *NIPS*, 2004. 3.3.2

Michiel P. de Looze, Lottie F. M. Kuijt-Evers, and Jaap van Dieen. Sitting comfort and discomfort and the relationships with objective measures. *Ergonomics*, 46(10):985–997, August 2003. 7.7

Amol Deshpande, Carlos Guestrin, Sam Madden, Joseph Hellerstein, and Wei Hong. Model-driven data acquisition in sensor networks. In *VLDB*, 2004. 2.2.1, 6.1.1, 7.1, 11, 12.1.1, 12.5.5, 15, 15.5.1, 15.6

Amol Deshpande, Samir Khuller, Azarakhsh Malekian, and Mohammed Toossi. Energy efficient monitoring in sensor networks. In *LATIN*, 2008. 1.1.1, 3.2.1, 5.1.1, 12, 12, 12.5.5, 12.6.2

A. Lahad; A. D. Malter; A. O. Berg; R. A. Deyo. The effectiveness of four interventions for the prevention of low back pain. *JAMA*, 272(1286-1291), 1994. 7.7

C. DiSalvo, J. Forlizzi, J. Zimmerman, B. Mutlu, and A. Hurst. The sensechair: The lounge chair as an intelligent assistive device for elders. In *Proceedings of the Conference on Designing for User Experiences (DUX'05)*, Fort Mason, San Francisco, CA, USA, 2005. 7.7.3

S. Dittmer and F. Jensen. Myopic value of information in influence diagrams. In *UAI*, pages 142–149, San Francisco, 1997. 3.1.2, 7, 15, 15.7.2

Pedro Domingos and Michael Pazzani. On the optimality of the simple bayesian classifier under zero-one loss. *Machine Learning*, 29:103–137, 1997. 15, 15.4.2

David Donoho. Compressed sensing. *IEEE Trans. on Information Theory*, 52(4):1289 – 1306, April 2006. 3.2.2

G. Dorini, P. Jonkergouw, and et.al. An efficient algorithm for sensor placement in water distribution systems. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006. 8.4.2

H. A. Dror and D. M. Steinberg. Robust experimental design for multivariate generalized linear models. *Technometrics*, 48(4):520–529, 2006. 10.10.2

Marco Duarte, Mark Davenport, Dharmpal Takhar, Jason Laska, Ting Sun, Kevin Kelly, and Richard Baraniuk. Single-pixel imaging via compressive sampling. *IEEE Signal Processing Magazine*, 25(2): 83–91, March 2008. 3.2.2

R. M. Dudley. *Real Analysis and Probability*. Cambridge University Press, 2 edition, 2003. 2

Richard Durbin, Sean R. Eddy, Anders Krogh, and Graeme Mitchison. *Biological Sequence Analysis : Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1999. 15.5.2

Emre Ertin. Gaussian process models for censored sensor readings. In *IEEE/SP 14th Workshop on Statistical Signal Processing*, 2007. 11.8.3

V. Federov and W. Mueller. Comparison of two approaches in the optimal design of an observation network. *Statistics*, 20:339–351, 1989. 3.1.2

V. V. Fedorov. *Theory of Optimal Experiments*. Academic Press, 1972. Trans. W. J. Studden and E. M. Klimko. 3.2.1

U. Feige, V. Mirrokni, and J. Vondrak. Maximizing non-monotone submodular functions. In *FOCS*, 2007. 10.10.4, 16.2

Uriel Feige. A threshold of ln n for approximating set cover. *Journal of the ACM*, 45(4):634 – 652, July 1998. 5.2, 5.3, 5.8, 5.2.3, 10.3.2, 10.4, B.3, B.5

M. L. Fisher, G. L. Nemhauser, and L. A. Wolsey. An analysis of approximations for maximizing submodular set functions - ii. *Math. Prog. Study*, (8):73–87, 1978. 5.6, 5.12, 12.2.1, 12.6.3, B.7, B.7

A. W. Fitzgibbon, M. Pilu, and R. B. Fisher. Direct least-squares fitting of ellipses. *IEEE T-PAMI*, 21(5): 476–480, May 1999. 4

P. Flaherty, M. Jordan, and A. Arkin. Robust design of biological experiments. In *Advances in Neural Information Processing Systems (NIPS) 19*, 2006. 6.5.4, 10.5.3, 10.5.3, 10.7.2, 10.9, 10.7.2, 10.10.2

Y. Freund, H. S. Seung, E. Shamir, and N. Tishby. Selective sampling using the query by committee algorithm. *Machine Learning*, 28(2-3):133–168, 1997. 3.3.2, 10.10.5

J. Friedman, T. Hastie, and R. Tibshirani. Additive logistic regression: a statistical view of boosting, 1998. URL `citeseer.ist.psu.edu/friedman98additive.html`. 7.7.1

H.T. Friis. A note on a simple transmission formula. *Proc IRE*, 1946. 11.8.3

S. Fujishige. *Submodular Functions and Optimization*. Elsevier, 2 edition, 2005. 5.1, 5.8

S. Fujishige. Polymatroidal dependence structure of a set of random variables. *Inform. Contr.*, 39:55–72, 1978. 7.2.1

Toshihiro Fujito. Approximation algorithms for submodular set cover with applications. *TIEICE: IEICE Transactions on Communications/Electronics/Information and Systems*, 2000. 5.2.3, 10.3.2, 10.11,

11.6, 12.3.1

Stefan Funke, Alex Kesselman, Fabian Kuhn, Zvi Lotker, and Michael Segal. Improved approximation algorithms for connected sensor cover. In *ADHOC*, 04. 11, 11.8.2

Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Co., 2003. B.4

Naveen Garg. Saving an epsilon: a 2-approximation for the k-mst problem in graphs. In *STOC*, 2005. 4, 11.4.4, B.6

G. Giakkoupis, A. Gionis, E. Terzi, and P. Tsaparas. Models and algorithms for network immunization. Technical report, C-2005-75, 2005. 8.4.2

M. Gibbs. *Bayesian Gaussian Processes for Regression and Classification*. PhD thesis, University of Cambridge, 1997. A.2

J. C. Gittins and D. M. Jones. A dynamic allocation index for the discounted multiarmed bandit problem. *Biometrika*, 66(3):561–565, December 1979. 3.1.2, 15.7.3

N. S. Glance, M. Hurst, K. Nigam, M. Siegler, R. Stockton, and T. Tomokiyo. Deriving marketing intelligence from online discussion. In *KDD*, 2005. 8.3.1

Amir Globerson and Sam Roweis. Nightmare at test time: Robust learning by feature deletion. In *ICML*, 2006. 10.5.5

Leslie Ann Goldberg and Mark Jerrum. Counting unlabelled subtrees of a tree is #p-complete. *LMS J Comput. Math.*, 3:117–124, 2000. 15.3.3

J. Goldenberg, B. Libai, and E. Muller. Talk of the network: A complex systems look at the underlying process of word-of-mouth. *Marketing Letters*, 12, 2001. 8.4.2

B. Goldengorin, G.A. Tijssen, and M. Tso. The maximization of submodular functions. Technical report, University of Groningen, Research Institute SOM, 1999. 5.5.2, 5.8

Daniel Golovin and Matthew Streeter. Online algorithms for maximizing submodular set functions. In *Submitted to SODA*, 2008. 10.11, 16.7

G. Golub and C. Van Loan. *Matrix Computations*. Johns Hopkins, 1989. 6.1.2

Teofilo F. Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoret. Comput. Sci.*, 38(2-3):293–306, 1985. ISSN 0304-3975. 10.10.3

H. H. Gonzalez-Banos and J. Latombe. A randomized art-gallery algorithm for sensor placement. In *Proc. 17th ACM Symposium on Computational Geometry*, pages 232–240, 2001. 3.1.1, 6, 12.6.1

R. B. Gramacy. *Bayesian Treed Gaussian Process Models*. PhD thesis, University of California, Santa Cruz, CA 95064, December 2005. Department of Applied Math & Statistics. 14.2.1, A.2

D. Gruhl, R. Guha, D. Liben-Nowell, and A. Tomkins. Information diffusion through blogspace. In *WWW*, 2004. 8.4.2

R. Gueli. Predator-prey model for discrete sensor placement. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006. 8.4.2

Anupam Gupta, Robert Krauthgamer, and James R. Lee. Bounded geometries, fractals, and low-distortion embeddings. In *FOCS*, 2003a. 1, 11.4, 11.4.1

Himanshu Gupta, Samir R. Das, and Quinyi Gu. Connected sensor cover: Self-organization of sensor networks for efficient query execution. In *MobiHoc*, 2003b. 11, 11.8.2

P. Guttorp, N. D. Le, P. D. Sampson, and J. V. Zidek. Using entropy in the redesign of an environmental monitoring network. Technical report, Department of Statistics. University of British Columbia., 1992. Tech. Rep. 116. 3.1.2

I. Guyon, J. Weston, S. Barnhill, and V. Vapnik. Gene selection for cancer classification using support vector machines. *Machine Learning*, 46:389–422, 2002. 7.7.1, 7.7.2, 7.10

M. G. Halender and L. Zhang. Field studies of comfort and discomfort in sitting. *Ergonomics*, 40(9): 895–915, 1997. 7.7.3

A. Haoui, R. Kavaler, and P. Varaiya. Wireless magnetic sensors for traffic surveillance. *Transportation Research C*, 16(3):294–306, 2008. 12

Thomas C. Harmon, Richard F. Ambrose, Robert M. Gilbert, Jason C. Fisher, Michael Stealey, and William J. Kaiser. High resolution river hydraulic and water quality characterization using rapidly deployable networked infomechanical systems (nims rd). Technical Report 60, CENS, 2006. 10.1, 10.7.1, 14.1, 14.6.1

T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning*. Springer, 2003. 6.1.2

David Heckerman, E. Horvitz, and B. Middleton. An approximate nonmyopic computation for value of information. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 15:292–298, 1993. 2.2.1, 3.1.2, 15, 15.1.1, 15.7.2

N. Heo and P. K. Varshney. Energy-efficient deployment of intelligent mobile sensor networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A*, 35(1):78–92, 2005. 3.2.2

D. Higdon, J. Swall, and J. Kern. Non-stationary spatial modeling. In Jos M. Bernardo, James O. Berger, A. P. Dawid, and Adrian F. M. Smith, editors, *Bayesian Statistics 6: Proceedings of the Sixth Valencia International Meeting*, volume 6. Oxford UP, 1998. A.2

N. J. Higham. Computing the nearest correlation matrix – a problem from finance. *IMA Journal of Numerical Analysis*, 22:329–343, 2002. A.2

D. Hobson. A powered aid for aligning the lower limb modular prosthesis. *Bulletin of Prosthetics Research*, Fall:10–18, 1972. 7.7.3

D. Hochbaum and D. Shmoys. A best possible heuristic for the k-center problem. *Mathematics of Operations Research*, 10(2):180–184, 1985. 10.10.3

D. S. Hochbaum and W. Maas. Approximation schemes for covering and packing problems in image processing and VLSI. *Journal of the ACM*, 32:130–136, 1985. 1.1.1, 3.1.1, 3.2.1, 5.1.1, 6, 6.5.2, 12, 12.6.1

Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58(301):13–30, 1963. B.3

S. C. H. Hoi, R. Jin, J. Zhu, and M. R. Lyu. Batch mode active learning and its application to medical image classification. In *ICML*, 2006. 10.10.5

H. Hotelling. Relations between two sets of variates. *Biometrika*, 28(3/4):321–377, 1936. 6.5.1

A. Howard, M. Mataric, and G. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem, 2002. 3.2.2

R. A. Howard. Information value theory. In *IEEE Transactions on Systems Science and Cybernetics (SSC-2)*, 1966. 2.3, 2.3.3, 3.1.2, 15.1, 15.1.1, 15.7.2

R. A. Howard and J.E. Matheson. *Readings on the Principles and Applications of Decision Analysis II*,

chapter Influence Diagrams, pages 719–762. Strategic Decision Group, Menlo Park, 1984. Reprinted 2005 in Decision Analysis 2(3) 127-143. 3.1.2, 15, 15.1.1, 15.7.2

Satoru Iwata, Lisa Fleischer, and Satoru Fujishige. A combinatorial strongly polynomial algorithm for minimizing submodular functions. *Journal of the ACM*, 48(4):761–777, 2001. URL `citeseer.ist.psu.edu/article/iwata00combinatorial.html`. 5.7

Shihao Ji, Ronald Parr, and Lawrence Carin. Non-myopic multi-aspect sensing with partially observable markov decision processes. *IEEE Transactions on Signal Processing*, 55(6):2720–2730, June 2007. 3.2.3

Michael Johanson, Martin Zinkevich, and Michael Bowling. Computing robust counter-strategies. In *NIPS*, 2007. 10.10.4

David S. Johnson, Maria Minkoff, and Stephen Phillips. The prize collecting steiner tree problem: theory and practice. In *SODA*, 2000. 10.6.4, 4, 11.5.1, B.6, B.6

A. Kapoor, S. Mota, and R. W. Picard. Towards a learning companion that recognizes affect. In *Proceedings from Emotional and Intelligent II: The Tangled Knot of Social Cognition, AAAI Fall Symposium*. AAAI, November 2001. 7.7.3

Ashish Kapoor, Eric Horvitz, and Sumit Basu. Selective supervision: Guiding supervised learning with decision-theoretic active learning. In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007. 3.1.2, 15, 15.7.2

Koushik Kar and Suman Banerjee. Node placement for connected coverage in sensor networks. In *WiOpt*, 2003. 11, 11.8.2

David Kempe, Jon Kleinberg, and Eva Tardos. Maximizing the spread of influence through a social network. In *Knowledge Discovery and Data Mining*, 2003. 5.2.1, 8, 8.4.1, 8.2, 8.4.1, 8.4.2, B.4

R. Kershner. The number of circles covering a set. *American Journal of Mathematics*, 61:665–671, 1939. 1.1.1, 3.1.1, 5.1.1

Samir Khuller, Anna Moss, and Joseph Naor. The budgeted maximum coverage problem. *Information Processing Letters*, 70(1):39–45, 1999. 5.2.2, B.1, B.2

H-.K. Kim, B. K. Mallick, and C. C. Holmes. Analyzing nonstationary spatial data using piecewise Gaussian processes. *Journal of the American Statistical Association*, 100(16):653–668, June 2005. A.2

S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598): 671–680, 1983. 5.3

C. Ko, J. Lee, and M. Queyranne. An exact algorithm for maximum entropy sampling. *Operations Research*, 43(4):684–691, 1995. 3.2.1, 6.2.1, 6.1, 14.2.1, 15.7.1, B.2

Thomas Kollar and Nicholas Roy. Efficient optimization of information-theoretic exploration in slam. In *AAAI*, 2008. 3.2.3

D. Koller and N. Friedman. *Structured Probabilistic Models*. Electronic Preprint, 2008. 2.2, 2.2.1, 14.5.2, 1

Farinaz Koushanfary, Nina Taft, and Miodrag Potkonjak. Sleeping coordination for comprehensive sensing using isotonic regression and domatic partitions. In *Infocom*, 2006. 3.2.1, 12, 12.1.3, 1, 12.5.5, 12.6.2

P. Kouvelis and G. Yu. *Robust Discrete Optimization and its Applications*. Kluwer Academic Publishers, 1997. 10.10.1

A. Krause and C. Guestrin. Optimal nonmyopic value of information in graphical models - efficient algorithms and theoretical limits. In *Proc. of IJCAI*, 2005a. 4, 5

A. Krause and C. Guestrin. Nonmyopic active learning of gaussian processes: An exploration–exploitation approach. In *ICML*, 2007. 4

A. Krause and E. Horvitz. A utility-theoretic approach to privacy and personalization. In *Proc. 23rd Conference on Artificial Intelligence (AAAI), Special Track on AI & the Web*, 2008. 17.2.3

A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of the Fifth International Symposium on Information Processing in Sensor Networks (IPSN)*, 2006a. 3, 6

A. Krause, J. Leskovec, S. Isovitsch, J. Xu, C. Guestrin, J. VanBriesen, M. Small, and P. Fischbeck. Optimizing sensor placements in water distribution systems using submodular function maximization. In *WDSA*, 2006b. 1.1.5

A. Krause, J. Leskovec, C. Guestrin, J. VanBriesen, and C. Faloutsos. Efficient sensor placement optimization for securing large water distribution networks. *Journal of Water Resources Planning and Management*, 136(6), 2008a. 8.1.5, 1, 6

A. Krause, A. Singh, and C. Guestrin. Near-optimal sensor placements in Gaussian processes: Theory, efficient algorithms and empirical studies. In *Journal of Machine Learning Research*, volume 9, 2008b. 10.7.2, 1, 2, 6

A. Krause, B. McMahan, C. Guestrin, and A. Gupta. Robust submodular observation selection. *To appear in the Journal of Machine Learning Research*, 2009. 3, 5

Andreas Krause and Carlos Guestrin. Near-optimal value of information in graphical models. In *UAI*, 2005b. 1, 5

Andreas Krause, Eric Horvitz, Aman Kansal, and Feng Zhao. Towards community sensing. In *IPSN*, 2008c. 12.1.1, 12.5.2, 17.2

R. Kumar, J. Novak, P. Raghavan, and A. Tomkins. On the bursty evolution of blogspace. In *WWW*, pages 568–576. ACM Press, 2003. doi: http://doi.acm.org/10.1145/775152.7775233. URL `http://doi.acm.org/10.1145/775152.7775233`. 8.4.2

John Lafferty, Andrew McCallum, and Fernando Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *ICML*, 2001. 15.2

N. Landwehr, M. Hall, and E. Frank. Logistic model trees. In *Proceedings of 14th European Conference on Machine Learning*, pages 241–252. Springer-Verlag, 2003. 7.7.1

Gilbert Laporte and Silvano Martello. The selective travelling salesman problem. *Disc. App. Math*, 26: 193–207, 1990. 10.6.4

N. Lawrence, M. Seeger, and R. Herbrich. Fast sparse gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems (NIPS) 16*, 2003. 6.5.3, 10.10.5

O. Ledoit and M. Wolf. A well-conditioned estimator for large-dimensional covariance matrices. *Journal of Multivariate Analysis*, 88(2):365–411, February 2004. A.2

Uri Lerner and Ronald Parr. Inference in hybrid networks: Theoretical limits and practical algorithms. In *UAI*, 2001. 15.8

J. Leskovec, L. A. Adamic, and B. A. Huberman. The dynamics of viral marketing. In *ACM EC*, 2006. 8.4.2

J. Leskovec, M. McGlohon, C. Faloutsos, N. S. Glance, and M. Hurst. Cascading behavior in large blog graphs. In *SDM*, 2007a. 8.1.5, 8.3.1, 8.4.2

Jure Leskovec, Andreas Krause, Carlos Guestrin, Christos Faloutsos, Jeanne VanBriesen, and Natalie Glance. Cost-effective outbreak detection in networks. In *13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2007b. 10.7.3, 12.5.4, 1, 6

Asaf Levin. A better approximation algorithm for the budget prize collecting tree problem. *Ops. Res. Lett.*, 32:316–319, 2004. 11.4.4

D. V. Lindley. On a measure of the information provided by an experiment. *Annals of Mathematical Statistics*, 27:986–1005, 1956. 2.3, 2.3.1, 3.1.2, 3.1.2, 3.1.2, 3.1.2, 3.3.2, 15.1, 15.1.1, 15.7.2

D. V. Lindley and A. F. M. Smith. Bayes' estimates for the linear model. *Journal of the Royal Statistical Society, Ser. B*, 34:1–18, 1972. 2.3, 6.1.2, 15.7.1

M. Littman, J. Goldsmith, and M. Mundhenk. The computational complexity of probabilistic planning. *Journal of Artificial Intelligence Research*, 9:1–36, 1998. 15.4.1

L. Lovasz. Submodular functions and convexity. *Mathematical Programming - State of the Art*, pages 235–257, 1983. 5, 5.1, 10.11

S. P. Luttrell. The use of transinformation in the design of data sampling schemes for inverse problems. *Inverse Problems*, 1:199–218, 1985. 3.1.2

D. MacKay. Information-based objective functions for active data selection. *Neural Computation*, 4(4): 590–604, 1992. 3.1.2, 3.2.1, 3.3.2, 10.10.5

D. J. C. MacKay. *Information Theory, Inference, and Learning Algorithms*. Cambridge UP, 2003. 1, 6.1.2

Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, 1990. 10.6.4

G. P. McCabe. Principal variables. *Technometrics*, 26(2):137–144, 1984. 6.5.1

M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, May 1979. 6.2.1

Alexandra Meliou, Andreas Krause, Carlos Guestrin, and Joseph M. Hellerstein. Nonmyopic informative path planning in spatio-temporal models. In *AAAI*, 2007. 10.6.4, 16.7

R. K. Meyer and C. J. Nachtsheim. Constructing exact D-optimal experimental designs by simulated annealing. *American Journal of Mathematical and Management Sciences*, 8(3-4):329–359, 1988. 3.2.1

E. Minieka. The m-center problem. *SIAM Rev*, 12(1):138–139, 1970. 10.10.3

M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. *Optimization Techniques, LNCS*, pages 234–243, 1978. 5.9, 5.4

T. J. Mitchell. An algorithm for the construction of "D-optimal" experimental designs. *Technometrics*, 16 (2):203–210, May 1974a. ISSN 00401706. 3.2.1

T.J. Mitchell. Computer construction of "D-optimal" first-order designs. *Technometrics*, 16(2):211–220, May 1974b. ISSN 00401706. 3.2.1

Nenad Mladenovic, Martine Labbé, and Pierre Hansen. Solving the p-center problem with tabu search and variable neighborhood search. *Networks*, 42(1):48–64, 2003. 10.10.3

B. Moghaddam, Y. Weiss, and S. Avidan. Spectral bounds for sparse PCA: Exact and greedy algorithms. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2005. 3.3.1

B. Moghaddam, Y. Weiss, and S. Avidan. Generalized spectral bounds for sparse LDA. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML)*, 2006. 3.3.1

L.C. Molina, L. Belanche, and A. Nebot. Feature selection algorithms: A survey and experimental evaluation. In *ICDM*, 2002. 15.7.4

M. Monette, R. Weiss-Lambrou, and J. Dansereau. In search of a better understanding of wheelchair sitting comfort and discomfort. In *Proceedings of the RESNA annual conference*, 1999. 7.7.3

Vijay S. Mookerjee and Michael V. Mannino. Sequential decision models for expert system optimization. *IEEE Trans. Knowl. Data Eng.*, 9(5):675–687, 1997. 15

Robert Morrow. *Wireless Network Coexistence*. McGraw-Hill Professional, 1 edition, August 2004. 11.8.3

Michael Munie and Yoav Shoham. Optimal testing of structured knowledge. In *Twenty-Third Conference on Artificial Intelligence (AAAI)*, 2008. 15.7.2

B. Mutlu, A. Krause, J. Forlizzi, C. Guestrin, and J. Hodgins. Robust, low-cost, non-intrusive sensing and recognition of seated postures. In *Proc. of the 20th ACM Symposium on User Interface Software and Technology (UIST)*, 2007. 6

M. Narasimhan and J. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. In *Advances in Neural Information Processing Systems (NIPS) 19*, 2006. 5.7, 5.8

Mukund Narasimhan and Jeff Bilmes. Pac-learning bounded tree-width graphical models. In *Uncertainty in Artificial Intelligence*, 2004. 5.7, 10.10.5

Mukund Narasimhan, Nebojsa Jojic, and Jeff Bilmes. Q-clustering. In *NIPS*, 2005. 5.7, 10.10.5

G. Nemhauser, L. Wolsey, and M. Fisher. An analysis of the approximations for maximizing submodular set functions. *Mathematical Programming*, 14:265–294, 1978. 1.1.1, 4, 5, 5.1, 5.1, 5.2, 5.6, 5.6, 6.3.2, 6.3.2, 6.3.3, 7.2.1, 7.2.2, 7.3, 7.3, 7.5, 4, 10, 17.1.2, B.1, B.2, B.4

G. L. Nemhauser and L. A. Wolsey. *Studies on Graphs and Discrete Programming*, chapter Maximizing Submodular Set Functions: Formulations and Analysis of Algorithms, pages 279–301. North-Holland, 1981. 5.5.1, 5.5.1, 6.3.5, 6.3.5

N-.K Nguyen and A. J. Miller. A review of some exchange algorithms for constructing discrete D-optimal designs. *Computational Statistics and Data Analysis*, 14:489–498, 1992. 3.2.1

D. J. Nott and W. T. M. Dunsmuir. Estimation of nonstationary spatial covariance structure. *Biometrika*, 89:819–829, 2002. 6.1.3, 6.6.2, 14, 14.5, A.3

A. O'Hagan. Curve fitting and optimal design for prediction (with discussion). *Journal of the Royal Statistical Society, Ser. B*, 40:1–42, 1978. 2.3.1, 3.1.2, 6.1.2

Avi Ostfeld and Elad Salomons. Optimal layout of early warning detection stations for water distribution systems security. *J. Water Resources Planning and Management*, 130(5):377–385, 2004. 8.4.2

Avi Ostfeld, James G. Uber, and Elad Salomons. Battle of water sensor networks (bwsn): A design challenge for engineers and algorithms. In *8th Annual Symposium on Water Distribution Systems Analysis*, Cincinnati, Ohio, 2006. 8, 8.2.1, 8.2.2, 8.2.3, 10.7.3

Avi Ostfeld, James G. Uber, Elad Salomons, Jonathan W. Berry, William E. Hart, Cindy A. Phillips, Jean-Paul Watson, Gianluca Dorini, Philip Jonkergouw, Zoran Kapelan, Francesco di Pierro, Soon-Thiam Khu, Dragan Savic, Demetrios Eliades, Marios Polycarpou, Santosh R. Ghimire, Brian D. Barkdoll, Roberto Gueli, Jinhui J. Huang, Edward A. McBean, William James, Andreas Krause, Jure Leskovec, Shannon Isovitsch, Jianhua, Carlos Guestrin, Jeanne VanBriesen, Mitchell Small, Paul Fischbeck, Ami

Preis, Marco Propato, Olivier Piller, Gary B. Trachtman, Zheng Yi Wu, and Tom Walski. The battle of the water sensor networks (bwsn): A design challenge for engineers and algorithms. *submitted to Journal of Water Resources Planning and Management*, 2008. 8.2.4, 10.1

C. J. Paciorek. *Nonstationary Gaussian Processes for Regression and Spatial Modelling*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, May 2003. 6.1.2, 14, A.2

Rina Panigrahy and Sundar Vishwanathan. An $\mathcal{O}(\log^* n)$ approximation algorithm for the asymmetric p-center problem. *Journal of Algorithms*, 27(2):259–268, 1998. 10.10.3

L. Paninski. Asymptotic theory of information-theoretic experimental design. *Neural Computation*, 17 (7):1480–1507, 2005. 3.1.2

C. H. Papadimitriou and M. Yannakakis. The complexity of tradeoffs, and optimal access of web sources. In *FOCS*, 2000. 2.5.1, 4

Christos H. Papadimitriou. *Computational Complexity*. Addison-Wesley, 1995. 15.4.1

James D. Park and Adnan Darwiche. Complexity results and approximation strategies for map explanations. *Journal of Aritificial Intelligence Research*, 21:101–133, February 2004. 15.4.1, 15.8

R. Pastor-Satorras and A. Vespignani. Immunization of complex networks. *Physical Review E*, 65, 2002. 8.4.2

J. Pilz, G. Spoeck, and M. G. Schimek. *Geostatistics Wollongong*, volume 1, chapter Taking account of uncertainty in spatial covariance estimation, pages 302–313. Kluwer, 1996. 10.10.2

Joelle Pineau, Geoff Gordon, and Sebastian Thrun. Anytime point-based approximations for large pomdps. *JAIR*, 27:335–380, 2006. 3.2.3

Ashok Kumar Ponnuswami and Subhash Khot. Approximation algorithms for the max-min allocation problem. In *APPROX*, 2007. 10.10.4, 12.6.3, 16.1

Robert Price and Paul R. Messinger. Optimal recommendation sets: Covering uncertainty over user preferences. In *AAAI*, 2005. 10.10.5

F. Pukelsheim. Information increasing orderings in experimental design theory. *International Statistical Review / Revue Internationale de Statistique*, 55(2):203–219, Aug. 1987. ISSN 03067734. 3.1.2

Maurice Queyranne. A combinatorial algorithm for minimizing symmetric submodular functions. In *SODA*, 1995. 5.7, 5.8, 10.10.5

J. Ross Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986. 15.7.4

Y. Radovilsky, G. Shattah, and S. E. Shimony. Efficient deterministic approximation algorithms for non-myopic value of information in graphical models. In *IEEE International Conference on Systems, Man and Cybernetics (SMC)*, volume 3, pages 2559–2564, October 2006. 15.7.2

S. Rajagopalan and V.V. Vazirani. Primaldual rnc approximation algorithms for set cover and covering integer programs. *SIAM Journal on Computing*, 28(2):525–540, 1998. 5.1.1

N. Ramakrishnan, C. Bailey-Kellogg, S. Tadepalli, and V. N. Pandey. Gaussian processes for active data mining of spatial aggregates. In *SIAM Data Mining*, 2005. 6, 6.2.2, 7

T.S. Rappaport. *Wireless Communication: Principles and Practice*. Prentice Hall, 2000. 11.8.3

C. E. Rasmussen and Z. Ghahramani. Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems (NIPS) 14*, 2002. A.2

C. E. Rasmussen and C. K. I. Williams. *Gaussian Process for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, 2006. 10.5.1, 14, 14.1, A.1

P. J. Ribeiro Jr. and P. J. Diggle. geoR: A package for geostatistical analysis. R-NEWS Vol 1, No 2. ISSN 1609-3631, 2001. 6.6.4

M. Richardson and P. Domingos. Mining knowledge-sharing sites for viral marketing. In *KDD*, 2002. 8.4.2

M. Richardson and P. Domingos. Markov logic networks. *Machine Learning*, 62:107–136, 2006. 16.6

H. Robbins. Some aspects of the sequential design of experiments. *Bulletin of the American Mathematical Society*, 58:527–535, 1952. 3.1.2, 15.7.3

T. G. Robertazzi and S. C. Schwartz. An accelerated sequential algorithm for producing D-optimal designs. *SIAM Journal of Scientific and Statistical Computing*, 10(2):341–358, March 1989. 5.4, 10.3.2

E. Rogers. *Diffusion of innovations (4th ed.)*. Free Press, 1995. 8.4.2

A. Rosenfeld and J. L. Pfaltz. Sequential operations in digital processing. *Journal of the ACM*, 13: 471–494, 1966. 1

L A Rossman. The epanet programmer's toolkit for analysis of water distribution systems. In *Annual Water Resources Planning and Management Conference*, 1999. 8.2.2, 10.7.3

J. Sacks and S. Schiller. *Statistical Decision Theory and Related Topics IV, Vol. 2*. Springer, 1988. 10.7.1, 10.10.2

Jerome Sacks, Susannah B. Schiller, and W.J. Welch. Design for computer experiments. *Technometrics*, 31(1):41–47, 1989. 3.1.2

P. D. Sampson and P. Guttorp. Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417):108–119, Mar. 1992. ISSN 01621459. A.2

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active learning of partially hidden markov models for information extraction. In *ECML/PKDD Workshop on Instance Selection*, 2001. 3.1.2, 15, 15.7.2

M. J. Scherer. User desires for wheelchairs. *Rehab Management*, 9(4):121–123, June-July 1996. 7.7.3

Alexander Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *J. Combin. Theory Ser. B*, 80(2):346–355, 2000. ISSN 0095-8956. 5.7

P. Sebastiani and H. P. Wynn. Maximum entropy sampling and optimal bayesian experimental design. *Journal of the Royal Statistical Society, Series B*, 62(1):145–157, 2000. 3.1.2

M. Seeger. Gaussian processes for machine learning. *International Journal of Neural Systems*, 14(2): 69–106, 2004a. 6.1.2

M. Seeger, C. K. I. Williams, and N. D. Lawrence. Fast forward selection to speed up sparse gaussian process regression. In *Proceedings of the Ninth International Workshop on Artificial Intelligence and Statistics (AISTATS)*, 2003. 6.5.3, 10.10.5

Matthias Seeger. Greedy forward selection in the informative vector machine. Technical report, University of California at Berkeley, 2004b. 10.10.5

S. Seo, M. Wallat, T. Graepel, and K. Obermayer. Gaussian process regression: Active data selection and test point rejection. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, volume 3, pages 241–246, 2000. 3.2.2, 3.3.2, 10.10.5, 14, 14.2.1

C.E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, July, October 1948. 2.3

G. Shaw. Wheelchair seat comfort for the institutionalized elderly. *Assistive Technology*, 3(1):11–23,

1992. 7.7.3

M. C. Shewry and H. P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14:165–170, 1987. 2.3.1, 3.1.2, 3.1.2, 6, 6.1.2, 14.1, 14.2.1

R. Sim and N. Roy. Global a-optimal robot exploration in slam. In *ICRA*, 2005. 3.2.3

Amarjeet Singh, Andreas Krause, Carlos Guestrin, William J. Kaiser, and Maxim A. Batalin. Efficient planning of informative paths for multiple robots. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 2204–2211, Hyderabad, India, January 2007. 10.6.4, 11.7, 11.7, 11.8.4, 16.3

V. Singhvi, A. Krause, C. Guestrin, J. Garrett, and H.S. Matthews. Intelligent light control using sensor networks. In *SenSys*, 2005. 15, 15.6.2, 6

S. Slijepcevic and M. Potkonjak. Power efficient organization of wireless sensor networks. In *ICC*, 2001. 12.6.2

R.D. Smallwood and E.J. Sondik. The optimal control of partially observable markov decision processes over a finite horizon. *Operations Research*, 21:1071–1088, 1973. 3.2.3, 15.1.1

D. M. Smith. Pressure ulcers in the nursing home, [review]. *Annals Int. Med.*, 123(6):43342, 1995. 7.7

E. Snelson and Z. Ghahramani. Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems (NIPS) 18*, 2005. 3.2.2, 6.5.3

E. Snelson, C. E. Rasmussen, and Z. Ghahramani. Warped Gaussian processes. In *Advances in Neural Information Processing Systems (NIPS) 16*, 2003. A.2

P. Sollich. Learning from minimum entropy queries in a large committee machine. *Physical Review E*, 53:R2060–R2063, 1996. 3.3.2, 10.10.5

C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *RSS*, 2005. 3.2.3

C. Stein. Estimation of a covariance matrix. Rietz Lecture, 39th Annual Meeting Institute of Mathematical Sciences. Atlanta, Georgia, 1975. A.2

M. L. Stein. Nonstationary spatial covariance functions. Technical Report 21, University of Chicago, 2005. A.2

Amos J. Storkey. Truncated covariance matrices and toeplitz methods in gaussian processes. In *ICANN99*, 1999. 6.4.2, 14.4.2

J. F. Sturm. Using sedumi 1.02, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software, special issue on interior-point methods*, 11(12):625–653, 1999. 6.6.5

M. Sviridenko. A note on maximizing a submodular set function subject to knapsack constraint. *Operations Research Letters*, 32:41–43, 2004. 5.2.2, 5.6, 6.3.3

Hong Z. Tan, Lynne A. Slivovsky, and Alex Pentland. A sensing chair using pressure distribution sensors. *IEEE/ASME Transactions on Mechatronics*, 6(3):261–268, 2001. 7.7, 7.7.1, 7.7.1, 7.7.2, 7.7.2, 7.7.3

R. Tibshirani. Regression shrinkage and selection via the lasso. *J. Royal. Statist. Soc B*, 58(1):267–288, 1996. 3.2.2

Simon Tong and Daphne Koller. Active learning for parameter estimation in bayesian networks. In *NIPS*, 2001. 15.7.4

S. Toumpis and G. A. Gupta. Optimal placement of nodes in large sensor networks under a general physical layer model. In *Proc. IEEE Communications Society Conference on Sensor and Ad Hoc Communications (SECON)*, 2005. 3.1.1

Volker Tresp. Mixtures of gaussian processes. In *NIPS*, pages 654–660, 2000. URL `citeseer.ist.psu.edu/tresp01mixtures.html`. A.2

J. A. Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Trans. Info. Theory*, 50 (10):2231–2242, October 2004. 3.2.2

P. D. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm. *Journal of Artificial Intelligence Research*, 2:369–409, 1995. 15

UC Berkeley, PATH and Caltrans. Freeway performance measurement system. http://pems.eecs.berkeley.edu/, 2005. URL `http://pems.eecs.berkeley.edu/`. 7.6.2, 12.5.1

Linda van der Gaag and Maria Wessels. Selective evidence gathering for diagnostic belief networks. *AISB Quart.*, 86:23–34, 1993. 3.1.2, 7, 15, 15.7.2

J.W. van Groenigen and A. Stein. Constrained optimization of spatial sampling using continuous simulated annealing. *J. Environ. Qual.*, 27:1078–1086, 1998. 10.7.1, 10.10.2

V. Vapnik. *The Nature of Statistical Learning Theory*. Springer, 1995. A.2

Vijay V. Vazirani. *Approximation Algorithms*. Springer, 2003. 2.6, 10.6.4, 11.1.1

Jan Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *STOC*, 2008. 5.6, 12.6.3

J. Watson, W. E. Hart, and R. Murray. Formulation and optimization of robust sensor placement problems for contaminant warning systems. In *Water Distribution System Symposium*, 2006. 10.10.3, 10.10.4

W. J. Welch. Branch-and-bound search for experimental design based on D-optimality and other criteria. *Technometrics*, 24(1):41–48, 1982. 3.2.1

M. Widmann and C. S. Bretherton. 50 km resolution daily precipitation for the pacific northwest. http://www.jisao.washington.edu/data_sets/widmann/, May 1999. 6.6.1, 10.7.1

D. P. Wiens. Robustness in spatial studies ii: minimax design. *Environmetrics*, 16:205–217, 2005. 10.7.1, 10.7.1, 10.10.2

Jason L. Williams, John W. Fisher III, and Alan S. Willsky. Performance guarantees for information theoretic active inference. In *AISTATS*, 2007. 12.2.1

I. H. Witten and E. Frank. *Data mining: practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann Publishers Inc., San Francisco, CA, 2nd edition edition, 2005. 7.7.2

L.A. Wolsey. An analysis of the greedy algorithm for the submodular set covering problem. *Combinatorica*, 2:385–393, 1982a. 5.7, 5.6, 10.3.2, 10.6.3, B.5

Laurence A. Wolsey. Maximising real-valued submodular functions: Primal and dual heuristics for location problems. *Mathematics of Operations Research*, 7(3):410–425, 1982b. 5.2.2

S. Wu and J. V. Zidek. An entropy based review of selected NADP/NTN network sites for 1983–86. *Atmospheric Environment*, 26A:2089–2103, 1992. 3.1.2

H. Xu, C. Caramanis, and S. Mannor. Robust regression and lasso. In *NIPS*, 2008. 10.8

D. Ylvisaker. *A Survey of Statistical Design and Linear Models*, chapter Design on Random Fields. North-Holland, 1975. 3.1.2

K. Yu, J. Bi, and V. Tresp. Active learning via transductive experimental design. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML)*, 2006. 3.1.2, 16.4

Tan H. Z., Ifung L., and Pentland A. The chair as a novel haptic user interface. In *Proceedings of the Workshop on Perceptual User Interfaces*, Banff, Alberta, Canada, 1997. 7.7, 7.7.1, 7.7.3

J. Zhang. Identifying factors of comfort and discomfort in sitting. *Human Factors*, 38, 1996. 7.7.3

F. Zhao, J. Shin, and J. Reich. Information-driven dynamic sensor collaboration for tracking applications. *IEEE Signal Processing*, 19(2):61–72, 2002. 3.2.1, 11.8.1, 12.6.2

Manli Zhu, Aleix M. Martinez, and Hong Z. Tan. Template-based recognition of static sitting postures. In *Workshop on Computer Vision and Pattern Recognition for Human Computer Interaction, CVPR*, 2003. 7.7, 7.7.1, 7.7.1, 7.7.2, 7.7.2, 7.7.3

Z. Zhu and M. L. Stein. Spatial sampling design for prediction with estimated parameters. *Journal of Agricultural, Biological and Environmental Statistics*, 11:24–49, 2006. 1, 6, 6.5.4, 14

J. V. Zidek, W. Sun, and N. D. Le. Designing and integrating composite networks for monitoring multivariate gaussian pollution fields. *Applied Statistics*, 49:63–79, 2000. 1, 6.2.2, 6.5.5, 6.7

D. L. Zimmerman. Optimal network design for spatial prediction, covariance parameter estimation, and empirical prediction. *Environmetrics*, 17(6):635–652, 2006. 1, 6, 6.5.4