Near-optimal Sensor Placements:
Maximizing Information while Minimizing Communication Cost

Andreas Krause
Carnegie Mellon University

Carlos Guestrin
Carnegie Mellon University

Anupam Gupta
Carnegie Mellon University

Jon Kleinberg
Cornell University

December 2005
CMU-CALD-05-110

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

When monitoring spatial phenomena with wireless sensor networks, selecting the best sensor placements is a fundamental task. Not only should the sensors be informative, but they should also be able to communicate efficiently. In this paper, we present a data-driven approach that addresses the three central aspects of this problem: measuring the predictive quality of a set of sensor locations (regardless of whether sensors are ever placed at these locations), predicting the communication cost involved with these placements, and designing an algorithm with provable quality guarantees that optimizes the NP-hard tradeoff. Specifically, we use data from a pilot deployment to build non-parametric probabilistic models called *Gaussian Processes* (GPs) both for the spatial phenomena of interest and for the spatial variability of link qualities, which allows us to estimate predictive power and communication cost of unsensed locations. Surprisingly, uncertainty in the representation of link qualities plays an important role in estimating communication costs. Using these models, we present a novel, polynomial-time, data-driven algorithm, *pSPIEL*, which selects Sensor Placements at Informative and cost-Effective Locations. Our approach exploits two important properties of this problem: *submodularity*, formalizing the intuition that adding a node to a small deployment can help more than adding a node to a large deployment; and *locality*, under which nodes that are far from each other provide *almost* independent information. Exploiting these properties, we prove strong approximation guarantees for our *pSPIEL* approach. We also provide extensive experimental validation of this practical approach on several real-world placement problems, and built a complete system implementation on 46 Tmote Sky motes, demonstrating significant advantages over existing methods.

Keywords: Gaussian Processes; Sensor Networks; Link Quality; Information Theory; Approximation Algorithms

1 Introduction

Networks of small, wireless sensors are becoming increasingly popular for monitoring spatial phenomena, such as the temperature distribution in a building [1]. Since only a limited number of sensors can be placed, it is important to deploy them at most informative locations. Moreover, due to the nature of wireless communication, poor link qualities, such as those between sensors which are too far apart, or even nearby nodes that are obstructed by obstacles such as walls or radiation from appliances, 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.

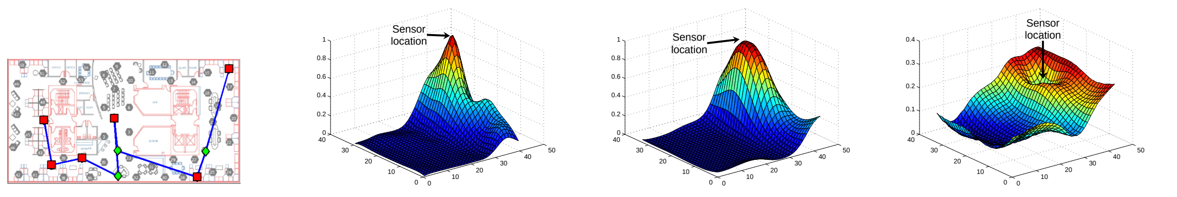
Existing work on sensor placement under communication constraints [2, 3, 4] has considered the problem mainly from a geometric perspective: Sensors have a fixed *sensing region*, such as a disc with a certain radius, and can only communicate with other sensors which are at most a specified distance apart. These assumptions are problematic for two reasons. Firstly, the notion of a *sensing region* implies that sensors can perfectly observe everything within the region, but nothing outside, which is unrealistic: e.g., the temperature can be highly correlated in some areas of a building but very uncorrelated in others (*c.f.,* Fig. 2(a)). Moreover, sensor readings are usually noisy, and one wants to make predictions utilizing the measurements of multiple sensors, making it unrealistic to assume that a single sensor is entirely responsible for a given sensing region. Secondly, 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 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 [5].

In order to avoid the *sensing region* assumption, previous work [6] established *probabilistic models* as an appropriate framework for predicting sensing quality by modeling correlation between sensor locations. In [7] we presented a method for selecting informative sensor placements, based on our *mutual information* criterion. We showed that this criterion led to intuitive placements with superior prediction accuracy compared to existing methods. Furthermore, we provided an efficient algorithm for computing near-optimal placements with strong theoretical performance guarantees; however, this algorithm does not take communication costs into account.

In this paper, we address the general 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 paper, 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, but also to predict communication costs.

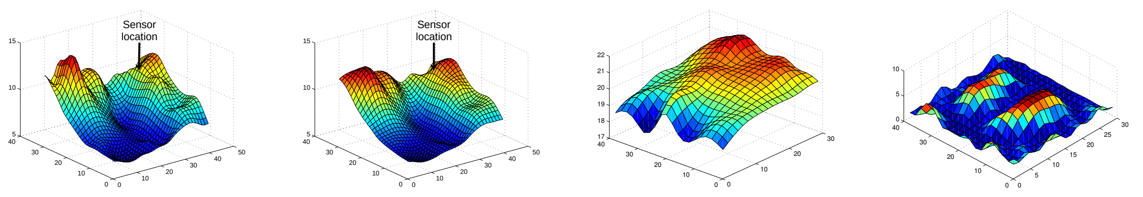
Balancing informativeness of sensor placements with the need to communicate efficiently can be formalized as a novel discrete optimization problem; it generalizes several well-studied problems, and thus appears to be a fundamental question in its own. We present a novel algorithm for this placement problem in wireless sensor networks; the algorithm selects sensor placements achieving a specified amount of certainty, with approximately minimal communication cost. More specifically, our main contributions are:

- A unified method for learning a probabilistic model of the underlying phenomenon and for the expected communication cost between any two locations 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*). Exploiting the concept of *submodularity*, this algorithm is guaranteed to provide near-optimal placements for this hard problem.
- Extensive evaluations of our proposed methods 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.
- A complete solution for collecting data, learning models, optimizing and analyzing sensor placements, realized on Tmote Sky motes, which combines all our proposed methods.



(a) *Example placement* (b) *Real link quality – node 41* (c) *GP link quality – node 41* (d) *Link quality variance – node 41*

Figure 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.



(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 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.

2 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 $\mathcal{V}$. 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 distribution, e.g., to actuate the air conditioning or heating system. Here, the sensing quality refers to our temperature prediction accuracy, and the communication cost depends on how efficiently the sensors communicate with each other, directly relating to the deployment lifetime of the network. More generally, we investigate the problem of solving optimization problems of the form

$$\min_{\mathcal{A}\subseteq\mathcal{V}} \mathrm{c}(\mathcal{A}) \text{ subject to } F(\mathcal{A}) \geq Q \tag{1}$$

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

$$\max_{\mathcal{A}\subseteq\mathcal{V}} F(\mathcal{A}) \text{ subject to } \mathrm{c}(\mathcal{A}) \leq B \tag{2}$$

for some *budget* $B > 0$. This optimization problems aims at finding the most informative placement subject to a budget on the communication cost, and is called the *maximization problem*. In this paper, we present efficient approximation algorithms for both the covering and maximization problems.

2.1 What is sensing quality?

In order to quantify how informative a sensor placement is, we have to establish a notion of uncertainty. We associate a random variable $X_s \in \mathcal{X}_\mathcal{V}$ with each location $s \in \mathcal{V}$ of interest; for a subset $\mathcal{A} \subseteq \mathcal{V}$, let $\mathcal{X}_\mathcal{A}$ denote the set of random variables associated with the locations $\mathcal{A}$. In our temperature measurement example, $\mathcal{V} \subset \mathbb{R}^2$ describes the subset of coordinates in the building where sensors can be placed. Our probabilistic model will describe a joint probability distribution over all these random variables. In order to make predictions at a location s, we will consider conditional distributions $P(X_s = x_s \mid \mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A})$, where we condition on all observations $\mathbf{x}_\mathcal{A}$ made by all sensors $\mathcal{A}$ in our placement. We use the conditional entropy of these distributions, $H(X_s \mid \mathcal{X}_\mathcal{A}) = -\int_{x_s, \mathbf{x}_\mathcal{A}} P(x_s, \mathbf{x}_\mathcal{A}) \log P(x_s \mid \mathbf{x}_\mathcal{A}) dx_s d\mathbf{x}_\mathcal{A}$ to assess the uncertainty in predicting X_s. Intuitively, this quantity expresses how "peaked" the conditional distribution of X_s given $\mathcal{X}_\mathcal{A}$ is around the most likely value, averaging over all possible observations $\mathcal{X}_\mathcal{A} = \mathbf{x}_\mathcal{A}$ the placed sensors can make. To quantify how informative a sensor placement $\mathcal{A}$ is, we use the criterion of *mutual information*:

$$F(\mathcal{A}) = I(\mathcal{X}_\mathcal{A}; \mathcal{X}_{\mathcal{V} \setminus \mathcal{A}}) = H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}}) - H(\mathcal{X}_{\mathcal{V} \setminus \mathcal{A}} \mid \mathcal{X}_\mathcal{A}). \quad (3)$$

This criterion expresses the expected reduction of entropy of all locations $\mathcal{V} \setminus \mathcal{A}$ where we did not place sensors, after taking into account the measurements of our placed sensors. We first proposed this criterion in [7], and showed that it leads to intuitive placements with superior prediction accuracy over existing approaches. Sec. 3 will explain how we model and learn a joint distribution over all locations $\mathcal{V}$ and how to efficiently compute the mutual information.

2.2 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 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

$$\int_\theta \frac{1}{\theta_{s,t}} p(\theta_{s,t}) d\theta_{s,t}. \quad (4)$$

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, with the edges labeled by their communication costs. We call this graph the *communication graph* of $\mathcal{V}$. In general, we assume that we can also place *relay nodes*, which do not sense but only aid communication. 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 $\mathcal{V}$. Finding this minimum cost tree to evaluate this cost function is called the *Steiner tree* problem; it is $\mathbf{NP}$-complete, but there exist very good approximations which we use. 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.

2.3 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 and the communication cost. 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. Fig. 2(c) and Fig. 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 step 2).

3 Predicting sensing quality

In order to achieve highly informative sensor placements, we have to be able to predict the uncertainty in sensor values at a location $s \in \mathcal{V}$, given the sensor values $\mathbf{x}_\mathcal{A}$ at some candidate placement $\mathcal{A}$. This is an extension of the well-known regression problem [8], where we use the measured sensor data to predict values at locations where no sensors are placed. The difference is that in the placement problem, we have to be able to predict not just sensor values at uninstrumented locations, but rather *probability distributions* over sensor values. *Gaussian Processes* are a powerful formalism for making such predictions. To introduce this concept, first consider the special case of the multivariate normal distribution over a set $\mathcal{X}_\mathcal{U}$ of random variables associated with n locations $\mathcal{U}$:

$$P(\mathcal{X}_\mathcal{U} = \mathbf{x}_\mathcal{U}) = \frac{1}{(2\pi)^{n/2}|\Sigma|} e^{-\frac{1}{2}(\mathbf{x}_\mathcal{U}-\mu)^T\Sigma^{-1}(\mathbf{x}_\mathcal{U}-\mu)}.$$

This model has been successfully used for example to model temperature distributions [1], where, every location in $\mathcal{U}$ corresponds to one particular sensor placed in the building. The multivariate normal distribution is fully specified by providing a mean vector μ and a covariance matrix Σ. If we know the values of some of the sensors $\mathcal{B} \subseteq \mathcal{U}$, we find that for $s \in \mathcal{U} \setminus \mathcal{B}$ the conditional distribution $P(X_s = x_s \mid \mathcal{X}_\mathcal{B} = \mathbf{x}_\mathcal{B})$ is a normal distribution, where mean $\mu_{s|\mathcal{B}}$ and variance $\sigma^2_{s|\mathcal{B}}$ are given by

$$\begin{aligned} \mu_{s|\mathcal{B}} &= \mu_s + \Sigma_{s\mathcal{B}}\Sigma^{-1}_{\mathcal{B}\mathcal{B}}(\mathbf{x}_\mathcal{B} - \mu_\mathcal{B}), && (5) \\ \sigma^2_{s|\mathcal{B}} &= \sigma^2_s - \Sigma_{s\mathcal{B}}\Sigma^{-1}_{\mathcal{B}\mathcal{B}}\Sigma_{\mathcal{B}s}. && (6) \end{aligned}$$

Hereby, $\Sigma_{s\mathcal{B}} = \Sigma^T_{\mathcal{B}s}$ is a row vector of the covariances of X_s with all variables in $\mathcal{X}_\mathcal{B}$. Similarly, $\Sigma_{\mathcal{B}\mathcal{B}}$ is the submatrix of Σ, only containing the entries relevant to $\mathcal{X}_\mathcal{B}$, and σ^2_s is the variance of X_s. $\mu_\mathcal{B}$ and μ_s are the means of $\mathcal{X}_\mathcal{B}$ and X_s respectively. Hence the covariance matrix Σ and the mean vector μ contain all the information needed to compute the conditional distributions of X_s given $\mathcal{X}_\mathcal{B}$. From (6) we learn that the posterior variance $\sigma^2_{s|\mathcal{B}}$ after making observations is never larger than the prior variance σ^2_s of X_s. The goal of an optimal placement will intuitively be to select the observations such that the posterior variance (6) for all variables becomes uniformly small. If we can make a set of measurements $(\mathbf{x}_\mathcal{U})_t$ of all sensors $\mathcal{U}$, we can estimate Σ and μ, and use it to compute predictive distributions for any subsets of variables. However, in the sensor placement problem, we must reason about the predictive quality of locations where we do not yet have sensors, and thus need to compute predictive distributions, conditional on variables for which we do not have sample data.

Gaussian Processes are a solution for this dilemma. Technically, a Gaussian Process (GP) is a joint distribution over a (possibly infinite) set of random variables, such that the marginal distribution over any finite subset of variables is multivariate Gaussian. In our temperature measurement example, we would associate a random variable $X(s)$ with each point s in the building, which can be modeled as a subset $\mathcal{V} \subset \mathbb{R}^2$. The GP $X(\cdot)$, which we will refer to as the *sensor data process*, is fully specified by a *mean function* $\mathcal{M}(\cdot)$ and a symmetric positive definite *Kernel function* $\mathcal{K}(\cdot,\cdot)$, generalizing the mean vector and covariance matrix in the multivariate normal distribution in the following way: For any random variable $X(s) \in \mathcal{X}$, $\mathcal{M}(s)$ will correspond to the mean of $X(s)$, and for any two random variables $X(s), X(t) \in \mathcal{X}$, $\mathcal{K}(s,t)$ will be the covariance of $X(s)$ and $X(t)$. This implies, that for any finite subset $\mathcal{B} = \{s_1, s_2, \ldots, s_m\}$, $\mathcal{B} \subseteq \mathcal{V}$ of locations variables, the covariance matrix $\Sigma_{\mathcal{B}\mathcal{B}}$ of the variables $\mathcal{X}_\mathcal{B}$ is obtained by

$$\Sigma_{\mathcal{BB}} = \begin{pmatrix} \mathcal{K}(s_1, s_1) & \mathcal{K}(s_1, s_2) & \dots & \mathcal{K}(s_1, s_m) \\ \mathcal{K}(s_2, s_1) & \mathcal{K}(s_2, s_2) & \dots & \mathcal{K}(s_2, s_m) \\ \vdots & \vdots & & \vdots \\ \mathcal{K}(s_m, s_1) & \mathcal{K}(s_m, s_2) & \dots & \mathcal{K}(s_m, s_m) \end{pmatrix},$$

and its mean is $\mu_{\mathcal{B}} = (\mathcal{M}(s_1), \mathcal{M}(s_2), \dots, \mathcal{M}(s_m))$. Using formulas (5) and (6), the problem of computing predictive distributions is reduced to finding the mean and covariance functions $\mathcal{M}$ and $\mathcal{K}$ for the phenomena of interest. In general, this is a difficult problem – we want to estimate these infinite objects from a finite amount of sample data. Consequently, in practice, often strongly limiting assumptions are made: it is assumed that the covariance of any two random variables is only a function of their distance (isotropy), and independent of their location (stationarity). A kernel function often used is the Gaussian kernel

$$\mathcal{K}(s, t) = \exp\left(-\frac{\|s - t\|_2^2}{h^2}\right). \tag{7}$$

These isotropy and stationarity assumptions lead to similar problems as encountered in the approach using geometric sensing regions, as spatial inhomogeneities such as walls, windows, reflections etc. are not taken into account. These inhomogeneities are however dominantly encountered in real data sets, as indicated in Fig. 2(a).

In this paper, we do *not* make these limiting assumptions. We use an approach to estimate nonstationarity proposed in [9]. Their method estimates a collection of stationary GPs with kernel functions of the form (7), each providing a local description of the nonstationary process around a set of reference points. These reference points are chosen on a grid or near the likely sources of nonstationary behavior. The stationary GPs are combined into a nonstationary GP, whose covariance function interpolates the empirical covariance matrix estimated from the initial sensor deployment, and near the reference points behaves similarly to the corresponding stationary process. Fig. 2(b) shows a learned non-stationary GP for our temperature data. Due to space restrictions, we refer to [9] for details.

Once we have obtained estimates for the mean and covariance functions, we can use these functions to evaluate our mutual information criterion. In order to evaluate Eq. (3), we need to compute conditional entropies $H(X_s \mid \mathcal{X}_{\mathcal{A}})$, which involve integrals over all possible assignments to the placed sensors $\mathbf{x}_{\mathcal{A}}$. Fortunately, there is a closed form solution: We find that

$$H(\mathcal{X}_{\mathcal{V}\setminus\mathcal{A}} \mid \mathcal{X}_{\mathcal{A}}) = \frac{1}{2}\log((2\pi e)^n \det \Sigma_{\mathcal{X}\setminus\mathcal{A}|\mathcal{A}}),$$

hence it only depends on the determinant of the predictive covariance matrix $\Sigma_{\mathcal{V}\setminus\mathcal{A}|\mathcal{A}}$. For details on efficient computation *c.f.,* [7].

4 Predicting communication cost

As discussed in Sec. 2.2, an appropriate cost measure to estimate communication cost is the expected number of retransmissions. If we have a probability distribution $P(\theta_{s,t})$ over transmission success probabilities $\theta_{s,t}$, Eq. (4) 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 Sec. 3, 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 where 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. Whereas in Sec. 3 we used GPs for regression, it is also possible to use GPs for classification[10]. In this *classification* setting, the predictions of the GP are transformed by the sigmoid 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,

```
**Input**: Covariance matrix $\Sigma_{\mathcal{V}\mathcal{V}}$, locations $\mathcal{C} \subseteq \mathcal{V}$
**Output**: Greedy sequence $(G_j)_j$
**begin**
    $\mathcal{C}_0 \leftarrow \emptyset$;
    **for** *j = 1* **to** $|\mathcal{C}|$ **do**
        $G_j \leftarrow \underset{\substack{G \in \mathcal{C} \setminus \mathcal{C}_{j-1}: \\ \bar{\mathcal{A}} = \mathcal{V} \setminus (\mathcal{A} \cup G)}}{\operatorname{argmax}} \frac{\sigma_G^2 - \Sigma_{G\mathcal{A}} \Sigma_{\mathcal{A}\mathcal{A}}^{-1} \Sigma_{\mathcal{A}G}}{\sigma_G^2 - \Sigma_{G\bar{\mathcal{A}}} \Sigma_{\bar{\mathcal{A}}\bar{\mathcal{A}}}^{-1} \Sigma_{\bar{\mathcal{A}}G}}$;
        $\mathcal{C}_j \leftarrow \mathcal{C}_{j-1} \cup G_j$;
    **end**
**end**
```

Algorithm 1: Greedy algorithm for maximizing mutual information.

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 Eqs. (5) and (6). Unfortunately, in the classification setting, the predictive distributions cannot be computed in closed form anymore, but one can resort to approximate techniques [10]. Using these techniques, we infer the link qualities by modeling the underlying GP $W(s,t)$. Intuitively, the binary observations will be converted to imaginary 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 Eqs. (5) and (6) 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 rather 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 even though the link qualities differ at different locations, the distance to the receiver (component 1) is the dominating feature. Fig. 1(c) shows an example of the predicted link qualities using a GP for our indoors deployment, and Fig. 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 (4). Assuming the predictive distribution for $W(s,t)$ is normal with mean μ and variance σ^2, we compute

$$\int \frac{1}{f(x)} \mathcal{N}(x;\mu,\sigma^2)dx = \int (1+\exp(-x))\mathcal{N}(x;\mu,\sigma^2)dx$$
$$= 1 + \exp(-\mu + \sigma^2),$$

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 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!

5 Problem structure in sensor placement optimization

We now address the covering and maximization problems described in Sec. 2. We will consider a discretization of the space into finitely many points $\mathcal{V}$, for example lying on a grid. For each pair of locations in $\mathcal{V}$, we define the edge cost as the expected number of retransmissions required to send a message between these nodes (since link qualities are asymmetric, we use the worse direction as the cost). The set of edges that have finite cost is denoted by E. The challenge in solving the optimization problems (1) and (2) is that the search space—the possible subsets $\mathcal{A} \subseteq \mathcal{V}$—is exponential; more concretely, the problem is easily seen to be $\mathbf{NP}$-hard as a corollary to the hardness of the unconstrained optimization problem [7, 3]. Given this, we seek an efficient approximation algorithm with strong performance guarantees. In Sec. 6, we present such an algorithm. The key to finding good approximate solutions is an understanding and exploitation of the problem structure.

Intuitively, the sensor placement problem satisfies the following diminishing returns property: The more sensors already placed, the less the addition of a new sensor helps us. This intuition is formalized by the concept of *submodularity*: A set function F defined on subsets of $\mathcal{V}$ is called *submodular*, if

$$F(\mathcal{A} \cup \{s\}) - F(\mathcal{A}) \geq F(\mathcal{B} \cup \{s\}) - F(\mathcal{B}), \tag{8}$$

for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$ and $s \in \mathcal{V} \setminus \mathcal{B}$. The function F is *monotonic* if $F(\mathcal{A}) \leq F(\mathcal{B})$ for all $\mathcal{A} \subseteq \mathcal{B} \subseteq \mathcal{V}$. With any such set function F, we can associate the following greedy algorithm: start with the empty set, and at each iteration add to the current set $\mathcal{A}'$ the element s which maximizes the *greedy improvement* $F(\mathcal{A}' \cup \{s\}) - F(\mathcal{A}')$, and continue until $\mathcal{A}'$ has the specified size of k elements. Perhaps surprisingly, if $\mathcal{A}_G$ is the set selected by the greedy algorithm (with $|\mathcal{A}_G| = k$) and if F is monotonic submodular with $F(\emptyset) = 0$, then $F(\mathcal{A}_G) \geq (1 - 1/e) \max_{\mathcal{A}:|\mathcal{A}|=k} F(\mathcal{A})$, i.e., $\mathcal{A}_G$ is at most a constant factor $(1 - 1/e)$ worse than the optimal solution [11]. In [7], we proved that our mutual information criterion is submodular and *approximately* monotonic: For any $\varepsilon > 0$, if we choose the discretization fine enough (polynomially-large in $1/\varepsilon$), then the solution obtained by the greedy algorithm is at most $(1 - 1/e)OPT - \varepsilon$. Algorithm 1 presents the greedy algorithm for mutual information; for details we refer the reader to [7]. However, this result only holds when we do not take communication cost into account, and does not generalize to the covering and maximization problems (1) and (2) we study in this paper. 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 (but still satisfying the quota), but which are closer together. In Sec. 7 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 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*.

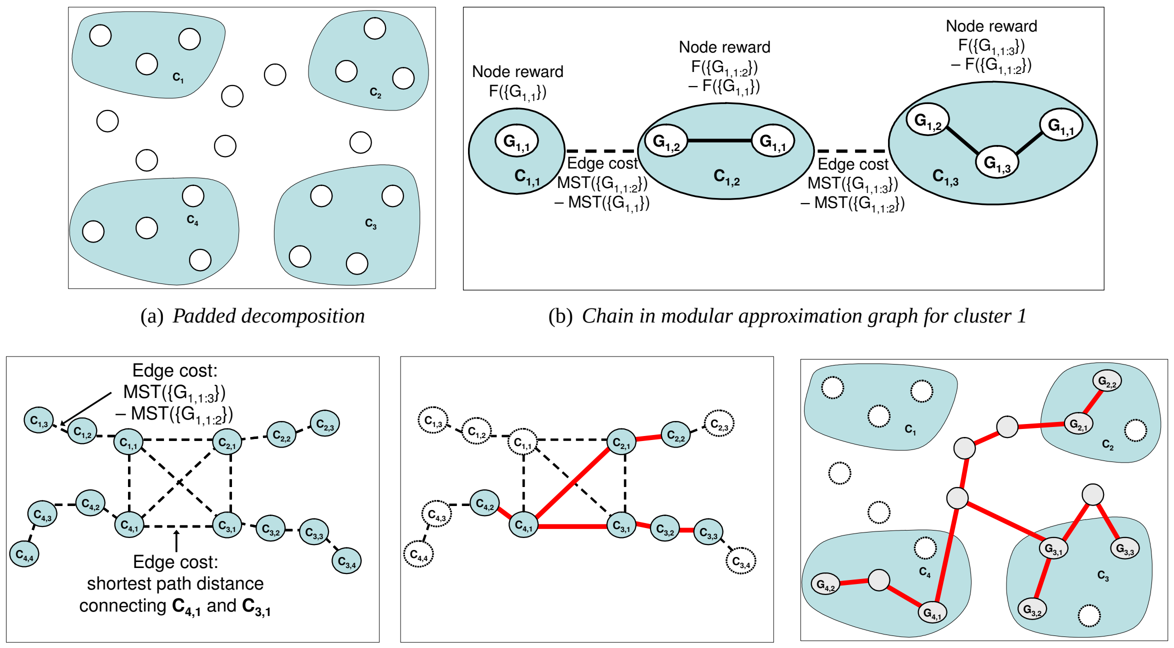
6 Approximation algorithm

In this Section, we propose an efficient approximation algorithm for selecting Padded Sensor Placements at Informative and cost-Effective Locations (*pSPIEL*). Our algorithm will exploit the problem structure via both the properties of *submodularity* and *locality* from Sec. 5. 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 clusters using Algorithm 2 (*c.f.,* Sec. 6.1, [12]), as in Fig. 3(a). All 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; furthermore, the well-separatedness and the locality property of F ensures that each cluster is approximately independent of the other clusters, and hence very informative.)

2. Use the greedy algorithm (Algorithm 1) within each cluster i to get an order $G_{i,1}, G_{i,2}, \ldots G_{i,n_i}$ on the nodes in this cluster. 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 Fig. 3(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.

3. Create a "modular approximation graph" $\mathcal{G}'$ from $\mathcal{G}$ by taking all these chains, and creating a fully connected graph on $G_{1,1}, G_{2,1}, \ldots, G_{m,1}$, the first nodes of each chain. The edge costs $(G_{i,1}, G_{i',1})$ correspond to the shortest path distances between $G_{i,1}$ and $G_{i',1}$, as in Fig. 3(c).


C₁
C₂
C₃
C₄
Node reward F({G₁,₁})
Node reward F({G₁,₁:₂}) – F({G₁,₁})
Node reward F({G₁,₁:₃}) – F({G₁,₁:₂})
G₁,₁
G₁,₂
G₁,₃
C₁,₁
C₁,₂
C₁,₃
Edge cost MST({G₁,₁:₂}) – MST({G₁,₁})
Edge cost MST({G₁,₁:₃}) – MST({G₁,₁:₂})
Edge cost: MST({G₁,₁:₃}) – MST({G₁,₁:₂})
Edge cost: shortest path distance connecting C₄,₁ and C₃,₁

(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 3: 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.

4. We now need to decide how to distribute the desired quota to the clusters. Hence, we approximately solve the Quota-MST problem (for the covering version) or the Budget-MST problem (for the maximization problem) on $\mathcal{G}'$ [13, 14], as in Fig. 3(d).

5. Expand the chosen edges of $\mathcal{G}'$ in terms of the shortest paths they represent in $\mathcal{G}$, as in Fig. 3(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.*, [12] for details). We prove the following guarantee:
Theorem 1. Given any graph $\mathcal{G} = (\mathcal{V}, E)$, and any (r, γ)-local monotone submodular function F, we can find a tree $\mathcal{T}$ with cost $\mathcal{O}(r \dim(\mathcal{V}, E)) \times \ell^*$, which spans a set $\mathcal{A}$ with $F(\mathcal{A}) \geq \Omega(\gamma) \times F(\mathcal{A}^*)$. The algorithm is randomized and runs in polynomial-time.

In other words, Theorem 1 shows that we can solve the covering and maximization problems (1) and (2) to provide a sensor placement for which the communication cost is at most a small factor (at worst logarithmic factor) larger, and for which the sensing quality is at most a constant factor worse than the optimal solution. [1] The proof can be found in our technical report [15]. In the rest of this section, we flesh out the details of the algorithm, giving more technical details and intuition about the performance of the algorithm.

6.1 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

[1]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.

```
Input: Graph $(\mathcal{V}, E)$, shortest path distance $d(\cdot,\cdot)$, $r > 0$, $\alpha \geq 64\dim(\mathcal{V}, E)$
Output: $(\alpha, r)$-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
            $\mathcal{U} \leftarrow \mathcal{U} \cup \{v\}$
        end
        $\pi \leftarrow$ random permutation on $\mathcal{U}$;
        $R \leftarrow$ uniform at random in $(r', 2r']$;
        foreach $u \in \mathcal{U}$ according to $\pi$ do
            $\mathcal{C}_u \leftarrow \{v \in \mathcal{V} : d(u,v) < R,$ and $\forall u' \in \mathcal{U}$ appearing earlier that $u$ in $\pi$ , $d(u',v) \geq R\}$;
        end
    until at least $\frac{1}{2}$ nodes $r$-padded;
end
```

Algorithm 2: Algorithm for computing padded decompositions.

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 of a padded decomposition is presented in Fig. 3(a).

If we put all nodes into a single cluster, we obtain a padded decomposition that is not very useful. Intuitively, 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 that are exactly r in size, but padded decompositions exist for arbitrary graphs for clusters sizes about a constant α larger, where α is $\Omega(\dim(\mathcal{V}, E))$ [12].

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., $\text{diam}(\mathcal{C}_i) \leq \alpha r$.

(ii) Each node $s \in \mathcal{V}$ is r-padded in the partition with probability at least ρ. (We say 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 paper, 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 2 [12].

In *pSPIEL*, for a fixed value of the locality parameter r, we gradually increase α, stopping when we achieve a partition, in which, after a small number of trials, at least half the nodes are r-padded. In expectation, the running time of this rejection sampling algorithm is polynomial. This rejection sampling is the only randomized part of our algorithm. How do we assure that, by stripping away all nodes which are not r-padded, we do not lose the most informative candidate locations? The following Lemma proves that this does not happen in expectation.

Lemma 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 $\Pr[X \in \mathcal{A}] \geq \rho$ for all $X \in \mathcal{B}$. Then $\mathbb{E}[F(\mathcal{A})] \geq \rho F(\mathcal{B})$.

Let $\mathcal{B}$ be the optimal solution for the covering or maximization problem, and let $\mathcal{A}$ denote the subset of r-padded nodes within $\mathcal{B}$. Then Lemma 2 proves that in expectation, there is a subset of nodes (with cost bounded by the cost of $\mathcal{B}$), which is at most a constant factor ρ worse than $\mathcal{B}$.

6.2 The greedy algorithm

After having sampled a padded decomposition, we run the greedy algorithm as presented in Algorithm 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 $G_{i,1}$, $G_{i,2}$, …, $G_{i_{n_i}}$ 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 [11] 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, …, j_m, where $\mathcal{C}^*_{k,j_k}$ denotes the optimal selection of j_k nodes within cluster k.

6.3 The modular approximation graph $\mathcal{G}'$

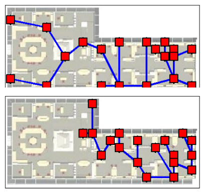
In step 3), *pSPIEL* creates the auxiliary *modular approximation graph* (MAG) $\mathcal{G}'$ from $\mathcal{G}$, whose nodes are the greedy sets $\mathcal{C}_{i,j}$. The greedy sets for cluster i are arranged in a chain with edge $e_{i,j}$ connecting $\mathcal{C}_{i,j}$ and $\mathcal{C}_{i,j+1}$ for 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 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, $\mathrm{c}_{MST}(\mathcal{C}_{i,k}) \leq \alpha\ r\ k$. Fig. 3(b) presents an example of a chain associated with cluster 1 in Fig. 3(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 $\mathcal{G}'$. Fig. 3(c) presents the modular approximation graph associated with the padded decomposition of Fig. 3(a).

6.4 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 (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 has 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 weight, 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 [13] for rooted Quota-MST, and $3 + \varepsilon$ (for any $\varepsilon > 0$) for unrooted Budget-MST [16]. We can use these algorithms to get an approximate solution for the covering and maximization problems in $\mathcal{G}'$. From Sec. 6.3, 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.) We omit all details here due to space limitations. Fig. 3(d) presents an example of such a Quota-MST solution.

6.5 Transferring the solution from $\mathcal{G}'$ back to $\mathcal{G}$

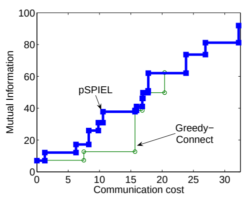
The Quota- or Budget-MST algorithms select a tree $\mathcal{T}'$ in $\mathcal{G}'$, which is at most a constant factor worse than the optimal such tree. We use this solution $\mathcal{T}'$ obtained for $\mathcal{G}'$ to select a tree $\mathcal{T} \subseteq \mathcal{G}$ thus: For every cluster i, if $\mathcal{C}_{i,j} \in \mathcal{T}'$ we mark $G_{i,1}$, …, $G_{i,j}$ in $\mathcal{G}$. We then select $\mathcal{T}$ to be an approximately optimal Steiner tree connecting all marked nodes in $\mathcal{G}$. E.g., we can compute an MST for the fully connected graph with all marked vertices, where the cost of an edge between s and t is the shortest path distance between s and t in $\mathcal{G}$. This tree $\mathcal{T}$ is the approximate solution promised in Theorem 1. (Fig. 3(e) presents the expansion of the Quota-MST from Fig. 3(d).)



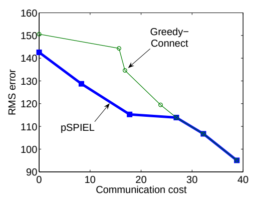
(a) *Placements*

Metric	M20	pS19	pS12
RMS	413.5	127.0	162.2
MAD	202.6	71.1	102.5
Pred. c.	24.4	19.9	15.3
Real c.	22.9	21.8	15.0

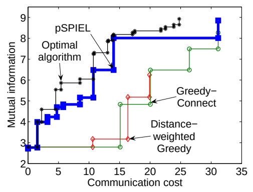
(b) *Costs and prediction qualities*



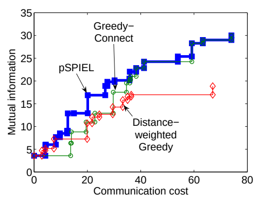
(c) *Cost-benefit for light data*



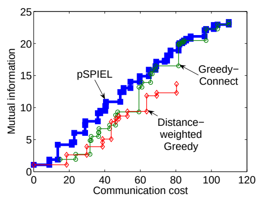
(d) *RMS error for light data*



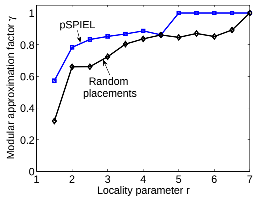
(e) *Small temperature data set*



(f) *Cost-benefit for temperature*



(g) *Cost-benefit for precipitation*



(h) (r, γ) *for temperature data*

Figure 4: Experimental results. (a) shows the expert placement (top) and a placement proposed by *pSPIEL*. (b) presents root-mean-squares and mean-absolute-deviation 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. (e) compares trade-off curves for a small subset of the temperature data. (f) shows tradeoff curves for the temperature GPs on a 10x10 grid. (g) compares tradeoffs for precipitation data from 167 weather stations. (h) compares the locality parameter r and the loss γ incurred by the modular approximation for the temperature GPs.

6.6 Additional implementation details

pSPIEL relies heavily on the monotonic 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 repear 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 Sec. 7.

7 Experiments

In order to evaluate our proposed method, we computed sensor placements for three real-world problems: Indoor illumination measurement, the temperature prediction task as described in our running example, and precipitation prediction in the Pacific Northwest.

7.1 Proof-of-concept study

Based on the TinyOS Surge application, we built a system for multi-hop collection of light data and message transmission logs, and combined it with our implementation of *pSPIEL*. 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 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. Fig. 4(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. Fig. 4(b) presents the results. Interestingly, the proposed placement (pS19) drastically reduces the prediction error to about a third. Our explanation for this result is, that our baseline deployment placed most sensors near the windows, which seemed to have mislead the prediction. Also, the prediction of the manual placement is even worse than simply predicting the mean light intensity, so a bad placement can be worse than no placement at all. Furthermore, *pSPIEL* decided not to explore the large western area. The reason is that this part of the lab was not occupied during the night, hence there was little fluctuation with indoor lighting – the variation in sunlight could be predicted from the deployment in the eastern part. We repeated the analysis for a 12 motes subsample (pS12), also proposed by *pSPIEL*. We also compared the predicted communication cost using the GPs with the measured communication cost. Fig. 4(b) shows that the prediction matches well to the measurement. Figs. 4(c) and 4(d) show that *pSPIEL* outperforms the Greedy heuristic explained below, both in the sensing quality and communication cost tradeoff, and also on the predictive RMS error.

7.2 Indoor temperature measurements

In our second set of experiments, we used an existing deployment (*c.f.,* Fig. 1(a)) of 52 wireless sensor motes to learn a model for predicting temperature and communication cost in a building. 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 1), and then connects the selected sensors using a Steiner tree approximation. The second heuristic, *Distance-weighted Greedy*, is inspired from an algorithm which provides near-optimal solutions to the Quota-MST problem [17]. It 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.,* [17]).

Fig. 4(e) 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. Fig. 4(e) indicates that the performance of *pSPIEL* is significantly closer to the optimal solution than any of the two heuristics. Fig. 4(f) 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 Fig. 4(f) 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* performs not much better than *Greedy-Connect*. This is due to fact, that for large quotas, too many nodes are ignored in step 1) of the algorithm, since they are not padded. Hence, *pSPIEL* will increase the locality constant r, until r is large enough that all nodes are padded. In this case, *pSPIEL* essentially reverts back to the *Greedy-Connect* algorithm. Also, the more nodes are placed, the less will communication cost be an issue. In the important region between 25% and 80% however, *pSPIEL* clearly outperforms the heuristics. Also, our results indicates 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 Fig. 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 $\mathcal{G}$, suggesting 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, the approximation factor is even higher.

7.3 Precipitation data

In our third application, our goal was to place sensors for predicting precipitation in the Pacific North-West. Our data set consisted of daily precipitation data collected from 167 regions during the years 1949–1994[18]. We followed the preprocessing from [7]. 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. Fig. 4(g) compares the sensing quality – communication cost tradeoff curves for selecting placements from all 167 locations. *pSPIEL* outperforms the heuristics up to very large quotas.

8 Conclusions

We proposed a unified approach for placing networks of wireless sensors. 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. 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.

Acknowledgements: We would like to thank Adrian Perrig for providing us with motes and Vipul Singhvi for helping with the sensors. This work was supported by NSF Grant No. CNS-0509383 and a gift from Intel Corporation.

References

[1] A. Deshpande, C. Guestrin, S. Madden, J. Hellerstein, and W. Hong, “Model-driven data acquisition in sensor networks,” in *VLDB*, 2004.

[2] H. Gupta, S. R. Das, and Q. Gu, “Connected sensor cover: Self-organization of sensor networks for efficient query execution,” in *MobiHoc*, 2003.

[3] K. Kar and S. Banerjee, “Node placement for connected coverage in sensor networks,” in *WiOpt*, 2003.

[4] S. Funke, A. Kesselman, F. Kuhn, Z. Lotker, and M. Segal, “Improved approximation algorithms for connected sensor cover,” in *ADHOC*, 04.

[5] A. Cerpa, J. L. Wong, L. Kuang, M. Potkonjak, and D. Estrin, “Statistical model of lossy links in wireless sensor networks,” in *IPSN*, 2005.

[6] N. A. Cressie, *Statistics for Spatial Data*. Wiley, 1991.

[7] C. Guestrin, A. Krause, and A. Singh, “Near-optimal sensor placements in gaussian processes,” in *ICML*, 2005.

[8] C. Guestrin, P. Bodik, R. Thibaux, M. Paskin, and S. Madden, “Distributed regression: an efficient framework for modeling sensor network data,” in *IPSN*, 2004.

[9] D. J. Nott and W. T. M. Dunsmuir, “Estimation of nonstationary spatial covariance structure,” *Biometrika*, vol. 89, pp. 819–829, 2002.

[10] L. Csato, E. Fokue, M. Opper, B. Schottky, and O. Winther, “Efficient approaches to gaussian process classification,” in *NIPS*, 2000.

[11] G. Nemhauser, L. Wolsey, and M. Fisher, “An analysis of the approximations for maximizing submodular set functions,” *Mathematical Programming*, vol. 14, pp. 265–294, 1978.

[12] A. Gupta, R. Krauthgamer, and J. R. Lee, “Bounded geometries, fractals, and low-distortion embeddings,” in *FOCS*, 2003.

[13] N. Garg, “Saving an epsilon: a 2-approximation for the k-mst problem in graphs,” in *STOC*, 2005.

[14] D. S. Johnson, M. Minkoff, and S. Phillips, “The prize collecting steiner tree problem: theory and practice,” in *SODA*, 2000.

[15] A. Krause, C. Guestrin, A. Gupta, and J. Kleinberg, “Near-optimal sensor placements: Maximizing information while minimizing communication cost,” CMU-CALD-05-110, Tech. Rep., 2005.

[16] A. Levin, “A better approximation algorithm for the budget prize collecting tree problem,” *Ops. Res. Lett.*, vol. 32, pp. 316–319, 2004.

[17] B. Awerbuch, Y. Azar, A. Blum, and S. Vempala, “New approximation guarantees for minimum-weight k-trees and prize-collecting salesmen,” *SIAM J. Computing*, vol. 28, pp. 254–262, 1999.

[18] 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.

9 Appendix

Proof of Lemma 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 with each 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

$$\mathbb{E}[F(\mathcal{A})] = E(\mathcal{P}_0) \geq E(\mathcal{P}^*) \geq \rho F(\mathcal{B}), \tag{9}$$

as required. □

In order to prove Theorem 1, let us consider the subset $\mathcal{A}^*$ spanned by the optimal tree, 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 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 that recoups a reasonable fraction of this “padded value”.

The following lemma will be useful in converting subtrees of $\mathcal{G}'$ back to solutions of our original problem.
Proposition 3. *Given any subtree $\mathcal{T}'$ of $\mathcal{G}'$ with weight W, it is possible to find a subtree $\mathcal{T} \subseteq \mathcal{G}$ spanning the same vertices $\mathcal{A}'$, with a total length no more than $\ell(\mathcal{T}')$, 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. □

Proposition 4. *If the graph $\mathcal{G}$ contains a subtree of length ℓ^* and value $F(\mathcal{A}^*)$, then there is a subtree $\mathcal{T}'$ of the graph $\mathcal{G}'$ that has length at most*

$$\ell^* \times (\alpha(r + 2) + 2) \tag{10}$$

and whose expected weight is at least

$$F(\mathcal{A}^*) \times (1 - e^{-1}) \times \rho \tag{11}$$

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 clusters $\mathcal{C}_1, \mathcal{C}_2, \ldots, \mathcal{C}_s$ be occupied. We start building $\mathcal{T}'$ by adding a spanning tree on the centers of the clusters that are occupied.

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

$$2(\alpha + 1)\,\ell^*. \tag{12}$$

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; i.e., the vertices $\{Z_i = G_{i,1}, G_{i,2}, \ldots, G_{i,n_i}\}$. Firstly, 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 (12) gives us 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

$$(1 - e^{-1})F(\overline{\mathcal{A}^* \cap \mathcal{C}_i}) \tag{13}$$

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 $(1 - e^{-1})\rho F(\mathcal{A}^*)$. □

Combining these results, we now prove a slightly more detailed statement of Theorem 1:
Theorem 5. *For the covering problem* (1), pSPIEL *will find a solution $\mathcal{T}$, with cost at most*

$$\kappa_{Quota}\,\ell^*\,(\alpha(r+2)+2) \tag{14}$$

and whose expected weight is at least

$$(1 - e^{-1})\,\gamma\rho F(\mathcal{A}^*), \tag{15}$$

where ℓ^* is the weight of the optimum tree $\mathcal{A}^*$. For the maximization problem (2), pSPIEL *will find a solution $\mathcal{T}$ with cost at most*

$$\ell^*\,(\alpha(r+2)+2) \tag{16}$$

and whose expected weight is at least

$$\kappa_{Budget}^{-1}(1 - e^{-1})\,\gamma\rho F(\mathcal{A}^*), \tag{17}$$

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 [13, 14]).

Proof. Proposition 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 3 handles the transfer of the solution to the original graph $\mathcal{G}$. Hence, in order to solve the covering problem (1) or optimization problem (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, unfortunately, there is no known constant-factor guarantee known for the rooted version. We can however exploit the structure of the MAG to still get an approximation guarantee and prove Theorem 1. We simply need to prune all nodes in $\mathcal{G}'$ which are further than $B = \ell^*\,(\alpha(r+2)+2)$ away from the core of $\mathcal{G}'$, and then run the unrooted approximation algorithm [14] on $\mathcal{G}'$. If this algorithm, started with budget $B = \ell^*\,(\alpha(r+2)+2)$ 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. □