Approximation of Graphical Probabilistic Models by Iterative Dynamic Discretization and its Application to Time-Series Segmentation

Lonnie Dale Chrisman

September 1996
CMU-CS-96-166

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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

Thesis Committee:
Tom Mitchell, Chair
Reid Simmons, Chair
Matthew Mason
Padhraic Smyth, University of California at Irvine/JPL

Supported by NASA-Jet Propulsion Laboratory under Fellowship Grant No. NGT-51039, and by NASA under Grant No. NAGW-1175. The views and conclusions contained in this thesis are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of NASA or the U.S. government.

Keywords: Iterative Dynamic Discretization, Model-based Time-Series Segmentation, approximation of graphical probabilistic models, continuous variables in probabilistic models, discretization of continuous variables, Bayesian networks, Markov fields, Gibbs sampling, focused Gibbs sampling, exact propagation, clique-marginal propagation, combination of Gibbs sampling and exact propagation, knowledge-based model construction, Hidden Segmented Semi-Markov Models (HSSMMs), Hidden Segmented Generalized Semi-Markov Models (HSGSMMs).

School of Computer Science

DOCTORAL THESIS
in the field of
Computer Science

Approximation of Graphical Probabilistic Models by Iterative Dynamic Discretization and Application to Time-Series Segmentation

LONNIE D. CHRISMAN

Submitted in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

ACCEPTED:

Reid D Simmons	9/27/96
THESIS COMMITTEE CHAIR	DATE
Tom M Mitchell	9/27/96
THESIS COMMITTEE CHAIR	DATE
Morris	10/24/96
DEPARTMENT HEAD	DATE

APPROVED:

R. Rashid	10/28/96
DEAN	DATE

Abstract

Most artificial intelligence applications must cope with uncertainty. Recent developments with graphical probabilistic models such as Bayesian networks have introduced useful methods for reasoning explicitly about degrees of uncertainty. This thesis explores a method called *iterative dynamic discretization* for approximating probabilistic inference in graphical networks. Continuous variables (or variables with enormous sample spaces) are replaced by discrete variables with a small number of possible values, and then the simplified discrete model is solved using exact propagation methods. The results of this computation are then used to find an improved discretization for the problem instance, and the process is iterated. The algorithm can be viewed as applying Gibbs sampling to the space of possible discretizations, obtaining a method for combining stochastic simulation methods with exact propagation. Alternatively, it can be viewed as an instance of approximate iterative knowledge-based model construction.

The thesis applies iterative dynamic discretization to a model-based time-series segmentation problem. A formalism for modeling qualitative signal shapes, durations, transitions, and uncertainty in multi-dimensional time series, called a Hidden Segmented Semi-Markov Model, is introduced and used to define a probabilistic model for the time-series segmentation task. This is converted to a graphical probabilistic model and solved by iterative dynamic discretization. Iterative dynamic discretization is found to require substantially fewer iterations to obtain a given level of performance compared to Gibbs sampling.

To my parents:

Leon Dale Chrisman
and
Linda Lucille Chambers

Contents

Acknowledgements

The completion of my Ph.D. is milestone that I would not have aimed for or reached had it not been for the great many people who have shaped who I am, encouraged me, or helped me along the way. I am especially grateful to:

- My parents. My parents deserve more credit than anyone for getting me where I am today. My father shaped my interests tremendously, exposing me to electronics and computers at a very young age and sharing his enthusiasm for science and technology. It is largely because of these early influences that I have ended up in a career of engineering and computer science. My mother is perhaps the most energetic and hard working person I have ever known (although my sister Lucinda is a clear contender for this distinction as well). It seems like she has always been actively involved with everything there is to be involved with and never stops to rest. She inspires me to work hard, try hard, to enjoy what I am doing, and to become involved in many things. It is from these values and interests that I have carved my path in life to this point.

- My family. For almost 13 years, my wife Debbie (who has been my wife for eight of those years) has given me tremendous support and encouragement. While I have been in graduate school, she has had to put up with my research-anxiety and thesis-induced depression, but has stood strong and never wavered in her support. Although I am listed as author, this milestone belongs partially to her as well. I am also grateful to my daughters, Brianna Sierra and Whitney Sequoia. There have been some very difficult times during the last few years, but during these they have provided me with a very positive light in my life. Without them in my life, I serious doubt I would have made it this far.

- My early role models. My peers and leaders in Scouting helped me develop many of the most valuable skills I now possess. Especially influential was an early scoutmaster, Wayne Franklin, who was an excellent teacher of leadership skills. Herb Pardula employed me as a computer programmer (at Systems for Automatic Test) while I was in high school and college and was largely my mentor. He provided me with very unique opportunities which greatly contributed to my engineering, programming, and management skills.

- My advisors and committee. Reid Simmons and Tom Mitchell have dedicated a enormous amount of their time and energy over the past eight years to advising and assisting me with my research, including the research reported in this thesis. They also gave me very unique and interesting opportunities to work on Mars and Lunar rovers and indoor mobile robots. They have guided me to all the research topics that I have done research on while at C.M.U. and have been absolutely instrumental in designing the program and agenda for my thesis research. They have discussed my research regularly with me every step of the way and help to correct my often incorrect or misconceived ideas.

  My thesis committee: Reid Simmons, Tom Mitchell, Matt Mason, and Padhraic Smyth, for reading this thesis, providing me with comments, and keeping me honest. I am also grateful to Glenn Shafer who was originally a member of my thesis committee before I changed topics and even took the time to fly out to Pittsburgh to attend my proposal. My minor advisor, Doug Tygar, also gave me great encouragement.

  I am also grateful to Stuart Russell, who was my advisor while I was a graduate student at U.C. Berkeley. It was my first exposure to the field of A.I., and I may have learned more about A.I. and Machine Learning during that year than I will ever learn again.

- Friends and Colleagues.

  I have benefited in many ways from an incredible number of friends and colleagues over the years. Here I will only list the most significant of these. Some pre-CMU friends that have had significant influence on me and my career path include: Jeff Vadasz, Allan Biegaj, Doug Evans, Tiffany Denny, and Eric Wefald. At CMU, Sven Koenig, Rich Goodwin, Rich Caruana, Hank Wan, Justin Boyan, Geoff Gordon, Jeff Jackson, Michael Littman, Oren Etzioni, Haym Hirsh, Sebastian Thrun, Fabio Cozman, and Jim Blythe have all significantly influenced my research at one point or another. Of these, Sven Koenig and Rich Goodwin have been especially important. Friends: Joe and Joyce Crowley, Gary Schaub, Bill Carr, Tom and Quoeta Coyne, Antonio Martinez, and Inmacula Garcia-Colavidas. Outside of CMU: Smadar Kedar, Mark Drummond, John Bresina, Leslie Pack Kaelbling, Tom Dean, Andrew McCallum, and Rich Sutton have all in one way or another impacted my research. Also Javier Alaman, Pablo Castells, Idoia Alarcon and other members of the Instituto de Ingeniería del Conocimento in Madrid. Thanks also my step-father Jack Chambers, who has worked in and often discusses robotics with me, and my late step-mother Nikki (nee Karengin) Chrisman, who was inspiring and a source of encouragement. I am also grateful to numerous other friends and colleagues who have also influenced me along the way.

- Sponsors

  The SELMON group at JPL provided me with a NASA graduate student fellowship, as well as with a source for the application area that motivated the initial research that led to this thesis. I am grateful for both of these. At JPL, Rich Doyle, Dennis DeCoste, Nick Rouquette, Padhraic Smyth and others provided comments and feedback at several stages during the work in this thesis.

  I have also been funded as a graduate student by a General Electric fellowship while at U.C. Berkeley, NASA Mars and Lunar Rover projects at CMU, and partially by the School of Computer Science at one point. For all of these, I am very thankful.

Chapter 1

Introduction

Most artificial intelligence applications are riddled with uncertainty. Information about the world state is generally incomplete, gaps exist in domain knowledge, sensors are noisy, and information sources are often unreliable. Indeed, many of the technologies that have emerged from Artificial Intelligence research can be viewed as alternative approaches for dealing with this ubiquitous uncertainty. Examples include machine learning algorithms, nonmonotonic reasoning formalisms, fuzzy logic, pattern recognition techniques, heuristic search, and others. Probabilistic and decision-theoretic techniques attempt to deal with uncertainty by modeling and reasoning about degrees of uncertainty explicitly. This thesis explores some aspects of probabilistic inference.

1.1 Thesis Overview

This thesis studies and develops algorithms for probabilistic inference by attacking a specific and challenging application problem: model-based time-series segmentation. Chapter 2 introduces a modeling formalism for expressing knowledge about time series. This formalism is called a Hidden Segmented Semi-Markov Model (HSSMM) and allows qualitative information about raw signal shapes to be expressed along with quantitative information about durations and explicit measures of uncertainty. The model also naturally accommodates multiple sensor streams. A HSSMM implicitly defines a prior probability distribution over the space of possible segmentations given a data stream. The segmentation task is to find the optimal segmentation — i.e., the segmentation with the maximum a posterior (MAP) probability given the time-series model and the time-series data. For this motivating application, finding a MAP configuration is the basic problem to be solved.

The HSSMM is a convenient and expressive modeling formalism from a user's perspective, but the associated optimization problem can be a formidable computational challenge. This creates an interesting domain for experimenting with sophisticated probabilistic inference techniques.

To obtain a solution to the computational challenge, Chapter 3 decomposes the huge optimization problem into a collection of much smaller optimization problems. The decomposition is based on probabilistic conditional independence and has foundations in recent work on graphical probabilistic models and Markov field theory. I refer to this process as *structural decomposition*. The techniques reduce, e.g., a 200-dimensional optimization problem into 200 three-dimensional optimization problems, thus enabling substantial benefit from leveraging existing structure. Furthermore, the decomposition is information-preserving, i.e., even though the computation challenge is substantially simplified, the answer to the decomposed problem is still an exact solution to the original problem.

Chapter 3 emphasizes the generality of decomposition and associated techniques. Although these techniques are not new to this thesis, casting the problem in terms of these ideas provides several advantages. First, it establishes a clear connection between the work in this thesis and contemporary conceptions of probabilistic inference. It also introduces a flexibility that would not be obtained if an algorithm to solve the HSSMM-based segmentation problem were simply described. Almost inevitably, any future application of the HSSMM formalism will find extensions of one form or another necessary. While it is impossible to

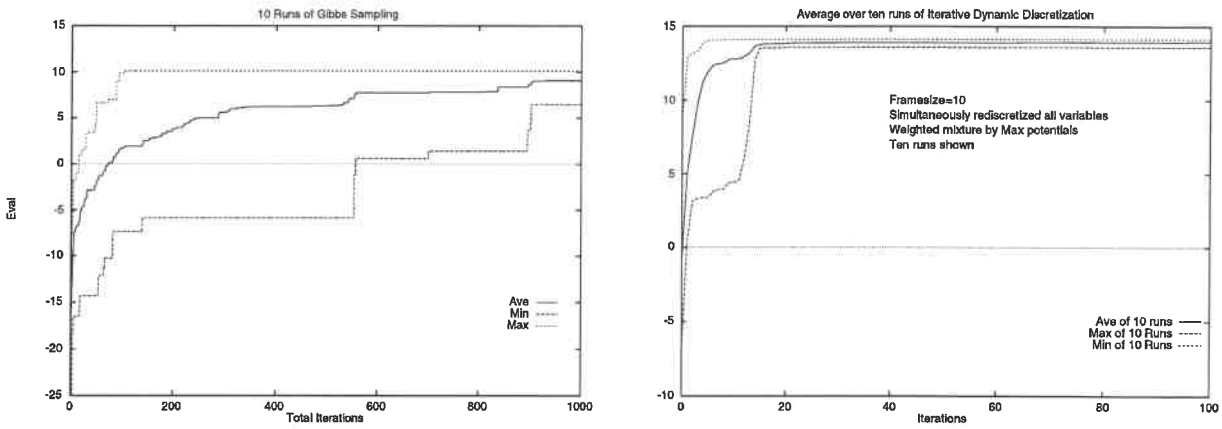


Figure 1.1: On the left is the performance of Gibbs sampling averaged over ten runs as a function of total iterations. On the right is the performance of a certain variant of iterative dynamic discretization on the same problem, also averaged over ten runs. Note that the x-axes are scaled differently. These experiments are described in Chapter 4.

predict in advance what these would be, the same process of structural decomposition can once again be applied in an almost mechanical fashion to support whatever extensions arise in the future. Finally, these techniques are more broadly applicable to other problems.

Structural decomposition simplifies the computational problem tremendously, but for the time-series segmentation problem, the simplification is not sufficient to obtain a computationally feasible algorithm. This is due to the fact that the variables in the problem are continuous (i.e., real-valued) and the distributions lack convenient conjugacy properties. For this hurdle, I turn to approximate techniques based on discretization.

Chapter 4 introduces *iterative dynamic discretization* in which continuous variables in a model are replaced with discrete variables taking on a small finite number of possible values. The resulting decomposed finite models can be solved using exact methods, but the answer obtained (i.e., the segmentation found) is only optimal relative to the discretization. It may not be optimal for the original continuous model. Thus, it is important to find good discretizations.

Choosing a good discretization is a hard problem. A good discretization is one whose optimum is nearly optimal for the original continuous problem. It would be unreasonable to assume that this can be done well on the first attempt, so instead, the whole process is iterated. The important idea is:

> *Information learned by solving a problem using the current discretized model is utilized for choosing the next discretization of the original model.*

It is the central thesis of this thesis that this can be done effectively.

This iterative process strives to improve the discretization on any given iteration. Besides being iterative, the process is also *dynamic* in that information about the current problem instance (including the currently observed sensor streams) have an important influence on the choice of discretization.

Iterative dynamic discretization is obviously an approximation technique, and the exploration of the algorithm in this thesis is largely empirical. Chapter 4 demonstrates that the algorithm can find good solutions in orders of magnitude fewer iterations than Gibbs sampling, a competing approximation technique (see Figure 1.1). An evolving implementation and experimentation with the algorithm has molded the algorithm into its current form. While it is a novel algorithm, it is closely connected to many different existing approximate techniques for probabilistic inference, from Gibbs sampling to knowledge-based model construction. These many connections create several perspectives on the contributions of this thesis.

The remainder of the introduction chapter reviews many of the basic foundations of probabilistic inference and existing techniques, and outlines where this work fits and the contributions it makes.

		stain-pos		stain-neg	
		react-pos	react-neg	react-pos	react-neg
male	benign	0.0144	0.0336	0.0576	0.1344
	malignant	0.1620	0.1620	0.0180	0.0180
female	benign	0.0168	0.0392	0.0672	0.1568
	malignant	0.0540	0.0540	0.0060	0.0060

Figure 1.2: A doctor's prior degrees of belief.

1.2 Probabilistic Inference

- A meteorologist says, "There is a 60% chance of rain tomorrow."
- A doctor tells a patient, "There is a 30% chance your tumor is malignant."
- An expert system reports that there is a 0.25 probability of a large oil deposit at a certain location.
- A mission control operator reports, "the water spray boiler is *probably* frozen, and if so, the primary auxiliary power unit is *unlikely* to start for reentry without further intervention."
- My car's idling problem is *most likely* due to a faulty oxygen sensor.
- As the phone rings, Joe says, "that is *probably* my wife calling."

Probabilistic inferences abound in everyday life. All the statements above explicitly express a degree of belief or uncertainty, and all these estimates are somehow based on a deeper or more general knowledge tailored to the particular situation. The methods behind such inferences are of great interest to artificial intelligence researchers and practitioners, both because they are so common in naturally occurring applications, and because explicitly reasoning about uncertainty can improve the robustness and quality of intelligent systems.

Bayesian probability theory provides one well-studied formalism for describing inferences of this nature. The basic idea is that an agent has a store of background prior knowledge (or an appropriate prior *model*), and the agent derives inferences such as the above by combining this information with observations specific to the current situation. An agent's prior background knowledge encodes a mutually exclusive and exhaustive set of all possible situations (called the *frame of discernment* or *sample space*), and a relative degree of belief for each situation. Observations serve to narrow the possibilities while preserving relative degrees of belief, thus providing a clear and precise basis for manipulating and reasoning about degrees of uncertainty.

For example, a doctor may model a situation in the following way. A tumor is either malignant or benign, and stain and reaction tests on a biopsy may come out either positive or negative. Furthermore, a patient is either male or female. This creates 16 possible situations. The first important part of the doctor's model is that these 16 situations are the only ones possible. Note that this model already assumes a tumor actually exists, etc.

For the second part of the model, the doctor assigns relative likelihoods to each of these joint situations. The table in Figure 1.2 shows a possible assignment of probabilities. Now, knowing the patient to be male and finding the stain test result to be positive, the doctor can narrow the possibilities to four: $\langle male, benign, stain\text{-}pos, react\text{-}pos\rangle$, $\langle male, benign, stain\text{-}pos, react\text{-}neg\rangle$, $\langle male, malignant, stain\text{-}pos, react\text{-}pos\rangle$, and $\langle male, malignant, stain\text{-}pos, react\text{-}neg\rangle$. The doctor thus infers that the odds are $0.1620 + 0.1620 : 0.0144 + 0.0336 = 324 : 48$ that the tumor is malignant, or stated in terms of probabilities, that there is an 87% chance that the tumor is malignant. The probability after the evidence is incorporated is the *posterior* probability.

This style of reasoning is the basic foundation of Bayesian probabilistic inference. The simplicity of this style of inference makes the technique very attractive as a computational tool, but it also highlights a few of the fundamental limitations.

The Bayesian probabilistic formalism has the following fundamental limiting factors:

- The situations identified by a model must be exhaustive and mutually exclusive.
- If a probability is to be assigned to a proposition, or if any evidence is observed, the proposition or evidence must be expressed as a subset of the possible situations delineated/discerned by the model.
- The numbers (the prior) must come from somewhere.

Of these, the first two are unarguably the most limiting, but it is hard to see how any computational framework would avoid the same (or analogous) fundamental limitations. The ability to discern situations, and the exhaustiveness of a model, are both enhanced by using larger models, i.e., models that allow for a greater number of possible situations. The nature of artificial intelligence applications makes models with enormous numbers of possibilities of great interest, and many techniques are aimed at such situations. The third limitation, the origin of the prior, and even the nature of the prior, are the topic of great debate throughout the literature, and the source for many divergent philosophies.

Many adherents of the Bayesian methodology feel that reasonable priors are not too difficult to come by in many applications, and that reasonable priors (even if not incredibly precise) usually lead to reasonable posteriors ([Pradhan *et al.*, 1995]). It is also common to use *noninformative* priors when little relevant knowledge is available. An example is to assign all possible situations the same probability. The term "noninformative" is a great misnomer since the set of possible situations that a model identifies encodes much about relative probabilities even when the numerical assignments are supposedly noninformative. Nevertheless, it is generally accepted that the numerical assignments do allow one to easily vary the degree of specificity or diffuseness of knowledge, so that some pragmatic application of noninformative priors is sometimes reasonable.

Modern usage of the term *probabilistic inference* generally refers to a situation such as that described here, with the important point being that inferences are made from a prior. The term *statistical inference* is used more generally to include activities such as hypothesis testing, regression, maximum likelihood estimation, etc.

1.3 Historical Perspective

Probabilistic and decision-theoretic approaches were among the earliest to receive attention by artificial intelligence researchers ([Simon, 1952b], [Simon, 1952a], [Simon, 1955], [Peirce, 1956], [Warner *et al.*, 1961]). These techniques explicitly and quantitatively model degrees of uncertainty, and offer (at least in theory) many advantages and enable many capabilities. However, researchers quickly recognized the intractability of the techniques, and also to some extent representational limitations, and largely became disenchanted with such approaches ([Simon, 1955], [Gorry and Barnett, 1968], [McCarthy and Hayes, 1969], [Szolovits and Pauker, 1978]). As a result, the artificial intelligence field saw a very limited use of probabilistic techniques during the 1960's and 1970's. However, during this time, a small number of A.I. researchers and practitioners did continue to use and propound the advantages of probabilistic techniques ([Minsky and Selfridge, 1961, Nilsson, 1965, Wallace and Boulton, 1968, Gorry and Barnett, 1968, Ginsberg, 1969, Munson, 1971, de Dombal *et al.*, 1972, Duda and Hart, 1973, Gorry *et al.*, 1973, Jacobs and Kiefer, 1973, Feldman and Yakimovsky, 1974, Coles *et al.*, 1975, Shortliffe and Buchanan, 1975, Shortliffe, 1976, Duda *et al.*, 1976, Good, 1977, Feldman and Sproull, 1977, Cheeseman, 1985]).

The sentiment towards probabilistic techniques in A.I. turned around in the 1980's and techniques that explicitly represent uncertainty continue to be a very integral part of mainstream A.I. in the 1990's. There are many reasons for the revitalization of interest in probabilistic techniques, one of the most important being the discovery and maturation of graphical probabilistic modeling formalisms. Among these, the directed graphical representation, referred to by various names including *Bayesian network, belief network, influence diagram, directed Markov field, recursive graphical model, probabilistic causal network*, and *directed inference network*, is perhaps the best known.

Graphical probabilistic models date as far back as the work of Sewall Wright in the 1920's ([Wright, 1921], [Wright, 1934]), but went entirely unrecognized for many decades. They were largely reinvented in the fields of decision analysis and statistics during the 1970's and early 1980's ([Good, 1961], [Howard, 1968], [Howard, 1970], [Miller *et al.*, 1976], [Speed, 1979], [Dawid, 1979], [Dawid, 1980], [Howard and Matheson,

1984a], [Lauritzen, 1982], [Kiiveri *et al.*, 1984], [Lauritzen *et al.*, 1984]), and partially by researchers in Artificial Intelligence in the late 1970's ([Shortliffe and Buchanan, 1975], [Shortliffe, 1976], [Duda *et al.*, 1976]). The most notable progenitor of A.I.'s modern Bayesian network was the PROSPECTOR system ([Duda *et al.*, 1976]), which used a directed graph representation based on conditional probabilities to encode knowledge and make inferences in a manner similar to rule-based systems. These works utilized the graphical representation primarily as a method for conveniently representing, assessing, and modularizing probabilistic knowledge.

In the 1980's, a new idea emerged: That the graphical structure of probabilistic knowledge can also "structure" computation and thereby allow probabilistic and decision-theoretic problems to be solved efficiently. The idea again dates back to the PROSPECTOR ([Duda *et al.*, 1976]) and MYCIN ([Shortliffe and Buchanan, 1975], [Shortliffe, 1976]) systems where attempts were made to chain together probabilistic rules in a fashion similar to the way in which logical rules in a rule-based system are chained together during inference. These early systems did not always adhere to the principles of probability theory and experienced some problems as a result ([Horvitz and Heckerman, 1986],[Heckerman and Horvitz, 1988]). Judea Pearl introduced a clean and well-founded Bayes message-passing scheme for directed poly-trees in [Pearl, 1982] and initiated the modern conception of using conditional independence-based structure to organize computation. Further advancements extended these propagation algorithms to more general structures ([Spiegelhalter, 1986], [Pearl, 1986b], [Shachter, 1986], [Lauritzen and Spiegelhalter, 1988], [Pearl, 1988], [Andersen *et al.*, 1989], [Lauritzen and Wermuth, 1989], [Shenoy and Shafer, 1990], [Jensen *et al.*, 1990b], [Jensen *et al.*, 1990a]). These advancements addressed the earlier sentiment regarding the intractability of probabilistic and decision-theoretic techniques, and has resulted in a renewal of interest in applying probabilistic approaches to problems of interest within Artificial Intelligence.

Graphical decompositions of probabilistic knowledge can be leveraged in order to achieve computational efficiency in many applications, often meaning the difference between a feasible implementation and absolute hopeless intractability. However in general, most interesting inference problems with graphical probabilistic models are known to be NP-hard ([Rosenthal, 1975], [Cooper, 1987], [Cooper, 1990b], [Verma and Pearl, 1993], [Shimony, 1994]), even if the answer is only to be approximated ([Dagum and Luby, 1993]). The actual complexity of a given problem generally depends on the precise graphical structure of the probabilistic knowledge. Graphical decomposition can significantly simplify a problem of probabilistic inference, but it is not always, by itself, enough.

1.4 Graphical Probabilistic Models

Once we become concerned with large models, the structure of knowledge becomes critical, and typically the dominant concern in one form or another. In fact, [Shafer and Pearl, 1990, Page 3] state "that probability theory is more fundamentally concerned with the structure of reasoning and causation than with numbers." The model in Figure 1.2 does not indicate any explicit structure — there is simply a single number attached to each possible situation. This kind of explicit enumeration of situations clearly does not scale reasonably with the number of variables that define the possible situations.

Bayesian networks and other graphical models are now commonplace in artificial intelligence. The space of possible situations are described by a finite set of variables, $\{\mathbf{x}_1, .., \mathbf{x}_n\}$. Each variable has a set of possible values, $\Omega_{\mathbf{x}_i}$, and thus the space of possible situations is $\Omega = \Omega_{\mathbf{x}_1} \times .. \times \Omega_{\mathbf{x}_n}$. If Ω is finite, then $\mathcal{F} = 2^{\Omega}$ is the space of possible *events* (an event is a set of situations). In the above example, there are four binary variables leading to 16 possible situations. An assignment of a value to every variable is called a *joint configuration*, or just *configuration*, a term I will use repeatedly throughout this thesis. The space of possible configurations is the cross product of the space of values each variable can take on. Events such as observations and the propositions of interest are typically expressed as instantiations of variables in the model.

A graphical probabilistic model consists of a graph, $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, whose nodes, $\mathcal{V} = \{\mathbf{x}_1, .., \mathbf{x}_n\}$, are the variables of a model and whose edges, $\mathcal{E} \subset \mathcal{V} \times \mathcal{V}$, represent dependencies. Also included in the model are collections of local probability assignments, whose form depends on the type of graphical model.

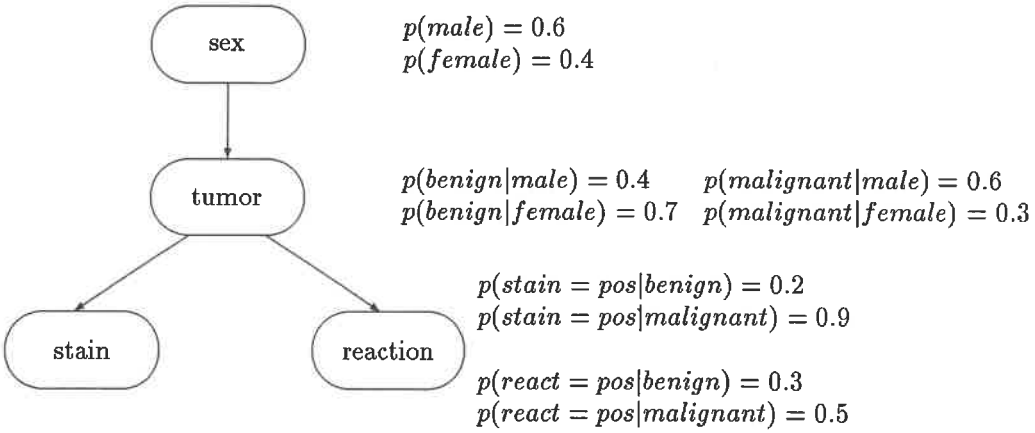


Figure 1.3: A Bayesian Network.

1.4.1 Bayesian Networks

The most popular graphical probabilistic model is the Bayesian network, in which the graph is a directed acyclic graph and the local probability assignments consist of a probability table at each node conditioned on the node's parents. Let $\mathbf{pa}(\mathbf{x}_i)$ be the parents of a node $\mathbf{x}_i$ in the graph, $\mathbf{pa}(\mathbf{x}_i) \subset \mathcal{V}$. The probability table at each node is denoted $p(\mathbf{x}_i|\mathbf{pa}(\mathbf{x}_i))$. The example from Figure 1.2 can be expressed as the Bayesian network shown in Figure 1.3.

The probability of every joint configuration is encoded in a Bayesian network as the product of all conditional probabilities corresponding to the configuration, i.e.,

$$p(\mathbf{x}_1, .., \mathbf{x}_n) = \prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{pa}(\mathbf{x}_i)) \tag{1.1}$$

For nodes with no parents, $p(\mathbf{x}_i)$ is used in this product. For example, in Figure 1.3,

$$\begin{aligned} p(male, benign, stain\text{-}pos, react\text{-}pos) &= p(male)p(benign|male)p(stain\text{-}pos|benign)p(react\text{-}pos|benign) \\ &= (0.6)(0.4)(0.2)(0.3) = 0.0144 \end{aligned}$$

Thus, the Bayesian network implicitly encodes the joint probability of every possible configuration. With the standard procedure for Bayesian updating, this completely defines the semantics of a Bayesian network — it is a distribution over a space of joint configuration specified as a product of probability tables.

For any event $A \in \mathcal{F}$, the probability of A (assuming Ω to be finite) is simply the sum of all configurations consistent with A, i.e.,

$$p(A) = \sum_{x \in A} p(x)$$

Note that x here is a joint configuration. The statement "$\mathbf{x}_3 = v$" specifies the event $A = \Omega_1 \times \Omega_2 \times \{v\} \times \Omega_3 \times .. \times \Omega_n$. A *marginal distribution* is simply the distribution over events involving a single variable, and is therefore obtained by "summing out" all other variables in a model. It is customary to write $p(\mathbf{x}_1)$ to denote the marginal of p to all events on $\mathcal{F}_{\mathbf{x}_1}$. Similarly, marginals may involve two or more variables, and one writes simply $p(\mathbf{x}_1, \mathbf{x}_2)$, etc., for the distribution marginalized to events involving only $\mathbf{x}_1$ and $\mathbf{x}_2$.

If A and B are two events, the conditional probability of A given B, defined when $p(B) > 0$, is

$$p(A|B) = \frac{p(A \cap B)}{p(B)} \tag{1.2}$$

This is the probability obtained by zeroing out the probability of all events inconsistent with B while maintaining relative probabilities of events consistent with B (of course, ensuring that $p(\Omega) = 1$). In a

Bayesian network, if a conditional probability appearing in a node's table is defined, it is easy to see that the value in the table always agrees with the conditional probability obtained from (1.2). This is a direct consequence of the acyclicity of the graph.

Probabilistic conditional independence plays a very important role in the study and use of graphical probabilistic models. A variable $\mathbf{x}_1$ is conditionally independent of a variable $\mathbf{x}_2$ given an event $A \in \mathcal{F}$ when $p(x_1|x_2, A) = p(x_1|A)$ for all $x_1 \in \mathcal{F}_1$ and $x_2 \in \Omega_2$. Equivalently, $p(x_1, x_2|A) = p(x_1|A)p(x_2|A)$. It is said that $\mathbf{x}_1$ is simply *independent* of $\mathbf{x}_2$ when this holds for the event $A = \Omega$. Note that $p(\cdot|\Omega) = p(\cdot)$.

Independence relationships do *not* define the Bayesian network, but they do fall out as a result of the definition. These are often referred to as *Markov properties*. For example, the *local Markov property* states that a variable is independent of all nondescendants given an assignment to its parents. The *(directed) global Markov property*[1] [Lauritzen *et al.*, 1990] (equivalently, d-separation [Pearl, 1988]) describes the more general set of conditional independence relationships shared by all distributions consistent with a given graphical structure. See [Lauritzen *et al.*, 1990, Frydenberg, 1990].

The directions of arrows in a Bayesian network do *not* imply causality; however, there is much interest in Bayesian networks with arrows that do correspond to causality. It is often useful to interpret each variable as being a stochastic function of its parents. Such an interpretation often allows one to write down directed dependencies directly for an application. These topics are discussed in more depth in Chapter 3.

There are many good introductions to Bayesian networks including [Heckerman and Horvitz, 1988, Pearl, 1988, Charniak, 1991, Henrion *et al.*, 1991, Jensen, 1993].

1.4.2 Undirected Networks (Markov Fields)

A second common graphical model is known as a *Markov Field*, *Markov Network*, *Gibbs System*, etc., and utilizes an undirected dependency graph. In an undirected graph, the edge $(\mathbf{x}_i, \mathbf{x}_j)$ is in $\mathcal{E}$ exactly when $(\mathbf{x}_j, \mathbf{x}_i) \in \mathcal{E}$, and self-loops are not allowed (i.e., $(\mathbf{x}_i, \mathbf{x}_i) \notin \mathcal{E}$). The basic properties underlying undirected graphical models are in many ways at the heart of nearly all graphical probabilistic models.

As with the Bayesian network, the global probability distribution is an implicit function of many local tables associated with the network. Unlike the Bayesian network, these tables are not comprised of conditional probabilities, and indeed, the entries might not even be probabilities, at least not initially. Each table is associated with a totally-connected[2] subset of variables and specifies a potential value for each combination of joint assignments the variables can take on. A maximal totally-connected subset of variables is called a *clique*, and limiting tables only to cliques is equally general. The joint probability of a full configuration is the (normalized) product of the corresponding entry from each table. A normalization may be necessary since the products might not sum to 1.

In general, it is difficult to interpret the numbers in individual tables in an undirected graph. Only the product of the tables is interpreted as a distribution[3]. Also, some care must be taken to ensure that at least one joint assignment is assigned a nonzero probability or one ends up with consistency problems. Because of these considerations, it is usually inconvenient to specify knowledge directly in terms of an undirected graph. However, because of the great generality — i.e., the joint probability must simply be expressed in product form — the undirected graph is generally the most useful of the graphical probabilistic models when knowledge is expressed in some application-specific form, but where graphical models are to be exploited later for computation, interpretation, etc. This is more or less the case in this thesis.

---

[1] The directed global Markov property is slightly more complicated to state, and so it not explained in the text at this point. It states that two sets of variables in a directed graph are conditionally independent given a third set of variables when the third set separates the two sets in the (undirected) *moralized ancestrial subgraph*. The moralized ancestrial subgraph is obtained by deleting all nodes from the original graph that have no descendants in one of the three sets (the ancestrial graph), connecting any two nodes with a common child node (moralizing = "marrying parents"), and dropping the direction of edges.

[2] The Hammersley-Clifford theorem shows that assignments to totally-connected subsets of variables is the most general representation that results in a consistent probability distribution when the distribution is everywhere positive [Besag, 1974]. Although the same is not quite true when there are zeros ([Moussouris, 1974]), assignments to complete subsets are generally used in general ([Pearl, 1988, Lauritzen and Spiegelhalter, 1988]).

[3] Unless the tables agree on all possible marginals, in which case the tables are said to be *consistent* and the tables then correspond to the marginals for those variables [Dawid and Lauritzen, 1993].

Conditional independence relations that hold for all distributions expressible as a given graph can be easily read off an undirected graph. Two sets of variables are conditionally independent given a third when all paths from one set to the other passes through the given (third) set of variables in the graph. This is called the *global Markov property.* For example, any variable is independent of all other variables given its immediate neighbors. This is called the *local Markov property.*

Equation 1.1, which defines the joint distribution of a Bayesian network, is in a product form; therefore, knowledge expressed as a Bayesian network can quickly be mapped onto an undirected graphical model. The undirected graph is obtained by connecting ("marrying") all nodes with a common child in the directed graph and then removing the directions of all arrows. The corresponding undirected graph has one table per node, with each table involving only the node and the nodes that were its parents in the original directed graph; however, any tables appearing on a common clique can be combined (multiplied together) and assigned to the clique. Such a graph is called the *moralized graph* of the Bayesian network. Since the distribution is unchanged by moralization, no new independence assertions are introduced from this operation. Algorithms developed for undirected graphs can thus be applied to solve Bayesian networks by applying them to the moralized graph ([Lauritzen and Spiegelhalter, 1988]).

1.4.3 Perspective

Much existing work on graphical probabilistic models treats the graphical model as the primary knowledge representation language. It is a language a knowledge engineer might use directly to encode domain knowledge for an application. Used in this way, it is not surprising that many find the Bayesian network a much more attractive formalism than the undirected network since the probabilities are easy to interpret, etc.

Although graphical probabilistic models often function well as a primary knowledge representation language, this use barely scratches the surface of their potential applicability. In most of the applications I have personally encountered, a standard Bayesian network has not matched the form of knowledge that is natural for the problem, or it was deficient in other respects. For example, in the time-series segmentation task discussed later it is natural allow for an arbitrary horizon, which could not be represented as a Bayesian network unless one had an infinite number of nodes. Furthermore, the time-homogenous structure of the model would not be made explicit in a Bayesian network, therefore making other modeling approaches more appropriate. However, graphical structure is ubiquitous, often arising when application-specific knowledge (in other forms) is brought to bear on a particular task. The techniques already in existence for graphical models often provides immediate solutions to these types of problems that arise in practice. One encodes knowledge in a domain specific formalism, and then by recognizing graphical structure, utilizes these tools obtain efficient algorithms. In this way, graphical models and the supporting theory can provide computation tools even if they do not provide the natural representation for expressing domain knowledge. The time-series application in this thesis certain demonstrates this point.

In contrast to much existing literature, it is this latter use I greatly emphasize in this thesis, particularly in Chapter 3. For me, graphical models are not as much primary knowledge representation formalisms as they are extremely useful and widely applicable computational tools.

1.5 Exact Solution Techniques

As mentioned in Section 1.3, much of the current interest in graphical probabilistic models is due to the fact that algorithms can take advantage of the graphical structure for computational efficiency.

1.5.1 What can be computed?

A common probabilistic inference task is to compute the marginal probability on a variable given some value assignment to a subset of other variables.

Another common task is to find the maximum a posteriori (MAP) configuration given assignments to some of the variables.

These two tasks cover a huge space of problems that arise in probabilistic inference applications, including those that appear in this thesis.

There are other possible inference tasks that are not touched by this thesis. For example, if one has a causal graph in the form of a Bayesian network (remember, not all Bayesian networks are causal), one might ask what would happen to the probability at $\mathbf{y}$ if an exogenous force suddenly made $\mathbf{x}$ true ([Pearl, 1994, Pearl, 1995b, Pearl, 1995a, Pearl, 1996]). One should realize that this is not the same task as computing the conditional probability $p(\mathbf{y}|\mathbf{x})$, which describes what to expect of $\mathbf{y}$ when $\mathbf{x}$ *happens* to be true, which is what the marginal posterior distribution describes. Another task is to find an optimal course of action relative to some utility model. This is the realm of *influence diagrams* ([Howard and Matheson, 1984a, Shachter, 1986]), and is not discussed here. However, many of the same ideas for solving networks readily transfer to influence diagrams ([Shenoy, 1992, Jensen *et al.*, 1994]).

1.5.2 Existing Algorithms

In [Lauritzen and Spiegelhalter, 1988], an algorithm for exact computation on graphical models is introduced. The algorithm computes probabilities by propagating potentials on a secondary undirected graphical structure, which has subsequently become known as the *junction tree*. The junction tree is compiled from the original graph through a process of moralization and triangulation. These graph theoretic concepts are explained in Section 3.1. [Jensen *et al.*, 1990a] extends the algorithm to gracefully incorporate multiple observations at once, and the result is often referred to as *Jensen's algorithm* or the `Hugin` *algorithm*. Furthermore, [Dawid, 1992] shows that the same basic algorithm can be applied to find the MAP configuration. Currently, this propagation algorithm is state-of-the-art and the most popular algorithm in use. The propagation approach is detailed in Chapter 3.

A second method for solving graphical networks is the *node-removal algorithm* of [Shachter, 1986, Shachter, 1990]. This algorithm performs a series of arc reversals and node removals until a single node remains, at which point the marginal probability of interest can be read off.

A third exact solution method is the *cutset-conditioning* algorithm ([Pearl, 1986a], [Pearl, 1988, Pages 204–210], [Zhang and Poole, 1992], [Becker and Geiger, 1994], [Darwiche, 1995], [Díez, to appear], [Suermondt and Cooper, 1990], [Peot and Shachter, 1991]). This algorithm locates a subset of variables that breaks all undirected loops. When these variables are clamped to a given value, the subgraph of the remaining nodes is a poly-tree, so that the efficient poly-tree algorithm of [Pearl, 1986b, Pearl, 1988] can be applied to solve the subproblem. This procedure must be repeated for every possible assignment of the cutset variables, and the results are then averaged appropriately.

A fourth method solves marginals symbolically by explicitly manipulating the symbolic summations that define marginals and simplifying the expressions by utilizing conditional independence relationships ([Chang and Fung, 1991, Shachter *et al.*, 1990, D'Ambrosio, 1990, D'Ambrosio, 1994, Castillo *et al.*, 1995a]).

There are, of course, minor variations to all these approaches.

What is somewhat surprising is that all of these exact approaches are essentially equivalent. Each algorithm can be seen as peeling off one variable at a time. This is fairly obvious in the case of node removal algorithms, and not too hard to see for the symbolic algorithms (see also [Lauritzen and Spiegelhalter, 1988, Comments by W. S. Kendall]). The clique-tree approach utilizes a triangulation as the basis for propagation. A well-known result ([Golumbic, 1980]) is that a graph is triangulated exactly when there is a node ordering in which the nodes can be removed one at a time, such that whenever a node is removed, the remaining neighbors of that node are totally connected. Thus, a triangulation is equivalent to a node-peeling order, so the equivalence is reasonably clear. The equivalence of cutset-conditioning is nontrivial to see, but is described nicely in [Shachter *et al.*, 1991, Shachter *et al.*, 1994]. Essentially, the use of a cutset is equivalent to a triangulation obtained by connecting each cutset variable to every other variable in the network[4]. Since only some of the possible triangulations can be obtained in this fashion, and since this can result in unnecessary edges being added, the cutset-conditioning methods are always inferior to the clique-marginal propagation approach.

In short, although many algorithms have been devised, they all seem to be equivalent to, or at least special cases of, the clique-tree propagation algorithm(s). These propagation algorithms are very general, very useful, and relatively easy to understand. They are covered in detail in Chapter 3.

[4] Recent terminology calls this *global (cutset) conditioning*. Local cutset conditioning methods ([Díez, to appear]) connect cutset variables to subsets of nodes, but the same type of transformation to clique trees still applies.

In addition to exact solution techniques, there are many approximation algorithms for tasks involving graphical networks. Some of these are reviewed in Section 1.8.

1.6 Real-Valued Variables

Parametric Representations

When each variable in a network is discrete with only a finite number of possible values, it is possible to use a *nonparametric* representation for the probabilities or potentials. In other words, we can store a separate probability for each possible value of the variables. When a variable can take on an infinite number of possible values, an explicit enumeration of probabilities is no longer possible, and it becomes necessary to turn to parametric representations.

A *parametric* representation defines a family of distributions, where the distributions within the family are indexed by a small number of parameters. For example, the Bernoulli distribution on the positive integers, $p(x) = \binom{n}{x} q^x (1-q)^{n-x}$, is defined by the two parameters $n \in \mathbb{Z}^+$, $q \in [0,1]$. Instead of having to store a probability for every possible value of x, one needs to only store n and q to recall the entire distribution.

Densities

On uncountable spaces, the probability density function (rather than the probability) is generally parameterized. The reader not familiar with general probability theory (i.e., on uncountable sample spaces) may find a text on the topic such as [Billingsley, 1986] useful; however, the reader should find most of this thesis understandable with only a basic understanding of concepts such as conditional distributions (densities) and integration (i.e., of densities to obtain distributions). An example of a parametric family on $\Re^n$ are the multi-variate Gaussian distributions, parameterized by a $1 \times n$ vector μ and the $n \times n$ covariance matrix Σ, with density given by (here x is a $1 \times n$ vector):

$$p(x) = \frac{1}{\sqrt{2\pi|\Sigma|}} e^{-\frac{1}{2}(x-\mu)\Sigma^{-1}(x-\mu)^T}$$

On uncountable (e.g., real-valued) sample spaces, the probability density is used where the probability is used with countable or finite sample spaces. For example, for a distribution represented on an undirected graphical network, the *density* must be expressed in a product form, and the result of propagation is a marginal density (in parametric form, of course).

Conjugacy

Parametric representations introduce an extra complication for graphical network representations: A conjugacy requirement. A family of distributions is conjugate if when any distribution from the family is updated after evidence is observed, the resulting posterior distribution is a member of the same family. Exact solution methods for graphical networks require an even stronger form of conjugacy: if information is propagated between nodes in the graph, an updated local density function after each individual propagation step must still belong to the same local parametric family (Section 3.2.5).

Gaussian distributions possess these conjugacy properties. When a network of real-value nodes is used to encode an n-dimensional Gaussian distribution (n is the number of variables), one can view the network structure as constraining certain covariances to be zero. Propagation methods for Gaussian networks are given in [Shachter and Kenley, 1989, Andersen *et al.*, 1993, Chang and Fung, 1991, Lauritzen, 1992, Geiger and Heckerman, 1994b].

A *CG-distribution* is a hybrid of a Gaussian distribution and a discrete distribution involving both real-valued and discrete variables. For any assignment to all discrete values, the distribution on the remaining continuous variables is Gaussian. The CG-distribution is also conjugate and propagation algorithms on graphical networks have been developed [Lauritzen and Wermuth, 1989, Olesen, 1993].

The network conjugacy requirement can be quite limiting. Many applications require distributions that do not belong to a simple conjugate parametric family. These may still have a product form and therefore

be representable using a graphical network, but exact methods cannot be applied to solve such cases. The time-series segmentation problem studied in this thesis is such a case.

Various approximation algorithms can be applied to solve certain tasks even when a conjugacy requirement does not hold. This thesis develops such an algorithm, which operates by discretizing real-valued variables so that nonparametric techniques can be applied to the discrete values.

1.7 Iterative Dynamic Discretization

A natural way to handle real-valued variables with general (i.e., not necessary conjugate) distributions is to replace continuous variables with a finite number of possible values. I refer to this as *discretization.*

If a continuous variable has a compact (i.e., bounded) state space, one can simply pick the discrete values to use by spacing the points evenly over the range of the variable. However, often this will result in sparse coverage in critical areas and excess coverage of unimportant values. Also, uniform spacing of values cannot (with a finite number of values) be applied to noncompact sample spaces. These considerations suggest using a *nonuniform* discretization, where points are chosen with greater density in areas that are likely to be more critical for the task at hand.

The challenge in choosing a discretization is in picking values that are likely to be important for the task at hand. The appropriate values depend on the problem instance and vary with different sets of observations. Thus, a discretization should be dynamically chosen and tailored to the problem instance. This, however, creates a certain chicken-and-egg problem: we must determine what values are important without having already solved the task. If we knew the answer, we might have a reasonable basis for knowing what values are critical, but if we knew the answer, there would be no reason to discretize in the first place.

This thesis explores an iterative approach to the discretization challenge. A discretization is chosen initially, and the problem solved using that discretization. The solution (partially) indicates where the important values for the task are. Using this information, a new discretization is picked, concentrating points more densely in the areas where the previous iteration suggests may be the most critical, and therefore obtaining a (possibly) improved discretization. Thus, the discretization is refined iteratively and dynamically.

The iterative dynamic discretization algorithm is the topic of Chapter 4. Also, relationships between iterative dynamic discretization and existing approximation techniques are discussed in Section 1.8.

1.8 Approximate Solution Techniques

Exact solution techniques (Section 1.5) are often infeasible for certain problems. Even for a finite domain, exact solutions to most interesting classes of graphical probablistic inference tasks are known to be NP-hard. In some cases, approximation techniques may provide good solutions in reasonable time when exact solutions are infeasible. For example, [Dagum and Chavez, 1993] identify a wide class of graphical models that can be approximated to any degree of accuracy in polynomial time[5], but are NP-hard to solve exactly. Exact techniques are very sensitive to network dependency structure, while the same sensitivity is not necessarily shared by approximation algorithms[6]. One should keep in mind, however, that approximating answers to graphical network tasks is also NP-hard in most general cases [Dagum and Luby, 1993].

Conjugacy considerations are another reason approximation techniques are important. While conjugacy is necessary for exact solutions, it is not necessarily required for approximation algorithms. Thus, various approximate methods may provide algorithms when nonconjugacy otherwise prevents exact solutions.

Iterative dynamic discretization is an approximation technique. It shares many similarities with existing techniques. It can be viewed as a generalization of Gibbs Sampling[7] (a standard stochastic simulation method). Discretization is clearly a form of model simplification. Iterative dynamic discretization is an

[5] They can be solved in time polynomial to the size of the network description and $1/\epsilon$, where ϵ is the desired accuracy.

[6] Approximation methods may be quite sensitive to other factors, such as the magnitude of probabilities within a model.

[7] The standard Gibbs sampling algorithm (see Page 12 and Section 3.4) is a special case of iterative dynamic discretization, obtained when the discretization size is set to one, i.e., only one possible value is picked for each variable on each iteration.

instance of (approximate) Knowledge-Based Model Construction (KBMC), and the study in this thesis provides some insights into that endeavor. In fact, the KBMC perspective was among my earliest perspectives and one of the main motivations for this line of research. Finally, it can also be viewed as a combination of Markov chain Monte Carlo techniques (stochastic simulation) with exact propagation, harnessing the strengths of each. This section reviews several existing approximation techniques for graphical probabilistic networks along with perspectives relating these techniques to iterative dynamic discretization.

1.8.1 Stochastic Simulation

Many probabilistic inference tasks can be approximated using various forms of stochastic simulation, in which configurations are drawn according to the probabilities in the model, and the frequencies within the sample used as an approximation for the inference task. Two issues arise: how to generate the sample, and how to use a generated sample to approximate the inference task of interest.

Generating Samples

An algorithm that generates a set of *independently* drawn configurations and bases an approximation on this set is called a *Monte Carlo* algorithm. Generating samples independently is usually very difficult to do efficiently; the more efficient algorithms relax the independence of samples to obtain efficient ways to draw samples. These are referred to as Markov chain Monte Carlo algorithms. Although consecutive samples are dependent, the asymptotic sampling distribution matches the distribution of interest.

A Bayesian network (or any other graphical network) represents a joint probability distribution over the space of configurations, $p(x)$. It is straightforward to generate configurations randomly and independently for a Bayesian network according to p. We simply pick an assignment first for variables with no parents according to the marginal probabilities at those nodes, then for each variable with no uninstantiated parents we choose a value conditioned on the values of its parents. In a Bayesian network, this conditional probability is an entry in the table associated with that node. The process is repeated until all nodes are instantiated. By construction, the resulting configuration, x, has probability $p(x)$.

Most inference tasks of interest, however, are not queries about $p(\cdot)$, but instead are queries concerning $p(\cdot|e)$, where e (the observed evidence) is an assignment to some subset of variables. A Monte Carlo algorithm must therefore draw samples from $p(\cdot|e)$, not from $p(\cdot)$.

Logic sampling ([Henrion, 1988]) is one technique for drawing samples from $p(\cdot|e)$. Tentative samples are drawn from $p(\cdot)$ until a sample is found exactly matching e. Logic sampling requires on average $1/p(e)$ trials per sample generated. Because $p(e)$ is often very small, logic sampling is usually extremely inefficient, in fact, usually infeasible. Also, logic sampling cannot be applied when observations include continuous variables, since there is generally zero probability that a sample drawn from $p(\cdot)$ will match e exactly.

Likelihood weighing or *importance sampling* ([Fung and Chang, 1989, Shachter and Peot, 1989]) also draws samples from $p(\cdot)$, but utilizes every sample by weighting each sample according to how closely it matches e. Typically, observed nodes are instantiated first, then the procedure for drawing a sample from $p(\cdot)$ described above is followed. In practice, a few samples very close to e tend to dominate all others, so importance sampling is usually only slightly better than logic sampling.

Logic and importance sampling are representative of Monte Carlo techniques that have been attempted with graphical networks, and both are quite inefficient when $p(e)$ is small, as is typically the case. In general, generating *independent* samples is difficult to do efficiently.

Markov chain Monte Carlo (MCMC) methods provide ways to generate samples efficiently by sacrificing the independence between adjacent samples. Although consecutive samples are dependent, asymptotically the samples are drawn from the distribution of interest.

The simplest and most popular[8] MCMC algorithm for graphical networks is Gibbs sampling [Geman and Geman, 1984, Chavez and Cooper, 1990, Gelfand and Smith, 1990, Hrycej, 1990, York, 1992, Neal, 1993, Tierney, 1994, Gelfand, 1995]. Consecutive samples generated by Gibbs sampling differ in (at most) one variable (it is possible for the same configuration to be repeated twice in a row). To generate a new sample, a

[8] Gibbs sampling is certainly the most popular of the MCMC methods in the context of graphical probabilistic models. The Metropolis-Hastings algorithm is commonly used for numerical integration and may be in wider use overall.

single variable $\mathbf{x}_i$ is selected, and its new value drawn from $p(\mathbf{x}_i|\mathbf{x}_{j\neq i})$, where $\mathbf{x}_{j\neq i}$ denotes all other variables in the network. Because of the conditional independence encoded by a graphical network, the sampling operation simplifies to drawing a sample from $p(\mathbf{x}_i|\mathbf{x}_{mb(\mathbf{x}_i)})$, where $mb(\mathbf{x}_i)$ is the *Markov boundary* of $\mathbf{x}_i$ (the variables immediately adjacent to $\mathbf{x}_i$ in the moral graph). A different variable is changed for each new sample, and under some basic assumptions about p and the sampling procedure (see Section 3.4), the configurations are drawn asymptotically from $p(\cdot|e)$ no matter what configuration the process is started from. Thus, if enough samples are drawn, they can be used in the same fashion that Monte Carlo generated samples are used. Gibbs sampling is discussed in more detail in Section 3.4.

Gibbs sampling has a number of advantages. It is not sensitive to network structure, and thus can be readily applied to networks with high connectivity when exact methods would be infeasible. A very big advantage is that Gibbs sampling imposes no conjugacy requirements on p, and therefore can be applied to general (i.e., nonconjugate) models with continuous variables. On the down side, Gibbs sampling is highly sensitive to the magnitude of probabilities in a model, often converging very slowly when probabilities within the model are very close to zero. Also, it requires that the operation of drawing a value for $\mathbf{x}_i$ from $p(\mathbf{x}_i|\mathbf{x}_{j\neq i})$ can be implemented efficiently. Even though in a graphical network it only involves a local collection of variables, this operation can often be very difficult and can limit the applicability of Gibbs sampling.

Iterative dynamic discretization can be viewed as a variation of the Gibbs sampling algorithm; however, instead of applying Gibbs sampling on the space of possible configurations, iterative dynamic discretization applies Gibbs sampling to the space of possible discretizations. Since a discretization specifies several possible values for each variable, a single discretization corresponds to a set of several different configurations, so this is like applying Gibbs sampling at a meta-level. When the discretization size is set to one so that a discretization specifies a single value for each variable, the pure Gibbs sampling algorithm falls out as a special case.

Gibbs sampling is the most popular, and probably the most appropriate, of the MCMC techniques for graphical probabilistic models. Other MCMC techniques exist as well [Neal, 1993, Tierney, 1994, MCMC, 1996], but most have less of a connection to graphical probabilistic networks and to the work in this thesis. One MCMC algorithm, *simulated annealing* [Kirkpatrick *et al.*, 1983], is generally more appropriate than Gibbs sampling for optimization problems such as finding MAP configurations. Simulated annealing can be described as being essentially the same procedure as Gibbs sampling, but applied to the distribution $\alpha(T)p(\cdot|e)^{1/T}$, where $\alpha(T)$ is a normalizing constant and T is a temperature. Simulated annealing gradually reduces T while sampling is occurring. As T gets closer to zero, the highest probability configurations become more distinct so that the sampling asymptotically spends more time near the optimum. The higher initial temperatures result in higher mobility, thus promoting quicker access to the highest probability regions.

Approximating Answers

Monte Carlo and Markov chain Monte Carlo methods can be used to approximate a variety of probabilistic inferences. All utilize a sample of configurations drawn according the techniques discussed above, but the way in which these samples are utilized depends on the task being solved. The manner in which samples are used is the same regardless of whether the samples originate from Monte Carlo simulation or MCMC simulation[9].

Expectations are naturally estimated using sampled frequencies. If $v(x)$ is a function of the variables in a model, the expectation of v given e, $E_p[v|e]$, is approximated using the samples $x^1, .., x^k$ drawn from $p(\cdot|e)$ as

$$E_p[v|e] = \frac{1}{k}\sum_{j=1}^{k} v(x^j)$$

Another simple query one might estimate is a binary query such as "what is the probability that $x_i \in A$," for some set of values A. For this, one simply counts the fraction of times this event occurs in the sample. This is, of course, a special case of expectation.

[9] Some variations of MCMC utilize only some fraction of the samples, usually spaced apart in time to reduce their dependence. This is a minor difference that occurs in some versions of MCMC that would not occur in a pure Monte Carlo algorithm.

Estimating marginal distributions can introduce a few additional considerations. If the variable of interest takes on a finite number of values, one can simply return the frequency of those values in the sample. In many cases, it is better to use the sample to update a Dirichlet prior for the marginal distribution. This can be used to avoid troublesome zeros that would otherwise be assigned to values that occur with zero frequency in the finite sample.

Estimating a marginal over a continuous variable (or even a variable with a countably infinite sample space) introduces additional complications. In this case, the marginal must be expressed in a parametric form, and the parameters estimated from the sample frequencies. This can be accomplished by fitting the parametric form to the $\mathbf{x}_i$ component of the sample. However, such an estimate can be improved using a technique known as *Rao-Blackwellization* (based on the well-known Rao-Blackwell theorem, e.g., [Bickel and Doksum, 1977, Page 121]). The distribution on $\mathbf{x}_i$ given its neighbors, $\mathbf{y}$, largely constrains the shape that the marginal distribution on $\mathbf{x}_i$ can have. Instead of estimating $p(x_i)$ directly, one estimates $p(y)$, (denote the estimate of $p(y)$ by $\tilde{p}(y)$) and then obtains the estimate over x_i as

$$\tilde{p}(x_i) = \int p(x_i|y)\tilde{p}(dy)$$

As argument in [Gelfand and Smith, 1990, Page 402] using the Rao-Blackwell theorem shows that this estimate always has a smaller than or equal least squared error relative to the true marginal distribution. See also [Casella and Robert, 1996, McKeague and Wefelmeyer, 1995].

Another query that can be approximated using Monte Carlo or MCMC samples is that of finding a MAP configuration. One simply returns the highest probability configuration from those in the sample. For this optimization task, it is often appropriate to distort the distribution to accentuate higher probability configurations; however, the degree of accentuation must be traded off against the fact that accentuations often reduce the efficiency of sample generation and lower mobility (and thus convergence rates) for MCMC methods.

As discussed above, iterative dynamic discretization can be viewed as a variation of Gibbs sampling, applied at a meta-level. In this thesis, it is applied to the problem of locating a MAP configuration. However, no attempt is made to accentuate the underlying distribution (as is done with simulated annealing, for example). The fact that a single discretization contains a whole collection of configurations makes this a reasonable method for searching the space, even though it might be argued that Gibbs sampling is not the best option for optimization. The implementation also computes marginal probabilities. The marginal distributions are for continuous variables, but marginals computed are only for a finite number of possible values (those in the discretization), and are only relative to the discretization of other variables in the network. They can therefore be viewed as a discrete approximation to the continuous marginal distribution.

1.8.2 Model Simplification

Several existing approximation techniques can be described as approximating the original model by simplifying the network in some way. Exact techniques are then usually applied to the simplified network to obtain an approximate answer for the original problem.

Some simplifications involve altering the graphical structure of the network in some way, such as by removing edges or key independence assumptions [Kjaerulff, 1993, Kjaerulff, 1994]. The mean-field theory approximation approach of [Saul and Jordan, 1995, Saul *et al.*, 1996] can also be viewed as a structural alteration, where the mean-field theory is harnessed for determining how the probabilities in the simplified model are to be set. In this thesis I do not consider structural alterations.

A couple of papers ([Poole, 1993b, Draper and Hanks, 1994]) obtain anytime approximation algorithms for Bayesian networks by restricting attention to and solving a subnetwork, and then gradually increasing the size of the subnetwork with time. Each step of such an algorithm is also a version of structural abstraction.

Another simplification is to artificially set certain probabilities to zero [Jensen and Andersen, 1990, Castillo *et al.*, 1995b]. These techniques obtain a simplified model by setting small probabilities to zero. The inter-variable structure of the network remains the same, but fewer values must be considered within each clique, so by utilizing sparse representations of potential arrays, efficiency of exact solutions can be increased. This is clearly closely related to the use of a discretization, which essentially amounts to zeroing

out all but a finite number of possible values that a variable can take on so that exact methods can be efficiently executed.

Yet another simplification method is *sample space abstraction* [Wellman and Liu, 1994]. Here the space of possible values an individual variable can take on is partitioned into abstract values, thus resulting in an abstract model with a smaller sample space. Applied to continuous variables, the sample space can be partitioned into a finite number of values. Iterative dynamic discretization is obviously quite similar to this, except an individual point is used rather than a conglomeration of values. One can view the point as representing a region of neighboring points to make the correspondence closer, but the real distinction comes in how initial probabilities are assigned to the classes of values. [Wellman and Liu, 1994] grow their sample space abstractions with time, i.e., increasing the number of values, obtaining an anytime algorithm with each successive pass requiring more work. In this thesis, the discretization size is determined by the user and fixed. Sample space abstraction is, at least on a conceptual level, preferable in some ways to the sampling approach of discretization; however, it also causes a number of complications that are very difficult to overcome. Other papers that are relevant to sample space abstraction approaches include [Laskey, 1991, Laskey, 1993, Laskey and Lehner, 1994, Kim and Valtorta, 1995].

Knowledge-Based Model Construction can also be considered a form of model simplification, although it has motivations beyond simple model simplification, and so I review it separately in Section 1.8.3.

1.8.3 Knowledge-Based Model Construction

An ideal intelligent agent would possess a huge broad-based store of background and domain knowledge. It would then have the ability to bring this knowledge to bear on any particular problem it encounters within its domain. Because uncertainty permeates all aspects of the real world, the background knowledge base itself would consist of probabilistic relations. The ideal agent should bring all relevant background knowledge to bear on any given problem, and if the (probabilistic) background knowledge base is rich enough, this could mean that the ramifications of almost any bit of knowledge could be relevant.

This model of an ideal agent views even the most trivial query as a huge inference task, one in which all ramifications of the agent's huge knowledge base must be taken into account.

Humans are more-or-less in this situation. The store of knowledge in a person's brain is immense by any standard; however, when a person solves a problem that requires him to reason about uncertainty, he certainly does not consider the ramifications of all his knowledge. Instead, one typically recalls a small set of relevant facts and then considers the ramifications of those facts alone.

In a similar fashion, it is conceivable that a computerized intelligent agent might, when confronted with a specific inference task, use its broad background knowledge base to construct a small probabilistic model for solving that task. The small model is not just any small model — it is one that is dynamically tailored to the specifics of the situation and information received so far. This means that it should be built from the most relevant facts and consider the most relevant possibilities. Once a small model is obtained, the problem instance can be efficiently solved. This general idea has been given the name *Knowledge-Based Model Construction* (KBMC) [Wellman *et al.*, 1992, Breese *et al.*, 1994], and is a new and emerging research area within the A.I. uncertain reasoning community. Figure 1.4, taken from [Breese *et al.*, 1994] and [Wellman *et al.*, 1992], illustrates this process.

There are two obvious advantages to KBMC. First, it offers an approximation technique. By considering only a subset of the entire knowledge base, the answer computed is an approximation to the more faithful answer that takes all available knowledge into account. And second, it allows more general representations of knowledge in the background knowledge base than those that can be handled by available inference algorithms. For example, existing graphical probabilistic modeling formalisms are propositional in nature, and typically allow only a finite number of possible situations. In contrast, a general knowledge base may contain quantifiers and arbitrary knowledge about continuous quantities or a countably infinite number of situations. A KBMC approach builds a model without quantifiers or continuous variables by selecting only a finite number of possible (constant) bindings for quantifiers and a finite number of possible values for variables with an infinite number of possible values. The specific choices are made based on the specifics of the problem instance and information obtained so far about the situation, so that the finite-sized model is tailored to the problem at hand.

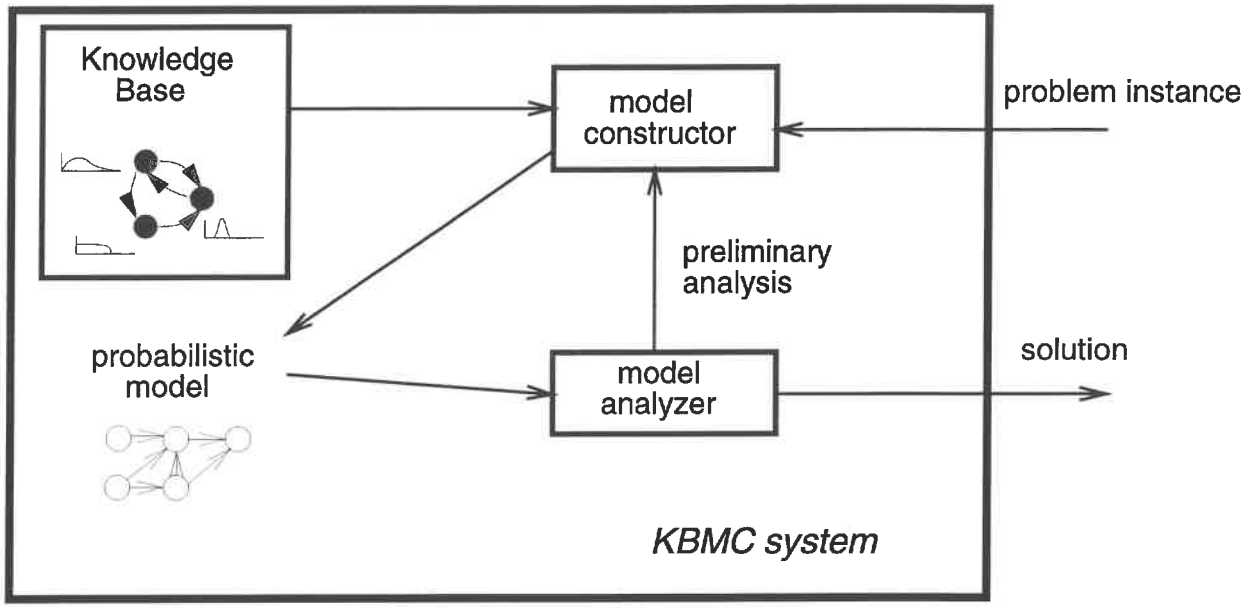


Figure 1.4: Knowledge-based Model Construction. Given a specific problem instance or query, the general domain knowledge is used to construct a probabilistic model which is then used to compute the answer. Information learned from solving the problem may be used to refine the constructed model for an improved answer.

Serious research on automating KBMC has only very recently emerged. In *The Foundations of Statistics*, [Savage, 1954] discussed how for any decision problem, one must construct a *small world* for that problem before formal techniques can be applied to analyze it. While the concepts are the same, it has not been until the 1990's that researchers have attempted to automate the process on a computer. Work along these lines include [Goldman and Charniak, 1990, Wellman, 1990, Breese, 1992, Wellman *et al.*, 1992, Laskey and Lehner, 1994, Xiang *et al.*, 1992, Goldman and Charniak, 1993, Poole, 1993a, Provan, 1993a, Provan, 1993b, Provan, 1994, Poh *et al.*, 1994, Saffiotti and Emkehrer, 1994, Druzdzel and Suermondt, 1994, Nicholson and Brady, 1994, Haddawy, 1994, Glesner and Koller, 1995, Haddawy *et al.*, 1995, Ngo and Haddawy, 1995b, Ngo *et al.*, 1995, Ngo and Haddawy, 1995a]. Related to this endeavor is the design of general quantified probabilistic logics for background knowledge representation formalisms (e.g., [Bacchus, 1993], [Bacchus *et al.*, 1994], [Bacchus and Grove, 1995]), and the fusing of multiple belief networks (e.g., [Shachter, 1991], [Matzkevich and Abramson, 1993]). Recommended reading for an initial introduction to this area are [Breese *et al.*, 1994], [Wellman *et al.*, 1992], [Breese, 1992], and [Haddawy, 1994].

The work cited in the previous paragraph predominantly considers the problem of constructing a propositional graphical probabilistic model from a more general knowledge base containing quantifiers. Almost universally throughout the work is a focus on constructing provably correct models — i.e., models where no relevant information is omitted (c.f., [Haddawy, 1994] and [Breese, 1992] where such correctness proofs are given as theorems). In some knowledge bases this is possible. For example, the knowledge base may contain a rule (this example inspired by the knowledge base of [Breese, 1992]):

$$\forall x, y. Pr(Calls(x, y) | Neighbor(x, y) \wedge Alarm(y, Ringing)) = 0.99$$

The rule says that if x and y are neighbors, and y's burglar alarm is sounding, then x will call y with probability 0.99. This rule holds for any possible binding of x and y. When a model is constructed, one or more rules (which becomes links in a Bayesian network) such as

$$P(Calls(Watson, Holmes) | Alarm(Holmes, Ringing)) = 0.99$$

are extracted and included in the model ($Neighbor(Watson, Holmes)$ is known). If the inference problem that is being solved is for Holmes to decide whether or not there has been a burglary at his house, it may

be possible to prove that the instantiation

$$P(Calls(Jack, Jill)|Alarm(Jill, Ringing)) = 0.99$$

is entirely irrelevant to the problem at hand; therefore, this latter rule can be omitted from the constructed model.

While the construction of provably correct models is a clear advance to the state-of-the-art, it skirts one of the most important issues of KBMC. In the general conception of KBMC, constructed models may lose information relative to the full knowledge base, and the effects of (and methods for coping with) this loss of information are of primary concern. When a broad knowledge base is sufficiently rich, nearly all facts may be somehow mutually relevant in some roundabout way. This is not to say that they are significant for any specific inference problem. For example, *Jack*'s call to *Jill* might increase the chances by some miniscule amount that *Watson* is unable to call *Holmes* due to overloaded telephone circuitry, and thus ultimately have a very small effect on the probability that *Holmes*'s house was burglarized given the available evidence. A technique that constructs a provably correct model would have to include the $Call(Jack, Jill)$ and related facts, despite their insignificant influence. Recall that the first reason listed above for considering KBMC is to obtain approximation methods. Provably correct models do not address this motivation for KBMC. Currently, little is known about the impact of information loss during KBMC or how it should be handled[10].

Understanding the loss of information during KBMC is not simply a matter of knowing the effect of information loss on the accuracy or fidelity of answers[11] — the understanding will likely have significant impact on the overall architecture and methods for KBMC. In particular, a system whose constructed models are provably correct has no reason to iterate on or re-evaluate the model construction step. However, the technology for analyzing deficiencies in constructed models and revising them appropriately may be crucial when information loss is possible.

Iterative Model Construction

As discussed above, KBMC may be employed as a method of approximation, and/or as a means for allowing a more expressive general knowledge base than specific inference algorithms can handle. In this thesis, it is utilized for both purposes.

There is a clear tradeoff between model construction and model analysis. A more effective model constructor greatly simplifies the effort required by a model analyzer, while a more powerful model analyzer lessens the burden placed on a model constructor. It is unreasonable in most circumstances to expect a model constructor to understand exactly what information is important or insignificant to a given problem instance in all cases up front. A more reasonable archetype is for the model constructor and model analyzer to interact iteratively — proposing a model, analyzing it, recognizing its deficiencies, refining and improving the model, re-analyzing it, and so on until a satisfactory model is obtained or time pressures dictate termination. This leads to the following central idea.

> *Information gained from solving a problem using a preliminary model can provide information about how the model should be refined.*

The general knowledge base used in this thesis consists of a stochastic process description with continuous random variables and an infinite horizon. To obtain an operational model, a finite horizon is chosen and a finite number of possible values are selected for each variable. I refer to this selection of possible values as *framing* or *discretization* — i.e., the process of selecting a *frame of discernment* (*sample space*) of a random variable. The resulting discrete-valued variables are assembled as a graphical probabilistic network, and posterior distributions are computed for the problem instance using the constructed model. It is possible, however, that the initial selection of possible values is poor for the current problem instance given the currently known information. The computed posteriors are used to suggest an improved set of possible

[10] Measures of information loss from approximate KBMC methods have been studied by [Laskey, 1991, Provan, 1993b, Provan, 1994, Laskey and Lehner, 1994]. However, these deal with a noniterative setting.

[11] Although the effect on fidelity is a topic of [Provan, 1993b, Laskey and Lehner, 1994].

values for each random variable, yielding an improved model for the task at hand. This process is then iterated. This is *iterative dynamic discretization* and it is the topic of Chapter 4. I call the more general idea of using posteriors to iteratively guide model construction and refinement *(Dynamic) Iterative Model Construction.*

The specifics of the knowledge base and construction process used in this thesis are in many ways quite different from other instances of KBMC in the literature. Most emphasis to date in previous published work on KBMC has been placed on using variations of probabilistic logics for general knowledge bases, with quantifiers and with rules having probabilities attached. The model construction process involves selecting instantiations for quantifiers and subsets of rules so that a propositional graphical network results. The model in this thesis is not a logic, and is not composed of rules. There are no explicit quantifiers (although it is possible to consider time and the handling of the infinite horizon as an instance of quantification, similar to [Haddawy *et al.*, 1995] and [Nicholson and Brady, 1994], but this is certainly not quantification in any general sense). And the model construction step focuses on variable discretization, but not on the structural design of the model. Indisputably, the work in this thesis does not consider many of the important aspects of KBMC that these previous works have focused on. Nevertheless, the thesis does address and study important issues for KBMC. Specifically, the thesis examines issues connected to the iterative construction of approximate models. The iterative model construction process is an important and virtually unstudied aspect of KBMC. Most existing work has focused on a one-time construction of a provably correct model, followed by the analysis of that model, *without* any feedback from the analysis back to the construction task. Also, this thesis plunges into experimentation with approximate-model construction (lossy KBMC) with the belief that experience and lessons learned here transfer to other instances of knowledge based construction of approximate models.

The Case-Study Approach

An objective of this thesis in this context is to explore the issues that arise when approximate probabilistic models are constructed from a broad probabilistic knowledge base. Again, the emphasis here is that information is lost during the model construction process. Conceivably, a lossy model construction process could cause arbitrary distortions to the conclusions derived from that model. Furthermore, before constructing a model, and without knowing the true answer in advance, it is very difficult (if not impossible) to know the real impact that omitting specific facts will have on the final result. These represent significant and difficult challenges to the enterprise of approximate-model construction.

Given what little is known about approximate-model construction, the issues and problems that arise when it is attempted, and the artistry necessary to make it work effectively, it seems that the best way to study this area at the present time is by way of a case study. This thesis takes a specific application — the model-based segmentation of multidimensional time series — and formalizes it as a large probabilistic inference problem (Chapter 2). The techniques of structural decomposition (Chapter 3) and Dynamic Iterative Discretization (Chapter 4) are then applied to construct an approximate model that can be used for computation in order to find a good segmentation. The case-study approach separates out, from the space of all conceivable artifacts of lossy model construction, only those that actually occur *in this specific application.* In this way, the work attempts to focus attention on the issues of greatest pragmatic concern, both for the current thesis as well as for the sake of future work in the area of KBMC.

The advantages of the case-study approach are demonstrated, for example, by the property of solution *mobility* (or *immobility*). In the time-series segmentation application, a sub-optimal model is often constructed during the earliest iteration(s). The information from that model and its solution are subsequently used (by iterative dynamic discretization) to re-assess the choices leading to the particular constructed model. The surprise from the case-study is that often each individual choice in the model's construction is nearly optimal in the context of all the other choices that have been made, even though the complete set of choices has substantial room for improvement. The result is that as each choice is re-evaluated in the context of the current model, no significant change occurs, and iterative model construction stays in a highly stable local minimum[12].

[12] Theoretical results guarantee that the algorithm will eventually escape the local stability if run long enough, but immobility greatly slows rates of improvement.

It is hardly surprising that an iterative algorithm can get stuck in a local minimum. However, quite a lot can be understood about the nature of these stable sub-optimal solutions in a particular domain, and I have found that algorithm variations can have a dramatic impact on the mobility. With some study, it is apparent that the problem is exacerbated in the time-series segmentation application by particular aspects of the formalization. This insight suggests a very simple addition to the iterative dynamic discretization algorithm, while maintaining the same local evaluation of individual choices. The modification (Section 6.7) in this case is rather specific to the time-series segmentation problem (i.e., a domain-dependent modification), but it is introduced to specifically counterbalance the application-specific property that is primarily responsible for the local model stability. The modification greatly improves mobility with dramatic impact on convergence rates.

The experience obtained from this case study suggests that mobility (or lack of it) is a significant concern for iterative model construction, but that certain properties of domain knowledge formalization may be largely responsible for the predominant negative effects. Once recognized, these may be compensated for through relatively simple means. At a minimum, anyone embarking on a lossy KBMC project would be wise to examine their knowledge formalism for similar characteristics that might lead to immobility, and to consider in advance what can be done to compensate for or eliminate these characteristics. The experience suggests that a future study, empirical or theoretical, to characterize mobility of iterative KBMC techniques and the conditions under which it is extreme, could be of considerable benefit to the endeavor of lossy KBMC. This one example demonstrates how the current case study may serve to provide a focus on the artifacts of real pragmatic concern. Similar issues are explored throughout the latter chapters of this thesis.

While developing and experimenting with iterative dynamic discretization, many of the basic ideas evolved to be increasingly similar to MCMC techniques. The evolution occurred largely in response to difficulties encountered and empirical behavior observed, and the implementation of the natural solutions to these obstacles along the way. Most of the resulting connections would exist as well for other instances of approximate, iterative KBMC, since they are more a result of the iterative and approximate nature of the process than to anything specific about this instantiation or application of iterative dynamic discretization. As such, these connections provide a number of potentially interesting insights and perspectives for KBMC. For example, notions of how to view and analyze asymptotic stability and correctness approximate iterative KBMC are suggested, which may eventually serve as analogues to the stronger versions of correctness in the noniterative case, such as those proved by [Haddawy, 1994] and [Breese, 1992]. This could potentially help in moving the study of approximate KBMC from a purely heuristic and empirical ground to a more solid theoretical foundation. It may also provide fruitful connections between approximate KBMC and other existing approximation techniques.

The case study explores a number of other issues. For example:

1. What artificial biases are introduced purely as the result of iterative model construction?

   For example, a number of artificial biases resulting from poor discretizations cause the segmentation algorithm to identify more transitions than it otherwise would.

2. When do these biases adversely bias other choices or future iterations of model construction?

   For example, a poor discretization of one transition time can cause the discretization of the following transition time to be even worse. Instances where artificial biases gets amplified in this manner are particularly deserving of attention.

3. How data-directed should/must model construction be?

   Exact (i.e., nonlossy) KBMC techniques such as [Breese, 1992] and [Haddawy, 1994] are entirely model-directed, with the constructed model being perfectly in agreement with the knowledge base. More generally, however, there are often heuristics based on the data, or subsets of the knowledge-base, that can be quickly evaluated to suggest appropriate parameters for the constructed models. The model construction of [Nicholson and Brady, 1994] is, for example, highly data-driven. However, reliance on data-directed methods generally introduces an additional bias, whereby the heuristics or partial knowledge used for model construction exerts an disproportionate amount of influence on the end result.

These are all instances of issues begging for additional case studies. The current case study provides a data point and at least some insight into each of these issues.

A case study has clear disadvantages as well. Strictly speaking, all conclusions are specific to the particular application. It must, at this point, be taken largely on faith that lessons learned here transfer to other applications and other manifestations of KBMC. A deep understanding of why a particular issue arises in this application can often help one to identify whether the same issue arises in another application; however, one should still be wary that issues that do not prove significant to this application could very well prove important in other applications. It is appropriate at this early stage to begin populating our understanding with lessons learned from case studies. It is also important to remain honest about the true generality of the lessons learned.

There are a number of things that can be done to increase the chances that lessons learned from this case study transfer widely to other instances of lossy KBMC. To these ends, every effort has been made in this work to use only the probabilistic knowledge contained in the general model while avoiding building application-specific heuristics into algorithms. Every effort is made to use general principles in algorithms rather than specific "hacks" that seem to work to empirically for time-series segmentation. These efforts help to ensure that the issues experienced are due to the handling of probabilistic knowledge, and that the techniques to deal with artifacts have the potential of being widely applicable beyond this specific application. In addition, the time-series application permits a great degree of variation — as experience with different variations grows, the breadth of the issues and lessons learned also grows, becoming (hopefully) more indicative of the scope of issues for lossy KBMC in general. These are general principles followed in this work in order to maximize the usefulness of this case study. Pragmatically, there are limits to how closely they can be followed, and it must still be remembered that this is still a single case study.

1.8.4 Hybrid Combinations

When it can be applied, exact propagation is usually superior to simulation and approximate methods — it produces exact answers, and without a need to iterate, can be much more efficient. The two primary factors that can prevent a reasonable use of exact propagation are (1) a complicated graphical structure, and (2) nonconjugate distributions. Exact methods do not suffer from, and in some cases can even benefit from ([Jensen and Andersen, 1990]), zero or close-to-zero probability configurations. Compare these attributes of exact propagation to those of Gibbs sampling. The efficiency of Gibbs sampling is generally insensitive to graphical structure (although it is sensitive to the size of the domain) and makes no conjugacy requirements on the distributions. Efficiency is adversely affected by low probability configurations, and zero probability configurations can destroy the irreducibility property that is necessary for correct results (see Section 3.4).

Because the properties of exact propagation and Monte Carlo schemes seem to complement each other so well, there is considerable promise for utilizing the strengths of both approaches by developing hybrid algorithms — combining exact with Monte Carlo methods.

To date, there appear to be have been two methods published in the existing literature for combining the two approaches: Blocking Gibbs [Hills and Smith, 1992, Jensen *et al.*, 1995] and Hybrid Propagation [Dawid *et al.*, 1994, Kjaerulff, 1995b]. This thesis presents two more approaches for combining the two: focused Gibbs (Section 3.4.1) and iterative dynamic discretization (Chapter 4). Again, this is one more possible perspective on this work — as a combination of stochastic approximation and exact inference (see Section 5.4).

Gibbs sampling changes the value for one variable at each step. The same method can be applied by grouping variables into blocks and then treating each block as if it were a variable, changing the value of one block at a time. This is called *Blocking Gibbs* (see Section 3.4.2). It is widely believed that blocking improves convergence rates (e.g., [Amit and Grenander, 1991, Kjaerulff, 1995b, Besag *et al.*, 1995, Roberts and Sahu, 1996]), with larger blocks resulting in larger convergence[13], but it also increases the computational difficulty each step. Specifically, "blocking moves any high correlation ... from the Gibbs sampler over to the random [value] generator [for each component]" [Seewald, 1992]. However, this is where exact methods may be of use,

[13]However, this is not universally true, for there are counter-examples in [Roberts and Sahu, 1996] that demonstrate that blocking can actually slow convergence in some cases.

and therefore result in a useful combination ([Jensen *et al.*, 1995]). Within each block, the graphical structure of the network within that block is utilized for the purpose of efficiently generating a sample. The block is solved (given an assignment to its values) using exact methods, and once these distributions have been obtained, a simple forward sampling strategy can be applied within the block to generate an independent sample from the correct distribution. [Jensen *et al.*, 1995] give empirical evidence that this combination can significantly improve the convergence rates over Gibbs sampling. Further, in their problem, continuous variables prevent direct application of exact methods, so the hybrid combination is useful. Blocking Gibbs is well-studied and is not necessarily tied to a combination of Gibbs with exact methods — it is just the paper by [Jensen *et al.*, 1995] that supplements Blocking Gibbs with exact propagation.

The second existing combination is called ***hybrid propagation*** and has been examined by [Dawid *et al.*, 1994, Kjaerulff, 1995b]. In this method, "universes" (usually cliques in a junction tree) are designated as being either exact or Gibbs. Messages are then passed using a control scheme similar to that used by exact propagation, except that Gibbs universes incorporate messages by simulating their local distribution and generating a list of possible local configurations. The full universe is then replaced by the list of local configurations — if the original universe was continuous, it becomes discrete, or if discrete, all ungenerated configurations become zero weighted. The sampled universe becomes an approximation of the original universe, and from that point exact propagation then treats it as if it were any other exact clique in the junction tree. The idea has been elaborated in [Dawid *et al.*, 1994, Kjaerulff, 1995b], but apparently it hasn't yet been studied to the point where empirical results are available.

Focused Gibbs sampling, introduced in Section 3.4.1 is a very simple idea that provides yet another way to combine Gibbs sampling with exact propagation. The idea is simply to apply Gibbs sampling to a subset of variables in a model (the focus set), sampling a new value for a variable at each step conditioned on the remaining variables in the focus set. Exact methods (propagation) are necessary to marginalize out the remaining variables from each step of Gibbs so that the procedure correctly implements Gibbs sampling on the focus set. See Section 3.4.1.

Finally, iterative dynamic discretization applies Gibbs sampling to the space of possible discretizations, rather than to the space of possible configurations. Each step of the Gibbs sampling produces a discrete set of configurations, and exact propagation can be applied to solve the discrete problem (at each iteration). Again, this is a method for combining exact propagation with Gibbs sampling (Section 5.4).

1.8.5 Optimization

For the particular problem of finding a MAP configuration, in addition to approximation techniques already reviewed, many standard optimization approaches can be applied. The task has been formulated as a linear programming problem [Santos, 1994, Li and D'Ambrosio, 1994], and as a heuristic search task [Dechter *et al.*, 1990, Henrion, 1991]. [Rojas-Guzmán and Kramer, 1993] apply genetic programming to the task. Although iterative dynamic discretization has little resemblance to this work, it does resemble genetic programming in that a single discretization could be viewed as a population of configurations. Changing discretizations over time corresponds to altering the population of configurations over time. If much of the power of genetic programming comes from maintaining a population of possibilities, this power is shared by iterative dynamic discretization. [Azevedo-Filho and Shachter, 1994] use an optimization technique called Laplace's method to search for MAP configurations. Finally, it should be repeated that simulated annealing is clearly a good substitute for Gibbs sampling for optimization tasks.

1.9 Time-Series Segmentation

One tangible result of this thesis is a model-based algorithm that segments a multi-dimensional time series into qualitatively distinct time intervals and labels signal shapes and context modes. A user specifies knowledge about the time series in the form of a Hidden Segmented Semi-Markov Model (H.S.S.M.M.), which can be thought of as a language for expressing probabilistic knowledge about durations, signal shapes, and transitions. Effectively, this model specifies an evaluation function over the space of possible segmentations, and the algorithm uses this evaluation to search for the optimal segmentation.

Real-world applications for multi-dimensional time-series segmentation abound. The stock market presents an obvious example of multi-dimensional time series data, where the historical stock prices for each individual stock constitute one dimension. Segmenting these histories may identify where qualitative changes in market conditions occur, which might then be useful for various financial analyses. However, stock market applications may not always be in the spirit of time-series segmentation as addressed in this thesis. A distinguishing feature of the segmentation task in this thesis is that it is *model-based*. It is aimed at applications where there exists a probabilistic model (albeit possibly a very weak and partially qualitative model) of the underlying system that generates the time-series data to be segmented. The appropriateness to stock market applications really depends on the availability of a suitable economic model of the market.

A complex system such as the Space Shuttle contains many sensors, each collecting data over time, and begs for a model-based approach since the underlying system is human designed and very well-understood. Identifying qualitative changes across suites of sensors can pinpoint context changes of various sorts — changes in mission phase, astronaut activity, or system configuration and conditions — that can then improve automatic monitoring or diagnosis capabilities. Segmentation of acoustic signals might be used to track or transcribe musical performances, with the score serving as an indirect source for an underlying model. Section 2.1 discusses the problem and potential applications in more detail.

The time-series segmentation task serves here as an interesting and useful case study for knowledge-based approximate-model construction. Finding the optimal segmentation of a time series entailed by the domain knowledge (the H.S.S.M.M.) is no trivial matter. The H.S.S.M.M. representation is very expressive, allowing continuous variables, nonconjugate continuous distributions, and specifies the behavior of the signals over an infinite horizon. In essence, the domain knowledge is just too complex to be used directly as a model during computations. Iterative dynamic discretization is employed to construct a discrete, finite model that is used for computation. After computations have been performed using a preliminary model, clues for how to improve the model are incorporated to construct a new model for computation. In this fashion, specific information about the specific data that is being segmented can incrementally be incorporated into the construction of the model used for computation, thus ultimately improving the model being used.

1.10 Contributions of Thesis

The primary thesis explored by this research is that information learned from solving a constructed discrete model can be used in an effective manner to select a new discretization. This is the central idea behind the iterative dynamic discretization algorithm. Application of iterative dynamic discretization to time-series segmentation produces a new algorithm for that task. Thus, new technologies introduced by this thesis include:

- Iterative dynamic discretization.
- A time-series model and segmentation algorithm.

The contributions of the iterative dynamic discretization algorithm can be viewed from many angles:

- As an (approximation) algorithm for handling continuous variables with nonconjugate distributions in graphical networks[14].
- As a method for combining the strengths of Gibbs sampling with exact propagation. Or as a method for accelerating convergence of MCMC methods.
- As an instance of approximate knowledge-based model construction. The studies in this thesis offer experience and lessons for the endeavor of iterative model construction.

The primary emphasis in this thesis is on general methods for probabilistic inference. However, the development was entirely directed at a single application area: model-based time-series segmentation. Time-series segmentation is of great interest by itself, and the thesis makes contributions in this area as well.

[14] The same techniques can also be applied to discrete variables with very large sample spaces.

First, the HSSMM is introduced in Chapter 2. The segmented nature of this model is unique and is a new contribution of this thesis. It may be convenient for many time-series modeling applications. Combined with the semi-Markov model, or the generalized semi-Markov model, this provides a very rich descriptive language of time-series behavior.

The general study of probabilistic inference techniques in this thesis endow the HSSMM formalism with a great deal of flexibility. Decomposition techniques (Chapter 3) and the conversion to graphical probabilistic network form make it relatively easy to generalize the model in domain-specific ways, for example, to leverage additional structure in the state space, generalize the transition model in various ways, and so on.

Structural decomposition along with iterative dynamic discretization provide a complete algorithm that utilizes the HSSMM to perform time-series segmentation. Thus, the thesis contributes a new algorithm for the time-series segmentation problem.

The two contributions to time-series segmentation, i.e., the HSSMM and the algorithm for solving the HSSMM, should really be viewed as distinct contributions. Even if one did not like the algorithm, for whatever reasons, or if something better comes along later, the HSSMM still provides a potentially useful modeling language for describing time-series behavior. If one prefers to apply pure Gibbs sampling, for example, and ignore the improved algorithms presented in later chapters, the HSSMM formalism could still be utilized. Conversely, if one has a probabilistic inference problem that is not modeled using an HSSMM, and may not even be a time-series segmentation problem, the algorithmic techniques presented in Chapters 3 and 4 can still be harnessed.

As a study of issues concerning probabilistic inference in general, the thesis makes contributions in a number of areas. The basic ideas behind structural decomposition in Chapter 3 are not new to this thesis; however, the thesis does provide a fairly unique perspective on one role these techniques can play as compared to the use that is dominant in the literature. Most existing literature treats graphical probabilistic models as formalisms for knowledge representation, while this thesis promotes the models as computational tools that are often useful for decomposing and solving problems in general. Chapter 3 generalizes current propagation algorithms via an easy to understand axiomatization. This is quite similar to an axiomatization given in [Shenoy and Shafer, 1990], but I believe the presentation here is far easier to understand and apply. It is also slightly different than the Shenoy-Shafer axiomatization. For example, Jensen's algorithm (also known as the `Hugin` algorithm) falls out of this axiomatization, but is not the same algorithm that results from the Shenoy-Shafer axiomatization.

Chapter 4 presents and studies the iterative dynamic discretization algorithm. A general framework for algorithms of this type is described and identifies the possible dimensions defining a specific variation of an iterative discretization algorithm of this type. A proof of asymptotic stability is given, showing that under suitable assumptions, iterative dynamic discretization always converges to a unique stable distribution of discretizations. In other words, the asymptotic behavior of the algorithm does not depend on the starting point (i.e., the initial discretization). The analysis also shows that the process is *recurrent*, meaning that the entire space of discretizations will, with probability 1, eventually be covered by the algorithm. Among other things, this means that the user can rest assured that if run long enough, the algorithm will eventually produce a configuration arbitrarily close to the optimum[15]. Although stability is proved, a useful characterization of the nature of the asymptotic behavior of the algorithm (i.e., precisely how the limiting behavior of the algorithm related to the distribution of interest) remains an open problem.

Finally, empirical evidence is given that iterative dynamic discretization can drastically (by a couple orders of magnitude) reduce the number of iterations required relative to Gibbs sampling (Figure 1.1). I examine a number of situations and troublesome phenomena that have been observed to occur, and which provide insight into the challenges of discretization and iterative model construction. I also investigate empirically the overall impact of various pieces of the algorithm.

1.11 Thesis Road Map

Chapter 2 begins by defining the time-series application domain and developing the probabilistic modeling formalism for describing time-series behavior (the HSSMM). It defines the time-series segmentation task

[15] Although this theoretical result says nothing about how long this will take.

is precise terms, and finally in Section 2.4 reviews previous approaches to time-series segmentation. All algorithm developments in later chapters are couched in terms of the time-series segmentation problem.

Chapter 3 decomposes the time-series segmentation problem defined in Chapter 2, turning a huge optimization task on a continuous k-dimensional space into k three-dimensional inter-related optimization tasks. This enormous simplification is based on conditional independence within the model and loses no information in the process. Propagation is used to communicate information between the three-dimensional subproblems such that afterwards, each problem can be optimized individually to obtain the globally optimum solution. It turns out, however, that even the three-dimensional optimization tasks are still too large to handle with exact methods. The solution to this comes in Chapter 4. Chapter 3 then discusses the methodology behind structural decomposition in general. A new and clean axiomatization is given which indicates when the propagation framework can be applied to harness structure to solve a particular problem. The axiomatization applies beyond the reaches of probability theory and can be interpreted as a general computational method for utilizing the structure that exists within a task for computational efficiency. A short discussion is given at the end of the chapter on the options that exist when structural decomposition is, by itself, insufficient for solving a probabilistic inference problem. This includes (Section 3.4) a thorough description of Gibbs sampling, and a description of a new and useful variation called *focused Gibbs sampling*.

Chapter 4 introduces the iterative dynamic discretization algorithm. With this algorithm, the subproblems identified in Chapter 3 can finally be solved (actually, the algorithm finds an approximate solution). First, a basic framework for choosing discretizations is described. This framework elucidates the important variations possible between iterative discretization algorithms in a way that makes for a natural understanding and comparison of alternative algorithms. An algorithm for randomly picking discrete points for a single variable based on the model, data, and current discretization of neighboring variables is developed in Section 4.4.2. Asymptotic stability of the algorithm is analyzed in Chapter 5, and empirical evidence given in Chapter 6 that iterative dynamic discretization can greatly improve convergence rates over pure Gibbs sampling. Section 6.11 describes specific difficult situations that arise when the algorithm is run. This is a great source for lessons to be learned from the case study, and suggests a number of improvements to the algorithm. Chapter 6 empirically examines the effect that several variations on the algorithm have on overall performance.

Chapter 7 discusses a number of issues related to the modeling of a time-series in a segmentation application. These issues are separate from the algorithmic issues of Chapters 3–6 that deal with searching for, or computing, a solution. For example, how does the uncertainty or fidelity of a time-series model impact the optimal segmentation? Questions of robustness such as this are related to a more general issue of how strong the prior expectations of a model are. Weak expectations lead to more data-directed solutions but also to more sensitivity to noise in the data, while strong expectations create an invariance to noise with a decreased sensitivity to unexpected occurrences. Some aspects of time-series behavior cannot be modeled using an HSSMM. These are discussed, and a generalization of the HSSMM is outlined. Also, some brief comments about the prospect of learning models from data are given. Chapter 7 touches a number of key issues to the endeavor of model-based time-series segmentation, and one of which is worthy of significant investigation, but each is considered only very briefly. The focus of this thesis is much more on inference and computation than on time-series modeling issues.

Chapter 2

Time-Series Segmentation

Multidimensional time series occur in many applications. Complex physical systems such as industrial plants, spacecraft, robots, and medical monitoring equipment often contain many sensors that continually produce streams of sensor readings. Financial markets produce rich sources of multidimensional time series used in a variety of investment analysis applications, as do trend analysis applications in environmental and socio-political domains. And the list goes on. In many of these domains, it can be useful to segment time series into time intervals identifying where qualitative changes in the behavior or shape of the time series occur.

This thesis develops and studies a probabilistic model-based segmentation algorithm. This exploration provides a case study of the issues involved in framing and solving a probabilistic computational model. This chapter discusses the segmentation problem in Section 2.1, and defines the probabilistic model used to guide the segmentation process in Section 2.2.

Time-series segmentation is, by itself, a topic of great interest. In this research, however, it serves more as an application area for the study of general probabilistic inference methods. The time-series segmentation algorithm developed in this thesis is a contribution, but the emphasis is much more on the study of general probabilistic inference techniques.

2.1 Problem Description

Time-series segmentation is one of the most fundamental problems of statistics. It is essentially a problem of detecting and localizing *change points*, i.e., points in time where the behavior of a system (abruptly) changes in some distinct way. Consider that most applications of statistics (in research studies, etc.) utilize tests for detecting changes or differences between two or more populations. Time-series segmentation extends this by detecting whether and *when* changes occur. Optionally, as done here, it can also including the identification of what the change is (e.g., assigning a meaningful label the states occupied between transitions).

Raw data often comes in the form of a time series, and it is often necessary to isolate segments where individual analyses can uniformly be applied. Abrupt changes in the behavior of a system invalidates many forms of analysis, so it is important that such changes be identified before any such analysis is applied. In some domains, the segmentation of a time series is a fundamental step in generating a higher level description of the data that can then be used for other purposes.

This thesis focuses on *model-based* segmentation of (multi-dimensional) time-series. An underlying model of the possible transitions, durations, signal characteristics between transitions, and degrees of uncertainty in all these factors is required. A model-based approach is best when there is some basis for expectations about a signal. *Multi-dimensional* refers to the fact that there are multiple related sensors, each generating its own time series on a shared time scale. While a model could specify only extremely weak expectations, and therefore perform in a very data-driven fashion (discussed in Section 7.1), this is clearly not a scenario where a model-based approach is most appropriate. Therefore, the approach described here is aimed at domains where a model of the underlying transition structure is (or could be made) available. This may be the case, for example, if the underlying system that generates the data is human designed or otherwise

well-understood. This thesis does not consider the task of learning a model from data.

2.1.1 Some Domain Examples

It is not difficult to find important uses for time-series segmentation. Since almost any raw sensor data comes in the form of a time-series, and since most complex systems exhibit qualitative changes of one form or another at various points in time, the problem arises naturally in almost any conceivable field.

Several applications in the medical domain have been developed based around a core technology of time-series segmentation. Several neurological disorders can be detected using the electroencephalogram (EEG). The EEG consists of 8 to 16 channels of electrical brain wave activity. The activity transitions through different phases as a patient passes through different sleep cycles or forms of mental activity. Certain neurological disorders result in anomalous patterns during some of the phases. Some disorders occur in short-lived phases that constitute a small fraction of a total EEG, so to detect and analyze these, it is critical to segment the EEG time-series to isolate the critical phases. Segmentation has been applied to EEGs in [Bodenstein and Praetorius, 1977, Ishii *et al.*, 1980, Basseville and Benveniste, 1983].

Diagnosis of heart conditions by way of the phonocardiogram (PCG) or the electrocardiogram (ECG) presents a domain very similar to the EEG, also benefiting from segmentation, and has been studied by [Stockman, 1982, Lee and Chou, 1989, Lee and Chou, 1990]. For example, the presence of murmurs is associated with whether a secondary pumping sound is heard following the primary heart beat, something best detected when the signal is separated into segments of pumping sounds, silence, etc. Note that these medical examples have a very good basis for expectations about transitions, and are thus very natural candidates for a model-based approach to segmentation.

The analysis or understanding of speech and music from audio waveforms presents another rich source of time-series segmentation problems. Transcription from audio-signals to notes on a score is essentially a problem of time-series segmentation. Because music has considerable regularity, a model-based approach would seem to be very appropriate. A problem with an even greater basis for expectations is score following, an important ability of an intelligent accompaniment device, for example, where a score provides very strong expectations about when and how transitions occur ([Dannenberg, 1984, Vercoe, 1984]). A major portion of speech recognition concerns segmentation — for example, segmenting a signal into phonemes, and expectations have long been known to be key to good performance at this task ([Li and Gibson, 1996]).

The stock market is a natural domain for many time-series analyses. It is a great source for multi-dimensional time-series data, with each stock comprising a single dimension, and inter-relationships existing between stocks in common industries, etc. Model-based approaches are appropriate for some of these analyses, and are not very appropriate for others. The ability to segment stock market data, for example to identify passing trends, has potential to improve numerous forms of analysis or historical understanding of market conditions that would otherwise not be possible (at least not automatically). The ability to quickly detect an abrupt change, such as a swing from a bear market to a bull market, could (if possible) be very profitable for an investor. Depending on the style of analysis, the information available, and the economic models utilized, model-based approaches for segmentation may or may not be the best choice for financial analysis. A financial method that integrates market fundamentals with quantitative technical indicators is an example that calls out for a model-based approach since the relationship between fundamental and technical indicators must somehow be established.

These examples demonstrate some of the many potential uses for time-series segmentation. In addition to these, I discuss one additional use of time-series segmentation, within a Space Shuttle monitoring application, in depth. This example further illustrates how uses for time-series segmentation can arise. It is also significant in that it was the example that motivated the use of time-series segmentation as the driving application for the research reported in this thesis.

2.1.2 Tracking Context in a Automatic Monitoring System

When the Space Shuttle (or other NASA mission) is in flight, a large number of human mission-control operators on Earth monitor signals produced by the thousands of sensors on board the shuttle, looking for abnormalities or signs of developing threats to the mission. The cost to maintain the large number of human

operators for this task comprises a substantial fraction of the total cost of a Shuttle mission. For this reason, computerized tools to assist in this monitoring task could reduce operations costs considerably.

The SELMON project at the Jet Propulsion Laboratory ([Doyle, 1995, Doyle *et al.*, 1993, Doyle and Fayyad, 1991]) targets the Shuttle monitoring application. The system attempts to focus an operator's attention on the most informative and critical sensors. If done well, this can allow a single operator to monitor more sensors, thus reducing the overall personnel requirement and cost. It may also help in crisis situations by helping an operator focus on the most important sensors when hundreds of alarms are simultaneously sounding. A similar system for this domain was developed by [Horvitz *et al.*, 1992, Horvitz and Barry, 1995].

The key to automatic monitoring is the ability to differentiate between normal and abnormal signals. The most trivial way to detect abnormalities is by using simple thresholds. For example, if the oil pressure drops below a certain threshold, an indicator light (an alarm) comes on. The idea can be generalized by applying thresholds to metrics derived from the raw sensor data, for example, by sounding an alarm when a temperature rises or falls at an excessive rate (thresholding the signal's time-derivative).

The main problem with simple thresholds (and many other approaches) is that they lack context-dependence ([Doyle, 1995]). What is normal for any given sensor often depends on a system's operating mode (takeoff, orbit, re-entry, etc.), what activity is in progress (are the astronauts sleeping, exercising, space-walking, etc.), and what environmental factors are present (is the Shuttle in the earth's shadow or exposed to the sun?). When fixed thresholds are set loose enough to catch only those readings that are anomalous in *all* contexts, many problems can go undetected. On the other hand, if fixed thresholds are set tighter than this, too many excess alarms are generated by normal conditions. Truly effective use of thresholds or other methods requires an analysis to take context into account, adjusting thresholds depending on the current operating context.

When context is totally observable, adjusting for context is conceptually simple. For the Shuttle, the phase of the mission (takeoff, orbit, re-entry, etc.), is a critical piece of context, but is also totally observable since the current phase of the mission is something that is always known to the system. In fact, the SELMON system already addresses context-dependence for directly-observable context. On the other hand, partially observable contexts present a much greater challenge. For example, the current activity of the astronauts is not usually observable, but, for example, may drastically impact oxygen consumption and thus be a critical consideration when monitoring the functioning of life-support subsystems.

For an illustrative example, consider the task of monitoring engine performance in an automobile. There is no sensor to detect whether the car is driving uphill or on level ground. However, engine ping while cruising on flat ground, or the absence of engine ping while accelerating uphill, are both indications of abnormalities (in both cases, they indicate an engine tune up is needed). Determining whether engine ping is normal or abnormal requires knowledge of context — the slope of the roadway — but is not directly observable. However, many other sensors in the car are influenced by the same context, e.g., instantaneous fuel consumption, suspension alignment, engine R.P.M., drive shaft torque, etc., each of which provides some clues about the important context. By simultaneously incorporating all these clues, one might plausibly track the relevant context, even when one abnormality (e.g., engine ping) exists. Note that considerable amounts of uncertainty are inherently part of the problem of tracking partially observable context.

A module to track partially-observable context would provide a means to improve the monitoring abilities of SELMON, and thus provided a well-motivated application domain for experimenting with probabilistic inference. Tracking context modes amounts to tracking qualitative changes in system behavior — i.e., time-series segmentation — with the added need to label the operating modes in a meaningful way. This latter requirement, along with the fact that the Shuttle is a well-understood system and that tracking should function well even in the presence of a few anomalies, strongly suggests the utilization of a model-based approach. Thus, a model-based multi-dimensional time-series segmentation problem naturally arises in this potentially profitable context-dependent monitoring application.

While initially motivated by the Shuttle monitoring application, this thesis does not address most of the monitoring concerns, largely because these are addressed already by SELMON, and largely because the scope of the thesis should (and must) be narrowed appropriately. To match well with the requirements of a context-tracker in the context of a monitoring system such as SELMON, the basic problem to be solved is to produce, at any moment in time, a description of the single most-likely context history over time. SELMON would have little trouble incorporating a description in this form. Had the motivating

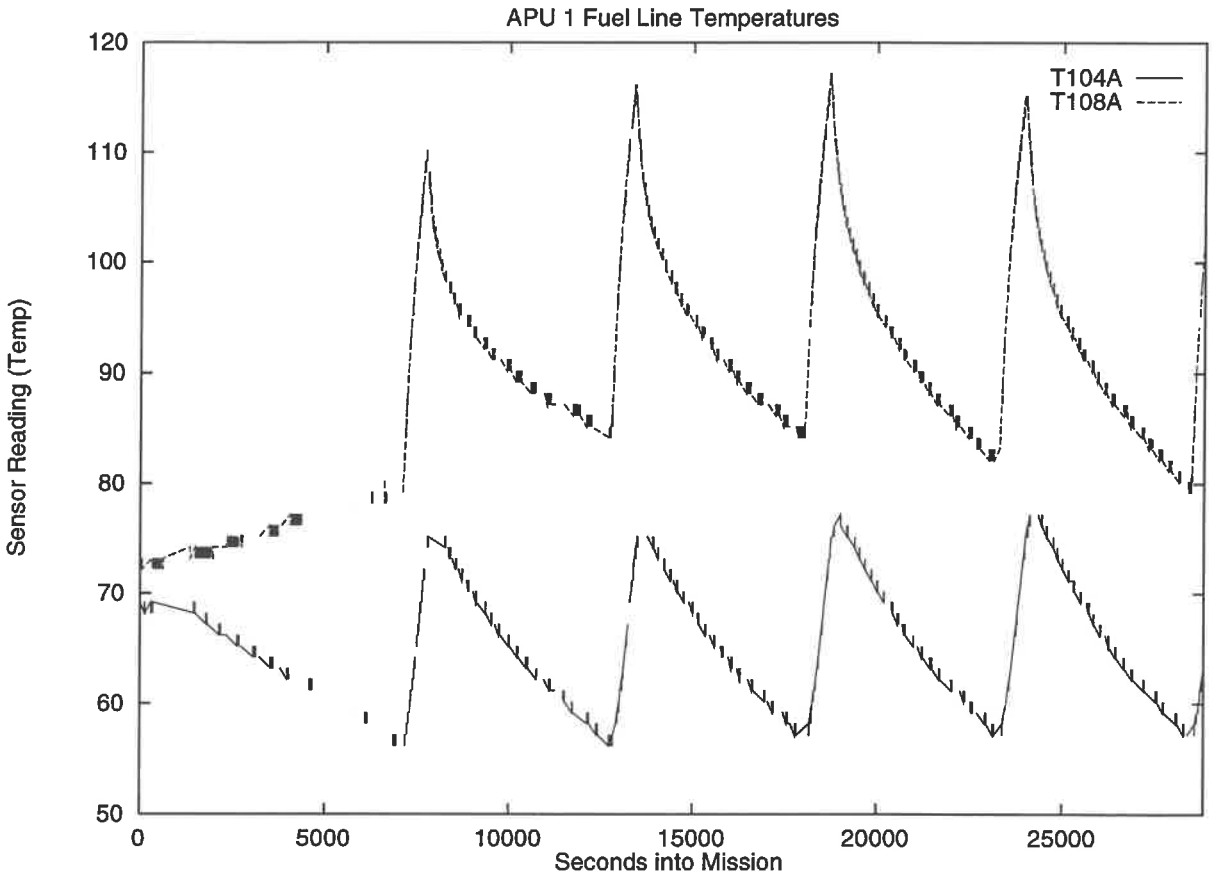


Figure 2.1: A two dimensional time series. The data shows fuel line temperature readings for the Space Shuttle's first Auxiliary Power Unit (A.P.U. #1) during the 55th Shuttle mission.

example been different, I might have concentrated the research more directly on problems like evaluating the probability of key propositions. Because of considerations such as this, the use of a motivating example has had some important impact on the nature of problems emphasized in the work reported in this thesis.

2.2 Hidden Segmented Semi-Markov Models

Model-based segmentation requires a model of the process generating a time series. This section presents a language for expressing such a model, the language of Hidden Segmented Semi-Markov Models (HSSMMs).

Consider the time series shown in Figure 2.1. The graph shows the evolution of two fuel line temperature readings, $T104A$ and $T108A$, during a flight of the Space Shuttle. A natural way to describe the evolution of this time series is shown in Figure 2.2. In Figure 2.2, the system begins in the *startup* state where $T104A$ decreases while $T108A$ slowly increases. After about 7,000 seconds, the system transitions to the *heat up* state where it stays for about 1,000 seconds while both temperatures rapidly increase. Then, in the *cool down* state, $T104A$ decreases more or less linearly while $T108A$ exponentially decays, with this mode lasting for about 5,000 seconds. The system then alternates between the *heat up* and *cool down* states indefinitely.

With a little further elaboration, the description shown in Figure 2.2 can be considered to be a model of the time series, and such a model can be used to guide the segmentation of a time series such as the one shown in Figure 2.1. In fact, the description of Figure 2.2 demonstrates all the basic components of the HSSMM.

There are two problems with using Figure 2.2 literally as a model of the time series. First, it is not operational — the components must be stated more precisely. For example, what is meant by *about 7,000*

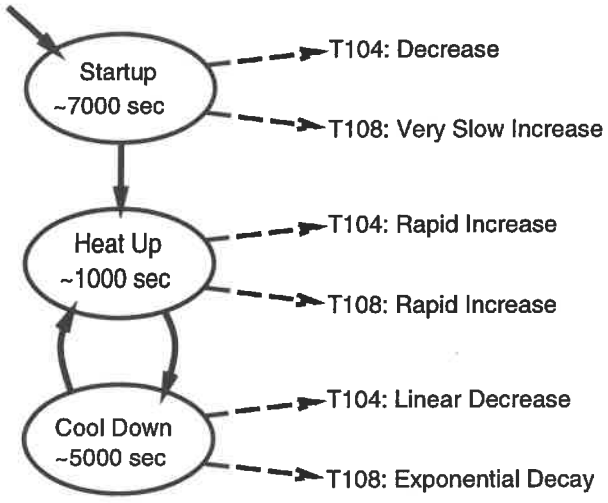


Figure 2.2: A simple time series description.

seconds? Would 10,000 seconds qualify? Similarly, what signals qualify as *slowly increasing*, *exponentially decaying*, or *linearly decreasing*? When data is noisy (unlike the exceptionally clean data of Figure 2.1), deciding whether signals qualify as one or more of these shapes may involve a judgement call, and therefore in an implementation precise definitions are necessary. A second problem with using Figure 2.2 literally as a model is that the transitions may be too rigid. It is possible that the same sensor may evolve slightly differently on a different shuttle mission, and if this is a possibility, it is desirable to model transitions as being stochastic. By addressing these deficiencies — i.e., allowing probabilistic transitions and operationalizing the imprecise components — the HSSMM is obtained.

2.2.1 The Transition Process

Figure 2.3 shows a more precise description of the transitions corresponding to the time series in Figure 2.1. The figure generalizes the deterministic transitions of Figure 2.2 to stochastic transitions, and it replaces the approximate *waiting times* with fully specified distributions over the possible waiting times for each state. The signal shapes corresponding to the right hand side of Figure 2.2 are not shown in Figure 2.3 — these will be covered in Section 2.2.2.

The model in Figure 2.3 consists of a state space $S = \{start\_up, heat\_up, cool\_down\}$, transition probabilities a_{s_i,s_j}, an initial state distribution b_{s_0}, and waiting-time distributions for each state, $c_{s_i}(\Delta t)$, where $s_i, s_j \in S$ and $\Delta t \in (0, \infty)$. Note that S is discrete and finite and Δt is a continuous random variable. These components are described below.

A model of this form is called a *(continuous-time) semi-Markov model.* The *semi-Markov property* means that *at the instant* following a transition, the (new) state of the process summarizes everything there is to know about the process. A process with this property is called a *semi-Markov process.* This is in contrast to the *Markov property*, which means that at *any* moment in time, the state of the system summarizes everything there is to know. The Markov property implies the semi-Markov property, and a continuous time Markov model can be represented as a semi-Markov model with exponential waiting-time distributions. On the other hand, a semi-Markov process (with nonexponential waiting time distributions) is not Markov since the amount of time that the system has been "waiting" in the current state provides additional information about when the next transition will occur. Because the state space is discrete and finite while time is continuous in these models, it is in general not possible to exactly replicate a semi-Markov process using a Markov process with an expanded state space[1]. Semi-Markov models are useful when one wishes to model

[1]Of course, a discrete-time semi-Markov process with bounded waiting times can always be converted to an equivalent Markov model with an expanded state space by simply including waiting time in the modified state space. However, even in

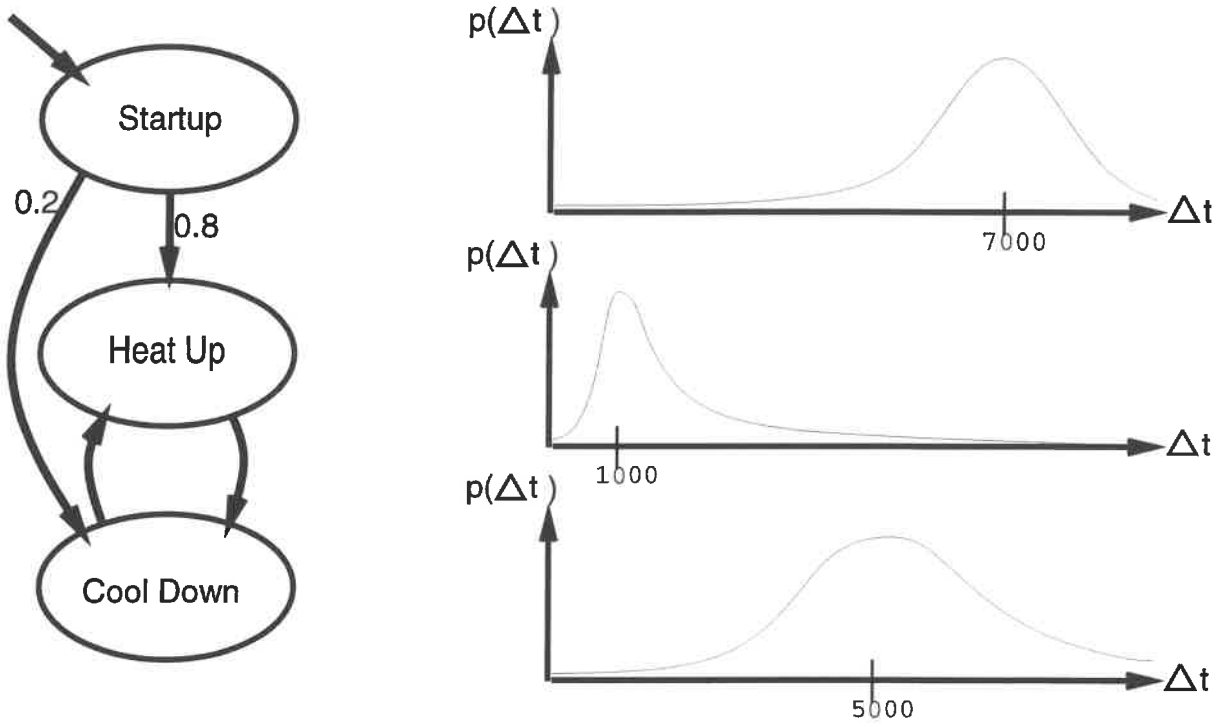


Figure 2.3: A Semi-Markov Model description.

general information about durations within a system. A very good reference on semi-Markov models is [Howard, 1971, Vol. 2].

The transition probabilities, a_{s_i,s_j} specify the probability of transitioning to state s_j whenever a transition initiates from state s_i. In Figure 2.3, $a_{start\_up,heat\_up} = 0.8$, $a_{start\_up,cool\_down} = 0.2$, $a_{heat\_up,cool\_down} = a_{cool\_down,heat\_up} = 1$, and $a_{s_i,s_j} = 0$ for all other $s_i, s_j \in S$. It is required that $0 \leq a_{s_i,s_j} \leq 1$ and $\sum_{s_j} a_{s_i,s_j} = 1$. A transition from mode s_i to itself is called a *virtual transition* ([Howard, 1971, Vol. 1, Chapter 10]). Virtual transitions are permitted in the models here (i.e., a_{s_i,s_i} may be positive). When a virtual transition occurs, the state does not change, but the time series may undergo a qualitative change at that moment. For example, a sensor that linearly increases from state s_i may experience a slope change following a virtual transition from s_i to s_i.

In addition to transition probabilities, it is necessary to specify the initial occupancy distribution over S at time $t = 0$ when the process is started. This is denoted by b_{s_0}, the probability that the system is in state s_i at $t = 0$, for $s_i \in S$. It is required that $0 \leq b_{s_0} \leq 1$ and $\sum_{s_0} b_{s_0} = 1$. There is an implicit assumption here that a state was indeed entered exactly at $t = 0$. When the starting moment is arbitrary, this may be a bad assumption; however, for the sake of simplicity, it will be assumed throughout the remainder of this thesis (although the assumption is quite easy to relax). When the starting moment is indeed arbitrary, it may be desirable to replace b_{s_0} by the steady-state occupancy distribution[2]. In Figure 2.3, $b_{start\_up} = 1$, $b_{heat\_up} = b_{cool\_down} = 0$.

The final components of the model are the waiting-time distributions, $c_{s_i}(\Delta T)$. For each state s_i, an arbitrary probability density function over $\Delta t \in (0, \infty)$ may be specified. It must be the case that $0 \leq c_{s_i}(\Delta t)$

---

this case there may be computational advantages in certain types of problems for maintaining the semi-Markov representation. See [Howard, 1971, Vol. 2, Chapter 10] for more details.

[2]The steady-state occupancy distribution is uniquely defined when the semi-Markov model is *ergodic*, and is further a function simply of the mean waiting times at each state, even if the waiting times themselves are arbitrary distributions. Again, see [Howard, 1971].

and $\int c_{s_i}(\Delta t)d(\Delta t) = 1$. The segmentation algorithm developed in later chapters does not assume that the distribution is in a known parametric form. The implementation of a distribution requires the distribution to support the queries `pdf(Δt)`, `cdf(Δt)`, `Variance()`, and `RandomVariate()`, where `pdf` returns the probability density, `cdf` returns the cumulative probability density, and `RandomVariate` returns a sample (a value for Δt) drawn independently and randomly from the distribution. It is also important that these queries can be computed very efficiently.

Often, one may wish to use a Gaussian waiting-time distribution, for example when it is known that the waiting time is "about 1,000 seconds." However, the Gaussian distribution extends into $\Delta t \leq 0$, and so is not a possibility. There are two good alternatives to a Gaussian: A truncated Gaussian, and a gamma distribution. The truncated Gaussian is simply a Gaussian distribution chopped off at $t = 0$ and normalized. A gamma distribution is a standard parametric distribution covered in any good statistics text. It is parameterized by two real numbers, $\alpha > 0$ and $\beta > 0$, and has the density

$$c(\Delta t) = \frac{e^{-\Delta t/\beta}(\Delta t)^{\alpha-1}}{\beta^{\alpha}\Gamma(\alpha)}$$

for $\Delta t > 0$. It has a mean of $\alpha\beta$, a mode of $\max\{0, (\alpha - 1)\beta\}$, and a variance of $\alpha\beta^2$. Special cases include $\alpha = 1$ (the exponential distribution with decay rate and mean $1/\beta$), α even and $\beta = 2$ (chi-squared distribution with $\alpha/2$ degrees of freedom), and the limiting case $\alpha \longrightarrow \infty$ (the Gaussian distribution). When $\alpha > 1$, it is a right-skewed unimodal distribution (i.e., *mode* < *mean*) and a reasonable substitution when one desires a Gaussian-like shape.

Waiting-time distributions can be generalized slightly to *holding-time distributions*, $c_{s_i,s_j}(\Delta t)$. A holding-time distribution augments waiting-time distributions with a dependence on the target state of a transition. The waiting-time distribution is obtained from the holding time distribution by $c_{s_i}(\Delta t) = \sum_{s_j} c_{s_i,s_j}(\Delta t)a_{s_i,s_j}$. Because notation is slightly simplified and the extra dependence was not utilized in the applications studied, only waiting-time distributions are used in the remainder of this thesis. The generalization to holding times presents no conceptual challenge. In fact, the structural decomposition in Chapter 3 is uneffected by this generalization, so the only real pragmatic difference is the increased complexity of notation. The issues of convergence and mobility that are discussed in Chapters 4–6 may be effected by the precise choices of numerical values, but not by any structural change created by using the more general holding-time distributions.

A semi-Markov model may be simulated as follows. At $t = 0$, select a state s_i according to b_{s_0} and then select a waiting time Δt_1 according to $c_{s_i}(\Delta t_1)$. After Δt_1 time has elapsed, pick a state s_j according to a_{s_i,s_j} and transition to s_j. Then pick a new waiting time, Δt_2 according to $c_{s_j}(\Delta t_2)$ and remain in state s_j until another Δt_2 time elapses. Repeat this process indefinitely. The simulation of a semi-Markov model produces a sequence of states and transition times. Methods for computing many of types inferences concerning a semi-Markov process are given in [Howard, 1971].

2.2.2 The Segmented Observation Model

At any given time in a time-series segmentation problem, the underlying state of the process generating the time series is not directly observable. The observation model elucidates the relationship between the underlying state and the time series that are actually observed. This section describes the observation model, including the concept of shape recognizers and some of the issues that must be addressed to obtain a fully meaningful model.

The right hand half of Figure 2.2 describes how the qualitative properties of the observed time series depend on the underlying state. However, the description in that figure is too qualitative. Time series consist of streams of real numbers (sensor readings) — they are not provided in terms of the qualitative shape descriptors. The observation model must precisely describe what constitutes a *rapid increase*, *linear decrease*, or *exponential decay*.

The segmented observation model operationalizes the meanings of these qualitative shapes by way of functions called *shape recognizers*. A shape recognizer, d_q for shape q takes as input a time series X^v for sensor v and two time points, t_1 and t_2. It returns a value, $d_q(X^v, t_1, t_2) > 0$, indicating the degree to which the given time series in the interval from t_1 to t_2 is representative of the underlying shape q. The greater the

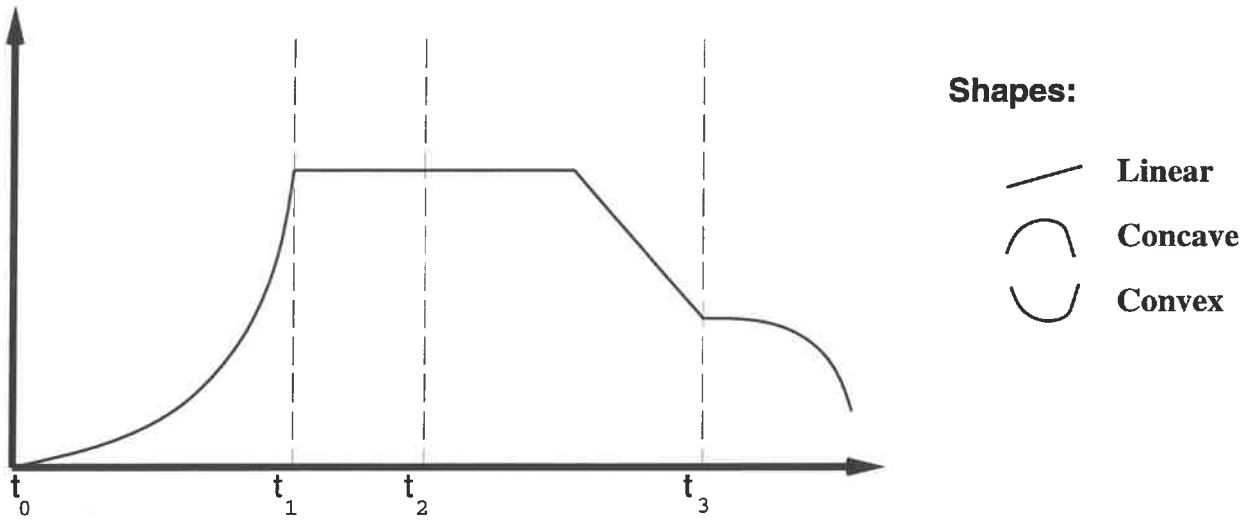


Figure 2.4: A Time Series with Three Shape Recognizers: *linear*, *concave*, and *convex*.

value, the more representative the time series is. It must be that $d_q(X^v, t_1, t_2) > 0$, and it is often convenient to enforce $d_q(X^v, t_1, t_2) \leq 1$ as well, so that $d_q = 1$ indicates a perfect exemplar of q.

Consider the time series in Figure 2.4, and suppose there are three shape recognizers: d_{linear}, $d_{concave}$, and d_{convex}. In the interval from t_0 to t_1, the time series is not very linear, so $d_{linear}(X, t_0, t_1)$ returns a number close to zero. It is an even worse example of a concave shape, so $d_{concave}(X, t_0, t_1)$ is smaller still. On the other hand, it is a very good example in that interval of a convex shape, so $d_{convex}(X, t_0, t_1) \approx 1$.

We can also evaluate $d_q(X, t_0, t_2)$ for each of the three shapes. In this case, the signal within that interval matches none of the shapes, so all three return small values when evaluated on that interval.

Notice that the time series on the interval $[t_2, t_3]$ in Figure 2.4 is *not* a good representative of the linear shape. Therefore, $d_{linear}(X, t_2, t_3)$ returns a small value. This highlights the difference between the segmented observation model and more standard observation models found in the literature on hidden expanded state Markov and hidden semi-Markov models ([Guedon, 1992], [Russell and Moore, 1985], [Levinson, 1986], [Guedon and Cocozza-Thivent, 1990], [Cook and Russell, 1986], [Huang and Jack, 1989], [Gupta *et al.*, 1987]). Classical observation models operate by sliding a small window over discrete time points in the signal. In each window, an estimation of the properties in that window is made, and then (this is the important part), the assumption is made that the properties of each window are independent of the other windows once the underlying state is known. In the $[t_2, t_3]$ interval of Figure 2.4, a classical technique for judging linearity would judge that all but one of the small windows in that interval look extremely linear. The single anomaly would be attributed to noise, and the overall linearity score would be high. It is only by evaluating the segment over the entire interval that we can recognize that it is not a good exemplar of a pure linear shape, one with a single qualitative characterization during the entire interval. The segmented observation model is a new contribution of this thesis.

Consider how a shape recognizer evaluates the "goodness" of a time series within an interval. A shape recognizer for linear shapes (regardless of the slope) operates, for example, by fitting a line to the data within the given interval and computing the residual ϵ. The mean squared error residual, equal to the sum squared error divided by $n - 2$, where n is the number of data points in the interval and 2 is the number of degrees of freedom when fitting a line, is used to achieve an invariance to the number of data points in the interval. The shape recognizer then returns $\alpha e^{-\alpha\epsilon}$, where $1/\alpha$ is the expected residual when the shape really is linear (this form was chosen for its simplicity but is otherwise arbitrary). The parameter α determines the sensitivity of the recognizer to noise, a smaller α will return favorable scores even when the signal is corrupted by a lot of noise or the fit is poor. A larger α will be much more stringent. Similar techniques can be used for log-linear shapes (e.g., exponential increases or decays) by doing the same analysis after taking

the logarithm of the sensor data[3].

A problem that arises in the implementation of a shape recognizer is the problem of what to return when there are an insufficient number of data points to reasonably determine whether the data matches the given shape. A linear shape requires at least two points in the interval to fit a line, and convex or concave recognizers require at least three points. Additional points are required to obtain reasonable estimates of residuals. However, $d_q(X^v, t_i, t_{i+1})$ must be defined on all intervals, whether or not data is present in that interval. The specific way in which this is handled has been observed to have a significant effect on the evaluation of segmentations and, consequently, on the performance of the segmentation algorithm as a whole.

In some domains, this sparse data problem can be ignored. For example, if the probability of a transition lasting less than 1 minute is zero, and in any given minute at least 10 data points will always be observed, then intervals with less than 10 data points are never relevant, and so it does not matter what is returned. However, many cases, including the gamma distribution, allow arbitrarily short transitions with nonzero probability. The precise evaluation when little or no data points are in an interval can artificially bias the segmentation to prefer or avoid arbitrarily short transitions. Thus, it is usually important to handle the situation with few data points appropriately.

A method that has been used in this work is the following. Suppose $1/\alpha$ is the expected residual when the underlying shape is really present. When there is an insufficient number of points to compute a residual, $1/\alpha$ is returned. Otherwise, a weighted average of $1/\alpha$ and (e.g.) $\alpha e^{-\alpha\epsilon}$ is returned, where the average is weighted more and more towards the latter as the number of data points increases. This is easily accomplished by using a straightforward Bayesian estimation of ϵ with a Dirichlet prior. The Dirichlet prior adds one additional parameter to the shape definition. In the thesis, this extra parameter was simply hard coded (to 3), so that ϵ was estimated as $(3/\alpha + sum\_squared\_error)/(n-2+3)$, where again, $n-2$ is the number of data points in the interval less the number of free parameters in the regression fit. In other words, this is equivalent to pretending there are three additional points, all of which are at the expected residual, that get averaged into the real data points. This (admittedly arbitrary) solution was sufficient to eliminate obvious instances of undesirable segmentations resulting from sparse data, and therefore alleviated the need to pursue a study of this problem further. It remains, however, a general problem that could benefit from the development of a more solid statistical foundation.

The formulation of shape recognizers is very general. Analysis within a single time interval could, alternatively, be performed in the frequency domain, for example. This may be useful for characterizing oscillations as shapes. There are some efficiency concerns, however. The shape recognizers are called often by the segmentation algorithm, so they must be relatively efficient. Autoregressive moving average models (ARMA) also form an interesting and natural class ([Lütkepohl, 1993]).

The shape recognizer functions, d_q, serve to precisely specify the meaning of the qualitative shape labels in Figure 2.2. The full specification of the observation model is completed with the mapping from state to shape for each sensor. Let $sh(s_i, v)$ be a function that returns the shape of sensor v from state s_i. It is allowed for several distinct states to map to the same shape. For notational convenience, it is convenient to write $d_{s_i}(X^v, t_1, t_2)$ as a shorthand for $d_{sh(s_i,v)}(X^v, t_1, t_2)$.

Putting the transition and observation models together, we obtain the complete Hidden Segmented Semi-Markov Model. The HSSMM is thus defined by the four components:

a_{s_i,s_j}	: Transition Probabilities
b_{s_0}	: Initial occupancy distribution
$c_{s_i}(\Delta t)$	: Waiting-Time Distributions
$d_{s_i}(X^v, t_1, t_2)$	: Shape Recognizers

With the time-series now modeled, the next section considers how this is to be used in the context of time-series segmentation.

---

[3] This works when the exponential curve asymptotes to zero. For example, such a characterization may be natural for stock market data where investors are concerned with rate-of-return rather than absolute stock prices.

2.2.3 k-Length Segmentations

A *segmentation of length* k is a listing of the first k states and the first $k+1$ transition times:

$$\lambda = \langle t_0, s_0, t_1, s_1, ..., s_{k-1}, t_k \rangle$$

As discussed above, here it is always assumed that $t_0 = 0$. Each t_i is a real value, and each $s_i \in S$. These transition times have no relation to the discrete times at which data is sampled.

The model described in the previous sections allows us to define the "goodness" of individual segmentations, and ultimately, the notion of an optimal segmentation. With a precise evaluation function, a well-defined search task is therefore obtained, and the task of segmentation becomes one of optimization. In this section, I discuss what makes a good evaluation function in time-series segmentation problems.

Data Fit

The shape recognizers of the previous section evaluate the "goodness" of a single segment in a time series. We obtain an indication of how well the time series fits a given segmentation by multiplying together the evaluation ("goodness") of each individual segment for each sensor. Doing so yields

$$d(X|\lambda) = \prod_{i=0}^{k-1} d_{s_i}(X, t_i, t_{i+1}) = \prod_{i=0}^{k-1} \prod_{v} d_{s_i}(X^v, t_i, t_{i+1}) \tag{2.1}$$

where λ is a k-length segmentation.

Trajectory Probability

The $d(X|\lambda)$ metric indicates how well the observed data matches a given segmentation. The overall goodness of a segmentation is not, however, solely a function of how well the segmentation fits the data. It is also important to consider the probability, $P(\lambda)$, that the particular sequence of state transitions would occur if the model were simulated. This probability must take into account the probability of transitions (as given by a_{s_i,s_j}) as well as the probability for the duration of each transition. Because time is continuous, the "probability" of any precise state trajectory is typically zero, so $P(\lambda)$ must be a probability density. However, densities are sensitive to time-scale distortions. In other words, if one distorts time in different ways, different measures for density are obtained, and thus, different criteria for measuring $P(\lambda)$. Alternative density measures may be appropriate for different tasks, and thus it is important to take care in selecting an appropriate density measure for the segmentation task. To emphasize this point, consider the following apparent paradox.

Example: A semi-Markov model consists of two states, s_1 and s_2, with $a_{s_1,s_2} = a_{s_2,s_1} = 0.9$, and $a_{s_1,s_1} = a_{s_2,s_2} = 0.1$. The waiting-time distribution in s_1 is bell-shaped with mean 1000 (seconds) and standard deviation of 1.0. The waiting-time distribution in s_2 is also bell-shaped with mean 1000 but with standard deviation 100.0. If the system is started in s_1 and run for 4 transitions, what is the most probable trajectory?

Most people would (intuitively) agree that the most probable trajectory alternates between s_1 and s_2, spending exactly 1000 seconds in each state. What is the probability (density) of this trajectory? The obvious measure is:

$$\begin{aligned} P(\lambda_1) &= c_{s_1}(1000) \cdot a_{s_1,s_2} \cdot c_{s_2}(1000) \cdot a_{s_2,s_1} \cdot c_{s_1}(1000) \cdot a_{s_1,s_2} \cdot c_{s_2}(1000) \cdot a_{s_2,s_1} \\ &= \frac{1}{\sqrt{2\pi}} \cdot 0.9 \cdot \frac{1}{100\sqrt{2\pi}} \cdot 0.9 \cdot \frac{1}{\sqrt{2\pi}} \cdot 0.9 \cdot \frac{1}{100\sqrt{2\pi}} \cdot 0.9 \\ &= 1.66 \times 10^{-6} \end{aligned}$$

Compare this to the trajectory that repeatedly stays in state s_1, each time for exactly 1000 seconds:

$$\begin{aligned} P(\lambda_2) &= c_{s_1}(1000) \cdot a_{s_1,s_1} \cdot c_{s_1}(1000) \cdot a_{s_1,s_1} \cdot c_{s_1}(1000) \cdot a_{s_1,s_1} \cdot c_{s_1}(1000) \cdot a_{s_1,s_1} \\ &= \frac{1}{\sqrt{2\pi}} \cdot 0.1 \cdot \frac{1}{\sqrt{2\pi}} \cdot 0.1 \cdot \frac{1}{\sqrt{2\pi}} \cdot 0.1 \cdot \frac{1}{\sqrt{2\pi}} \cdot 0.1 \\ &= 2.53 \times 10^{-6} \end{aligned}$$

Hence the apparent paradox: the intuitively most probable trajectory is actually not the most probable. In this case, the chances of the system staying in s_2 for exactly 1000 seconds (or for some arbitrarily close amount to 1000 seconds) is so much less than the chances of it staying in s_1 for exactly 1000 seconds that the trajectory that stays always in s_1 appears more probable.

The apparent paradox is a result of the particular choice of density used when computing $P(\lambda)$ above. The density used in the example is certainly reasonable, but it is inappropriate for the segmentation problem since it would introduce an unwanted bias on states with low-variance waiting times. For the segmentation task, we do not want waiting time variances to influence trajectory probabilities as it did above. We do, however, want the waiting-time distributions to influence trajectory probabilities, for a trajectory containing a very unlikely transition duration (for example, excessively long) should be assigned a low probability.

Before identifying the appropriate density function for $P(\lambda)$, consider first the probability when time is discrete. Let $Pr(\lambda)$ be the probability of the discrete-time parse, rather than the probability density. Discrete-time distributions for waiting time are obtained by discretizing Δt. Suppose we discretize by dividing time into units of length $\delta > 0$ where $\delta \approx 0$. Then $Pr(\Delta t|s) = \delta c_s(\Delta t)$. In this case, a measure of density is

$$\begin{aligned} P(\Delta t|s) &= \lim_{\delta \to 0} Pr(\Delta t|s)/\delta \\ &= c_s(\Delta t) \end{aligned}$$

However, this is not the only possible way to define density. Another possibility is to discretize the Δt in $c_s(\Delta t)$ by discretizing units of standard deviation. Now let $\delta \approx 0$ be the number (fraction) of units of standard deviation per discrete chunk of time. Here time is discretized differently for each state. Now, $Pr(\Delta t|s_i) = \delta\sigma(c_{s_i})c_{s_i}(\Delta t)$, where $\sigma(c_{s_i})$ is the standard deviation of the waiting time distribution from s_i. As $\delta \to 0$, we again approach continuous time, but now the measure of density becomes

$$\begin{aligned} P(\Delta t|s_i) &= \lim_{\delta \to 0} Pr(\Delta t|s)/\delta \\ &= \sigma(c_{s_i})c_{s_i}(\Delta t) \end{aligned}$$

This measure of density is clearly insensitive to differences in variance in waiting-time distribution, and reflects the rarity of a transition time rather than the absolute probability. The apparent paradox in the above example does not appear with this choice of density — the trajectory that alternates states is indeed the most likely, and in fact, it is exactly 9^4 times more likely than the trajectory that continually transitions to the same state. There are an unlimited number of other possible densities that can legitimately be identified, but this is the one that appears appropriate for the segmentation task. Thus, the probability (density) of a particular sequence of state transitions is given by

$$P(\lambda) = b_{s_0} \prod_{i=0}^{k-1} a_{s_{i+1},s_i}\sigma(c_{s_i})c_{s_i}(t_{i+1} - t_i) \tag{2.2}$$

where $\sigma(c_i)$ is the standard deviation of the distribution $c_{s_i}(\cdot)$.

Overall evaluation

The overall evaluation of a segmentation is the normalized product of data fit and trajectory probability, given by

$$\begin{aligned} P(\lambda|X) &= \alpha P(\lambda)d(X|\lambda) \\ &= \alpha b_{s_0} \prod_{i=0}^{k-1} a_{s_i,s_{i+1}}\sigma(c_{s_i})c_{s_i}(t_{i+1} - t_i) \prod_v d_{s_i}(X^v, t_i, t_{i+1}) \end{aligned} \tag{2.3}$$

where α is a normalization constant equal to $\alpha = (\int_\lambda P(\lambda)d(X|\lambda)d\lambda)^{-1}$. The normalization constant depends only on the HSSMM and the data, and so is a constant with respect to the space of possible parses. Because

we are usually only interested in the relative goodness of segmentations (with the data and model fixed), the normalization constant is optional and can be ignored if desired.

By interpreting $d_{s_i}(X^v, t_i, t_{i+1})$ as the probability of observing the data $X^v[t_i, t_{i+1}]$ given that the process is in state s_i during the interval $[t_i, t_{i+1}]$, the evaluation function in (2.3) is precisely what is obtained from Bayes's rule. Our evaluation function is slightly more flexible than this, however, since we do not absolutely require $d_{s_i}(X^v, t_i, t_{i+1})$ to truly be a probability distribution[4]. Under the Bayes' rule interpretation, $1/\alpha$ is the probability of seeing the time series observed as determined by the model. We will adopt the interpretation that $d(X|\lambda)$ reflects relative probability densities, such that $d(X|\lambda_1)/d(X|\lambda_2)$ is the ratio of the probability (density) of observing exactly data X given λ_1 versus λ_2.

2.3 The Search Task

The metric $P(\lambda|X)$ provides an evaluation of a k-length segmentation given time-series data. A segmentation λ_1 is considered better than λ_2 of the same lengthwhenever $P(\lambda_1|X) > P(\lambda_2|X)$. I do not have a method to compare segmentations of different lengths. The segmentation task is defined in this section in such a way so as to avoid the need to consider segmentations of differing lengths.

Let Λ_k denote the space of all k-length segmentations, and let

$$\lambda_k^* = \arg\max_{\lambda \in \Lambda} P(\lambda|X) \tag{2.4}$$

be the optimal k-length segmentation. The optimal segmentation is

$$\lambda^* = \lim_{k \to \infty} \lambda_k^*$$

This infinite-length segmentation would not only segment existing data in the time series, but would also predict the most likely future transitions. Even if prediction is not a concern, defining the optimal segmentation without reference to k in a mathematically sound manner requires, in general, taking k to arbitrarily large values. This is because an HSSMM may allow the possibility of an arbitrarily quick transition with small probability. Therefore, even if the time series is short, Λ_{1000} may contain a segmentation with very rapid transitions that only covers half the data. The probability of 1,000 rapid transitions may be miniscule, but positive probability nonetheless, and if the time-series data is sufficiently ill-behaved or the shape recognizers sufficiently stringent, there is still a possibility that a segmentation with 1,000 rapid transitions may outperform all other 1,000-length segmentations. As k increases, these rare cases become less and less significant, but the point remains that for any finite k, all segmentations in Λ_k are not guaranteed to cover the time series[5]. It is for this reason that the mathematically sound rationale for defining the optimal segmentation is in terms of the limit as $k \to \infty$.

Pragmatically, however, we are not concerned with infinite-length segmentations. A more down-to-earth criteria is to find λ_k^* such that t_k in λ_k^* is beyond the final time stamp in the time series. Intuitively, shortening $\lambda_{k'}^*$, $k' > k$, to the first k transitions is likely to yield λ_k^*. Thus, we can formalize the segmentation task as follows:

Segmentation Task: Find $\lambda_k^* = \arg\max_{\lambda \in \Lambda_k} P(\lambda|X)$ for some k such that t_k in λ_k^* is greater than that time stamp of any data point in the time series X.

---

[4] When $d_{s_i}(X^v, t_i, t_{i+1})$ is really a probability density function, then integrating over all possible time series should yield the value 1.0. Such an integral would require a measure μ over the space of possible segmentation, but due to the fact that certain shape recognizers effectively partition possible segmentations into equivalence classes, an appropriate definition for μ is anything but obvious. If a particular measure were adopted, then enforcing the integration to 1.0 would greatly complicate implementation, and unnecessarily so, since there is really no reason that such a precise specification is necessary. However, there appears to be no problem in interpreting the ratios of $d_{s_i}(X, t_i, t_{i+1})/d_{s_j}(X, t_i, t_{i+1})$ as reflecting relatively probability densities.

[5] When there is a positive lower bound on the minimum possible transition time, then it would be possible to cast the optimization task in terms of the space Λ_k for some finite k.

The model-based segmentation task is now precisely defined as an optimization task. The space of possible segmentations is, however, enormous. For example, Λ_{100} is a 200-dimensional unbounded continuous space. Without additional structure, searching such a space would be entirely infeasible. However, due to the structure inherent in a HSSMM, the space of possible segmentations is highly structured, and it is possible to take tremendous advantage of this structure to perform the optimization. The following chapters develop an algorithm for performing this optimization.

2.4 Approaches to Segmentation (Literature Review)

There are many different ways to approach the time-series segmentation problem, with an endless variety of different possible algorithms. Many of these potential directions have been explored in existing literature, resulting in an enormous number of variations of algorithms for the segmentation problem. The time-series segmentation problem also appears under the labels of the *change-detection problem*, *detection of (abrupt) structural change in time-series*, *switching regressions*, and *detection of nonstationarity of time-series* (note that nonstationary time-series are more general than time-series whose changes in stationary properties is isolated to occasional abrupt points ([Lin and Teräsvirta, 1994])). It is also closely related to *intervention analysis* ([Box and Tiao, 1975], [Abraham, 1980]) where one tests for evidence of a change in the characteristics of a time-series at a known time point where an intervention took place (for example, when a new economic policy was adopted by the government). It differs from intervention analysis in that in segmentation analysis, the time of the potential transition is unknown.

Existing algorithms can be compared or classified among many possible dimensions. Two methods that seem to be radically different in most aspects may share a commonality in the technique used to solve one particular subproblem, and because a number of distinct subproblems tend to arise, there are a large number of ways to compare various techniques. An implementation may combine ideas from several techniques.

This section reviews some of the existing literature on time-series segmentation. A number of fundamental approaches are reviewed, although these are not always mutually exclusive. The review in this section is by no means comprehensive. It is intended simply to provide the reader with a broad picture of the various techniques and fundamental ideas that have received significant attention.

2.4.1 Sequential Sliding Window Techniques

The methods that would best be described as sequential sliding (or growing) window techniques comprise the most frequently used and most extensively studied techniques for time-series segmentation. There are a huge number of variations that fall within this category.

At the highest level, these algorithms function by considering a subset of the data, usually starting at time $t = 0$, and successively examining data up to a further and further horizon into the future. At each succession, the subset of data to that horizon is analyzed to detect whether a (single) transition has occurred. When it is decided that a transition has occurred within that horizon, the actual time of the transition, t^*, is then localized and the whole process starts over at the beginning as if t^* is now $t = 0$. Thus, one of the most distinguishing aspects of these techniques is that once they detect and localize a transition, they commit to that decision and continue from there.

In practice, algorithms that use these techniques often append additional techniques to alter previous commitments, for example by utilizing clustering methods to combine one or more successive time intervals found by a pure sequential technique into a single time interval ([Brandt, 1982]). The use of a post-clustering method suggests over-segmenting initially, preferring methods that err on the side of detecting extra transitions. Of course, the utilization of post-clustering methods is not unique to sequential windowing techniques.

Sequential techniques typically fall into one of two categories ([Basseville, 1980b], [Chu, 1995]): *residual-based* methods (a.k.a., one-window methods), and *two-window methods*. Residual-based methods require only a model of the signal before transition (such as linear), and then detect when the data deviate from that model. Two-window methods require two models — corresponding to before and after the transition — and use these in concert to detect the transition. A variant of two-window methods that is somewhat closer to residual-based methods in spirit leaves the second model unspecified, but detects deviations in shape by

```
// Time series to be segmented is X[m..n] = {(t_i, x_i) : i = m, ..., n and t_i < t_{i+1}}

    L = 2; // Line fitting has 2 free parameters.
    i = m;
    j = i + L;
    k = 1;
    while (j <= n) { // Evaluate Window X[i..j]

        ℓ = BestFitLine(X[i..j]);
        ε = AveResidualPerPoint(ℓ, X[i..j]);
        ++j;
        if (ε > threshold) {
            tr[k++] = i = LocalizeTransition(X[i..j]);
            j = i + L;
        }
    }
// Transition times are returned in array tr[1..k].
```

Figure 2.5: A simple residual-based algorithm to segment a time series into piecewise-linear segments

using errors in predictions on the second window based on the fit obtained to data in the first window. These are reviewed separately below as *prediction-based methods.*

Residual-Based Methods

As an example, Figure 2.5 describes a residual-based algorithm to segment a time series into piecewise-linear segments. The main idea is simply to grow a *single* window anchored at the beginning of the time series until a transition is found. At each window size, the residual of the best fit line is computed (`AveResidualPerPoint` is the cumulative residual divided by the number of points), and a transition is detected when the average residual per point over that interval exceeds a prespecified threshold. After a transition has been detected, the actual time of the transition must be localized since a large residual indicates only that the transition occurs somewhere within the window, but does not indicate exactly where the change occurs. Once a transition (also called a *change point* in the literature) is determined, the entire process is repeated using only the time-series data from the most recent change point forward (i.e., anchoring the left edge of the window to the most recently determined change point).

Almost all residual-based algorithms are variations on the algorithm in Figure 2.5; however, many of these variations have been studied in excruciating depth. I will review each of these variations in turn.

The first possible variation on residual-based methods is the choice of the within-segment process model. The algorithm in Figure 2.5 finds the best fit line, but of course `BestFitLine` can be replaced by virtually any process model. Normally a maximum likelihood criteria is used as the basis for fitting the free parameters of the model to the data (in the line fitting case, the free parameters were slope and intercept). Recall that the problem of fitting an order-p polynomial to data is an instance of linear regression since the independent variables can be taken to be $\mathbf{x} = \langle x, x^2, x^3, ..., x^p \rangle$ such that $y = \mathbf{m} \cdot \mathbf{x} + b$ (where $\mathbf{m}$ is a vector). Residual-based segmentation methods based on linear regression are developed or used in [Inselman and Arsenal, 1968, Brown *et al.*, 1975, Ferreira, 1975, Sen, 1983, Dufour, 1982]. Some of the most popular and powerful classes of process models in the modern literature are the *stable autoregressive moving average process models* ([Box and Jenkins, 1976, Kotz *et al.*, 1982, Makhoul, 1975, Lütkepohl, 1993]). Papers that develop residual-based segmentation algorithms based on the assumption that each segment behaves like a stationary stable autoregressive process include [Segen and Sanderson, 1980] and [Basseville, 1980b].

A second variation to the basic algorithm is that the average residual measure can be replaced by different

test statistics. Although a natural metric, mean residual per point has a number of drawbacks. Detecting a small change in mean requires a large window, but large windows cause detection of large changes in average residual to be delayed, requiring more data after the change before they can be detected ([Page, 1954]). Furthermore, it is difficult to assess the significance of a departure of this measure from zero ([Brown *et al.*, 1975, pg. 151]). This latter point is of interest since it is preferable to base the detection criteria on a significance level rather than some arbitrary (and hard to interpret) threshold. In fact, the problem of measuring significance, and of determining the power of a statistical test, has probably received more attention in the statistics and economic literature than any other aspect of time-series segmentation. To address these concerns, a number of different test statistics have been studied in the literature. In many cases, statistical significance tests and/or assessments of the power of the statistics have been derived.

Statisticians prefer to set a significance level, α, for the probability of making a Type I error, in this case, the probability of detecting a transition that is not really there (the null hypothesis being that the data is generated from a single process model), and then detecting transitions based on this significance level. This essentially amounts to using a more principled basis for setting the threshold in Figure 2.5. A survey of some tests appears in [Zacks, 1983]. Because "experience has shown that ... the plot of ordinary least squares residuals, or the plot of their squares, against time is not a very sensitive indicator of small or gradual changes in [slope]" [Brown *et al.*, 1975, pg. 151], it is standard practice to base significance tests on *cumulative sums* (cusums) of residuals or squared residuals ([Page, 1954], [Barnard, 1959]). However, the mathematical form of a cumulative sum of residuals (or cusum of squared residuals) is quite nasty and does not allow any straightforward derivation of standard statistical tests (although [Ploberger and Krämer, 1992] does develop a test based on ordinary residuals). For this reason, it is more common to use cusums of *recursive residuals*, where a squared recursive residual at time t is defined to be

$$w_t^2 = S_t - S_{t-1}$$

S_t is the cusum of squared residuals for the best fit line through the points up to time t ([Brown *et al.*, 1975]). When standard residuals are independent and distributed normally with mean zero and variance σ^2 (under the null hypothesis that all data is generated from a line), then the recursive residuals are also independent and distributed normally with mean zero and variance σ^2, and in addition, the recursive residuals have nice mathematical properties[6]. These have lead to reasonable statistical tests for detecting transitions ([Brown *et al.*, 1975, Bauer and Hackl, 1978, Sen, 1983]).

A third variation to the algorithm is the choice of method for localizing the time of a transition. This is often referred to as the *change-point localization problem*. The detection of a transition by a residual-based method does not provide much information about where the transition occurs other than that it occurs somewhere within the window being examined. Thus, after detection, it is usually necessary to localize the time of the transition. Typically, this is done by employing a two-window method (described below under "two-window methods"). For example, after detection that a transition occurs between $t = 1$ and $t = T$, [Brown *et al.*, 1975] apply Quandt's log-likelihood ratio ([Quandt, 1958], [Quandt, 1960]) to every pair of windows $([1..r], [r+1..T])$, as r varies from $1 + L$ to $T - L$ (where L is the number of points necessary to apply the metric and is based on the number of free parameters in the regression). The value of r that minimizes the metric taken as the transition time.

Finally, a fourth variation to the algorithm is the windowing technique. In the algorithm displayed in Figure 2.5, the window is anchored at the latest determined transition point and *grown* forward in time. The other popular variation is to *slide* a fixed-width window forward in time. An increasing window size can cause problems for metrics such as cumulative (or average) residual since large changes will have less effect as the window grows ([Page, 1954]). A fixed-width sliding window does not have this problem, but introduces the extra problem of selecting a window width. A sliding window must be smaller than the time scale of transitions, yet large enough to have the power to detect transitions.

The big advantage of residual-based methods is that the technique requires only a model of the behavior within a segment (i.e., before a transition) in order to detect a transition. In comparison, two-window methods also require a model of the signal behavior after the transition. The main drawback is a lack of power to detect certain kinds of changes. [Krämer *et al.*, 1988, pg. 1362] say: "If structural change is

[6] Specifically, the covariance between successive cusums has a manageable mathematical form.

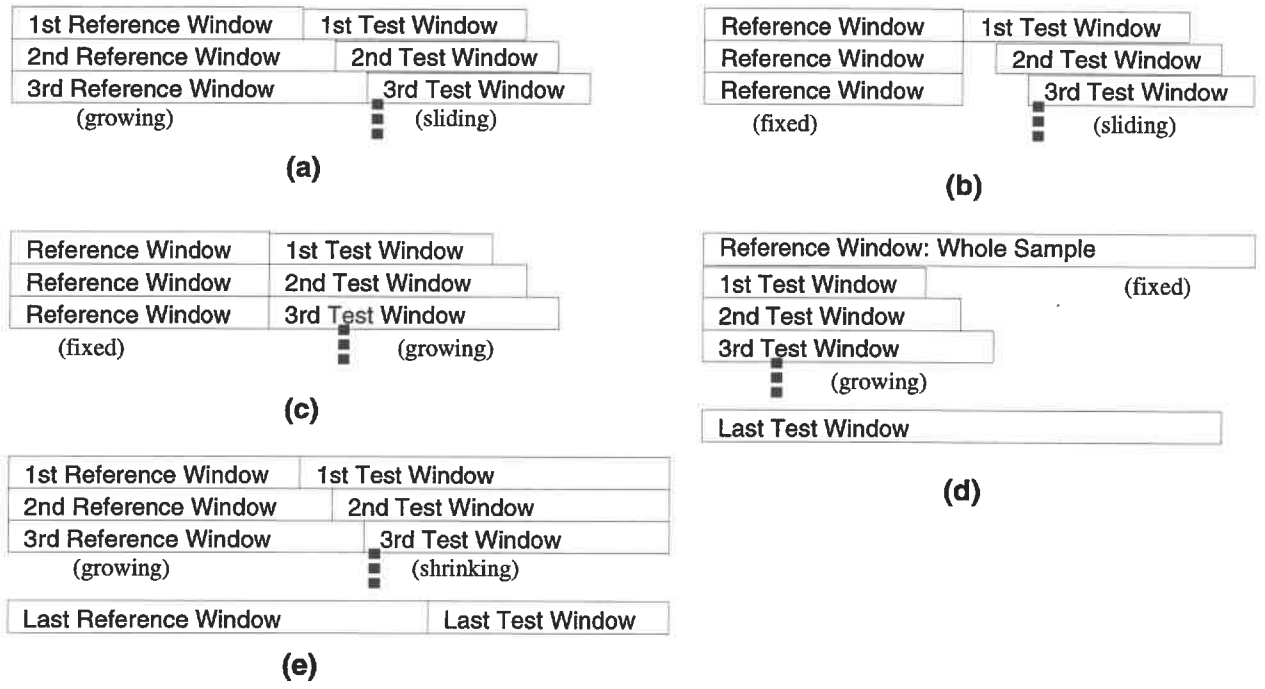


Figure 2.6: Methods for selecting reference and test windows in a two-window method. Diagram taken in part from [Chu, 1995].

orthogonal to the regressors or occurs rather late in the sample period, no version of the cusum test will detect it." In general, additional modeling information, when available, has the potential to increase the power of segmentation methods.

Two-Window Methods

Two-window methods are generally more powerful than residual-based methods. The extra power comes at the cost of requiring two models: A *reference model* for the signal before the transition, and a *test model* for the signal after the transition. The methods operate by measuring the differences between the two models, i.e., comparing the probability the data was produced from a single reference model versus the probability the data was produced by the first model up to time r, and then by a second model from then on.

Two-window methods operate by maintaining two windows, called the *reference window* and the *test window*, over the time-series. The methods then consider the hypothesis that the data is generated by the reference model inside the reference window and is then generated by the test model inside the test window.

Two fundamental issues underlie two-window methods: how to select the reference and test windows, and how to measure the differences between the two models.

There are several reasonable ways to select the reference and test windows. For example, [Appel and Brandt, 1983], [Basseville and Benveniste, 1983], and [Chu, 1995] grow the reference window, as was done with the window in Figure 2.5, and then append a fixed-width test window at the end of the reference window. As the reference window grows, the test window slides. This is depicted in Figure 2.6(a). Alternatively, [Bodenstein and Praetorius, 1977, Ishii *et al.*, 1979, Ishii *et al.*, 1980] and [Chu, 1995][7] use a fixed width stationary reference window and slide a fixed-width test window forward in time as in Figure 2.6(b). [Appel and Brandt, 1983] also utilize the scheme in Figure 2.6(c) where the reference window is fixed and stationary and the test window is grown. This is used by [Appel and Brandt, 1983] in iteration with that of Figure 2.6(b) only during their change point localization optimization. However, the combination could be used in a

[7] [Chu, 1995] analyzes both of the techniques shown in Figure 2.6(a) and 2.6(b).

spirit closer to that of a residual-based method, where the test window plays the role of the residual-based method's window, but where the ability to detect a deviation is enhanced by the information obtained from within the test window. The scheme depicted in Figure 2.6(d) is used by [Ploberger *et al.*, 1989]. Finally, Figure 2.6(e) depicts the configuration where the reference window is successively grown while the test window is simultaneously shrunk. As was briefly mentioned in the previous subsection, [Brown *et al.*, 1975] use this scheme to perform the change-point localization. In fact, this is the most natural scheme for change-point localization, and even techniques (e.g., [Appel and Brandt, 1983]) that use one of the other window management techniques to detect a transition may make use of this scheme to localize the time of the transition. In addition to localization uses, [Deshayes and Picard, 1986] and [Andrews, 1993] use this scheme in a straight two-windows approach.

Two-window sequential segmentation methods initially anchor the reference window to $t = 0$ and grow/slide the reference and test windows around using one of the schemes of Figure 2.6 until a transition is detected. After a transition detected, the time of the transition is localized (note that in the schemes of Figure 2.6(b-d), the time of the transition is not obvious), and the anchor is moved to this new location and the whole process repeated. Again, a single transition is detected at a time, and once found, sequential techniques permanently commit to that transition time. It should also be pointed out that the window management techniques of Figure 2.6(a-c) are readily applicable to an implementation where segmentation occurs while data arrives incrementally, while techniques 2.6(d-e) are only applicable to a batch scenario where all the data is available in advance (they are also quite useful for change-point localization, as noted above).

Once window positions have been selected, the difference between the models in the two windows must be measured. There is a plethora of possible measures that have been proposed for this task. The earliest was the Quandt's log-likelihood ratio ([Quandt, 1958], [Quandt, 1960]). First, a measure of the data fit assuming that no transition occurs is obtained by fitting the reference model to the data in both windows. Next, a measure of the data fit assuming that the data in the reference window was produced by the reference, while the data in the test window was produced by the test process, is obtained by fitting the reference model to the data in (only) the reference window, and fitting the test model to the data in the test window. Quandt's criteria takes the log of the ratio between the probability of the observations given the single model to the probability of the observations given two models. A small value of this metric indicates a transition. The distribution, and therefore a statistical test for this metric is derived in [Andrews, 1993]. This full procedure is repeated for every possible transition time considered. [Appel and Brandt, 1983] generalize this to autoregressive processes. The likelihood-ratio is very computationally intensive ([Basseville, 1980a]), and problems may arise near the edges when one window contains very little data ([Deshayes and Picard, 1986],[Basseville, 1980b]). A wide variety of other statistics have been proposed, in many cases in an attempt to approximate the likelihood ratio with less computation, but in some cases based on other motivations ([Bodenstein and Praetorius, 1977, Sen, 1980, Basseville and Benveniste, 1983, Hawkins, 1987, Ishii *et al.*, 1979, Ploberger *et al.*, 1989, Andrews, 1993]). A review of tests for changes in the parameters of linear regression, along with a categorization of the types of changes that can be detected and the appropriate corresponding statistical tests, is given in [Pesaran *et al.*, 1985].

As was the case with residual-based methods, the choice of model class for each window arises with two-window methods as well. For example, using linear models (fitting best fit lines in each window) an algorithm can detect a change in the slope of the data (e.g., [Sen, 1980]). However, the Autoregressive Moving Average (ARMA) models again are among the most popular. Notable papers that develop or use two-window ARMA models include [Basseville, 1980b, Appel and Brandt, 1983, Chu, 1995].

Prediction-Based Methods

Another sequential two-window approach is to fit a model in a reference window, and then compare the predictions made by that model over a test window. Like two-window techniques, this requires the management of both a reference and a test window, but like the residual-based methods, only one model is fit. [Pesaran *et al.*, 1985] reviews prediction-based statistical tests for detecting changes in linear regression models. Methods for autoregressive and ARMA models are developed in [Box and Tiao, 1976, Lütkepohl, 1988, Lütkepohl, 1989], [Lütkepohl, 1993, Sections 4.6 and 11.4.3], and references therein. [Lütkepohl, 1988, pg. 268] claims that "one advantage of this approach is that only a few data points are needed after the time

point or period where the [transition] is suspected. ... Another practical advantage is that no hypothesis is required on the precise form of structural change." However, "changes in the model of different kinds may not be easily distinguishable" [Box and Tiao, 1976], and in reality assumptions about the nature of the change are required. For example, a change in variance will go undetected if one is only comparing the mean error between forecast and data. Statistical tests exist for detecting a change in variance ([Pesaran *et al.*, 1985]), but the choice to invoke such a test implicitly assumes something about the nature of the change. Therefore, it is reasonable to view prediction-based models as a special case of two-window methods.

2.4.2 Transition Recognition Methods

The window-based techniques of the previous section detect a transition by comparing the goodness of fit of data within one or more intervals of time. An alternative is to directly recognize salient characteristics of transition points without paying much attention to the uniformity of data characteristics between transitions.

Often the local extrema and inflection points in the graph of a function correspond to our own intuitions about where abrupt transitions seem to occur. There are points where the first or second derivative of a function with respect to time are zero. Therefore, one approach is to find time points where the first or second derivative of the data with respect to time is zero. Generally this requires smoothing the data first.

Another approach to recognize transitions directly is to train a neural net to recognize transitions as a function of a window of data surround a possible transition, or in the case of a recurrent neural net, as a function of some signal history. Again, the salient characteristic is that the recognition of the transition is based on the shape of the signal at the transition (e.g., bend/kink recognition, etc) rather than on a change in the uniformity of some signal parameter over a time interval.

Domain-Dependent Transitions

A system for segmenting an acoustic waveform of spoken digits so that digits can then be individually recognized is developed in [Rabiner and Sambur, 1976]. They note a number of characteristics specific to this domain ([*ibid* p.171–2]):

1. "An interval of unvoiced speech or silence ... denotes the beginning or end of a digit, i.e., there are no internal unvoiced or silent regions within the 10 digits (0-9)."

2. "Local minima of the energy contour ... are strong indications of digit boundaries."

They train a classifier to classify each 10 millisecond interval as either voiced, unvoiced, or silence. Then based on these labels, the design a number of rules to recognize digit boundaries directly. In this application, success was possible because of domain-dependent properties of transitions that were identified.

Scale-Space Filtering

Scale-space filtering ([Witkin, 1983]) is a technique to recognize and classify inflection points at different time resolutions. Generally, a signal must be smoothed before it is possible to identify inflection points. This may be because data points occur only at discrete times, so that smoothing has the effect of turning the time-series into a continuous time series, or because discontinuities exist in the underlying continuous-time function. Smoothing of data is generally accomplished by convolving the time series with a Gaussian signal. The amount of smoothing that is actually obtained depends on the variance of the Gaussian used. A larger variance results in a smoother signal with fewer inflection points.

Smoothing introduces two effects: qualitative simplification (e.g., the removal of inflection points) and spatial distortion (shifts in the actual time-position of inflection points). The technique of scale-space filtering (a) prioritizes inflection points — those that appear at coarser smoothings are considered to be more significant transition points, and (b) localizes the actual time of a transition (removes shifts that result from smoothing). The original paper, [Witkin, 1983], is a very good exposition of the technique.

Neural-Net Recognition

In some domains, a reasonably precise description of the features of a transition may be possible, as is the case in the above examples. However, in other domains, even though a transition may have highly salient features, knowing exactly how to describe these features may be difficult. For these, one approach is to train a time-delay neural network to recognize the salient features that define a transition. A window of time-series data is fed into the neural network, and the network outputs 1 if it is a transition, 0 otherwise (or perhaps a grade between 0 and 1 if it is uncertain).

In the digit recognition task of [Rabiner and Sambur, 1976], some parameters of the feature recognizer were tuned from training data. In this sense, that system already resembles such a system. However, the transition characteristics there were largely already defined and the parameter adjustment was more of a fine tuning.

In speech recognition, certain stop constant-phonemes provide a certain natural transition point. These are phonemes such as `/b d g k p t/`, usually consisting of closure, explosion, and aspiration phases. Not all phoneme boundaries are delineated by stop-constants, but for some tasks these may represent a reasonable marking of transitions within a stream of continuous speech. Time-delay neural networks have been trained to recognize stop-constant phonemes by [Hampshire and Waibel, 1990, Waibel *et al.*, 1989].

Although the basic idea is rather obvious, other examples of neural-bet transition-recognition methods seem to be rare in the literature. This may simply be an underexplored area, or it may be an indication that domains with salient transition features are uncommon.

2.4.3 Clustering and Labeling Techniques

Labeling techniques provide a fairly crude method for detecting transitions and obtaining very rough estimates or constraints on the transition times. They are often used in combination with other techniques, for example to simplify the problem the other technique has to solve.

The idea behind labeling techniques is simply to abstract the signal from a real-valued time series into a string of labels. The time-series is partitioned into fixed-sized windows, and then each window is classified and assigned a label. A change in label may indicate a transition, or some combination of rules based on the label assignments can be used to identify transitions (see Section 2.4.6). However, the precise time of the transition is typically under-specified since it is usually only known to occur somewhere within the two windows surrounding a label change.

The two issues behind labeling techniques are how to use the labels to perform the segmentation, and how to produce or learn to produce the labels. Learning to label segments of time series is an unsupervised learning problem.

One method for labeling a signal is to use a hierarchical clustering algorithm. [Lee and Chou, 1989] compute the dispersion of an ARMA process within each window, and then apply a hierarchical clustering algorithm to assign labels. A change in labels indicates a change point and provides rough constraints on the time of the change point. They then apply a dynamic programming algorithm (Section 2.4.5) to precisely localize the transition times. The technique is applied in [Lee and Chou, 1990] to segment phonocardiograms (PCGs) for use in the diagnosis of heart ailments.

2.4.4 Gated Experts

Approaches in this category are based on various competitive neural-net architectures where subnetworks compete for segments of the signal. Once a mapping from subnetworks to data points is established, this mapping partitions the time series into segments. This type of approach is in many ways an example of a clustering or labelling technique, but the methods used appear somewhat different to other variations of clustering.

Competitive neural net training is employed by [Pawelzik *et al.*, 1996] to learn signal shapes that can be used to assign labels to windows in a labeling scheme. Using the assumption that transitions occur rarely, so that adjacent data points belong with high probability to the same class, they are able to train a radial basis net of the type studied by [Moody and Darken, 1989] so that distinct neurons respond to distinct signal shapes. A similar approach is taken by [Kohlmorgen *et al.*, 1994] and [Weigend *et al.*, 1995] using an adaptive

mixture of local experts network ([Jacobs *et al.*, 1991], also called *measure fields* by [Marroquin, 1995]). In this approach, the neural network is competitively trained to map each data point of the time-series to a subnetwork. When applied to segmentation ([Kohlmorgen *et al.*, 1994]), a bias is necessary to encourage the assignment of adjacent points to the same subnetwork.

2.4.5 Dynamic Programming Methods

Instead of detecting transitions one at a time and then committing to those choices, as is done with sequential segmentation techniques, dynamic programming approaches offer reasonable methods for optimizing over all transitions in a time series simultaneously. The technique as applied to time-series segmentation appears earliest in [Bellman and Roth, 1969].

Suppose $g(t)$ is a piece-wise stationary signal consisting of N segments, where $g_i(t)$ is the segment from time t_{i-1} to t_i, $i = 1, .., N$, and where each linear segment has been fit to the data within that segment. It is desired to find the best-fit piece-wise stationary signal from some class of piece-wise stationary signals over the time interval $[t_0..t_N]$, where t_0 and t_N are given, and t_i for $0 < i < N$ are to be determined.

Let $error(g, t_i, t_j)$ be a measure of the difference between $g(t)$ and the actual time series between times t_i and t_j. This might be the cumulative residual, cumulative squared residual, or the sum of maximum deviation on each segment, for example. Assume that whatever the error function, it is obtained by summing the error on each individual segment, so that

$$error(g, t_0, t_i) = error(g, t_0, t_{i-1}) + error(g_i, t_{i-1}, t_i)$$

In the spirit of dynamic programming, let $F_i(t_i)$ be the error of the best fit i-segment curve between time t_0 and time t_i with the final endpoint of the i^{th} segment occurring at time t_i. F_i can be written as the recurrence

$$F_i(t_i) = \min_{0<t_{i-1}<t_i} \left[F_{i-1}(t_{i-1}) + \min_{g_i} error(g_i, t_{i-1}, t_i) \right] \tag{2.5}$$

The dynamic programming algorithm begins by computing $F_1(t_1)$ for all possible values of t_1 by simply fitting the curve to the data in $[t_0..t_1]$. Once F_1 is computed, then $F_2(t_2)$ is computed for all possible values of t_2 using (2.5). This is repeated until finally $F_N(t_N)$ is computed, but in the final case, it need only be computed for the known t_N.

To implement this algorithm, time must be discretized. Once discretized, $F_i(t_i)$ can be stored as an array where t_i takes on only a finite number of possible values. [Bellman and Roth, 1969] impose a uniformly spaced grid.

The algorithm as shown does not impose the requirement that the fitted curve be continuous at transition points. This extra condition is accommodated by including the y value of the curve at each transition point in $F_i(t_i, y_i)$, the error of the best fit i-segment curve ending at the point (t_i, y_i). $F_i(t_i, y_i)$ is now a two-dimensional array, and the y dimension must also be discretized. The algorithm is given in [Bellman and Roth, 1969].

Many variations of the dynamic programming approach are possible. First of all, many variations on the error measure used above are easily accommodated. Other techniques can be applied as a preprocessing step to narrow the range of possible transition times and/or the set of possible within-segment shapes in order to streamline the dynamic programming optimization step. For example, [Lee and Chou, 1989] apply a hierarchical clustering method (see Section 2.4.3) to identify the number of change points, constrain their locations, and determine the best fit segment. This essentially eliminates the minimization in (2.5) over g_i and reduces the range for t_i to a small (two-window) region.

The basic dynamic programming algorithm above assumes the number of segments is known. In most situations, the number of segments is not known and must also be estimated. Estimating the number of segments is complicated by an over-fitting problem — a greater number of segments always results in a better data fit, while a good segmentation keeps the number of segments to a minimum. A minimum description length principle ([Rissanen, 1978], [Rissanen, 1986]) can be easily incorporated into the dynamic programming algorithm to determine a good number of segments. One simply computes F_i up to a sufficiently large i, and then applies the MDL criterion to each i to find the best length. An approach using a measure very similar to Rissanen's information measure to choose the number of segments in a piece-wise polynomial

regression problem appears in [Brailovsky, 1992]. The Vapnik-Chervonenkis dimension of the piece-wise polynomial regression problem is derived in [Brailovsky and Kempner, 1992].

Dynamic programming is applied to a Bayesian formulation of the problem in [Djurič *et al.*, 1992], where a prior distribution (essentially a uniform prior over all k-length segmentations) is specified, and a goodness of fit given a particular segmentation has the form of a product of data fit on each individual segment (note that this is the form in Equation (2.1)). They are able to express the maximum a posteriori parse in a (very complex) dynamic programming form. The paper uses an autoregressive model for each segment. The Bayesian formulation is similar to the formulation in this thesis, but the prior used in that work is not adjustable. [Ostendorf and Roukos, 1989] segment speech to phonemes using dynamic programming on a fairly simple Bayesian formulation.

2.4.6 Model-Driven Methods

Model-driven methods take as input, in addition to the time-series data, a formal model describing the evolution of the time series or the process underlying it. A segmentation algorithm then attempts to find the interpretation (i.e., segmentation) with maximal agreement to the model. Model-based approaches overlap many of the already discussed literature, and often deciding whether a given work should be called model-based is a grey area. Almost every algorithm has parameters that must be set, and sometimes reasonable arguments can be made that the parameters settings used are modeling a specific process even when it might not be immediately obvious from the outset. The most distinctive characteristic of a model-based method is, therefore, not so much in the end result, but rather, in the approach used to get there. In these approaches one typically concentrates on encoding (or learning) domain-specific knowledge about the time series or its underlying process prior to applying the algorithm to actual time-series segmentation tasks of interest.

Several model-based methods utilize probabilistic models, including the methods of this thesis. The specific form of the probabilistic may vary considerably, but generally these models implicitly encode a distribution over the possible segmentations given time-series data, and the algorithm's task is to find the most probable segmentation given the data (called the maximum a posteriori (MAP) segmentation).

A good example of a probabilistic approach is [Sclove, 1983]. In this paper, a time-series is modeled as a Hidden Markov Model (HMM). At any moment, a time-series is considered to be in one of n possible states. Time is modeled discretely, such that at synchronous points in time, transitions between states occur according to prespecified transition probabilities. Typically there is a rather high probability of staying in the same state. At each synchronous instant, a data point for the time-series is generated according to a distribution conditioned only on the current state. With a HMM model of this form, a Viterbi algorithm ([Forney, 1973]) can be applied to find the most likely segmentation. (Note that the Viterbi algorithm is a dynamic programming algorithm, highlighting a connection to the dynamic programming methods reviewed in Section 2.4.5.) Those transitions from a state to itself are filtered out, and all transitions where the state actually changes are the points where transitions occur.

The time-series segmentation formulation in [Sclove, 1983] bears a distinct similarity to speech recognition methods based on HMMs (see e.g., [Rabiner, 1989]). Clearly, segmentation (into phonemes or words) is an intrinsic part of speech recognition, so this is not surprising. With such an connection, many of the speech recognition methods could also be considered model-based segmentation methods. However, due to the size of that literature, I do not review that literature from that perspective here. Many of the relevant references are already cited elsewhere in this thesis in specific contexts.

The use of an HMM-based method has certain limitations. The model itself encodes some very strong assumptions about the nature of the time-series process — most specifically, that the process is Markovian. Individual data points within the time series are independent of each other given the underlying state. This means that even when two points occur within the same state, there is little opportunity to model additional regularity between those points. Also, transitions are Markovian, so that state durations can only be modeled as geometric (i.e., exponential) distributions. This is again quite limiting in many cases. The latter limitation has been addressed by a number of works in speech recognition, for example, by utilizing Hidden semi-Markov models (HSMMs). A good review of these methods is [Guedon, 1992].

The HSSMM introduced in this chapter is yet a further generalization of HSMMs, where the segmented observation process eliminates the limitation that points within the same state be independent given the

state.

Non-probabilistic model-based approaches are possible as well, although due to the abductive nature of segmentation, there can be conceptual difficulties that don't arise with a probabilistic MAP approach. Waveform parsing systems ([Cox *et al.*, 1972, Stockman *et al.*, 1976, Stockman, 1982]) are one such example of a non-probabilistic model-based method. These approaches even have roots in the HEARSAY speech recognition system [Reddy *et al.*, 1973]. In waveform parsing systems, the time-series is described by a Chomsky-style grammar. The waveform itself must be preprocessed to convert the series into a symbolic series, i.e., a string, with each "letter" denoting a localized shape such as `CUP`, `CAP` or `S-TURN` [Stockman, 1982]. Segmentation is then a parsing task. Although waveform parsing systems have been pursued quite seriously, it does have several obvious downsides. First, the technique itself does not address the potentially difficult step of symbolizing the time series (which itself could be considered a segmentation task in some cases). Second, ambiguities (multiple possible parses) are possible, but the technique itself provides no basis for choosing between these. And finally, noise in the process or the symbolization step can create significant difficulties.

2.5 Summary

Time-series segmentation is the task of finding points in time where abrupt qualitative changes occur in a signal. The model-based approach developed in this chapter assumes that expectations in the form of knowledge about states, durations, transitions, signal shapes within a state, and associated uncertainties in all these are known in advance. This knowledge about the time series is encoded as a Hidden Segmented Semi-Markov Model, a formal time-series model developed in Section 2.2. The segmented observation part of the HSSMM is novel, while the transition portion utilizes a semi-Markov model. The segmentation task becomes one of finding the most probable segmentation given the time-series data. Subsequent chapters consider the algorithms for finding this most probable segmentation.

Chapter 3

Structural Decomposition

Chapter 2 formulated the time-series segmentation problem as an optimization task over the space of possible segmentations. Although enormous, the space of possible segmentations is highly structured. By utilizing this structure the search task can be decomposed into several smaller search tasks, and thus greatly simplified.

The ability to take advantage of structure is key to most large inference problems. The methodology used in this chapter to decompose the time series segmentation problem is quite general and can be applied similarly to a wide variety of inference and estimation problems to obtain, in a very systematic fashion, an efficient algorithm from a given problem formulation. It is often even more useful for guiding the formulation of a problem in the first place, since many of the tradeoffs between structural assumptions and efficiency are made explicit.

The method in this chapter is rooted in the theory of graphical Markov fields ([Lauritzen *et al.*, 1984], [Lauritzen and Spiegelhalter, 1988]). The theory is not new to this thesis and has been applied to many applications in the literature ([Heckerman *et al.*, 1995], [Noormohammadian and Oppel, 1993]). However, the generality of the theory is often under-appreciated. For example, most existing statistical estimation and inference algorithms that utilize conditional independence to simplify a large problem, but otherwise solve the problem exactly, fall out automatically as special cases by applying the methodology to their specific problem formulations ([Smyth *et al.*, 1996, Levy *et al.*, 1996]). While most such algorithms have been developed without (explicit) recourse to graphical Markov field theory, there is little doubt that familiarity with and use of the theory would have, in most cases, expedited the development of the algorithms by reducing the reliance on creative inspiration. In short, it is a very useful methodology to have in one's toolkit.

Section 3.1 demonstrates the methodology by decomposing the problem of finding the optimal k-length segmentation for a given k. It is also shown in Section 3.1.3 how to decompose the computation of marginal posterior probabilities, such as computing the posterior probability distribution for the time of the fourth transition, or the probability that $s_5 = \textit{heatup}$ (s_5 is the state between the fifth and sixth transitions). Section 3.2 then discusses the methodology for structural decomposition in greater generality. Finally, leveraging structure is a very powerful and important idea, but it is not always, by itself, sufficient, nor is it the only source of power one should turn to when solving a problem. The structural decomposition of the HSSMM is a huge win, but it is not enough by itself to make the search task feasible. Section 3.3 considers this point, a problem which is addressed more comprehensively in Chapter 4.

3.1 Decomposing the HSSMM.

This section decomposes the task of finding an optimal k-length segmentation, λ_k^*, given an HSSMM and time-series data, X. Equation (2.4) identifies the optimal k-length segmentation as

$$\lambda_k^* = \arg\max_{\lambda \in \Lambda} P(\lambda|X) \tag{3.1}$$

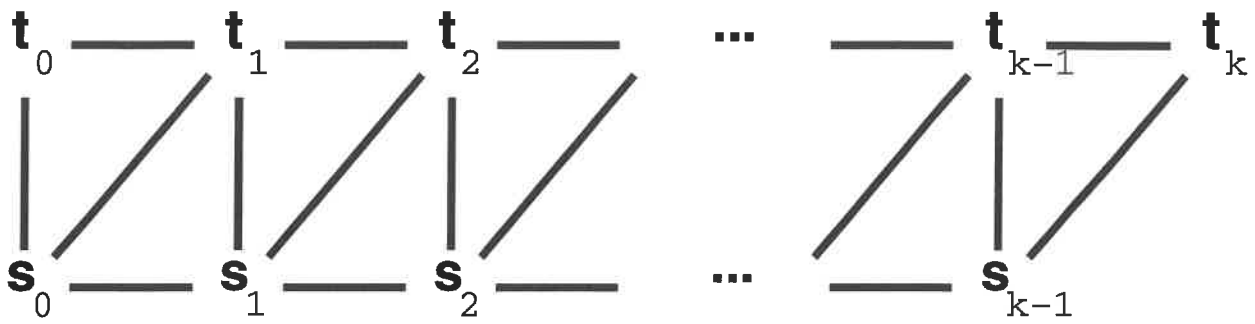


Figure 3.1: Dependencies between the random variables of a segmentation.

where $\lambda = \langle t_0, s_0, ..., s_{k-1}, t_k \rangle$, $t_0 = 0$, and from Equation (2.3),

$$P(\lambda|X) = \alpha b_{s_0} \prod_{i=0}^{k-1} a_{s_i,s_{i+1}} \sigma(c_{s_i}) c_{s_i}(t_{i+1} - t_i) \prod_v d_{s_i}(X^v, t_i, t_{i+1}) \tag{3.2}$$

where $\alpha = \alpha(X)$ is a constant when X is given.

Notice that the evaluation function in (3.2) is a product of terms where each term involves only a small number of variables. For example, the term $c_{s_3}(t_4 - t_3)$ involves only three random variables: t_3, s_3, t_4, as does $d_{s_3}(X^1, t_3, t_4)$. This is a significant source of structure and is a first indication that there may be structure in the problem that can be efficiently utilized. However, it is also necessary to examine the interdependencies between the variables.

In Figure 3.1, the random variables in this problem are shown with an edge between two variables if both appear together in a single term of (3.2). For example, since s_1 and s_2 appear together in a_{s_1,s_2}, there is an edge between s_1 and s_2. Because t_1, s_1, and t_2 appear in $c_{s_1}(t_2 - t_1)$, there are edges between all pairs of t_1, s_1, and t_2. Figure 3.1 represents the dependencies between the variables in a segmentation. Structure exists because the variables are not totally interconnected.

Figure 3.1 depicts a *graphical Markov field.* Suppose the function $P(\lambda|X)$ is given as in (3.2), and suppose also that somehow the time of the second transition, t_2, and the state just prior to that transition, s_1, are revealed. Then learning anything additional about the time of the first transition, t_1, does not provide any additional information about s_2, the state just following t_2. This property is referred to as the *Markov property*, and hence the term *graphical Markov field.* In general, if a set of nodes, **C**, blocks all paths between node sets **A** and **B**, and the values for all variables in **C** are given, then further knowledge about the variables in **A** can provide no further information about the variables in **B**. A Markov field makes *independencies* of this form explicit in a visual, graphical form. Graphical Markov properties are studied in depth in [Lauritzen *et al.*, 1990] and [Frydenberg, 1990].

It is also possible to view the dependencies of (3.2) in terms of a directed graph. Directed dependency graphs are often intuitively appealing when the direction of arrows corresponds with some notion of causality, whereby the parents of a node are considered to be that node's direct "causes" ([Pearl, 1988]). It is possible to convert Equation (3.2) into a directed graph as follows:

1. Impose any total ordering on the random variables.

2. For each term in (3.2), insert a directed arc between all pairs of variables in the term, with the direction determined by the ordering.

For any ordering, a directed acyclic graph results, but few of these would be considered "causal", since for example, most contain arrows pointing backwards in time (e.g., an arc from t_i to t_j where $j < i$). By using, for example, the ordering where t_i precedes s_i and s_i precedes t_{i+1}, and by taking X^v as a constant (not a random variable, therefore it does not appear in the graph), the directed dependency graph shown in

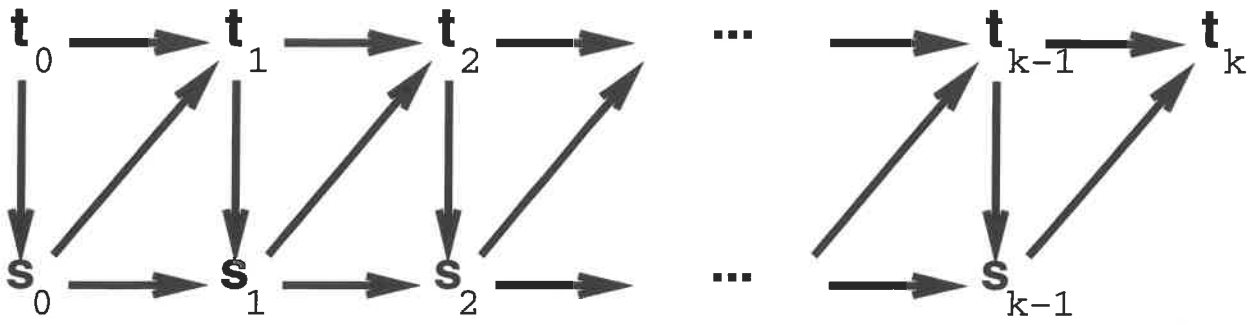


Figure 3.2: Directed dependencies between the variables of a segmentation.

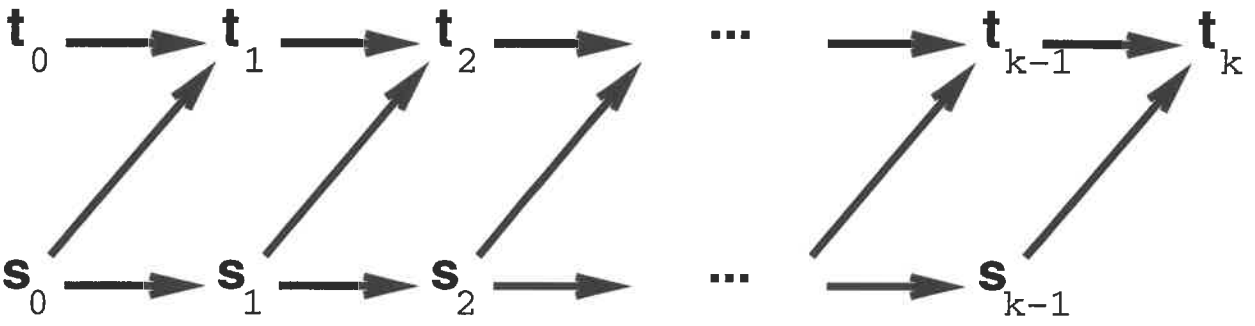


Figure 3.3: Directed dependencies for $P(\lambda)$ with no time series data considered.

Figure 3.2 results. By picking other possible orderings that agree with the time order, the direction any of the arcs between s_i and t_i can be reversed.

The need to include arrows between s_i and t_i arises from the data term in (3.2). If we consider just $P(\lambda)$, the probability of a segmentation with no data given, these arrows can be omitted since $c_{s_i}(t_{i+1} - t_i)$ can be equivalently expressed as a conditional probability distribution $P(t_{i+1}|t_i, s_i) = c_{s_i}(t_{i+1} - t_i)$. When a term is a conditional probability distribution, it is only necessary to include arcs from each variable on the right hand side of the bar to the variable on the left hand side of the bar — in this case, an arc from t_i to t_{i+1} and an arc from s_i to t_{i+1}. Thus, the dependencies for $P(\lambda)$ alone, as given by Equation (2.2), is shown in Figure 3.3. The graph shows that without data, t_i and s_i are independent given s_{i-1}, while this is not the case when time-series data is given (Figure 3.2).

We can also treat $d(X|\lambda)$ as a conditional probability distribution and explicitly include a node in the graph corresponding to the observed data (X), as shown in Figure 3.4. This makes it possible to draw dependencies as directed without including the arrows between s_i and t_i. However, Figure 3.4 introduces a number of dependencies that are not present in this problem. Because the data distribution is a product form (2.1), the data dependencies can be broken down further. This is depicted in Figure 3.5. The notation X_i^v denotes the data between t_i and t_{i+1} for sensor v.

The various dependency graphs provide various alternative ways of viewing the structure of the problem. These can provide a model designer with a visual representation that can be of great assistance in understanding the limitations of a model, and help in spotting alterations that can have significant computational advantages. The literature on graphical probabilistic models gives an enormous amount of attention to the question of what independence properties are explicitly encoded by graphical dependency structures. For the remaining purposes in this thesis, these issues are irrelevant, but it is interesting to examine exactly what independence relationships are encoded by these various dependency graphs. All three of the dependency graphs considered above (Figures 3.5, 3.2, and 3.1) all encode precisely the same conditional independence

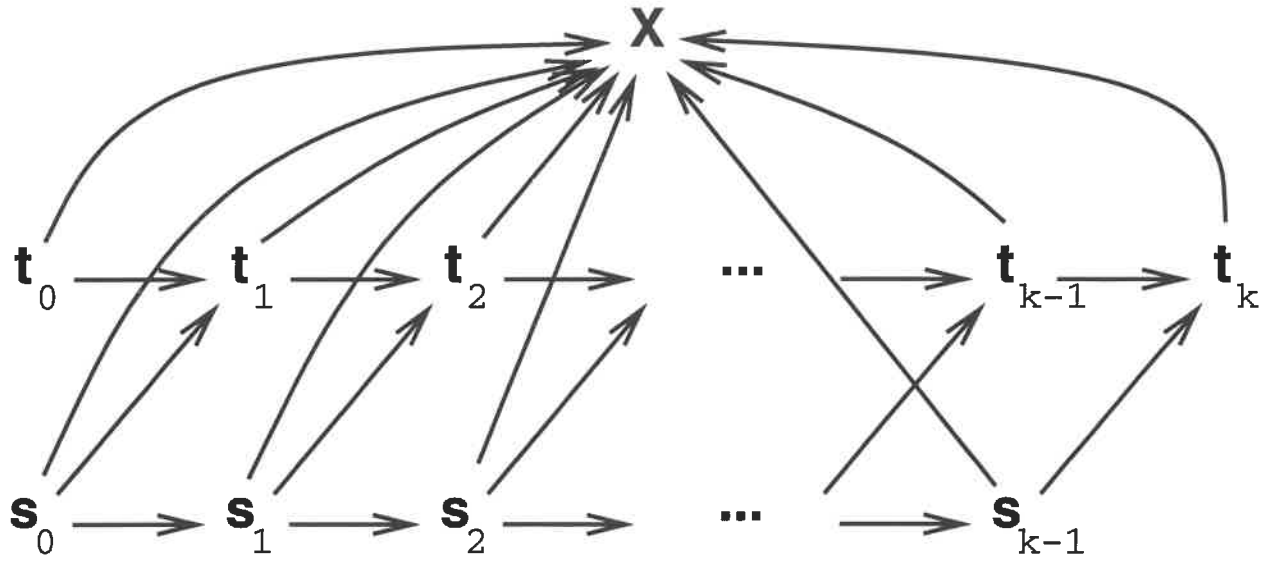


Figure 3.4: Directed dependencies including an observed data node, X.

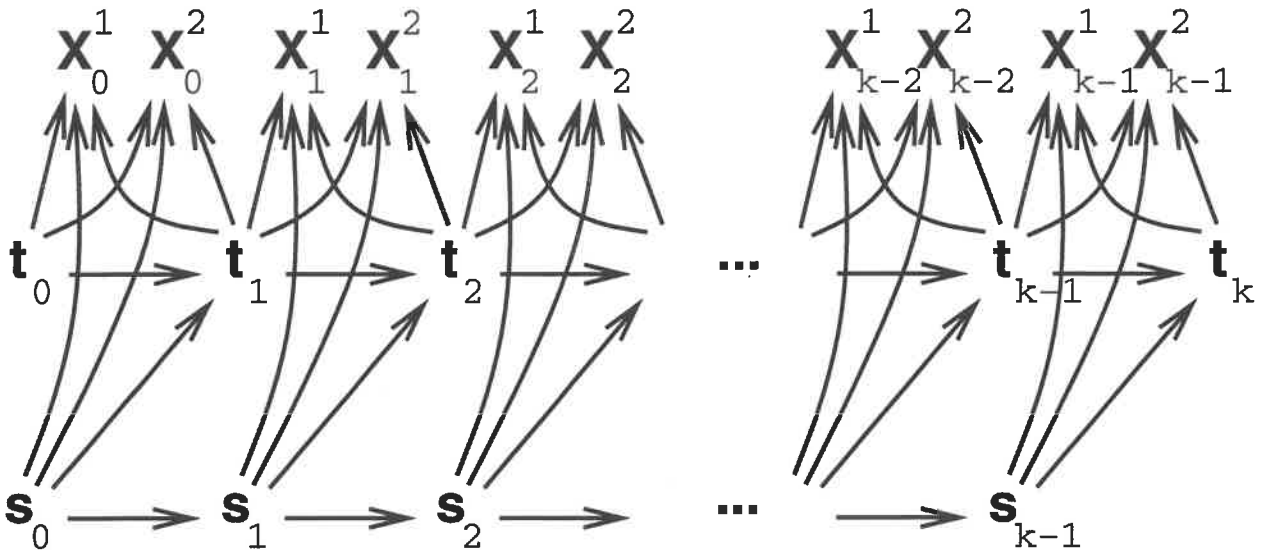


Figure 3.5: Directed dependencies between the variables of a segmentation and the data. Note that all the X_i^v variables are observed (given).

Rejection Sampling

A second approach is rejection sampling. This approach is applicable when a distribution $g(x)$ is available that approximates $f(x)$ and from which independent samples can be efficiently drawn. A value $c > 0$ is assumed, and the following procedure is used to generate a sample $x \sim f$:

```
loop
        x ~ g
        u ~ U                    U = uniform dist. on [0,1].
until cug(x) ≤ f(x)
return x
```

The sample x returned is distributed according to $\alpha \min\{f(x), cg(x)\}$, where α is a normalization factor. If c is chosen so that $f(x) \leq cg(x)$ for all x, then the sample generated is distributed according to f. The existence of a c such that $f \leq cg$ basically requires g to have heavier tails and sharper infinite peaks than f. The smaller c is, the more efficient the procedure is (i.e., fewer samples are rejected), so it is very important for g to be a good approximation to f. It is often useful to tradeoff the fidelity of the approach by lowering c to where $f \leq cg$ does not hold everywhere in order to obtain computational efficiency. This is especially reasonable when g would serve as a reasonable approximation to f in the application considered. In general, this tradeoff must be evaluated on a case by case basis. Since our f is already a subjective estimate, some distortion in the interest of efficiency is often acceptable.

Product Form Rejection Sampling

One variation of rejection sampling is useful when f is in a product form. Suppose $f(x) = cg(x)h(x)$, where g is a pdf, h is a nonnegative function such as a pdf, and c is a (typically unknown) normalization constant. Let $\hat{h}$ be a value such that $\hat{h} \geq h(x)$ for all x. Then the following procedure generates $x \sim f$:

```
loop
        x ~ g
        u ~ U
until uĥ ≤ h(x)
return x
```

The efficiency of this procedure depends crucially on the shape of $h(x)$ relative to g, and on how close $\hat{h}$ is to $\sup_x h(x)$. For example, if 99.9% of the area under $h(x)$ occurs in a region where $g(x)$ is nearly zero, then less than one point per thousand will be accepted. It can become extremely inefficient when the modes of g and h are very disparate, or when h is very spiky (i.e., nearly zero everywhere except for a few narrow spikes). However, when used with care, this technique can be very useful.

Mixture Sampling

Another technique for generating random variates, which is of great importance for the methods in this chapter, can be applied when f is expressed as a mixture of densities and random variates can be efficiently generated from each component density. Let

$$f(x) = \sum_i c_i g_i(x) \qquad \sum_i c_i = 1$$

Pick a value i with probability c_i, then sample $x \sim g_i$. It follows that $x \sim f$. Therefore, it is possible to sample from mixture distributions very efficiently. For comparison, the inversion method is incredibly inefficient in this case.

4.4 Estimating f

This section considers the problem of estimating f.

Given a full continuous probabilistic model, any given value for $\mathbf{x}$ either is or is not the value occuring in the optimum MAP configuration. Therefore, the knowledge available entails that the probability of $\mathbf{x}$ being the optimum is everywhere either zero or one. This is to emphasize that $f(x)$, the probability (density) that $\mathbf{x}$ is the value in the optimum MAP configuration, must be a subjective estimate formed with limited resources based only on the computation that has been completed thus far. It is not a probability distribution entailed by the available knowledge. By the nature of it being subjective, there is no single correct way to construct $f(x)$ from the partial computations completed to this point. The section examines a number of possible schemes for constructing $f(x)$. Once constructed, $\mathbf{x}$ can be discretized based on $f(x)$ as already discussed in Section 4.2.

It is important to note that $f(x)$ is a continuous distribution defined for all real-values of $\mathbf{x}$. Intermediate computations are performed by discretizing all continuous variables and computing probabilities as if the discrete values are the only possible values. Therefore, the information available for constructing $f(x)$ is largely discrete, but this must be used to construct a continuous $f(x)$. This is accomplished in various ways by the various methods discussed below.

Throughout this section, continuous variables appear without hats, and their discretized counterparts appear with hats. For example, $\hat{\mathbf{x}}$ is the discretized version of the continuous variable $\mathbf{x}$, and $\Omega_{\hat{\mathbf{x}}}$ is the finite set of possible values for $\hat{\mathbf{x}}$.

Section 4.4.1 begins with a relatively simple method for estimating f based on a variable's parents' discrete posteriors (the estimate is denoted f_{pp}). It is included for two reasons. First, it is perhaps the simplest and most straightforward approach that is possible. This provides a point of comparison for motivating more sophisticated methods for obtaining f. Second, it provides the reader with a simple starting point for understanding how an estimate for f might be obtained.

Since it would seem to be the simplest approach, one might consider how the algorithm would perform if the values were either selected from a uniform distribution or if they were spaced uniformly (i.e., at equal distances, which is equivalent to using the area partitioning method on a uniform distribution). Näively the uniform distribution would seem to be simplest and most obvious choice for f; however, this choice creates more problems than it solves. The t_i's (the random variables) are unbounded variables. It is not possible to specify a (proper) uniform distribution on the real-line, so using a uniform distribution requires bounds on the possible values for t_i. Doing so places an upper bound on the greatest waiting time, yet no such upper bound exists in the HSSMM in general. For example, the tail of a gamma distribution has positive probability all the way to infinity. Different HSSMMs may run at different time scales, so any method of choosing the boundaries for a uniform distribution would have to examine the waiting-time distribution in the HSSMM. Once this step is taken, it is much simpler to use the waiting-time distribution directly, rather than develop some artificial procedure for extracting a reasonable upper bound from the distributions. Since this is what the parents' posteriors method does, it is far more reasonable to consider the parents' posterior method to be the simplest and most straightforward technique.

Section 4.4.2 introduces a better technique that utilizes the posteriors from all the neighbors of a variable (the variable's Markov boundary). The estimate produced from this technique is denoted f_{mbp}. It is based on the simple idea that we can equate f to be the variable's (estimated) posterior probability. This is not how f is defined, but it is perhaps the most natural heuristic to try. Quite a few sophisticated techniques are developed in Section 4.4.2 to make f_{mbp} usable, and together these form the bread and butter of the approach. From there, a number of minor variations are also explored.

4.4.1 Using Parents' Posteriors

Suppose some discretization has previously been chosen for all variables in the model, and the marginal posteriors computed using the nonparametric propagation algorithm of Chapter 3. The means by which this iterative process is bootstrapped is considered later in Section 4.5. Based on the information available from the posteriors of the parents of $\mathbf{x}$, the variable $\mathbf{x}$ is to be (re)discretized. $\mathbf{x}$ has parents $\mathbf{y}_1$ and $\mathbf{y}_2$, connectively denoted as just $\mathbf{y}$, as shown in Figure 4.3.

Denote the *discrete* values of $\mathbf{y}_1$ and $\mathbf{y}_2$ by $\Omega_{\hat{\mathbf{y}}} = \Omega_{\hat{\mathbf{y}}_1} \times \Omega_{\hat{\mathbf{y}}_1}$. The hat denotes the discretized version of the continuous variable. Then $\Omega_{\hat{\mathbf{y}}}$ is a set with a finite number of elements, and as a result of propagation, $Pr(\hat{y}|data) = Pr(\hat{y}_1, \hat{y}_2|data)$ is readily available, i.e., the joint marginal posterior probability over $\Omega_{\hat{\mathbf{y}}}$ given

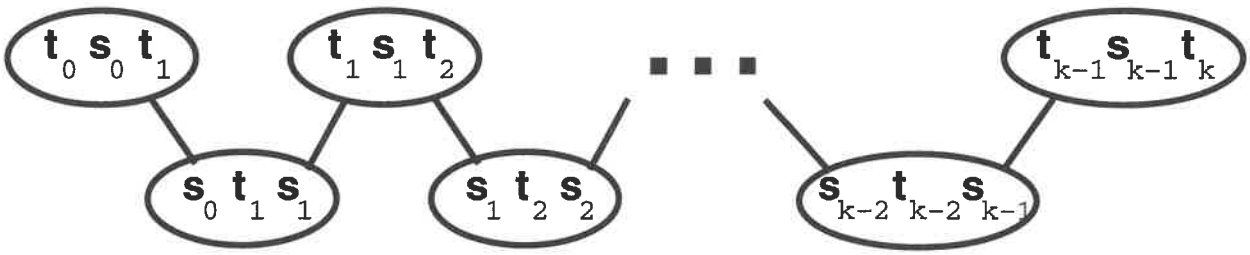


Figure 3.6: The junction tree for the segmentation problem.

relationships (assuming that time-series data is always given). Even when time-series data is not given, as in Figure 3.3, the computational situation is the same since all existing algorithms for solving directed probabilistic networks would add a dependency between t_i and s_i anyway (a result of t_i and s_i being in each other's Markov blanket [Pearl, 1988, pg. 97]). It is the author's opinion that for this problem, the undirected dependency graph is the most natural expression of (in)dependencies, the easiest to interpret, and the easiest to deal with.

While dependency graphs illustrate the structure of a problem, for the purposes of computation, the graphical representation shown in Figure 3.6 is more useful. This representation is called a *junction tree* ([Jensen *et al.*, 1990b], [Jensen and Jensen, 1994], others), *cluster tree* ([Shachter *et al.*, 1994], [Draper, 1995]), or *Markov tree* ([Shenoy and Shafer, 1986]). Nodes in Figure 3.6 represent subsets of variables, such that the graph has the following properties:

1. It is an undirected tree (no loops).
2. Any subset of variables appearing together in a single term of (3.2) also appear together in some node of the junction tree.
3. For any two nodes with vertex subsets $\mathbf{A}$ and $\mathbf{B}$, all nodes on the path between the two nodes contains all the variables in $\mathbf{A} \cap \mathbf{B}$.

The third property is called the *junction-tree property.*

It is usually more direct to write the dependencies in a problem first in the form of Figure 3.1. Transforming Figure 3.1 to Figure 3.6 can then be done mechanically, as discussed below. Because the motivations here are purely computational, Figure 3.6 is the graph of interest (for exact methods). The use of the junction tree was pioneered by [Lauritzen and Spiegelhalter, 1988] and [Jensen *et al.*, 1990a].

The significance of Figure 3.6 is that it decomposes the huge $2k$-dimensional search problem into $2k$ related 3-dimensional search problems. It is far more tractable to solve 200 3-dimensional optimizations than it is to solve one 200-dimensional optimization task. The reduction is possible because of the structure identified in Figure 3.1. To perform the optimization, each node of Figure 3.6 communicates local information about its optima with its neighbors and incorporates local information from its neighbors into its own optimization task. The effect is the propagation of information down the tree and back.

Triangulating Dependency Graphs

The problem of going from a dependency graph to a junction tree is a well understood problem [Lauritzen and Spiegelhalter, 1988]. Three steps are involved (each explained subsequently):

1. If there are directed dependencies, construct the *moral graph.*

   The moral graph is undirected.

2. Triangulate the (undirected) graph.

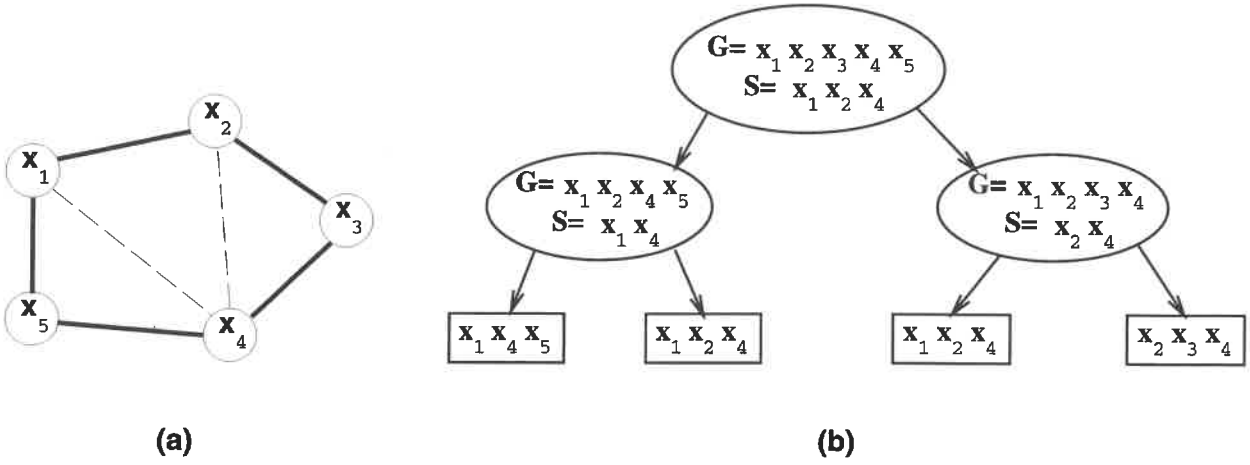


Figure 3.7: (a) A nontriangulated graph (without dashed edges) and a possible triangulation (with dashed edges). (b) A recursive decomposition.

3. Extract the junction tree from the triangulated graph.

The purpose of Step 1 is to find an undirected graph encoding all the dependencies in the original graph ([Kelly, 1988]). In general, it will encode additional dependencies as well (i.e., it will not encode certain independencies). The moral graph provides an easy way of finding such a graph. If the dependency graph is already undirected, this step is skipped.

The *moral graph* is obtained from the original directed acyclic dependency graph by inserting edges between all nodes sharing a common child, then replacing all directed edges by undirected ones. If the dependency graph contains directed and undirected edges, but has no cycles containing one or more directed edges, then it is called a *chain graph* [Frydenberg, 1990], and any nodes are married that have children belonging to the same *connected component* (i.e., when there is an undirected path between the children). See [Frydenberg, 1990] for more on chain graphs. A moral graph is undirected. Note that the moralization step is quite trivial.

Figure 3.1 shows the moral graph for both the dependency graphs in Figures 3.2 and 3.3. In Figure 3.2, the parents of each node in the graph are already connected, so moralization is accomplished simply by dropping the direction of edges. In Figure 3.3, the parents of t_i must be "married" by adding an edge between t_{i-1} and s_{i-1}, after which, the direction of arrows are dropped.

An undirected graph is *triangulated* whenever every cycle of length four contains a *chord*, i.e., an edge between two nonadjacent vertices in the cycle ([Golumbic, 1980]). The short circuit is sometimes called a *chord* and the term *chordal* is synonymous for triangulated. The graph in Figure 3.1 is triangulated.

If a graph is not triangulated (an adjective), it can always be triangulated (a verb) by adding the appropriate (undirected) edges. Even though this is not necessary in Figure 3.1, it is necessary to complete a triangulation step in general, and so I discuss the process in detail here. Each time an edge is added, a new dependency is introduced (equivalently, an independency is removed). Since structure is a source of computational power, it is desirable to minimize the number of edges added and to carefully select which edges are added. For example, without the dashed edges, the graph in Figure 3.7(a) is not triangulated, but the graph containing the dashed edges is triangulated. If edges from x_2—x_5 or x_3—x_5 are added, the graph is still triangulated, but these are unnecessary and reduce computational efficiency.

One way of finding a good triangulation is to recursively decompose the dependency graph ([Golumbic, 1980, Cooper, 1990a]). Finding a triangulation is then equivalent to finding a decomposition (Figure 3.7(b)). An undirected graph is *decomposable* when it is complete, or when there exists two sets of vertices, $\mathbf{A}$ and $\mathbf{B}$, such that $\mathcal{V} = \mathbf{A} \cup \mathbf{B}$, $\mathbf{A} \cap \mathbf{B}$ is complete[1], and $\mathbf{A} \cap \mathbf{B}$ separates $\mathbf{A}$ from $\mathbf{B}$ ([Golumbic, 1980,

[1] $\mathbf{A} \cap \mathbf{B}$ is complete when the nodes in $\mathbf{A} \cap \mathbf{B}$ are totally connected.

Lauritzen *et al.*, 1984, Dawid and Lauritzen, 1993]). A decomposition is proper if $\mathbf{A}, \mathbf{B} \neq \mathcal{V}$. A classical result ([Golumbic, 1980]) is that an undirected graph is decomposable if and only if it is triangulated. Thus, triangulation is equivalent to choosing a recursive decomposition. A recursive decomposition of $\mathcal{G}$ is a binary tree, with each branch being a recursive decomposition subgraphs $\mathcal{G}_{\mathbf{A}}$ and $\mathcal{G}_{\mathbf{B}}$ respectively. The leaves of a recursive decomposition are the cliques of the triangulated graph. The triangulation is obtained by simply connecting any two nodes belonging to a common leaf of the recursive decomposition.

A second way to find a triangulation is by way of a node peeling order. A *perfect numbering* is a numbering of the nodes of an undirected graph, $\{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n\}$, such that $neighbors(\mathbf{x}_i) \cap \{\mathbf{x}_{i+1}, ..., \mathbf{x}_n\}$ is complete. A graph *admits a perfect numbering* if there exists a perfect numbering of its vertices. In such a graph, one node at a time can be peeled off the graph with the property that when a node is removed, all its remaining neighbors are totally connected. Another classical result is that a graph is triangulated if and only if it admits a perfect numbering. This also highlights the importance of a triangulation — because a peeled node's neighbors are totally connected, they can absorb the information from the node (marginalized to the remaining subgraph). To triangulate an untriangulated graph, one can simply specify an ordering for the nodes — this ordering will be the perfect numbering. Then, simply remove the first node and add in any edges required to connect the node's neighbors in the remaining graph. Then remove the second node, connect its neighbors, etc. The edges added during this process are the triangulation edges (the dashed edges in Figure 3.7, and they can then be added to the original graph. For the graph in Figure 3.7, a node ordering that results in the given triangulation is x_5, x_1, x_4, x_3, x_2.

An optimal triangulation is one that minimizes the computational complexity of propagation (discussed next in Section 3.1.1). Roughly, this amounts to finding a triangulation that results in the minimal-sized cliques. When variables are discrete and finite-valued, the optimal triangulation must also consider the number of assignments that each clique can take on, so that instead of simply minimizing the number of variables in the largest clique, the optimal triangulation should minimize the total number of assignments that any clique can take on. However, finding an optimal triangulation is NP-hard [Arnborg *et al.*, 1987], so one must turn to heuristic techniques for locating a triangulation. Some heuristics have been compared in [Kjaerulff, 1990].

In some cases, a person may be designing a formalism for a particular application, in which case it can be helpful to consider how dependencies will be triangulated. Removal of certain problematic dependencies may dramatically reduce the complexity of a resulting triangulation. For example, if any one of the (undashed) dependencies in Figure 3.7(a) could be eliminated, the graph would be triangulated with no clique having more than two variables. Conversely, there may be certain candidate dependencies in the formalism that the engineer might be worried about adding, since additional dependencies may increase computational complexity. However, if these dependencies must be added anyway in order to triangulate the graph, then these dependencies will not create an additional computational cost. In this way, an engineer has guidance as to which features of a formalism come with a high computational price, and which can be added without a computational penalty. Consider, for example, the option of using holding-time distributions (Page 31) instead of waiting-time distributions in the HSSMM formalism. Would this generalization increase the computational complexity of finding a solution? The question can be answered by considering the impact on the triangulated graph. A holding-time distribution renders t_i dependent on t_{i-1}, s_{i-1}, and s_i. This amounts to the graph in Figure 3.2 but with the arrows from $t_i \rightarrow s_i$ reversed (i.e., from $s_i \rightarrow t_i$), which in turn yields the triangulated moral graph in Figure 3.8. As compared with Figure 3.1, this graph contains larger cliques (of size 4), and thus results in an increase in required computation.

After an undirected triangulated dependency graph is obtained, a junction tree for the graph can be quickly constructed [Jensen and Jensen, 1994]. The cliques of the graph become the nodes of the junction tree. The edges connecting cliques of the junction tree must be specified in a way that ensures the junction-tree property — i.e., that the path between any two nodes in the junction tree contains the intersection of the two nodes at every step (Page 51). This can be ensured by adding the largest separators first in a greedy fashion (see [Jensen and Jensen, 1994]). Unlike the task of finding a triangulation, the extraction of a junction tree from an already triangulated graph can always be done efficiently (i.e., in polynomial time). For Figure 3.1, the linear structure of the graph yields only one possible junction tree, so this step is especially trivial.

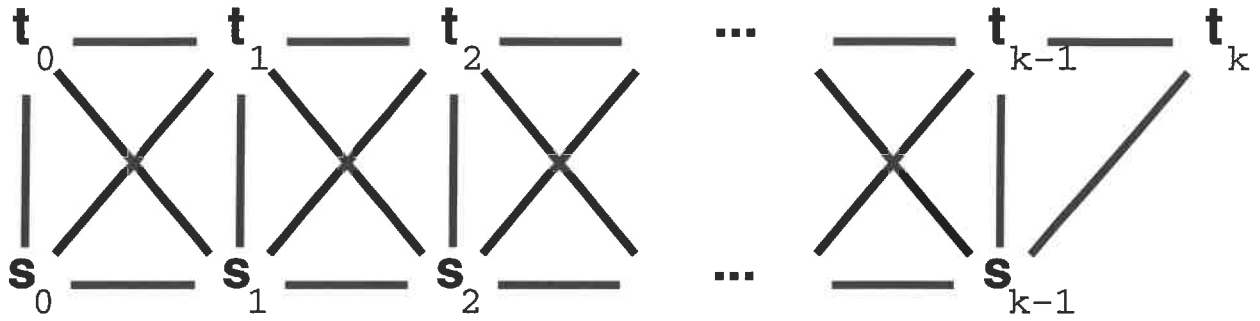


Figure 3.8: Moralization of dependencies introduced by including waiting-time distributions in the HSSMM.

3.1.1 Propagation

Figure 3.6 decomposes the huge $2k$-dimensional optimization task into $2k$ related 3-dimensional search problems. The single global evaluation function of (3.2) is decomposed into several local evaluation functions called *potentials* — one for each node and one for each edge in Figure 3.6. Each node passes information about its potential function to its neighbors, and whenever a node receives a message from a neighbor, it modifies its potential function accordingly. This process of message passing is called *propagation*. After propagation converges, each individual node in Figure 3.6 can be optimized locally such that when put together, the global maximum is obtained.

Denote the potentials of each node of Figure 3.6 as $\psi_{t_i,s_i,t_{i+1}}$ and $\psi_{s_i,t_{i+1},s_{i+1}}$. For each edge, define potentials $\vartheta_{s_i,t_{i+1}}$ and ϑ_{t_i,s_i}. For example, $\psi_{t_1,s_1,t_2}(t_1,s_1,t_2)$ is a function mapping the times of the first and second transitions and the state between those times to a real number. It is notationally convenient to write $\psi(t_1,s_1,t_2)$, omitting the subscripts since the arguments make it clear which function is being referred to. At each step during the propagation, each node in Figure 3.6 has exactly one potential function (ψ) associated with it. Similarly, each edge has also exactly one potential (ϑ). Initially set

$$\begin{aligned}
\psi(t_i,s_i,t_{i+1}) &= \sigma(c_{s_i})c_{s_i}(t_{i+1}-t_i)\prod_v d_{s_i}(X^v,t_i,t_{i+1}) \\
\psi(s_i,t_i,s_{i+1}) &= a_{s_i,s_{i+1}} \qquad\qquad i>0 \\
\psi(s_0,t_0,s_1) &= b_{s_0}a_{s_0,s_1} \\
\vartheta(\cdot) &= 1
\end{aligned} \tag{3.3}$$

Note that with this initialization,

$$P(\lambda|X) \propto \frac{\prod\psi}{\prod\vartheta} \tag{3.4}$$

where $\prod\psi = \psi(t_0,s_0,t_1)\prod_{i=1}^{k-1}\psi(s_{i-1},t_i,s_i)\psi(t_i,s_i,t_{i+1})$, $\prod\vartheta = \vartheta(s_i,t_{i+1})\vartheta(t_i,s_i)$, and $\frac{0}{0}$ is taken to be 0.

To begin the propagation, the node containing $\{t_0,s_0,t_1\}$ propagates information to its neighboring node, $\{s_0,t_1,s_1\}$, as follows:

$$\begin{aligned}
\vartheta'(s_0,t_1) &= \frac{1}{\mu}\max_{t_0}\psi(t_0,s_0,t_1) \\
\psi'(s_0,t_1,s_1) &= \frac{\psi(s_0,t_1,s_1)\vartheta'(s_0,t_1)}{\vartheta(s_0,t_1)}
\end{aligned}$$

with $\frac{0}{0}=0$. These new potentials, $\vartheta'(s_0,t_1)$ and $\psi'(t_0,s_0,t_1)$, replace the corresponding previous potentials.

The constant μ used in the propagation step is arbitrary and can be any positive value. The reason for

introducing it is to prevent numerical underflow problems in an actual implementation. It is useful to use

$$\mu = \sum_{s_0,t_1} \vartheta'(s_0,t_1)$$

This keeps the magnitudes of the values of a potential function within a reasonable range so that the numbers do not underflow by enforcing that ψ' is always normalized. The μ factor was introduced by [Jensen, 1994]. The same trick, termed scaling, was introduced into the related Baum alpha-beta procedure by [Levinson *et al.*, 1983] (see also [Rabiner, 1989] and Devijver's algorithm [Devijver, 1985]), and is also the distinguishing component of Derin's algorithm [Askar and Derin, 1981].

The propagation from any other node to a neighboring node occurs in exactly the same manner as the above. Once the potentials are initialized, the propagation process simply starts from the leftmost node in Figure 3.6 and each successive node propagates to the neighbor on its right. When the rightmost neighbor is reached, the process reverses: The rightmost node propagates to its left neighbor, it propagates to its left, and so on, until the leftmost node is reached. After this completes, the resulting potential functions are guaranteed to converge such that any additional propagations at that point would not alter the potentials ([Dawid, 1992]). The full propagation algorithm is written out in Figure 3.9. When the propagation completes, we have

$$\begin{aligned}\psi(t_i,s_i,t_{i+1}) &\propto \max_{\lambda}\{P(\lambda|X) : \lambda[\mathbf{t}_i,\mathbf{s}_i,\mathbf{t}_{i+1}] = \langle t_i,s_i,t_{i+1}\rangle\}\\ \psi(s_i,t_{i+1},s_{i+1}) &\propto \max_{\lambda}\{P(\lambda|X) : \lambda[\mathbf{s}_i,\mathbf{t}_{i+1},\mathbf{s}_{i+1}] = \langle s_i,t_{i+1},s_{i+1}\rangle\}\end{aligned} \tag{3.5}$$

3.1.2 Local Optimization

Once the propagation completes, each potential is truly local — that is, a global optimization can be obtained by optimizing each individual node in Figure 3.6 individually since each maximum in (3.5) corresponds to the global optimum. If there is only one optimal segmentation (in other words, there are not two or more segmentations that are tied), then each local frame will have a unique optimum. We can find the values t_0, s_0, and t_1 for the optimum segmentation by locating the optimum value in ψ_{t_0,s_0,t_1}. The remaining variables can be determined by optimizing the other local frames. If there is more than one optimum segmentation, then the multiple maxima at the various frames must be matched in the obvious fashion to obtain a valid global optimum. For example, after finding a maximum for t_0, s_0, and t_1, the potential ψ_{s_0,t_1,s_1} is optimized subject to the given values for s_0 and t_1, and so on. The result is an HSSMM-specific variation of the Viterbi algorithm ([Forney, 1973]).

3.1.3 Computing Marginal Distributions

When $P(\lambda|X)$ in Equation (3.2) is interpreted as a probability distribution, it is often useful to compute marginal posterior distributions for individual variables of the segmentation. For example,

$$P(t_1|X) = \int_{\Lambda_{\mathbf{t}_1}} P(\lambda|X)d\lambda \qquad \text{where } \Lambda_{\mathbf{t}_1} = \{\lambda : \lambda[\mathbf{t_1}] = t_1\}$$

is the posterior distribution over the possible times for the first transition,

$$P(s_0|X) = \int_{\Lambda_{\mathbf{s}_0}} P(\lambda|X)d\lambda \qquad \text{where } \Lambda_{\mathbf{s}_0} = \{\lambda : \lambda[\mathbf{s_0}] = s_0\}$$

is the probability distribution over the possible starting states, etc. Such estimations are often used in EM-statistical learning algorithms ([Dempster *et al.*, 1977, Rabiner, 1989]) and are also utilized in Chapter 4 for discretizing continuous variables.

The same methodology that was used to decompose the optimization problem can also be applied to decompose the computation of marginal distributions. The potentials for each node and edge in Figure 3.6

// Rightward propagation:

For $i = 1$ **to** $k-1$ {

$\mu = \sum_{s_{i-1},t_i} \max_{t_{i-1}} \psi(t_{i-1}, s_{i-1}, t_i)$

$\vartheta'(s_{i-1}, t_i) = \frac{1}{\mu} \max_{t_{i-1}} \psi(t_{i-1}, s_{i-1}, t_i)$

$\psi'(s_{i-1}, t_i, s_i) = \frac{\psi(s_{i-1},t_i,s_i)\vartheta'(s_{i-1},t_i)}{\vartheta(s_{i-1},t_i)}$ (3.6)

Replace ψ by ψ' and ϑ by ϑ'.

$\mu = \sum_{t_i,s_i} \max_{s_{i-1}} \psi(s_{i-1}, t_i, s_i)$ [Note that ψ is ψ' from (3.6).]

$\vartheta'(t_i, s_i) = \frac{1}{\mu} \max_{s_{i-1}} \psi'(s_{i-1}, t_i, s_i)$

$\psi'(t_i, s_i, t_{i+1}) = \frac{\psi(t_i,s_i,t_{i+1})\vartheta'(t_i,s_i)}{\vartheta(t_i,s_i)}$

Replace ψ by ψ' and ϑ by ϑ'.

}

// Leftward propagation:

For $i = k-1$ **downto** 1 {

$\mu = \sum_{t_i,s_i} \max_{t_{i+1}} \psi(t_i, s_i, t_{i+1})$

$\vartheta'(t_i, s_i) = \frac{1}{\mu} \max_{t_{i+1}} \psi(t_i, s_i, t_{i+1})$

$\psi'(s_{i-1}, t_i, s_i) = \frac{\psi(s_{i-1},t_i,s_i)\vartheta'(t_i,s_i)}{\vartheta(t_i,s_i)}$

Replace ψ by ψ' and ϑ by ϑ'.

$\mu = \sum_{s_{i-1},t_i} \max_{s_i} \psi(s_{i-1}, t_i, s_i)$

$\vartheta(s_{i-1}, t_i) = \frac{1}{\mu} \max_{s_i} \psi(s_{i-1}, t_i, s_i)$

$\psi(t_{i-1}, s_{i-1}, t_i) = \frac{\psi(t_{i-1},s_{i-1},t_i)\vartheta'(s_{i-1},t_i)}{\vartheta(s_{i-1},t_i)}$

Replace ψ by ψ' and ϑ by ϑ'.

}

Figure 3.9: Propagation of potentials for optimization.

// Rightward propagation:
For $i = 1$ **to** $k-1$ {

$$\mu = \sum_{t_{i-1}, s_{i-1}, t_i} \psi(t_{i-1}, s_{i-1}, t_i)$$
$$\vartheta'(s_{i-1}, t_i) = \frac{1}{\mu} \sum_{t_{i-1}} \psi(t_{i-1}, s_{i-1}, t_i)$$
$$\psi'(s_{i-1}, t_i, s_i) = \frac{\psi(s_{i-1}, t_i, s_i)\vartheta'(s_{i-1}, t_i)}{\vartheta(s_{i-1}, t_i)} \qquad (3.7)$$

Replace ψ by ψ' and ϑ by ϑ'.

$$\mu = \sum_{t_i, s_i} \sum_{s_{i-1}} \psi(s_{i-1}, t_i, s_i)$$ [Note that ψ is ψ' from (3.7).]
$$\vartheta'(t_i, s_i) = \frac{1}{\mu} \sum_{s_{i-1}} \psi'(s_{i-1}, t_i, s_i)$$
$$\psi'(t_i, s_i, t_{i+1}) = \frac{\psi(t_i, s_i, t_{i+1})\vartheta'(t_i, s_i)}{\vartheta(t_i, s_i)}$$

Replace ψ by ψ' and ϑ by ϑ'.

}
Normalize $\psi(t_{k-1}, s_{k-1}, t_k)$.
// Leftward propagation:
For $i = k-1$ **downto** 1 {

$$\mu = \sum_{t_i, s_i, t_{i+1}} \psi(t_i, s_i, t_{i+1})$$
$$\vartheta'(t_i, s_i) = \frac{1}{\mu} \sum_{t_{i+1}} \psi(t_i, s_i, t_{i+1})$$
$$\psi'(s_{i-1}, t_i, s_i) = \frac{\psi(s_{i-1}, t_i, s_i)\vartheta'(t_i, s_i)}{\vartheta(t_i, s_i)}$$

Replace ψ by ψ' and ϑ by ϑ'.

$$\mu = \sum_{s_{i-1}, t_i} \sum_{s_i} \psi(s_{i-1}, t_i, s_i)$$
$$\vartheta(s_{i-1}, t_i) = \frac{1}{\mu} \sum_{s_i} \psi(s_{i-1}, t_i, s_i)$$
$$\psi(t_{i-1}, s_{i-1}, t_i) = \frac{\psi(t_{i-1}, s_{i-1}, t_i)\vartheta'(s_{i-1}, t_i)}{\vartheta(s_{i-1}, t_i)}$$

Replace ψ by ψ' and ϑ by ϑ'.

}

Figure 3.10: Propagation for the computation of marginal posterior distributions.

are initialized as specified in (3.3), and the potentials are propagated as before except that edge potentials are updated using summation rather than maximization, e.g.:

$$\vartheta'(s_i, t_{i+1}) = \frac{1}{\mu} \sum_{t_i} \psi(t_i, s_i, t_{i+1})$$

The full propagation is written out in Figure 3.10. It is easy to verify that each propagation step preserves the invariant of (3.4), i.e.,

$$P(\lambda|X) \propto \frac{\prod \psi}{\prod \vartheta}$$

so that the *relative evaluation* of any two segmentations is never altered by a propagation step. Even the constant of proportionality is preserved by each propagation step.

Because the invariant is preserved, immediately following the *CollectEvidence* (i.e., the rightward propagation) step, the potential $\psi(t_{k-1}, s_{k-1}, t_k)$ differs from $P(t_{k-1}, s_{k-1}, t_k)$ only by the constant factor α.

Thus, this is a convenient time to evaluate the normalization constant:

$$\alpha^{-1} = \int \psi(t_{k-1}, s_{k-1}, t_k) dt_{k-1}\ ds_{k-1}\ dt_k$$

Indeed, this is the same α that appears in Equation (3.2). Having computed α, the potential $\psi(t_{k-1}, s_{k-1}, t_k)$ can be replaced by

$$\psi'(t_{k-1}, s_{k-1}, t_k) = \alpha \cdot \psi(t_{k-1}, s_{k-1}, t_k)$$

With this substitution (and dropping the prime), the new invariant

$$P(\lambda|X) = \frac{\prod \psi}{\prod \vartheta}$$

is established and preserved by the *DistributeEvidence* (i.e., leftward propagation) step. The clique-marginal propagation algorithm with a normalization step inserted between the *CollectEvidence* and *DistributeEvidence* steps is known as *Jensen's algorithm* or the **Hugin** *algorithm* ([Jensen *et al.*, 1990a], [Jensen *et al.*, 1990b]). The algorithm without the normalization step is best credited to [Lauritzen and Spiegelhalter, 1988].

After the propagation has completed, each local potential is the marginal over these variables, e.g.,

$$\psi(t_0, s_0, t_1) = \int_{\Lambda'} P(\lambda|X) d\lambda$$
$$\Lambda' = \{\lambda : \lambda[\mathbf{t_0}, \mathbf{s_0}, \mathbf{t_1}] = \langle t_0, s_0, t_1 \rangle\}$$

The marginal posterior distribution for an individual variable is obtained by selecting any node potential containing that variable and summing or integrating out the remaining variables, for example,

$$P(t_i|X) \propto \sum_{s_i} \int \psi(t_i, s_i, t_{i+1}) dt_{i+1}$$

In other words, after propagation, the marginal distributions are obtainable from a localized integration. Note that since $P(t_i|X)$ is a probability density function, the constant of proportionality is simply the value that yields $\int P(t_i|X) dt_i = 1$.

3.1.4 Potential Representation

This section has demonstrated how the HSSMM-based segmentation problem can be decomposed into smaller local subframes which can then be optimized individually. There is one critical aspect of the propagation process that has not yet been addressed: how ψ and ϑ are to be represented. The above exposition refers to these in the abstract, and provided that they can be represented (exactly), both before and after propagation updates, the above description continues to hold. However, since these potential functions have an infinite domain, their representation is problematic. This is considered in Section 3.3 and Chapter 4.

Although the algorithm to segment time-series data might be of interest to some readers, the more important thing one should come away with from this exposition is a familiarity with the general methodology that was used to obtain the decomposition and the associated propagation algorithm. Therefore, the next section examines the general methodology.

3.2 General Methodology

Suppose someone has a time-series segmentation problem that is almost (but not quite) perfect for the HSSMM approach, but some assumption in the HSSMM formalism must be altered. For example, perhaps the assignment of sensor shapes to states should be nondeterministic, the use of waiting-time distributions needs to be replaced by holding time distributions, there is a desire to weaken the semi-Markov property of the transition model (see Section 7.2.1 of Chapter 7), or the evaluation criteria is to be changed. An algorithm

to segment a time series using an HSSMM does not reveal how such alterations are to be handled. However, an understanding of the general methodology of structural decomposition does provide such guidance.

The general methodology of structural decomposition is very powerful and applicable far beyond time-series segmentation. It is also quite simple and quite clean. For this reason, it is more important to obtain a familiarity with the methodology from this exposition than it is to add a time-series segmentation algorithm to one's collection of algorithms. This section describes the decomposition process in general terms.

One should consider decomposing a large inference problem into smaller subproblems whenever the knowledge involved appears to contain significant structure, particularly if the structure arises from any sort of irrelevance or conditional independence (not necessarily probabilistic independence) between pieces of knowledge. Without a well-organized methodology, discovering just how to effectively utilize this structure is no easy task. The general methodology discussed here can provide significant guidance to a wide class of inference problems.

The most valuable benefit from this methodology may actually be something that has not, to my knowledge, been discussed and certainly has not been emphasized in related published literature: The methodology can be of tremendous assistance in designing a problem formulation. Real applications do not come with fixed and predefined formulations, and it is the task of formulating a computationally tractable representation of an application problem that is often the most difficult and most critical step in its solution. Because of the graphical nature involved in the expression and manipulation of dependencies, graphical Markov field theory helps one differentiate between problematic versus easy to handle dependencies, helps one isolate pieces requiring simplification, and suggests possible assumptions that would yield high efficiency returns [Smyth *et al.*, 1996]. While most of the mathematics underlying this theory is not new to this thesis[2], the message and style of using the theory for guiding problem formation is not similarly emphasized elsewhere.

3.2.1 Overview

To use the methodology, it is necessary to cast one's problem in the appropriate terminology. This involves expressing possible situations in terms of joint assignments to variables, expressing knowledge by way of potentials, expressing the inference problem as a marginal function, and expressing the structuring of knowledge in terms of a combination function. These components of the methodology are introduced in Section 3.2.2.

Given a formalization of a problem in the appropriate terms, the methodology can then be applied to obtain an inference algorithm that utilizes structure in the problem for computational advantage. Applying the methodology consists of the following steps:

1. Choose an appropriate set of variables to describe the possible situations (configurations).
2. Draw a dependency graph indicating dependencies between variables (Section 3.2.3).
3. Triangulate the graph (Section 3.2.3).
4. Extract a *junction tree*, which is then used as the basis for propagation (Section 3.2.4).
5. Choose a (conjugate) representation for local potentials (Section 3.2.5).
6. Define the initialization of local potentials.

In some applications of the methodology, it may be appropriate to automate all or some of the above steps. For example, the knowledge underlying a Bayesian network or chain graph may be encoded in an expert system, and thus to perform an inference the system may perform all the above steps without human intervention. The same may hold in other cases whenever a problem formulation is cast in stone. However, here I am advocating a slightly different use for the methodology, whereby the above steps will typically be done by hand by a human. Certain algorithms may assist the human in some of the steps, such as triangulation or extraction of a junction tree, but once again, but the emphasis here is the usefulness

[2]The specific axiomatization is an original contribution, although it does borrow tremendously from [Shenoy and Shafer, 1990] and [Dawid, 1992].

of the methodology when designing a problem formulation initially. In a great many applications of the methodology, the need to dynamically triangulate the graph at run time, for example, is not essential, and thus these steps can be hand designed as appropriate. This was the case for the HSSMM.

3.2.2 Expressing a Problem

In order to apply the methodology, an application, the knowledge available, and the problem to be solved must be expressed in terms of a joint set of variables, a potential function, a marginal function, and a combination function satisfying certain properties (axioms). These are first introduced here in a very formal and precise fashion, followed by a less formal discussion of how these components should influence one's problem formulation. The axioms below are similar to those introduced in [Shenoy and Shafer, 1990], although most readers should find the presentation given here to be greatly simplified and easier to comprehend. The two axiomatizations are slightly different.

Let $\mathcal{V} = \{\mathbf{x}_1, ..., \mathbf{x}_k\}$ be a set of variables taking on possible values from $\Omega_1, ..., \Omega_k$. No assumptions are made here about the finiteness or countability of Ω_i. A joint assignment, $\mathbf{x} = \times_{i=i..k}\mathbf{x}_i$, takes on a value from $\Omega = \Omega_1 \times ... \times \Omega_k$. If $\mathbf{A} \subset \mathcal{V}$ is a subset of variables ($\mathbf{A} = \{\mathbf{x}_{i_1}, ..., \mathbf{x}_{i_\ell}\}$), then $\mathbf{x}_\mathbf{A}$ denotes $\mathbf{x}_{i_1} \times ... \times \mathbf{x}_{i_\ell}$. The space of all possible subsets of variables is $2^\mathcal{V}$. If $\mathbf{x}$ and $\mathbf{y}$ are joint assignments, we say that $\mathbf{x}$ and $\mathbf{y}$ *agree* on $\mathbf{A}$ when $\mathbf{x}_\mathbf{A} = \mathbf{y}_\mathbf{A}$.

A *potential* ψ is a function on Ω. Commonly it maps from Ω to real numbers in $[0, 1]$, but in general it might map to complex numbers, fuzzy numbers, bounds, vectors, or, with a stretch of the imagination, something symbolic such as sentences or concepts. The only requirement[3] on this function is on its domain, i.e, that it be Ω. A set of potential functions is denoted using Ψ. The space of all possible potential functions is Υ.

A *marginal function*, $m : \Upsilon \times 2^\mathcal{V} \longrightarrow \Upsilon$, takes a potential function and a subset of variables as input, and returns a potential function with the property that the potential function is constant over all assignments that agree on the given subset of variables. Stated precisely, this is the first axiom

Axiom 1 (Locality) $m(\psi, \mathbf{A})(\mathbf{x}) = m(\psi, \mathbf{A})(\mathbf{y})$ *whenever* $\mathbf{x}_\mathbf{A} = \mathbf{y}_\mathbf{A}$.

We also assume that the marginal function obeys the following:

Axiom 2 (Reflexivity) $\psi = m(\psi, \mathcal{V})$, *where* $\mathcal{V}$ *is the full set of variables.*

Axiom 3 (Consonance) $m(m(\psi, \mathbf{A}), \mathbf{B}) = m(\psi, \mathbf{A} \cap \mathbf{B})$

We say a potential function is *local to* $\mathbf{A}$ whenever $\psi = m(\psi, \mathbf{A})$.

When $\psi_\mathbf{A}$ and $\psi_\mathbf{B}$ are two local potentials (local to $\mathbf{A}$ and $\mathbf{B}$ respectively), we say they are *consistent* when $m(\psi_\mathbf{A}, \mathbf{B}) = m(\psi_\mathbf{B}, \mathbf{A})$.

A *combination function* accepts two arguments, both of which are sets of potential functions, and returns a potential function. $c(\Psi_1; \Psi_2)$ returns the combination of the potentials in Ψ_1 *modulated by* the potentials in Ψ_2. When $\Psi_2 = \emptyset$, we will also write simply $c(\Psi_1)$. Typically, the combination function can equally well be referred to as a (generalized) product function, whereby the potentials in Ψ_1 are multiplied together and the result is divided by each of the potentials in Ψ_2. In general, this may be the standard arithmetic product, various fuzzy product functions, evidential combination formulas, etc. A combination function must obey the following axiom whenever the potentials in $\Psi_1 \cup \Psi_2 \cup \Psi_3 \cup \Psi_4$ are consistent:

Axiom 4 (Associativity) $c(\Psi_1 \cup \Psi_2; \Psi_3 \cup \Psi_4) = c(\{c(\Psi_1; \Psi_3)\} \cup \Psi_2; \Psi_4)$

The marginal and combination functions must be interrelated according to the following axiom:

Axiom 5 (Extraction)

$$m(c(\{\psi_1, \psi_2\}; \{m(\psi_1, \mathbf{B})\}), \mathbf{B}) = \psi_2$$

whenever $\psi_2 = m(\psi_2, \mathbf{B})$.

[3] Although a conjugacy requirement is introduced on local potentials in Section 3.2.5.

These axioms imply the following convenient consequence.

Proposition 1 (Cancellation) $c(\Psi_1 \cup \Psi_2; \Psi_2 \cup \Psi_3) = c(\Psi_1; \Psi_3)$

Potentials, marginal and combination functions, and the above axioms provide a framework for stating and thinking about inference problems. First, the methodology requires one to view inference problems and knowledge bases in terms of a collection of variables. Typically, a complete situation is described fully by an assignment of a value to all variables. An inference problem consists of deducing information about some subset of those variables when only partial information about the complete situation is known. Typically, the variable-based representation is just one of several possible alternatives (for example, [Shafer *et al.*, 1987] use a partition-based representation), the choice between them being often arbitrary. A need for quantification can complicate the variable-based representation and require additional machinery, not considered here (see [Breese, 1992], [Poole, 1993a], [Haddawy, 1994]). For the HSSMM, the variables are those that define a segmentation.

Local Potentials

As stated above, a potential ψ is local to a subset of variables $\mathbf{A}$ whenever $\psi = m(\psi, \mathbf{A})$. It is more efficient to represent a local potential than it is to represent a global potential, especially when $\mathbf{A}$ contains a small number of variables, since it can be stored in terms of a function depending only upon the supporting variables, $\mathbf{A}$.

Although it is not explicit in the axioms or in the theorems that follow, a potential may (optionally) remember its supporting variables. One way to view this is that $\mathbf{A}$, the set of supporting variables, is simply one of the parameters that defines ψ. For example, ψ may be stored (parametrized) internally as $\langle \phi, \mathbf{A} \rangle$, where $\phi : \Omega_{\mathbf{A}} \longrightarrow Range(\psi)$, and $\mathbf{A}$ is a subset of variables. The potential's value is then $\psi(\mathbf{x}) = \phi(\mathbf{x}_{\mathbf{A}})$. Below we will write $loc(\psi)$ for the supporting variables when a local potential representation is used, i.e., $loc(\psi) = \mathbf{A}$ when $\psi := \langle \phi, \mathbf{A} \rangle$.

One very important thing to note is that the marginal and combination functions may explicitly access the identity of the supporting variables. This is necessary, for example, to define marginal probability in the standard fashion.

By leaving this aspect of the representation explicit in the axioms, the notation is greatly simplified and the axioms are even slightly more general then they would otherwise be since the theorems that follow do not require this particular choice of local representation.

Interpretation of Components

A potential is essentially a representation of knowledge — both background knowledge and knowledge about the particulars of a problem instance combined. Again, the view of knowledge as a mapping from variable assignments to a value is not as much a limitation as it is simply a way of viewing knowledge. The general framework does not stipulate the type of value that a potential maps to — it may be truth values, probabilities, fuzzy measures, or even nonnumbers. Thus, this view of knowledge is quite flexible. For the HSSMM, we use as a potential the evaluation function of (3.2), which as discussed earlier, may be interpreted as a probability density.

The marginal function serves a number of purposes. Most importantly, it usually extracts the "answer" to an inference problem from a joint potential. The joint potential represents all global knowledge available, the marginal function extracts out the information about some local item of interest (i.e., knowledge about a subset of variables taken locally). The framework requires this local knowledge to also be expressed in the form of a potential function, but being local it is more readily interpreted. For the HSSMM, two marginal functions were introduced:

$$m(\psi, \mathbf{A})(\mathbf{x}) = \max_{\mathbf{y}:\mathbf{y}_{\mathbf{A}}=\mathbf{x}_{\mathbf{A}}} \psi(\mathbf{y})$$

used to find the optimal configuration, and

$$m(\psi, \mathbf{A})(\mathbf{x}) = \int_{\substack{\mathbf{y} = \mathbf{x}_{\mathbf{A}} \\ \mathbf{y} \in loc(\psi)}} \psi(\mathbf{y}) d\mathbf{y} \tag{3.8}$$

$$loc(m(\psi, \mathbf{A})) = loc(\psi) \cap \mathbf{A}$$

used to find local marginals in Section 3.1.3. The integration in (3.8) becomes a summation when Ω is countable. The function *loc* is the local potential function discussed above. The required properties of the marginalization function, Axioms 1–3, are quite straightforward.

Finally, the methodology utilizes a combination function. The global knowledge in ψ is formed as a combination of various local (consistent) facts, and it is the combination function that determines how these local pieces of information join together to form the global knowledge entailed by ψ. It is also by way of the combination function that structure gets introduced. If ψ were to be simply specified as an amorphous function (a black box), mapping variable assignments to potential values, there would be no opportunity to leverage structure that might be inherent in the problem. By combining pieces of knowledge together using the combination function, there is an opportunity for structure to exist within the knowledge.

In the axiomatization developed here, the combination function $c(\Psi_1; \Psi_2)$ takes two sets of potentials. Those in Ψ_1 are combined, while those in Ψ_2 denote knowledge that is to be factored out of the combination (see Proposition 1). The second argument gives the framework some additional flexibility, and allows us to match the well-known propagation algorithm developed by [Lauritzen and Spiegelhalter, 1988] and [Jensen and Jensen, 1994] exactly. A similar axiomatization introduced by [Shenoy and Shafer, 1990] does not include a second argument to the combination function and results in a slightly different propagation algorithm. For example, the algorithm of [Jensen and Jensen, 1994] is not a special case of [Shenoy and Shafer, 1990], but is a special case of the axiomatization discussed here. Axiom 4 expresses a natural requirement: that the combination of knowledge does not depend on the order in which is combined.

The combination function and marginal function cannot be chosen in a totally disconnected fashion. Axiom 5 expresses succinctly the relationship that must exist between these two functions. It states that on any "universe" $\mathbf{B}$ (terminology of [Jensen and Andersen, 1990] and [Jensen *et al.*, 1990b]) on which ψ_2 is local, combining ψ_1 with ψ_2 and then factoring out the local effect of ψ_1 in $\mathbf{B}$ yields the same local potential ψ_2 on $\mathbf{B}$. It is rather surprising that such a simple relationship between the marginal and combination functions is all that is necessary for the methodology to be applicable.

3.2.3 Graphical Markov Representation

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be an undirected graph with vertices $\mathcal{V}$ and edges $\mathcal{E} \subset \{\{\alpha, \beta\} : \alpha, \beta \in \mathcal{V}, \alpha \neq \beta\}$. The vertices of our graphs correspond to the random variables $\mathcal{V}$ above, hence the dual use of $\mathcal{V}$. A *path* of length L from α_0 to α_L is a sequence of at least two vertices, $\alpha_0, \alpha_1, ..., \alpha_L$, such that $\{\alpha_i, \alpha_{i+1}\} \in \mathcal{E}$. A *cycle* is a path with $\alpha_0 = \alpha_L$. A subset of vertices, $\mathbf{S}$, is said to *separate* $\mathbf{A}$ from $\mathbf{B}$ when all paths from any node of $\mathbf{A}$ to any node of $\mathbf{B}$ contain a node in $\mathbf{S}$. If $\mathbf{A}$ is a subset of vertices, the graph $\mathcal{G}_\mathbf{A}$ induced by $\mathbf{A}$ is the subgraph $\mathcal{G}_\mathbf{A} = (\mathbf{A}, \mathcal{E}_\mathbf{A})$ where $\mathcal{E}_\mathbf{A}$ is the set of edges in $\mathcal{E}$ with both endpoints in $\mathbf{A}$ (i.e., $\mathcal{E}_\mathbf{A} = \{\{\alpha, \beta\} : \alpha, \beta \in \mathbf{A}\} \cap \mathcal{E}$). A subset of vertices, $\mathbf{A}$, is called *complete* when all pairs of vertices in $\mathbf{A}$ are connected. If $\mathbf{A}$ complete and is not a subset of a larger complete set of vertices, then $\mathbf{A}$ is called a *clique*. The set of all cliques in $\mathcal{G}$ is denoted by $\mathcal{C}$.

A pair of vertex subsets, $(\mathbf{A}, \mathbf{B})$, *decomposes* $\mathcal{G}$ when $\mathcal{V} = \mathbf{A} \cup \mathbf{B}$, $\mathbf{A} \cap \mathbf{B}$ is complete, and $\mathbf{A} \cap \mathbf{B}$ separates $\mathbf{A}$ from $\mathbf{B}$. The decomposition is called *proper* if $\mathbf{A}, \mathbf{B} \neq \mathcal{V}$. A graph $\mathcal{G}$ is *decomposable* when it is complete, or if there exists a proper decomposition $(\mathbf{A}, \mathbf{B})$ into decomposable subgraphs $\mathcal{G}_\mathbf{A}$ and $\mathcal{G}_\mathbf{B}$. A well known graph theoretic result (e.g., [Golumbic, 1980]) is that a graph is decomposable if and only if it is *triangulated* (also called *chordal*), that is, if all cycles of length $L \geq 4$ contain a short-circuiting edge (a *chord*) between two nonconsecutive vertices in the cycle. Any graph can be converted to a *triangulated* graph by adding edges, but finding the optimal triangulation is in most cases NP-hard ([Arnborg *et al.*, 1987]).

Definition 1 *A potential ψ is called* Markov *with respect to an undirected graph $\mathcal{G}$ whenever $\mathcal{G}$ is complete, or when for any decomposition* $(\mathbf{A}, \mathbf{B})$

$$\psi = c(\{m(\psi, \mathbf{A}), m(\psi, \mathbf{B})\}; \{m(\psi, \mathbf{A} \cap \mathbf{B})\})$$

Theorem 1 *Suppose for each clique $\mathbf{C} \in \mathcal{C}$ of a decomposable graph $\mathcal{G}$, a potential $\psi_\mathbf{C}$ that is local to $\mathbf{C}$ is given, and that these potentials are pairwise consistent. Then there is a unique Markov potential, ψ, having these marginals (i.e., such that $m(\psi, \mathbf{C}) = \psi_\mathbf{C}$).*

This theorem generalizes Theorem 2.6, Theorem 3.9, and Theorem 4.6 of [Dawid and Lauritzen, 1993].

3.2.4 Propagation

A *junction tree* $\mathcal{J} = (\mathcal{C}, \mathcal{E})$, corresponding to a graph $\mathcal{G}$, also sometimes called a *Markov tree* ([Shenoy and Shafer, 1990], [Shafer *et al.*, 1987]) a *cluster tree* ([Shachter *et al.*, 1994], [Draper, 1995]), or a *join tree* ([Maier, 1983]), is an undirected tree where:

1. Each node corresponds to a subset of vertices of $\mathcal{G}$.

2. The vertices of any clique in $\mathcal{G}$ appear together in at least one node of $\mathcal{J}$.

3. Every node on a path in the tree contains a superset of the intersection of the path's endpoints.

The last property is termed the *junction tree property.*

A junction tree can always be efficiently constructed from a decomposable graph ([Jensen and Jensen, 1994]). A nondecomposable graph must first be triangulated before a minimally-sized junction tree can be efficiently extracted.

The results in this section assume that the junction tree has a finite number of nodes.

A junction tree is useful for efficiently computing marginals by locally propagating information between the nodes of the junction tree. Each node of the junction tree, $\mathbf{C}$, maintains a potential $\psi_{\mathbf{C}}$ that is local to $\mathbf{C}$. Denote the set of these by $\Psi_{\mathcal{C}}$ (i.e., $\Psi_{\mathcal{C}}$ has one member per node in the junction tree). In addition, each edge in the junction tree maintains a potential $\psi_{\mathbf{A}\cap\mathbf{B}}$ that is local to both its endpoints (a *separator potential*). We say that a subset of variables is a separator when it is the intersection of two adjacent nodes in a junction tree, and denote by $\mathcal{S}$ the set of all separators in the junction tree. Note that there is one separator per edge. Denote the set of all separator potentials by $\Psi_{\mathcal{S}}$.

The local potentials in $\Psi_{\mathcal{C}} \cup \Psi_{\mathcal{S}}$ are initialized so that

$$\psi = c(\Psi_{\mathcal{C}}; \Psi_{\mathcal{S}}) \tag{3.9}$$

where ψ is the potential of interest.

Suppose $\mathbf{A}$ and $\mathbf{B}$ are neighbors in the junction tree. Then a propagation from $\mathbf{A}$ to $\mathbf{B}$ occurs as follows:

$$\psi'_{\mathbf{A}\cap\mathbf{B}} = m(\psi_{\mathbf{A}}, \mathbf{A}\cap\mathbf{B}) \tag{3.10a}$$

$$\psi'_{\mathbf{B}} = c(\{\psi_{\mathbf{B}}, \psi'_{\mathbf{A}\cap\mathbf{B}}\}; \{\psi_{\mathbf{A}\cap\mathbf{B}}\}) \tag{3.10b}$$

after the propagation, the local potentials $\psi_{\mathbf{B}}$ and $\psi_{\mathbf{A}\cap\mathbf{B}}$ are replaced by $\psi'_{\mathbf{B}}$ and $\psi'_{\mathbf{A}\cap\mathbf{B}}$.

Theorem 2 *A propagation step does not change ψ, i.e., $c(\Psi_{\mathcal{C}}; \Psi_{\mathcal{S}}) = c(\Psi'_{\mathcal{C}}; \Psi'_{\mathcal{S}})$.*

A full propagation is accomplished as follows. First, pick any node of the junction tree as the *root.* Let d be the maximum distance between the root and any other node. Second, perform a *collect evidence* step: for each node at distance d from the root, propagate from it to its neighbor at distance $d-1$. Then for each node at distance $d-1$, propagate from it to its neighbor at distance $d-2$, and so on, until the root's neighbors have propagated to the root. Third, perform a *distribute evidence* step: From the root, propagate potentials outward to the leaves, first from the root to its neighbors, then from each node at depth 1 to each of its neighbors at depth 2, and so on down to depth d.

Theorem 3 *After a full propagation, $\psi_{\mathbf{C}} = m(\psi, \mathbf{C})$ for all local potentials $\psi_{\mathbf{C}}$ in the junction tree.*

3.2.5 Representation

Axioms 1–5 ensure that the computation of marginals is well-defined (Theorem 1) and can be carried out by propagation (Theorem 3). In essence, these axioms characterize the general properties sufficient to utilize these Markov field techniques to decompose and solve a problem. There is, however, an additional concern that has not yet been addressed: on a computer, local potential functions must be represented. Until the required properties of the representation are considered, the axiomatization is really not complete.

It is useful to think of each local potential function as being specified by a set of parameters. For example, $\mathbf{A}$ might be a set of n real-valued variables, and the local potential on this subset of variables, $\psi_{\mathbf{A}}$, may be represented as a n-dimensional Gaussian distribution (as in [Lauritzen, 1992], [Shachter and Kenley, 1989], [Andersen *et al.*, 1993], and [Chang and Fung, 1991]). In this case, $\psi_{\mathbf{A}}$ is parameterized by an n-dimensional mean vector and an $n \times n$ covariance matrix. Denote by $\Upsilon_{\mathbf{A}}$ the family of all potentials that can be represented at $\mathbf{A}$ — i.e., the space of all n-dimensional Gaussian distributions. With this representation, the propagation step outlined in the previous section can only succeed if at every step of the propagation process, $\psi_{\mathbf{A}}$ is in $\Upsilon_{\mathbf{A}}$.

The potential representation is characterized by a family of potentials for each node, $\{\Upsilon_{\mathbf{C}} : \mathbf{C} \in \mathcal{C}\}$, and for each separator, $\{\Upsilon_{\mathbf{S}} : \mathbf{S} \in \mathcal{S}\}$, in a junction tree. We therefore have the following requirement on the representation of potentials.

Axiom 6 (Conjugacy) *Suppose $\mathbf{A}, \mathbf{B} \in \mathcal{C}$ are the variable subsets for two adjacent nodes in a junction tree. Let $\Upsilon_{\mathbf{A}}$ be the family of representable potentials at $\mathbf{A}$, and let $\Upsilon_{\mathbf{A}\cap\mathbf{B}}$ be the family of representable separator potentials at $\mathbf{A} \cap \mathbf{B}$. Then for any $\psi_{\mathbf{A}} \in \Upsilon_{\mathbf{A}}$, $\psi_{\mathbf{A}\cap\mathbf{B}} \in \Upsilon_{\mathbf{A}\cap\mathbf{B}}$, and $\psi'_{\mathbf{A}\cap\mathbf{B}} \in \Upsilon_{\mathbf{A}\cap\mathbf{B}}$,*

$$m(\psi_{\mathbf{A}}, \mathbf{A} \cap \mathbf{B}) \in \Upsilon_{\mathbf{A}\cap\mathbf{B}}$$
$$c(\{\psi_{\mathbf{A}}, \psi'_{\mathbf{A}\cap\mathbf{B}}\}; \{\psi_{\mathbf{A}\cap\mathbf{B}}\}) \in \Upsilon_{\mathbf{A}}$$

It is clear from a direct comparison with (3.10) that Axiom 6 ensures that all potentials during the computation can be represented.

When the domain of $\mathbf{x}_{\mathbf{A}}$ is discrete and finite, then a nonparametric representation is an option where each potential is stored as an array with one element for each possible value of $\mathbf{x}_{\mathbf{A}}$. Axiom 6 is trivially satisfied for a nonparametric representation. However, in general, it is much more difficult to characterize when this property is satisfied in any truly general form.

3.2.6 The Shenoy-Shafer Axioms

An axiomatization similar to the one I have given here appears in [Shenoy and Shafer, 1990]. Their axiomatization is equally useful for demonstrating the generality of propagation techniques. It also influenced the axiomatization presented above significantly.

The axiomatization given above differs from the Shenoy-Shafer axioms for a couple reasons. First, their axioms, as presented in [Shenoy and Shafer, 1990], are very difficult to understand. The description that I have given simplifies the exposition tremendously, and as such make the axioms much more usable for most readers. Second, if you apply the Shenoy-Shafer axioms to a standard probabilistic network, the algorithm that results is not the popular Jensen's (a.k.a., `Hugin`) algorithm. Since I utilized the `Hugin` algorithm in Section 3.2.4, it was easier to describe the general methodology in terms of an axiomatization that produces the same algorithm. Finally, neither set of axioms is fully subsumed by the other set, so it is plausible that there are systems that decompose using my axioms but do not decompose using the Shenoy-Shafer axioms. However, in most cases with mild additional assumptions, the Shafer-Shenoy style axiomatization is more general.

To enable a comparison between the two axiomatizations, I will restate the Shenoy-Shafer axioms in a form comparable to the axioms stated in the previous section. This differs considerably from the presentation in [Shenoy and Shafer, 1990], and it may take a reader a while to see that the presentation here is essentially the same. Again, I believe the way in which I have presented them here is far easier to understand. In fact, the axioms as presented here actually generalize those in [Shenoy and Shafer, 1990], but the fundamental ideas behind them are the same.

The Shafer-Shenoy axioms also make use of a marginal function, m, satisfying Axioms 1, 2, and 3, and a combination function, denoted here as $\otimes$ and taking only one argument and returning a potential function, satisfying the following axioms:

Axiom 7 (Associativity) $\otimes(\Psi_1 \cup \Psi_2) = \otimes(\{\otimes(\Psi_1)\} \cup \Psi_2)$

Axiom 8 (Distributivity)

$$m(\otimes(\{\psi_1, \psi_2\}), \mathbf{B}) = \otimes(\{m(\psi_1, \mathbf{B}), \psi_2\})$$

whenever $\psi_2 = m(\psi_2, \mathbf{B})$.

Axiom 9 (Identity) $\psi = \otimes(\{\psi, \psi\})$

In other words, Axioms 7 and 8 replace Axioms 4 and 5. Axiom 9 is a minor additional assumption used to prove Theorem 5 and assumed by [Shenoy and Shafer, 1990].

In terms of the axioms, the most notable difference is that the Shenoy-Shafer combination function does not require a second argument. Essentially this means that a division-like operation does not need to be supported (c.f., Proposition 1). [Shenoy and Shafer, 1990] did not consider the issue of conjugacy (Axiom 6).

The more distinct difference between the axiomatizations appears in the propagation algorithm associated with the axioms. Since the Shenoy-Shafer axioms have no requirement that a division operation is supported, and since division is a basic operation used during the Jensen/Hugin propagation algorithm, it is clear that the specifics of the propagation algorithm enabled by the Shenoy-Shafer axioms must be different.

The basic control structure for Shafer-Shenoy propagation is the same as with clique tree propagation, and like clique-tree propagation, Shenoy-Shafer propagation also occurs over a junction-tree graph. However, the important local information in Shafer-Shenoy propagation exists on edge potentials rather than on clique potentials. In addition, two potentials are maintained on each edge, one in each direction (it is not necessary to keep clique potentials during propagation, although it can be done for improved efficiency). After propagation completes, the potentials in each direction are not equal — instead, each summarizes the information originating from each side of the edge, so together they have all the information relevant to that edge.

Suppose $\mathbf{A}$ is an edge (i.e., $\mathbf{A} = \mathbf{C}_1 \cap \mathbf{C}_2$ for two adjacent clique nodes $\mathbf{C}_1$,$\mathbf{C}_2 \in \mathcal{C}$ in the junction tree). Because the junction tree is a tree, the edge naturally separates the graph into two components (i.e., if the edge were removed, the junction tree become two disconnected components). Let $\mathbf{B}_1$ and $\mathbf{B}_2$ denote these components. Let the notation $\psi_{\mathbf{B}_1 \to \mathbf{A}}$ denotes a local potential on the edge $\mathbf{A}$ corresponding to side $\mathbf{B}_1$. Edge $\mathbf{A}$ has two local potentials, $\psi_{\mathbf{B}_1 \to \mathbf{A}}$ and $\psi_{\mathbf{B}_2 \to \mathbf{A}}$, one in each direction.

Initialization

The initial potential, in the form of $\psi = \otimes(\{\psi_{\mathbf{C}} : \mathbf{C} \in \mathcal{C}\})$, is assigned to the local edge potentials by setting

$$\psi_{\mathbf{B} \to \mathbf{A}} = m(\psi_{\mathbf{C}}, \mathbf{A})$$

where $\mathbf{C}$ is the clique in $\mathbf{B}$ attached to the edge $\mathbf{A}$ in the junction tree.

Propagation

Figure 3.11 shows an edge $\mathbf{A}$ with $\mathbf{B}$ denoting the portion of the junction tree to one side, and $\mathbf{C}$ denoting the clique node adjacent to $\mathbf{A}$. Clique node $\mathbf{C}$ has edges $\mathbf{A}, \mathbf{A}_1, .., \mathbf{A}_K$, and each edge $\mathbf{A}_i$ is associated with a portion of the junction tree $\mathbf{B}_i$. The basic operation for updating $\psi_{\mathbf{B} \to \mathbf{A}}$ is:

$$\psi'_{\mathbf{B} \to \mathbf{A}} = m(\otimes(\{\psi_{\mathbf{B}_i \to \mathbf{A}_i} : i = 1, .., K\} \cup \{\psi_{\mathbf{B} \to \mathbf{A}}\}), \mathbf{A}) \tag{3.12}$$

And the important property of an individual propagation step is the following.

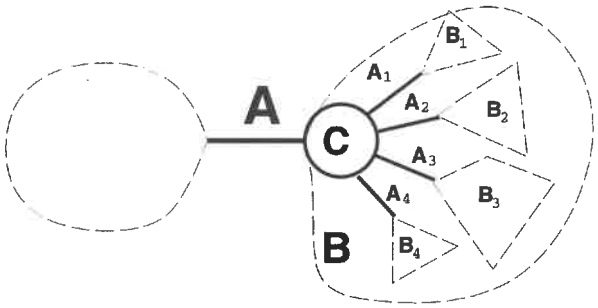


Figure 3.11: A portion of a junction tree. $\mathbf{A}$ is an edge in the junction tree, $\mathbf{C}$ a node (clique) touching the edge, $\mathbf{A}_1, .., \mathbf{A}_K$ the other edges from $\mathbf{C}$, and $\mathbf{B}$ the subgraph on that side of $\mathbf{A}$. The potential $\psi_{\mathbf{B}\to\mathbf{A}}$ is to be updated.

Proposition 2 *Let* $\psi_{\mathbf{B}_i} = \otimes(\{\psi_{\mathbf{C}} : \mathbf{C} \in \mathbf{B}_i\})$, *where* $\psi_{\mathbf{C}}$ *are the initial local potentials. If* $\psi_{\mathbf{B}_i\to\mathbf{A}_i} = m(\psi_{\mathbf{B}_i}, \mathbf{A}_i)$ *for all edges* $\mathbf{A}_i$ *shown in Figure 3.11, then after the propagation step in (3.12),*

$$\psi'_{\mathbf{B}\to\mathbf{A}} = m(\psi_{\mathbf{B}}, \mathbf{A})$$

where $\psi_{\mathbf{B}} = \otimes(\{\psi_{\mathbf{C}} : \mathbf{C} \in \mathbf{B}\})$ *is the initial potential on subtree* $\mathbf{B}$.

Note that whenever $\mathbf{B}_i$ is a leaf, then the initialization ensures that $\psi_{\mathbf{B}_i\to\mathbf{A}_i} = m(\psi_{\mathbf{B}_i}, \mathbf{A}_i)$ automatically.

As before, a full propagation consists of selecting a root node in the junction tree, propagation all potentials from the leaves to the root, and then propagation all potentials from the root back down to the leaves.

Theorem 4 *After a full proposition,* $\psi_{\mathbf{B}\to\mathbf{A}} = m(\psi_{\mathbf{B}}, \mathbf{A})$ *for any edge potential.*

Once these potentials are updated, the potential on any clique is immediately available by simply combining the local potentials on all incoming edges.

Theorem 5 *Let* $\mathbf{C}$ *be a clique with edges* $\mathbf{A}_1, .., \mathbf{A}_K$, *and associated subtrees* $\mathbf{B}_1, .., \mathbf{B}_K$. *Let* $\psi_{\mathbf{B}_i\to\mathbf{A}_i}$ *denote the edge potentials after propagation has completed. Then*

$$m(\psi, \mathbf{C}) = \otimes(\{\psi_{\mathbf{B}_i\to\mathbf{A}_i} : i = 1, .., K\})$$

From $m(\psi, \mathbf{C})$, the marginal for any variable of interest can be obtained.

Discussion

There are two useful ways to store potentials. The first way stores the information using a single potential per clique and edge. The second way uses two potentials on each edge. The two representations result in two different forms for the basic propagation step. Other than that, propagation in either case is roughly the same.

For probability propagation, [Shachter *et al.*, 1994] gives a nice description of the two different forms of propagation. Pearl's poly-tree propagation algorithm results from applying the Shafer-Shenoy propagation to a poly-tree Bayesian network. The clique marginal algorithm falls out from the axiomatization given in Section 3.2.2 of this thesis. The two axiomatizations (the one introduced by this thesis plus the Shafer-Shenoy axioms), appear to span the propagation algorithms found in the literature.

It may seem that having a second axiomatization so close to the Shenoy-Shafer axioms is inconsequential. My personal experience has been that it is extremely helpful. In another line of research, not directly related to work in this thesis, I attempted (unsuccessfully) over the course of a couple years to derive an exact propagation algorithm for lower and upper probabilities. A number of complications exist that make exact propagation difficult, and indeed up until the publication of [Chrisman, 1996], no exact algorithms for this

problem had been published[4]. In hindsight, considering the many unusual properties of lower probabilities, it is actually quite surprising that exact propagation is possible at all for lower probabilities. Shortly after finding the axioms introduced here, by using the representation and updating framework in [Chrisman, 1995] I finally discovered an exact propagation algorithm for 2-monotone lower probabilities. The algorithm appears in [Chrisman, 1996]. Since I now know how it can be done, it is plausible that I would have little trouble obtaining an analogous propagation algorithm from the Shenoy-Shafer axioms. Regardless of whether this is the case, it was the existence of a second axiomatization and the alternative perspective it offers that allowed me to discover a solution. I believe this experience alone demonstrates the usefulness of a second set of axioms.

3.3 When Structural Decomposition is Insufficient

In many cases, structural decomposition alone cannot reduce a problem into a feasible-to-solve size. In many cases, the dependency structure between variables is too extensive, resulting in cliques with a large number of variables. This renders the propagation scheme intractable since it scales exponentially with the number of variables in the largest clique[5]. In other cases, the individual variables may take on too many values so that a nonparametric potential representation is infeasible, or the variables may be continuous with nonconjugate distributions. In all of these cases, it is almost always necessary to turn to approximation techniques.

Many common approximation techniques for probabilistic models are based on Monte Carlo simulation schemes. At the current time, the most popular Monte Carlo technique for graphical probabilistic models is Gibbs sampling ([Geman and Geman, 1984], [Pearl, 1987], [Chavez and Cooper, 1990], [Gelfand and Smith, 1990], [Gelfand, 1995]). Gibbs sampling is a special case of Markov chain Monte Carlo (MCMC) sampling ([Neal, 1993], [Tierney, 1994]), and is reviewed in Section 3.4. It is a very general technique that is insensitive to the dependency structure between variables and does not rely on any conjugacy assumptions about distributions. Also, its simplicity is appealing. It applies much more generally than just to probabilistic models, but it is especially well-suited to graphical probabilistic models with their very localized dependency structure. To use pure Gibbs sampling, it is only necessary to identify the dependency structure between variables, and not necessary to triangulate graphs or find a junction tree. While this makes it that much easier to apply, it also means that the structural decomposition that is possible cannot be fully leveraged.

In Section 3.4.2, I review some existing techniques for combining structural decomposition with Gibbs sampling ideas in an attempt to harness the relative strengths of each. While I do not utilize these techniques in this thesis, the methods in Chapter 4 can be viewed as a new method for combining Gibbs style sampling with propagation, so the review provides a relative point of comparison, as well as acquainting the reader with previously proposed methods.

Although they are mentioned in Section 3.4, I do not review other Monte Carlo techniques such as logic sampling ([Henrion, 1988]) and importance sampling ([Shachter and Peot, 1989]). While these and related approaches have received substantial attention in the Artificial Intelligence literature, straight logic sampling cannot be applied to problems with continuous variables (as in the current case) and both are highly inefficient when the probability of observations is low. Because they are not used directly in this thesis, I do not review them in any detail here. It should be noted, however, that importance sampling can be quite useful in probabilistic contexts other than approximating graphical networks.

When structural decomposition is insufficient because variables are continuous with nonconjugate distributions, or because individual variables take on too many values, another approximation technique is *iterative dynamic discretization*. This technique is the topic of Chapter 4. It is designed to leverage the results of structural decomposition while implementing an iterative approximation scheme with similarities to Gibbs sampling.

---

[4] [Tessem, 1992] published an approximation to exact propagation for a very special case, and [Fertig and Breese, 1993, Breese and Fertig, 1991] published an exact algorithm for an even more special case, while [Cano *et al.*, 1991] published an algorithm for a less tractable and more general case. See [Chrisman, 1996] for details.

[5] More exactly, it scales linearly with the number of possible values for a clique assignment, which in turn scales exponentially with the number of variables.

3.4 Gibbs Sampling

Gibbs sampling occasionally goes by the name *straight simulation* ([Pearl, 1987, Pearl, 1988, Chavez and Cooper, 1990, Dagum and Horvitz, 1993, Monti and Cooper, 1996]), although the name Gibbs Sampling is far more predominant. A *Gibbs system* is equivalent to a Markov field with everywhere positive probability ([Moussouris, 1974]), which is no doubt the source of the name, but since the fundamental procedure is not tied only to Gibbs systems the naming is somewhat unfortunate and straight simulation would be a better name. Another good name might be *one-variable-at-a-time simulation.* However, since the name Gibbs sampling is now so highly entrenched in existing literature, it would be confusing to call it by any other name.

The Gibbs sampling algorithm is the simplest of the Markov chain Monte Carlo algorithms ([Neal, 1993]). A description of the algorithm appeared as early as [Hastings, 1970], but it first gained prominence with the publication of [Geman and Geman, 1984], where it was applied to image restoration (with a Gibbs system, and probably the original source of the name). It is a very natural approximation to apply to graphical probabilistic models, and its use was in this context was introduced by [Pearl, 1987], described also in [Pearl, 1988, Section 4.4.3]. Theoretical convergence rates for Bayesian networks are analyzed in [Chavez and Cooper, 1990] and [Dagum and Horvitz, 1993]. The technique was popularized in the statistics community with the publication of [Gelfand and Smith, 1990] and is now among the most popular inference method. Its wide applicability and simplicity have contributed to its popularity.

Suppose $\mathbf{x} = \langle \mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n \rangle$ is a set of random variables, and one wishes to compute inferences over the joint probability distribution $p(x_1, x_2, ..., x_n)$. If the variables are continuous, then $p(\cdot)$ is a probability density, if they are discrete, it is a probability distribution. Denote the joint probability space by $(\Omega, \mathcal{F})$, where Ω is the set of all possible *joint* configurations, and $\mathcal{F}$ is a σ-algebra on Ω (in the discrete case, $\mathcal{F} = 2^{\Omega}$). The inferences of interest are usually expectations relative to p, or marginal distributions such as $p(x_1)$, $p(x_2)$, etc.

Monte Carlo techniques approximate inference by drawing joint configurations[6] at random from p, and averaging over the sample. The more samples drawn, the closer one can expect the sample frequencies to match the distribution p.

However, in most cases of interest, drawing independent samples from the distribution of interest is very difficult. To see this, consider a Bayesian network. The distribution of interest is $p(x_1, ..., x_n) = P(x_1, ..., x_n|e)$, where e is observed evidence, often located in the leaves of the network, and $P(x_1, ..., x_n)$ is the joint prior distribution on the network without any evidence. While it is trivial in a Bayesian network to generate independent samples from $P(x_1, ..., x_n)$[7], it is almost always very difficult to generate independent samples from $P(x_1, ..., x_n|e)$, the distribution of interest. One method termed *logic sampling* is to sample from $P(x_1, ..., x_n)$, and then throw away all samples that don't match the evidence exactly ([Henrion, 1988]). This requires an average of $1/P(e)$ iterations per sample, which is generally unacceptable. A small improvement is to use importance sampling ([Shachter and Peot, 1989]), which enforces $x_i = e_i$ for each observed variable x_i when the sample is being generated, and then weights each sample according to the probability of the evidence given the other sampled variable values. However, a few samples generally dominate importance sampling averages — generally those that would probably have been kept during logic sampling — making it only a small improvement over logic sampling.

Markov chain Monte Carlo methods overcome the difficulty of drawing independent samples from a distribution of interest by sacrificing the independence of samples to obtain an efficient sampling procedure. A sequence of *dependent* samples are drawn from the distribution of interest in a way that guarantees that the asymptotic distribution of the samples is the same as the distribution of interest. By sacrificing independence, efficient sampling procedures can often be obtained.

The *Gibbs sampling algorithm* works by changing the value of only one component in a configuration at a time. An initial assignment to all the variables is first obtained, perhaps arbitrarily[8]. In one iteration,

[6] A *joint configuration* is simply an assignment of a value to every variable in the network.

[7] One simply samples each variable in the order indicated by the directed arcs of the Bayesian network, with each sample based on the distribution conditioned on the value already selected for that node's parents.

[8] One may also consider employing an expensive method such as logic sampling or rejection sampling for obtaining the first

a single variable, x_i, is selected and a new value for x_i is drawn at random from $p(x_i|x_{j\neq i})$. All the other variables retain their previous values. Thus, two consecutive samples produced by Gibbs sampling differ at most by one component.

One simple form of Gibbs sampling visits each variable in the network sequentially. On the first iteration, the value of x_1 is resampled from $p(x_1|x_2, ..., x_n)$. The second iteration samples x_2 from $p(x_2|x_1, x_3, x_4, ..., x_n)$. The n^{th} iteration samples x_n from $p(x_n|x_1, ..., x_{n-1})$, and the $(n+1)^{th}$ iteration again samples x_1 from $p(x_1|x_2, ..., x_n)$ and the process repeats until an acceptable approximation is obtained. Another variant of Gibbs sampling selects the variable to resample at random.

Each iteration of the Gibbs sampling algorithm can be described as a transition from one joint configuration, $x \in \Omega$, to another joint configuration $y \in \Omega$, according to a probability $h_i(x, Y)$, the probability of generating a state $y \in Y \in \mathcal{F}$ next given that the current configuration is $x \in \Omega$. The subscript on h_i identifies the iteration of the algorithm. The version of Gibbs sampling that visits each variable sequentially alternates between n different h_i's. The version that chooses the variable to resample at random can be characterized as using the same transition probability at every iteration. In the second case, the subscript can be omitted from h, and h is said to be *homogeneous*. A homogeneous h can also be obtained in the first case by considering one pass through all the variables to be a single iteration, so that[9] $h = h_1 \circ h_2 \circ ... \circ h_n$ is a homogeneous transition probability for each combined iteration. In fact, any repeating pattern of visits to all variables will yield a homogeneous transition probability for the process. I will call any sampling pattern that can be equated to a homogeneous transition probability at some level a *homogeneous sampling pattern* (nonstandard terminology). Both variants of Gibbs described above, cycling through each variable and selecting the variable at random, are examples of homogeneous sampling patterns.

The transition probability, h_i, together with the initial configuration or a description of the distribution from which the initial sample is drawn defines a *Markov chain*. Hence the name *Markov chain Monte Carlo* (MCMC). Gibbs sampling (i.e., changing one component at a time) is only one way to obtain an appropriate Markov chain, and so it is a special case of MCMC.

The m^{th} stage transition probability, $h_i^m(x, Y)$, is the probability of reaching Y in m steps when starting from x on the i^{th} iteration. It is simply given by $h_i^m = h_i \circ h_{i+1} \circ ... \circ h_{i+m-1}$. A *stationary distribution* π is one in which for all i

$$\pi(Y) = \int h_i(x, Y)\pi(x)dx$$

Once a system reaches a stationary distribution, it will remain there indefinitely. Any Markov chain formed by sampling one variable at a time from $p(x_i|x_{j\neq i})$ has p as a stationary distribution. In general a Markov chain may not have a unique stationary distribution, but in the case of Gibbs sampling, if variables are sampled in an appropriate order, there is a unique distribution and the process converges to it. The fundamental Gibbs sampling theorem ([Tierney, 1994]) is as follows:

Theorem 6 (Gibbs Sampling Theorem) *The asymptotic distribution of configurations visited by Gibbs sampling approaches p whenever*

1. *p assigns a positive probability density to every joint configuration.*
2. *All variables are visited in a homogeneous sampling pattern.*

Other variations on the theorem exist as well. [Geman and Geman, 1984] prove that for finite state spaces and everywhere positive probability, Gibbs sampling converges from all starting configurations as long as all variables are visited infinitely often (i.e., a homogeneous sampling pattern is not required). I have not found a proof in existing literature of convergence in this nonhomogeneous case with a general (nonfinite) state space.

It is often useful when applying Gibbs sampling to an application to understand the basic outline of the proof of the above theorem. As discussed above, p is a stationary distribution of the Markov chain

---

sample from the distribution of interest. If this extra work is done for the first sample, then the MCMC sampling process is started from the ergodic distribution, so the time one would otherwise have to wait for the process to reach its ergodic distribution is eliminated.

[9] $(h_i \circ h_j)(x, Y) = \int h_i(x, z)h_j(z, Y)dz$, and $\circ$ is left-associative.

resulting from Gibbs sampling. Also, because p is everywhere positive, any subset of states A with $p(A) > 0$ is reachable within a finite number of steps since we can change each component one at a time to match A. This condition is called p-irreducibility. A basic result of Markov chain theory ([Tierney, 1994, Theorem 1]) states that if h is p-irreducible and p is an invariant distribution of h, then h is positive recurrent (meaning p is the unique invariant distribution). Furthermore, the positivity of p ensures that h is also absolutely continuous with respect to p, and with this extra condition, h is *Harris recurrent* [Tierney, 1994, Corollary 1], meaning that every subset of configurations A with $p(A) > 0$ is visited infinitely often with probability 1. The stochastic nature of Gibbs sampling also ensures *aperiodicity*. The Aperiodic Ergodic Theorem ([Meyn and Tweedie, 1993]) ensures that a Harris recurrent, aperiodic Markov chain with invariant distribution p approaches p asymptotically (in total variation norm) from any starting configuration ([Tierney, 1994, Theorem 1]).

The primary operation performed during the Gibbs sampling algorithm is the operation of drawing a value of x_i from $p(x_i|x_{j\neq i})$. This is especially convenient in the case of a graphical probabilistic network since x_i is conditionally independent of all variables in the network given its Markov boundary. In an undirected graph, the *Markov boundary* is simply the set of variables that share an edge with x_i. For example, in the undirected network of Figure 3.1 on Page 48, the Markov boundary of t_1 is $\{t_0, s_0, s_1, t_2\}$. Therefore, $p(t_1|t_0, s_0, s_1, t_2, s_2, t_3, s_3, ...) = p(t_1|t_0, s_0, s_1, t_2)$. Thus, sampling only involves a local subset of variables. This usually means that the form of $p(x_i|x_{j\neq i})$ is considerably simplified. Gibbs sampling is therefore very well-suited for use in graphical probabilistic networks.

There are two requirements that must be met in order for Gibbs sampling to be usable:

1. The joint probability density must be everywhere positive.

2. It must be possible to sample from $p(x_i|x_{j\neq i})$ efficiently.

The first condition can often be relaxed in special cases by reasoning directly about the conditions discussed above that lead to Harris recurrence. This is usually just a matter of ensuring p-irreducibility (which is easier to verify) over the portion of the state space that has positive probability density ([York, 1992]), since the other conditions are pretty much automatically satisfied in a Gibbs sampling application. When only positive recurrence can be assured (e.g., when h is not absolutely continuous with respect to p), then the convergence theorem holds only for p-almost all starting configurations ([Tierney, 1994, Theorem 1]). Thus, the first requirement seldom restricts the applicability of Gibbs sampling to practical problems. However, even when p-irreducibility is obtained, the convergence rate of Gibbs sampling is critically dependent on the magnitude of the smallest probability density ([Chavez and Cooper, 1990, Dagum and Chavez, 1993]).

The second requirement can be much more difficult to overcome. Sampling from the conditional distribution, if it can be done efficiently at all, may require any number of advanced sampling techniques, such as those discussed in [Devroye, 1986]. As already mentioned, graphical probabilistic networks simplify this portion of the problem greatly by isolating the problem to a variable and its Markov boundary, but in general, the step can still be quite difficult. In the time-series segmentation domain, this step is highly nontrivial and is addressed in Chapter 4.

Once Gibbs sampling can be applied, the rate of convergence becomes the primary concern. On finite state spaces[10], Gibbs sampling is known to converge geometrically (for a fixed distribution) to the asymptotic distribution ([Geman and Geman, 1984], [Gelfand and Smith, 1990], [Neal, 1993]). As good as this sounds, the rate of convergence does not scale well with $1/p^*$, where p^* is the smallest joint probability (density) ([Chavez and Cooper, 1990, Dagum and Chavez, 1993, Dagum and Horvitz, 1993]). However, all known theoretical bounds are much too large to be of any pragmatic use anyway, so this is primarily academic. Therefore, meaningful evaluations of Gibbs sampling methods, and MCMC methods in general, are currently restricted to empirical investigations. Metrics to detect convergence are often used in practice and are an active area of study ([Brooks and Roberts, 1995, Cowles and Carlin, 1996]); however, these are by their very nature just heuristics.

[10] Some similar convergence are available for continuous state spaces, but involve complicated conditions such as compactness of the state space, various forms of measurability, etc [Liu, 1991, Tierney, 1994, Rosenthal, 1995a, Rosenthal, 1995b, Baxter and Rosenthal, 1995].

3.4.1 Focused Gibbs Sampling

This section describes a variant of Gibbs sampling that would seem to have many uses, but which I have not found mentioned previously in the literature. This variant provides a very natural way to combine Gibbs sampling with exact propagation techniques. The Markov chain Monte Carlo aspect of this algorithm is, however, pure Gibbs sampling.

The idea is to simply partition the variables in a model into two groups: $\mathbf{x} = (\mathbf{y}, \mathbf{z})$. The variables in $\mathbf{y}$ are sampled using Gibbs sampling, while for each setting of values to $\mathbf{y}$, the remaining distribution over $\mathbf{z}$ is computed using exact methods.

Let the sampled variables be $\mathbf{y} = \langle \mathbf{y}_1, \mathbf{y}_2, .., \mathbf{y}_n \rangle$. During one step of focussed Gibbs sampling, a variable $\mathbf{y}_i$ is selected, and a new value for $\mathbf{y}_i$ is drawn according to $p(y_i|y_{j\neq i})$. Thus, the stochastic part of the algorithm is the same as Gibbs sampling, but over the space $\Omega_{\mathbf{y}}$ rather than over the space $\Omega_{\mathbf{x}}$.

To obtain $p(\mathbf{y}_i|\mathbf{y}_{j\neq i})$, one must marginalize out the variables in $\mathbf{z}$:

$$p(y_i|y_{j\neq i}) = \int p(y_i, dz|y_{j\neq i}) = \int p(y_i|y_{j\neq i}, z)p(dz|y_{j\neq i})$$

Suppose a variable $\mathbf{y}_i$ has neighbors[11] $\mathbf{y}'$ and $\mathbf{z}'$, where $\mathbf{y}'$ is a subset of the variables comprising $\mathbf{y}$, and $\mathbf{z}'$ is a subset of the variables comprising $\mathbf{z}$. Then $p(y_i|y_{j\neq i}, z) = p(y_i|y', z')$, so

$$p(y_i|y_{j\neq i}) = \int p(y_i|y', z')p(dz'|y_{j\neq i})$$

To draw a value for y_i from this distribution, one can first draw a value z' from $p(z'|y_{j\neq i})$, and then draw a value for y_i from $p(y_i|y', z')$. Then $y_i \sim p(y_i|y_{j\neq i})$.

Focussed Gibbs sampling requires the extra operation (not required by pure Gibbs sampling) of drawing $z' \sim p(\mathbf{z}'|y_{j\neq i})$. It is for this operation that propagation is required. The sampled values $y_{j\neq i}$ are inserted into a propagation network as "evidence," and propagation used to update potentials. The distribution $p(y_i, y', z'|y_{j\neq i})$ is then simply the product of all clique potentials containing y_i divided by the product of all separator potentials containing y_i. To pick a value z' from $p(z'|y_{j\neq i})$, one simply picks $(y_i, y', z') \sim p(y_i, y', z'|y_{j\neq i})$ and then returns just z'.

Convergence properties for this process are the same as for pure Gibbs sampling since focussed Gibbs is an instance of pure Gibbs sampling, but applied to the submodel $p(y)$ rather than to the full model $p(y, z)$. It is commonly stated within the literature on Gibbs sampling that convergence is highly dependent on the number of variables (e.g., [Jensen *et al.*, 1995, Gelfand, 1995, Neal, 1995]). Doubling the number of variables often more than doubles the necessary run times. Focused Gibbs sampling restricts the Gibbs sampling part to a smaller set of variables than pure Gibbs sampling, handling the remaining variables with exact methods. Because of the smaller number of variables, and the potential to break troublesome correlation structures with a well placed selection of sampled variables, focused Gibbs sampling has the advantage of reducing the number of iterations required relative to full Gibbs sampling. The full and precise extent of this advantage in general has not yet been studied and left here as future work. Once again, the process is still Gibbs sampling, only on a smaller set of variables.

3.4.2 Blocking-Gibbs Sampling

Gibbs sampling changes the value of one variable at a time at each iteration. It is possible to reformulate the variables that define a model by grouping existing variables into *blocks*. In the reformulated model, each block acts as a variable, and Gibbs sampling can be applied directly to this "blocked" model. The result is called *Blocking Gibbs* ([Amit and Grenander, 1991, Hills and Smith, 1992, Liu *et al.*, 1994, Besag *et al.*, 1995, Jensen *et al.*, 1995, Roberts and Sahu, 1996]).

The convergence properties and rates of Gibbs sampling depend greatly upon the blocking structure of the underlying model. Generally, the less correlated individual blocks are, the faster Gibbs sampling converges. It is generally accepted that larger blocks lead to faster convergence ([Amit and Grenander, 1991,

[11] A neighbor here is a variable in $\mathbf{y}_i$'s Markov boundary.

Roberts and Sahu, 1996]), but the problem of drawing a value assignment for a block conditioned on all other variables in the model is typically more difficult for a larger block. For example, if this could be done efficiently for the block consisting of all variables in the model, it would be unnecessary to turn to a Markov chain technique to generate samples. Despite the common wisdom, there are some known cases where blocking can actually reduce convergence rates [Roberts and Sahu, 1996].

If one groups highly correlated variables into a single block, then the random value generator that must draw a value from the block is faced with a complex correlation structure, and that task becomes more difficult. Suppose, however, that a certain block contains a large number of variables, but the interrelationships on the subgraph consisting only of these variables has a convenient graphical structure (e.g., a poly-tree, or more generally, something with small cliques after it is triangulated). If the distributions on those variables have the appropriate conjugacy properties to permit the application of exact propagation, then the distribution within this block (conditioned on all variables outside the block) can be propagated exactly. After propagation, each clique contains the marginal distribution. Samples can then be drawn using forward sampling from this exact distribution. In other words, within a block, exact propagation may be used to enable to sampling of a value for that block. This idea was introduced by [Jensen *et al.*, 1995].

Blocking Gibbs typically uses nonoverlapping blocks, but this is not entirely necessary — blocks may also overlap [Jensen *et al.*, 1995]. For a strictly positive distribution, the irreducibility argument given in Section 3.4 still holds, so convergence is still guaranteed.

In some models containing zero probability configurations, pure Gibbs sampling may result in an irreducible Markov chain. In these cases, the asymptotic guarantees do not hold. However, often blocking can be applied to solve this problem by placing functionally dependent variables in the same block and restoring irreducibility.

Blocking Gibbs sampling is not used in this thesis, but has been reviewed here since it is an important issue in practice when Gibbs sampling is applied. Note that both focused Gibbs sampling and blocking Gibbs sampling are simply instances of the general Gibbs sampling procedure — they are simply variations in the component arrangement that is used.

3.5 Proofs (Chapter Appendix)

The proofs for the propositions and theorems appearing in Chapter 3 are given in this chapter appendix. The results are motivated and discussed in the text of the previous chapter sections, and no additional results appear here. The theorem statements are repeated (with the same numbering as in the text) and proofs given. The chapter appendix should be skipped by any reader not interested in the intricate details of the proofs.

Proposition 1 (Cancellation): $c(\Psi_1 \cup \Psi_2; \Psi_2 \cup \Psi_3) = c(\Psi_1; \Psi_3)$

Proof: Let $\Psi_1 = \{\psi_0\} \cup \Psi_1'$ and $\Psi_2 = \{\psi_{2,1}, ..., \psi_{2,m}\}$. By repeatedly applying Axiom 4,

$$c(\Psi_1 \cup \Psi_2; \Psi_2) = c(\{c(...\{c(\{c(\{\psi_0, \psi_{2,1}\}; \{\psi_{2,1}\}), \psi_{2,2}\}; \{\psi_{2,2}\}), ..., \psi_{2,m}\}; \{\psi_{2,m}\})\} \cup \Psi_1')$$

By setting $\mathbf{B} = \mathcal{V}$ in Axiom 5, applying Axiom 2, and renaming subscripts, we obtain

$$c(\{\psi_0, \psi_{2,1}\}; \{\psi_{2,1}\}) = \psi_0$$

Plugging this into the above equation m times obtains

$$c(\Psi_1 \cup \Psi_2; \Psi_2) = c(\{\psi_0\} \cup \Psi_1') = c(\Psi_1)$$

Finally, by Axiom 4

$$c(\Psi_1 \cup \Psi_2; \Psi_2 \cup \Psi_3) = c(\{c(\Psi_1 \cup \Psi_2; \Psi_2)\}; \Psi_3) = c(\{c(\Psi_1)\}; \Psi_3) = c(\Psi_1; \Psi_3)$$

□

Theorem 1: Suppose for each clique $\mathbf{C} \in \mathcal{C}$ of a decomposable graph $\mathcal{G}$, a potential $\psi_{\mathbf{C}}$ that is local to $\mathbf{C}$ is given, and that these potentials are pairwise consistent. Then there is a unique Markov potential, ψ, having these marginals (i.e., such that $m(\psi, \mathbf{C}) = \psi_{\mathbf{C}}$).

Proof: If $\mathcal{G}$ is complete, then the theorem trivially holds. Otherwise, pick one clique $\mathbf{C}_1$ and form two subsets of $\mathcal{V}$, $\mathbf{A} = \mathbf{C}_1$ and $\mathbf{B} = \bigcup(\mathcal{C} - \mathbf{C}_1)$. Then $(\mathbf{A}, \mathbf{B})$ decomposes $\mathcal{G}$ and $\mathcal{G}_{\mathbf{B}}$ is a smaller subgraph. Assume by induction that the theorem holds for $\mathcal{G}_{\mathbf{B}}$, and let $\psi_{\mathbf{B}}$ be the unique potential on $\mathcal{G}_{\mathbf{B}}$. Also set $\psi_{\mathbf{A}} = \psi_{\mathbf{C}_1}$.

The potential is $\psi = c(\{\psi_{\mathbf{A}}, \psi_{\mathbf{B}}\}; \{m(\psi_{\mathbf{A}}, \mathbf{B})\})$. By Axiom 5, $m(\psi, \mathbf{B}) = \psi_{\mathbf{B}}$. Since $\psi_{\mathbf{A}}$ and $\psi_{\mathbf{B}}$ are consistent, $m(\psi_{\mathbf{A}}, \mathbf{B}) = m(\psi_{\mathbf{B}}, \mathbf{A})$, so $\psi = c(\{\psi_{\mathbf{A}}, \psi_{\mathbf{B}}\}; \{m(\psi_{\mathbf{B}}, \mathbf{A})\})$ and again by Axiom 5, $m(\psi, \mathbf{A}) = \psi_{\mathbf{A}}$. So a potential with the given clique margins exists.

To show uniqueness, suppose ψ and ψ' are both Markov and agree on clique marginals (i.e., $m(\psi, \mathbf{C}) = m(\psi', \mathbf{C})$ for $\mathbf{C} \in \mathcal{C}$). Then $\psi_{\mathbf{A}} = \psi'_{\mathbf{A}}$, since $\mathbf{A}$ is a clique. By the induction, $\psi_{\mathbf{B}} = \psi'_{\mathbf{B}}$. And by Axiom 1, $\psi_{\mathbf{A}\cap\mathbf{B}} = \psi'_{\mathbf{A}\cap\mathbf{B}}$. Thus,

$$\begin{aligned}\psi' &= c(\{m(\psi', \mathbf{A}), m(\psi', \mathbf{B})\}; \{m(\psi', \mathbf{A} \cap \mathbf{B})\}) \\ &= c(\{m(\psi, \mathbf{A}), m(\psi, \mathbf{B})\}; \{m(\psi, \mathbf{A} \cap \mathbf{B}\})\}) \\ &= \psi\end{aligned}$$

The first line holds because ψ' is Markov. The second line because ψ and ψ' agree on $\mathbf{A}$, $\mathbf{B}$, and $\mathbf{A} \cap \mathbf{B}$ as established above. And the third line holds because ψ is Markov. Therefore ψ is unique. □

Theorem 2: A propagation step does not change ψ, i.e., $c(\Psi_{\mathcal{C}}; \Psi_{\mathcal{S}}) = c(\Psi'_{\mathcal{C}}; \Psi'_{\mathcal{S}})$.

Proof: First, note the following:

$$\begin{aligned}c(\{\psi'_{\mathbf{B}}\}; \{\psi'_{\mathbf{A}\cap\mathbf{B}}\}) &= c(\{c(\{\psi_{\mathbf{B}}, \psi'_{\mathbf{A}\cap\mathbf{B}}\}; \{\psi_{\mathbf{A}\cap\mathbf{B}}\})\}; \{\psi'_{\mathbf{A}\cap\mathbf{B}}\}) \\ &= c(\{\psi_{\mathbf{B}}, \psi'_{\mathbf{A}\cap\mathbf{B}}\}; \{\psi_{\mathbf{A}\cap\mathbf{B}}, \psi'_{\mathbf{A}\cap\mathbf{B}}\}) \\ &= c(\{\psi_{\mathbf{B}}\}; \{\psi_{\mathbf{A}\cap\mathbf{B}}\})\end{aligned}$$

The first line simply expands $\psi'_{\mathbf{B}}$ according to (3.10b). The second line uses Axiom 4. The third line applies Proposition 1.

Let ψ be the potential before propagation and ψ' be the potential after propagation. Then

$$\begin{aligned}\psi &= c(\{\psi_{\mathbf{B}}\} \cup \Psi_{\mathcal{C}-\mathbf{B}}; \{\psi_{\mathbf{A}\cap\mathbf{B}}\} \cup \Psi_{\mathcal{S}-\mathbf{A}\cap\mathbf{B}}) \\ &= c(\{c(\{\psi_{\mathbf{B}}\}; \{\psi_{\mathbf{A}\cap\mathbf{B}}\})\} \cup \Psi_{\mathcal{C}-\mathbf{B}}; \Psi_{\mathcal{S}-\mathbf{A}\cap\mathbf{B}}) \\ &= c(\{c(\{\psi'_{\mathbf{B}}\}; \{\psi'_{\mathbf{A}\cap\mathbf{B}}\})\} \cup \Psi_{\mathcal{C}-\mathbf{B}}; \Psi_{\mathcal{S}-\mathbf{A}\cap\mathbf{B}}) \\ &= c(\Psi'_{\mathcal{C}}; \Psi'_{\mathcal{S}}) = \psi'\end{aligned}$$

The first line simply writes out ψ as the Markov combination of its local potentials. The second line applies Axiom 4. The third line applies (3.13), and the fourth line reapplies Axiom 4. Thus $\psi = \psi'$ — the joint potential is unchanged by the propagation step. □

Lemma 1 *After a propagation step from* $\mathbf{A}$ *to* $\mathbf{B}$*, where* $\mathbf{A}$ *and* $\mathbf{B}$ *are neighbor nodes in the junction tree, the resulting* $\psi'_{\mathbf{A}\cap\mathbf{B}}$ *is consistent with* $\psi_{\mathbf{A}}$*. Furthermore, if before the propagation,* $\psi_{\mathbf{A}\cap\mathbf{B}}$ *and* $\psi_{\mathbf{B}}$ *were consistent, then the resulting* $\psi'_{\mathbf{B}}$ *is also consistent with* $\psi_{\mathbf{A}}$ *and with* $\psi'_{\mathbf{A}\cap\mathbf{B}}$*.*

Proof: By Axiom 3, $m(\psi'_{\mathbf{A}\cap\mathbf{B}}, \mathbf{A}) = \psi'_{\mathbf{A}\cap\mathbf{B}}$, and directly from (3.10a), $m(\psi_{\mathbf{A}}, \mathbf{A} \cap \mathbf{B}) = \psi'_{\mathbf{A}\cap\mathbf{B}}$, so by definition, $\psi'_{\mathbf{A}\cap\mathbf{B}}$ is consistent with $\psi_{\mathbf{A}}$.

Now, suppose $\psi_{\mathbf{A}\cap\mathbf{B}}$ and $\psi_{\mathbf{B}}$ are consistent before the propagation.

$$\begin{aligned}m(\psi'_{\mathbf{B}}, \mathbf{A} \cap \mathbf{B}) &= m(c(\{\psi_{\mathbf{B}}, \psi'_{\mathbf{A}\cap\mathbf{B}}\}; \{\psi_{\mathbf{A}\cap\mathbf{B}}\}), \mathbf{A} \cap \mathbf{B}) \\ &= m(c(\{\psi_{\mathbf{B}}, \psi'_{\mathbf{A}\cap\mathbf{B}}\}; \{m(\psi_{\mathbf{B}}, \mathbf{A} \cap \mathbf{B})\}), \mathbf{A} \cap \mathbf{B}) \\ &= \psi'_{\mathbf{A}\cap\mathbf{B}} \qquad \text{by Axiom 5.} \\ &= m(\psi'_{\mathbf{A}\cap\mathbf{B}}, \mathbf{B})\end{aligned}$$

so $\psi'_{\mathbf{B}}$ and $\psi'_{\mathbf{A}\cap\mathbf{B}}$ are consistent. □

Theorem 3: After a full propagation, $\psi_{\mathbf{C}} = m(\psi, \mathbf{C})$ for all local potentials $\psi_{\mathbf{C}}$ in the junction tree.

Proof: By the first part of Lemma 1, after the collect evidence step, all local edge potentials are consistent with the local potential at the end point furthest from the root. Then, as a result of this, by the second part of Lemma 1, after the distribute evidence step all neighboring nodes are consistent. Therefore, by Theorem 1, the local potentials equal the marginals of the final joint combined potential. Note that by Theorem 2, the final combined joint potential is equal to the initial combined joint potential. □

Shenoy-Shafer Axiomatization

Although the correctness of propagation within the Shenoy-Shafer framework is demonstrated in [Shenoy and Shafer, 1990], I have altered and generalized the presentation somewhat and therefore feel it is necessary to reprove the results as stated in this thesis.

Lemma 2 *Let $\psi_{\mathbf{B}_i}$ denote potentials local to $\mathbf{B}_i$. If $\mathbf{C}$ is a clique node in a junction tree that separates subtrees $\mathbf{B}_i$, $i = 1, .., K$, then*

$$m(\otimes(\{\psi_{\mathbf{B}_i} : i = 1, .., K\}), \mathbf{C}) = \otimes(\{m(\psi_{\mathbf{B}_i}, \mathbf{C}) : i = 1, .., K\})$$

Proof: For readability, I will use $\otimes$ in an infix manner here. First, consider subtrees $\mathbf{B}'_i = \mathbf{B}_i \cup \mathbf{C}$. These subtrees are also separated by $\mathbf{C}$, and $\psi_{\mathbf{B}_i}$ is also local to $\mathbf{B}'_i$. By the junction tree property, if $\mathbf{C}$ separates $\mathbf{B}'_i$ from $\mathbf{B}'_j$, then $\mathbf{C} \supset \mathbf{B}'_i \cap \mathbf{B}'_j$, therefore, $\mathbf{C} = \mathbf{B}'_i \cap \mathbf{B}'_j$ for $i, j \in \{1, .., K\}$.

First, consider only two of the subtrees, say $\mathbf{B}_1$ and $\mathbf{B}_2$.

$$\begin{aligned}
m(\psi_{\mathbf{B}'_1} \otimes \psi_{\mathbf{B}'_2}, \mathbf{C}) &= m(\psi_{\mathbf{B}'_1} \otimes \psi_{\mathbf{B}'_2}, \mathbf{B}'_2 \cap \mathbf{C}) && ; \mathbf{C} = \mathbf{B}'_2 \cap \mathbf{C} \\
&= m(m(\psi_{\mathbf{B}'_1}, \mathbf{B}'_2) \otimes \psi_{\mathbf{B}'_2}, \mathbf{C}) && ; \text{ by Axioms 1,3, 8} \\
&= m(m(\psi_{\mathbf{B}'_1}, \mathbf{B}'_1 \cap \mathbf{B}'_2) \otimes \psi_{\mathbf{B}'_2}, \mathbf{C}) && ; \psi_{\mathbf{B}'_1} \text{ local to } \mathbf{B}'_1, \text{ Axiom 3} \\
&= m(m(\psi_{\mathbf{B}'_1}, \mathbf{C}) \otimes \psi_{\mathbf{B}'_2}, \mathbf{C}) && ; \mathbf{C} = \mathbf{B}'_1 \cap \mathbf{B}'_2 \\
&= m(\psi_{\mathbf{B}'_1}, \mathbf{C}) \otimes m(\psi_{\mathbf{B}'_2}, \mathbf{C}) && ; \text{ Axioms 1,8}
\end{aligned}$$

Now, taking $\psi_{\mathbf{B}'_1} \otimes \psi_{\mathbf{B}'_2}$ as a single potential (i.e., $\mathbf{B}'_1 \cup \mathbf{B}'_2$ as a single subtree), Axiom 7 extends this to K subtrees. □

Proposition 2: Let $\psi_{\mathbf{B}_i} = \otimes(\{\psi_{\mathbf{C}} : \mathbf{C} \in \mathbf{B}_i\})$, where $\psi_{\mathbf{C}}$ are the initial local potentials. If $\psi_{\mathbf{B}_i \to \mathbf{A}_i} = m(\psi_{\mathbf{B}_i}, \mathbf{A}_i)$ for all edges shown in Figure 3.11, then after the propagation step in (3.12),

$$\psi'_{\mathbf{B}\to\mathbf{A}} = m(\psi_{\mathbf{B}}, \mathbf{A})$$

where $\psi_{\mathbf{B}} = \otimes(\{\psi_{\mathbf{C}} : \mathbf{C} \in \mathbf{B}\})$ is the initial potential on subtree $\mathbf{B}$.

Proof: First, let $\psi_{\mathbf{B}_i}$ denote the initial potential $\otimes(\{\psi_{\mathbf{C}'} : \mathbf{C}' \in \mathcal{C}, \mathbf{C}' \subset \mathbf{B}_i\})$, and similarly for $\psi_{\mathbf{B}}$. By the assumptions in the theorem, $\psi_{\mathbf{B}_i \to \mathbf{A}_i} = m(\psi_{\mathbf{B}_i}, \mathbf{A}_i)$.

In Figure 3.11, the junction tree property ensures that $\mathbf{A}_i = \mathbf{B}_i \cap \mathbf{C}$, so that

$$\begin{aligned}
m(\psi_{\mathbf{B}_i}, \mathbf{A}_i) &= m(\psi_{\mathbf{B}_i}, \mathbf{B}_i \cap \mathbf{C}) \\
&= m(m(\psi_{\mathbf{B}_i}, \mathbf{B}_i), \mathbf{C}) && \text{by Axiom 3} \\
&= m(\psi_{\mathbf{B}_i}, \mathbf{C})
\end{aligned}$$

Thus, from this and the assumptions of the theorem,

$$\psi_{\mathbf{B}_i \to \mathbf{A}_i} = m(\psi_{\mathbf{B}_i}, \mathbf{C}) \tag{3.13}$$

We have

$$\begin{aligned}
\psi'_{\mathbf{B}\to\mathbf{A}} &= m(\otimes(\{\psi_{\mathbf{B}_i\to\mathbf{A}_i}, i=1,..,K\}\cup\{\psi_{\mathbf{B}\to\mathbf{A}}\}),\mathbf{A}) && \text{; Equation (3.12)}\\
&= \psi_{\mathbf{B}\to\mathbf{A}}\otimes m(\otimes(\{\psi_{\mathbf{B}_i\to\mathbf{A}_i} : i=1,..,K\}),\mathbf{A}) && \text{; Axiom 8}\\
&= \psi_{\mathbf{B}\to\mathbf{A}}\otimes m(\otimes(\{m(\psi_{\mathbf{B}_i},\mathbf{C}) : i=1,..,K\},\mathbf{A}) && \text{; by (3.13)}\\
&= \psi_{\mathbf{B}\to\mathbf{A}}\otimes m(m(\otimes(\{\psi_{\mathbf{B}_i} : i=1,..,K\}),\mathbf{C}),\mathbf{A}) && \text{; by Lemma 2}\\
&= m(\psi_{\mathbf{C}},\mathbf{A})\otimes m(\otimes(\{\psi_{\mathbf{B}_i} : i=1,..,K\}),\mathbf{A}) && \text{; Init., Axiom 3, } \mathbf{A}=\mathbf{A}\cap\mathbf{C}\\
&= m(\otimes(\{\psi_{\mathbf{C}} : \mathbf{C}\in\mathcal{C}, \mathbf{C}\subset\mathbf{B}\}),\mathbf{A}) && \text{; Axiom 3 \& def. of } \psi_{\mathbf{B}_i}\\
&= m(\psi_{\mathbf{B}},\mathbf{A}) && \text{; Def. } \psi_{\mathbf{B}}
\end{aligned}$$

□

Theorem 4: After a full proposition, $\psi_{\mathbf{B}\to\mathbf{A}} = m(\psi_{\mathbf{B}},\mathbf{A})$ for any edge potential.

Proof: Assume a full propagation starts at the leaves, propagates to a root node in the junction tree, and then propagates away from the root node. Whenever an edge $\psi_{\mathbf{B}\to\mathbf{A}}$ is updated, the edges $\psi_{\mathbf{B}_i\to\mathbf{A}_i}$ in Figure 3.11 will have already been computed. Since initialization sets $\psi_{\mathbf{B}\to\mathbf{A}} = m(\psi_{\mathbf{B}},\mathbf{A})$ when $\mathbf{B}$ is a leaf, this means that the invariant in Proposition 2 always holds while the full propagation is taking place. □

Theorem 5: Let $\mathbf{C}$ be a clique with edges $\mathbf{A}_1,..,\mathbf{A}_K$, and associated subtrees $\mathbf{B}_1,..,\mathbf{B}_K$. Let $\psi_{\mathbf{B}_i\to\mathbf{A}_i}$ denote the edge potentials after propagation has completed. Then

$$m(\psi,\mathbf{C}) = \otimes(\{\psi_{\mathbf{B}_i\to\mathbf{A}_i} : i=1,..,K\})$$

Proof: Let $\mathbf{A}_i$ be all edges emanating from $\mathbf{C}$, and $\mathbf{B}_i$ the associated subtrees. Again let $\psi_{\mathbf{B}_i}$ and $\psi_{\mathbf{C}}$ refer to initial local potential assignments (before propagation). Let $\psi_{\mathbf{B}\to\mathbf{A}}, \psi_{\mathbf{B}_i\to\mathbf{A}_i}$ refer to edge potentials after the full propagation.

$$\begin{aligned}
\otimes(\{\psi_{\mathbf{B}_i\to\mathbf{A}_i} : i=1,..,K\}) &= \otimes(\{m(\psi_{\mathbf{B}_i},\mathbf{A}_i) : i=1,..,K\}) && \text{; By Theorem 4}\\
&= \otimes(\{m(\psi_{\mathbf{B}_i},\mathbf{C}) : i=1,..,K\}) && \text{; Since } \mathbf{C}\cap\mathbf{B}_i=\mathbf{A}_i\\
&= m(\otimes(\{\psi_{\mathbf{B}_i} : i=1,..,K\}),\mathbf{C}) && \text{; by Lemma 2}\\
&= m(\otimes(\otimes(\{\psi_{\mathbf{C}'} : \mathbf{C}'\in\mathcal{C}, \mathbf{C}'\subset\mathbf{B}_i\}) : i=1,..,K\}),\mathbf{C}) && \text{; Def. of } \psi_{\mathbf{B}_i}.\\
&= m(\otimes(\{\psi_{\mathbf{C}'} : \mathbf{C}'\in\mathcal{C}\}),\mathbf{C}) && \text{; by Axioms 7,9}\\
&= m(\psi,\mathbf{C}) && \text{; Initialization}
\end{aligned}$$

□

Theorem 6: The asymptotic distribution of configurations visited by Gibbs sampling approaches p whenever

1. p assigns a positive probability density to every joint configuration.
2. All variables are visited in a homogeneous sampling pattern.

Proof: Proof is given in text, immediately following theorem statement, or see [Tierney, 1994, Section 3]. □

Chapter 4

Iterative Dynamic Discretization

In Chapter 3, the inherent structure of the time-series segmentation formulation was harnessed to decompose an enormous optimization problem into a collection of smaller (3-dimensional) optimization problems. Although structural decomposition results in a substantial simplification, the resulting subproblems are still too large to solve exactly. The remaining difficulty in this case is due to the presence of continuous variables representing transition times (the t_i's). So, while each subproblem involves only three variables, there is still an infinite number of possible instantiations under consideration for each subproblem. Furthermore, the distributions involved in each local subproblem are not restricted to any simple parametric family. For example, the evaluation of data fit can jump around substantially as a function of proposed transition time, and the shape recognizers of the HSSMM (Section 2.2.2) must be called to evaluate each proposed transition time. This disqualifies the use of existing parametric schemes for handling real-valued variables in probabilistic graphical networks (e.g., [Kiiveri *et al.*, 1984], [Lauritzen and Wermuth, 1989], [Kenley, 1986], [Shachter and Kenley, 1989], [Whittaker, 1990], [Geiger and Heckerman, 1994a], [Buntine, 1994], [Driver and Morrell, 1995]).

This chapter introduces and explores the use of *iterative dynamic discretization*. The technique is applicable for solving general Bayesian networks with continuous variables and arbitrary distributions. The same technique can also be applied to a discrete variable with a very large number of possible values, although I do not experiment with such a case in this thesis. Iterative dynamic discretization selects a finite number of possible values for each continuous variable in a network and constructs an approximate model by substituting these finite-valued variables for the original continuous ones. Standard nonparametric methods can thus be applied to solve the discretized problem exactly. The choice of discretization is *dynamic* in that information available at run-time influences the choice of the possible values for each continuous variable, and it is also *iterative* in that the discretization is successively improved based on the information learned from previous iterations. It therefore qualifies as a special case of *iterative model construction* (Section 1.8.3).

4.1 Desired Discretization

A natural way of handling continuous variables in a probabilistic model is to finely and uniformly discretize the possible values that a variable can take on, and to use standard nonparametric techniques to compute quantities within the model. However, as the number of possible discrete values that a variable can take on grows, the complexity of inference algorithms for solving graphical networks also grows substantially, making the fine-grained uniform discretization an unattractive approach in most circumstances. Additionally, when a variable is not bounded, additional complications arise in choosing the range of the variable to discretize.

Often, a small number of discrete values may suffice as a replacement for a real-valued variable, provided that "good" discrete values are used. For example, if an unobserved variable is to represent body temperature, a uniform discretization of all temperatures between $28°C$ and $44°C$ at a $4°C$ resolution would not be an optimal choice. It would be better to place many values around $37°C$ spaced very closely together, with only a few values distant from $37°C$. This would allow us, with the same number of discrete points, to obtain a higher resolution where it is likely to matter. However, if a patient is already showing extreme flu

symptoms, it may be better increase the resolution around $40°C$ and decrease it elsewhere. This exemplifies a *dynamic nonuniform* discretization.

The challenge to choosing a nonuniform discretization is to identify where the resolution should be the greatest, and by how much. As the above example highlights, the appropriate discretization depends on the specific problem instance. However, the problem has an inherent chicken-and-egg flavor to it. If we could know in advance where the posterior probability mass would fall, we would discretize more finely where the posterior probability mass is most highly concentrated. Of course, if we know the posterior probabilities, there would be no need to discretize in the first place.

> *A good discretization has two desirable properties: The number of points is small and the points are placed where it matters.*

A primary technical challenge of this chapter, and to some extent Chapters 5 and 6, is to make the phrase "where it matters" in the above statement precise. Ultimately, "where it matters" is tied to algorithm performance — points should be chosen as to maximize the algorithm's ability and efficiency of finding good solutions. However, because I am casting the task in an *anytime* framework (term coined by [Dean and Boddy, 1988]), whereby the algorithm always makes available the best solution found so far and successively improves its solution over a nonpredetermined amount of time, notions of performance are clearly not unique. The maximization of short-term versus long-term algorithm performance is often in conflict, and the best tradeoff may depend on other circumstances surrounding the application.

Even if a precise performance evaluation can be appropriately specified for an application, the way in which this maps into the design decisions in an algorithm is seldom obvious, making a direct specification of a performance evaluation nonoperational and thus not-directly-usable in practice. A tenet of the discretization approach is that the concept of "where it matters" is a more down-to-earth concept that often can be made operational. This opens up the possibility for proposing multiple precise notions for what "where it matters" could mean, thus focusing the endeavor of developing discretization algorithms into an investigation of how various conceptions of "where it matters" translate into performance, either theoretically or empirically.

In the present case, the ultimate goal is to identify a global *maximum a posteriori* (or nearly MAP) configuration for the original continuous network. Ideally, the chosen discretization would include at least one such configuration. If so, it will be a MAP configuration in the discrete network and any exact algorithm on the discrete network will find it. This consideration nails down the notion of "where it matters" somewhat since universally, regardless of specific performance evaluations for other cases, any discretization containing a MAP configuration has chosen points where it matters. As more discretized values are included, the chance of including a good (nearly MAP) configuration increases (although this may have positive or negative impact on performance evaluations while the algorithm is searching through intermediate configurations). Frame size must be traded off against increasing computational requirements. Here I assume that the number of points in the discretization (m_i) is provided by the user. ([Wellman and Liu, 1994] addresses this tradeoff by increasing m_i over time in an anytime framework.) Our task is to find, for each variable $\mathbf{x}_i$, the m_i best points to include in the discretization.

4.2 A Framework for Selecting a Discretization

Consider how one might pick a reasonable discretization. Let $\mathbf{x}$ be a continuous variable. In the time-series segmentation application, $\mathbf{x}$ corresponds to a single transition time, t_i. Suppose $f(x)$ is a (subjective) probability density function encoding an estimation of what values appear promising. A larger value of $f(x)$ indicates that x is more promising, and $f(x_1) = 2f(x_2)$ would indicate that x_1 is twice as promising as x_1. A (possible) semantics for defining "more promising" precisely is that $f(x)$ is the belief that $\mathbf{x} = x$ is the optimum (MAP) configuration. For this to be mathematically meaningful, it is necessary to assume that a unique optimum MAP configuration exists, which does not necessarily have to be the case, but which I will assume so that this semantics is meaningful. The origin of such an estimate is a central topic of the remainder of this chapter, but for now simply suppose $f(x)$ is given. For example, in Figure 4.2 a possible $f(x)$ is plotted. Because $f(x)$ summarizes our knowledge about $\mathbf{x}$, it forms a natural basis for choosing a new discretization.

1. Choose an initial discretization for each continuous variable.
2. Initialize the parameters (i.e., the junction-tree potentials) of the discrete model.
3. Solve the discrete model using the exact methods of Chapter 3.
4. Post the optimum solution if this is the best found so far.
5. Select a variable, x_i, to rediscretize.
6. Estimate $f(x_i)$.
7. Use $f(x_i)$ to select a new discretization ($\Omega_{\hat{\mathbf{x}}_i}$) for $\mathbf{x}_i$. This yields a new discrete model.
8. Go to 2.

Figure 4.1: The iterative dynamic discretization algorithm (template).

The estimate $f(x)$ provides us with a framework for considering a whole collection of iterative dynamic discretization algorithms, and for understanding the relationship between possible algorithms. Different algorithms may differ in how they derive this estimate, how they use it to select a new discretization, or how they choose which variable(s) to rediscretize next.

4.2.1 The Algorithm Template

The iterative dynamic discretization algorithm (better described as the "algorithm template") is given in Figure 4.1. This description actually specifies a collection of algorithms. Each step of the algorithm template can be instantiated in multiple different ways (except, of course, Steps 4 and 8.). For example, there are different ways one might estimate $f(x_i)$ in Step 6, each of which yields a different algorithm. Different exact algorithms could be used for Step 3, but shouldn't have any influence on the end result. Here I very briefly consider possible choices for a few of the key steps of the algorithm to give the reader a basic idea of the variations possible. The remainder of this chapter explores possible variations in depth.

Choosing an Initial Discretization

(Step 1) In some domains (but not in the time-series segmentation domain), variables may have compact domains (i.e., the values they can take on is bounded), and any combination of value assignments to the individual variables denotes a legitimate configuration. In such a case, an obvious initial discretization would be to simply use a uniform discretization of each variable.

A better initial discretization may provide better solutions in a shorter amount of time, so one may be motivated to utilize more sophisticated methods for this step. A more sophisticated algorithm might take the distributions in the continuous model into account in some fashion. Also, since the variables in the time-series task are neither bounded nor logically independent in this sense, a slightly more sophisticated method is required.

Section 4.5 presents a growing method used by the remaining algorithms in this thesis. In this method, a network is grown successively, with each variable rediscretized based on what looks promising given the subnetwork grown to that point (the same estimate used in Step 6 of the algorithm). The growing method is the only variant for the initial discretization step explored in this thesis.

Selecting a Variable (or Variables) to Rediscretize

(Step 5) There are at least three natural methods for implementing Step 5.

A first method selects a single variable on each pass through the algorithm, selecting the variables in their natural order (i.e., x_1 on the first pass, x_2 on the second pass, etc.).

A second method is to select a single variable randomly each time through the loop.

A third method is to rediscretize all continuous variables in the network simultaneously. This has the computational advantage of having to perform only one propagation per total iteration (where one total iteration corresponds to rediscretizing every variable in the network once). However, it has the disadvantage that each rediscretization is based on less information than a one-by-one algorithm. The simultaneous method can introduce numerical instabilities that must be addressed, and may cause constraints between this step and other steps. In this thesis, we constrain the possible choices for Step 7 when simultaneously rediscretizing in order to ensure numerical stability. See Page 113 in Section 6.6.

In this thesis, all of these variations are explored at one time or another. The theoretical analysis of Chapter 5 covers the first and second methods (the third case is too difficult to analyze). The empirical investigation in Chapter 6 explores the first and third methods (the motivation for exploring the third method is the computational savings).

Estimating $f(x_i)$

(Step 6) The challenge here is to utilize the information obtained from solving or analyzing the discrete model to derive a heuristic estimate in the form of a function $f(x_i)$ for what values of x_i appear promising. Conceptually, Step 6 is the most difficult step of the algorithm.

Two basic methods are discussed in Section 4.4: the *parents' posteriors method* and the *Markov boundary's posteriors method.* These are then augmented by three possible alternative weighting schemes in Section 4.4.3 (uniform, sum, and max), yielding six total variations. The thesis, however, barely scratches the surface of all possible variations for this step.

The intuition behind the Markov boundary's posteriors method is to set $f(x_i)$ equal to the posterior distribution, $p(x_i|data)$. However, only a discrete version of the latter is available, so various approximations are introduced to estimate the posterior.

I essentially take the Markov boundary's posteriors method(s) as the most base method. The parents' posteriors method is introduced first as an expository aid to give the first-time reader an idea of how an estimate might be derived, and second, as a special case of the Markov boundary's posterior method that is applicable during the growing stage, thus simplifying the discussion of that step. Furthermore, because this is the most conceptually challenging step of the algorithm, the in-depth discussion of this step occurs after the in-depth discussion of using the estimate so that the reader has a better idea of what the estimate is about and how it is used before considering how it is estimated.

The theoretical and empirical analyses both utilize (only) the Markov boundary's posteriors method. The theoretical analyses covers all three weighting regimes (uniform, sum, max) as well as a wide range of other possibilities. The empirical investigations explore these three weighting regimes. Section 4.4.2 takes the "sum weighting" (a.k.a., discrete posterior weighting) as the most basic of the possible variations.

Using $f(x_i)$ to Pick a Discretization

(Step 7) Once an estimate of what looks promising is derived, it must be used to select a new discretization. A number of variations on this step are explored throughout the thesis.

The most basic method is *random sampling* and is discussed in Section 4.3. This is the only method covered by the asymptotic analysis, and is the method used in the "pure" version explored in Section 6.4 of the empirical investigation.

An alternative approach, area partitioning, is also discussed in Section 4.3, but not pursued in either the theoretical or empirical investigations.

A number of variations are possible for Step 7. The first is to keep the best value from the previous iteration, discussed in Section 4.3.1 and explored empirically in Section 6.5. A further variation, motivated by observed empirical behavior of the algorithm, is to include values that neighboring variables rate promising and is explored in Section 6.7.

The remaining sections of this chapter develop various possibilities for each step of the algorithm. Rather than present each step in the order they appear in Figure 4.1, the detailed discussions are presented in an order that is most convenient for exposition. This is roughly (but not exactly) the reverse order. For example, a thorough understanding of how f is to be used greatly aids in understanding the motivations

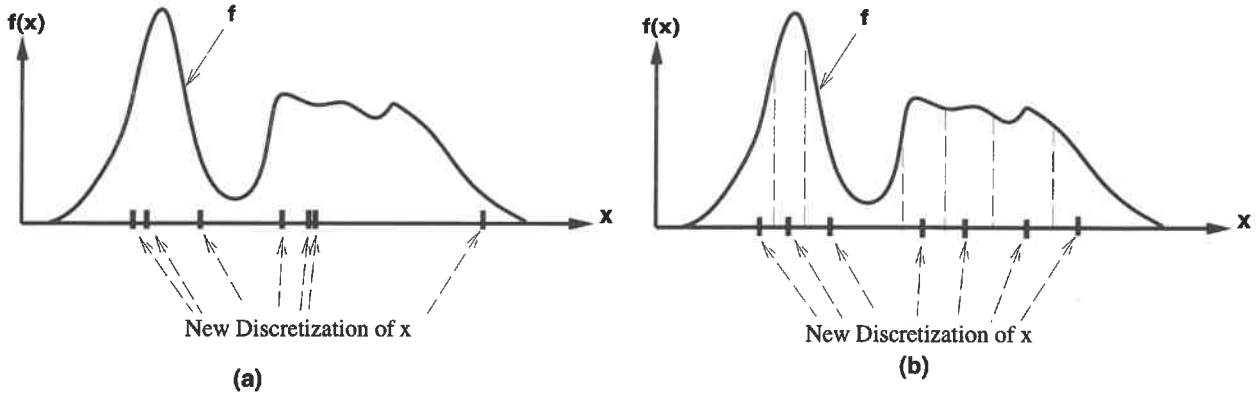


Figure 4.2: A subjective estimate for where the optimum value of $\mathbf{x}$ occurs is encoded as a pdf, shown above as $f(x)$. (a) A discretization of $\mathbf{x}$ chosen by picking $m = 7$ random values according to $f(x)$. (b) A discretization chosen by dividing the probability mass into $m = 7$ equal areas and selecting the median from each region.

and algorithms behind the estimation of f, and this in turn provides the background for understanding the growing process used to select the initial discretization.

4.3 Using an Estimate f

The most natural way to use $f(x)$ is to randomly choose m independent values for $\mathbf{x}$ according to the distribution $f(x)$. Thus, values of $\mathbf{x}$ are picked according to the probability of them being optimal. A discretization chosen in this fashion is shown in Figure 4.2(a).

A second seemingly reasonable way to use $f(x)$ is to partition the area under $f(x)$ into m equal areas and include one representative point (the median) from each of those areas in the discretization. A discretization chosen in this fashion is shown in Figure 4.2(b).

Both the random and area partitioning methods tend to concentrate points more densely where f is large, obtaining better resolution where it estimated to be most promising. However, the random method has several advantages over the area partitioning method. First, it is more widely applicable since the area partitioning method requires the possible values of a variable to be totally ordered, while the random method can be applied in general, for example to multi-variate or (unordered) categorical variables. Second, since the techniques are being applied in an iterative context, randomness leads to a more thorough exploration of all possibilities over repeated iterations, while area partitioning must rely on a change to f to explore alternatives across iterations. This makes the asymptotic behavior of the random method potentially easier to understand. Also, there are degenerate cases where a multimodal shape of f can trick area partitioning into selecting points where f is small (although these are not of much concern in practice). Third, when the methods are applied to arbitrary shaped distributions, area partitioning requires the numeric evaluation of a number of integrals (actually, of inverse integrals) in order to locate the desired points. This can be quite costly computationally. The random method requires only that samples can be drawn from f efficiently. While inverse integrations can also be used to accomplish this, many other methods for drawing values from arbitrary distributions exist ([Devroye, 1986]) and can be harnessed when they are more efficient.

In most situations we expect two very close values of $\mathbf{x}$ to have very similar posterior probabilities; therefore, it would be wasteful to pick two values very close together. This can happen with the random sampling method as evidenced in Figure 4.2(a). This observation would appear to give area partitioning at least one advantage over the random method. However, to the extent that this is a concern, the random method can be modified in a number of obvious ways to introduce a "spreading-out bias." For example, km points can be chosen ($k \geq 1$), sorted, and then only every k^{th} point kept (starting with the $\lceil k/2 \rceil^{th}$ point).

The end result has two points very close together only if $k+1$ or more draws are very close together, an event that becomes very unlikely with increasing k. As k is made larger, the spreading-out bias becomes stronger, approaching the area partitioning method with probability 1 as $k \longrightarrow \infty$. It should be noted that the use of such a bias distorts the effective distribution that the points are being drawn from as well as making the draws dependent. Also, like the area partitioning method, this use of a spreading out bias requires an ordered domain, and therefore is slightly less general than pure random sampling.

In the results reported in this thesis, I have used only the random method. During the course of development, I did utilize the area partitioning method quite extensively, but in the end, particularly over multiple iterations, I had better results with the random method. I then turned to techniques and particular choices of distributions that could not feasibly be used with the area partitioning method (because of the complexities of computing an integration on these distributions), but for which I was able to utilize the random methods. As a result, pragmatically the two became incomparable. During development, I have also experimented with various spreading-out biases with the random method and actually (counter to my intuition) found them to hurt performance. Roughly speaking, these biases tend to flatten peaks and fatten valleys and tails in the effective sampling distribution. The decrease in performance is an indicator that the waste due to points being chosen too close together is negligible compared to the distortions to the distributions introduced by spreading out biases. For larger m, these biases have virtually no effect. Since they only complicate the algorithms and no benefit has yet become apparent, I also limit attention in what follows only to the pure random method.

In a full probabilistic model, there are typically many real-valued variables that must be discretized. In the time-series segmentation application, all the transition times, $t_1, ..., t_k$, must be discretized. I examine two basic approaches. The first treats one variable at a time. f is estimated for a single variable, perhaps utilizing all information available in the full network, and the single variable is discretized. The estimate f is re-estimated (for the next variable to be rediscretized) at each step based on the new discretization of the previous variable. The second approach is to re-discretize all variables in the network on the same step. In general, we can imagine f to be a subjective distribution over the joint space of all the continuous variables, specifying the belief that the joint assignment is the optimum configuration. However, for practicality, the variables are treated independently. In other words, a separate $f(t_i)$ is estimated independently for each variable, so that each continuous variable can be discretized individually. This is akin to assuming that the joint f is the product of the individual f_i's, i.e., $f(t_1, ..., f_k) = \prod_i f(t_i)$.

The subjective probability estimate provides a general framework for exploring iterative discretization methods. A method is characterized by the way it derives $f(x)$. Different bases for obtaining $f(x)$ result in different discretization algorithms. These are compared in this chapter by using the random sampling method for selecting a discretization from $f(x)$. Clearly there are many possible approaches to discretizing a continuous variable that do not require an explicit representation of $f(x)$. However, a great many of these approaches are equivalent to algorithms that chose their discretization from an f estimate, and can thus be characterized by the effective f used in the equivalent algorithm. In this way, the framework based on $f(x)$ provides a common fabric that is useful conceptually and for comparing possible algorithms.

4.3.1 Keeping the Previous Best Configuration

Once potentials have been propagated for a given discretization, the optimal segmentation (configuration) relative to that discretization can easily be read off. Since the discretization process is iterative, it is worthwhile to include the best point from the previous iteration in a new discretization. When this is done, then the current best segmentation can only improve (or stay the same) with increasing iterations. This introduces one more variant for Step 7 of the algorithm.

There is another good reason for preserving the best point when reframing. In some cases, a subtle numerical instability can arise. For example, if all variables are (re)discretized simultaneously, it is possible (although somewhat rare) for the latest point chosen for transition i by the (re)discretization algorithm to occur before the earliest point for transition $i-1$. This is due to the random variation inherent in random sampling when the distributions for t_{i-1} and t_i overlap. The probability of a segmentation with $t_i < t_{i-1}$ is zero, so when this occurs, a singularity is encountered: all (discrete) segmentations evaluate to zero. Although a number of sophisticated algorithms are possible for preventing this singularity from occuring,

the method of keeping the best segmentation found so far is very easy to implement and provides a very robust solution to singularities of this nature. In all experiments later in this chapter where such a singularity is possible, the point from the best configuration is always kept.

4.3.2 Initializing Discrete Potentials

Once all the continuous variables in the model have been discretized, a discrete probabilistic model is constructed by filling in probabilities in the discrete model based on the probability densities in the continuous model. In actuality, it is most convenient to do this at the junction tree level. The potentials at each node of the junction tree are now simply nonparametric arrays with only a finite number of entries. These potentials are initialized according to (3.3, page 54), storing only the elements corresponding to the discretization. This initialization preserves the relative evaluations between the possible discrete segmentations. In fact, *any* initialization that preserves relative evaluations can differ from this initialization by only a constant factor.

Because it is important to maintain the relative evaluations (or at least the ordering of evaluations), this initialization appears to be the only reasonable choice for the optimization problem at hand. However, it does not compensate in any way for the spacing between discrete points. The initialization treats the discretization as if it is evidence that the true segmentation is known to be one of the segmentations allowed by the discretization (thus, we can talk of the optimal segmentation given the discretization). An alternative might be to initialize the potentials by regarding each point as representing a region of values, so that potentials on points in close proximity to other points are reduced (since they correspond to a smaller area), while potentials on isolated points are increased. Such an approach might be more appropriate for the task of estimating true posterior probabilities, but it also introduces a myriad of unsolved problems: Markov assumptions are violated and additional measures over the space of possible values are required but not available in any obvious way. Since there are other conceivable (albeit undeveloped) alternatives such as these for initialization, the point-based initialization used here is clearly one of the most distinguishing aspects of my approach.

4.3.3 Sampling from f

Once a subjective estimate f is obtained, the basic operation of the random method for selecting a discretization is that of drawing a sample randomly and independently according to f. Depending on the nature of f, this is not always an easy operation. However, for the techniques of this chapter to be applied, it is critical for this operation to be performed efficiently.

There are several approaches to drawing a sample (random variate) from an arbitrary distribution. The 840 page book by Devroye ([Devroye, 1986]) is the leading reference on the topic, and demonstrates the richness of this topic area in and of itself. All the material in this subsection is covered in that text. Section 4.4.2 considers the problem of drawing samples efficiently from the particularly complicated f that arises in the time-series segmentation problem.

When applying the techniques in this chapter to other domains, the problem of efficiently drawing a sample from f must be addressed. Here I will very briefly review some of the most basic tools for this that one should be aware of. This review is by no means comprehensive. A combination of all these methods for the implementation of the algorithm in this thesis (see the subsection "Sampling from f_{mbp}" in Section 4.4.2).

Inversion Method

The inversion method is based on the simple observation that if $F(x)$ is the cdf of $f(x)$, and $\mathbf{x}$ is univariate, then $x = F^{-1}(u)$ is a sample drawn from f when u is drawn uniformly from $[0, 1]$. This method works for any f, but requires an efficient evaluation of the inverse cdf. Note that if the cdf can be computed efficiently, a binary search can be used to compute F^{-1}, with the number of iterations being logarithmic in the desired resolution. For complicated f, the cdf usually requires numerical integration methods, and is often costly to evaluate.

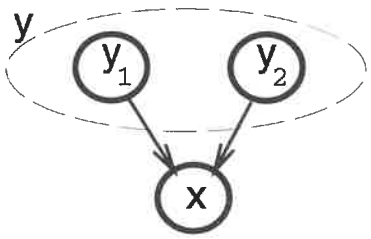


Figure 4.3: The continuous variable **x** is to be discretized.

Figure 4.4: A three variable model. The influence of **y** on **x** is very diffuse, while the influence of **x** on **z** is quite strong. The parents' posterior method does not use given information about **z** as effectively as it could.

the available time-series data and assuming the discrete values in the model are the only possible values for the variables. In addition, the original probabilistic model contains $p(\mathbf{x}|\mathbf{y})$ — the continuous distribution over $\Omega_{\mathbf{X}}$ conditioned on its parents. In the HSSMM, for example, $p(t_{i+1}|t_i, s_i) = c_{s_i}(t_{i+1} - t_i)$. These items of information can be combined as a mixture:

$$f_{pp}(x) = \sum_{\hat{y} \in \Omega_{\hat{\mathbf{y}}}} Pr(\hat{y}|data)p(x|\hat{y}) \tag{4.1}$$

This is a mixture of the possible distributions of **x** given its parents weighted by the posterior probability for each discrete value its parent's can take on as determined by the parents' current discretization. $f(x)$ obtained by (4.1) is a continuous distribution over $\Omega_{\mathbf{X}}$ which can be used to obtain a new discretization (Section 4.2). Random variates from f_{pp} can be generated using the mixture method (Section 4.3.3, Page 83). First, $\hat{y}$ is drawn according to the computed potentials $p(\hat{y}|data)$, then a random variate is drawn from $p(x|\hat{y})$. Recall from Chapter 2, Page 31, that the HSSMM requires that waiting-time distributions support efficient random variate generation, so f_{pp} is a very efficient method for discretizing a variable.

Drawbacks

The parents' posteriors method for obtaining $f(x)$ suffers from a number of problems. It does not make full use of evidence observed for descendants of **x**, it can be adversely biased by the particular nonuniform discretization of the parents, and it can over-commit to intermediate computations. The first two points are discussed here while the last point is discussed in Section 4.5.

A very simple example illustrates the first drawback. A three variable model is depicted in Figure 4.4, where **x**, **y** and **z** are all real-valued. **x** has one parent and one child. A continuous probabilistic model is specified by the following distributions:

$$\begin{aligned} p(y) &= GammaDist[\alpha = 10, \beta = 10](y) \\ p(x|y) &= GammaDist[\alpha = \frac{y}{1000}, \beta = 1000](x) \\ p(z|x) &= GammaDist[\alpha = x, \beta = 1](z) \end{aligned}$$

Furthermore, in the current problem instance, **z** is observed to have a value $z = 500$. The thing to note about the example is that the connection from **y** to **x** is very diffuse — **x** given **y** has a variance of $1,000 \cdot y$ with a mean of only y, while the connection from **y** to **z** is very strong, having a variance of x with a mean x.

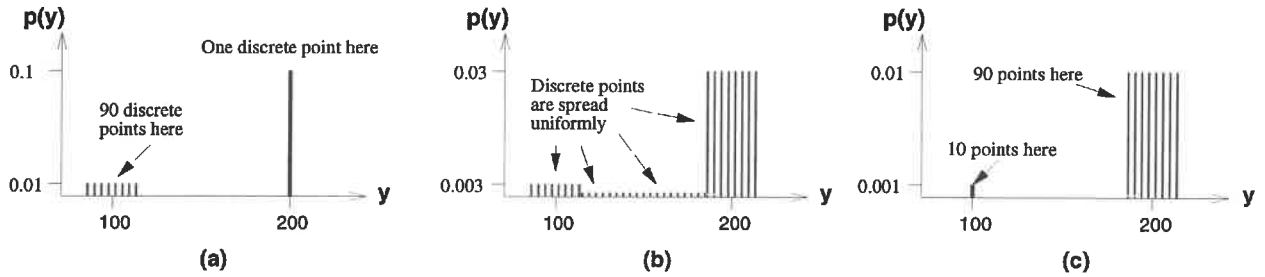


Figure 4.5: The discrete posteriors over $\mathbf{y}$ (the parent of $\mathbf{x}$). These posteriors are used by the parents' posteriors method to discretize $\mathbf{x}$. (a) 90 of the 91 discrete values for $\hat{\mathbf{y}}$ are located around $\hat{y} = 100$, while only one is located near $\hat{y} = 200$. However, the $\hat{y} = 200$ value is 10 times more likely than any of the values near $\hat{y} \approx 100$. (b) The discretization is spread more uniformly. (c) A discretization is concentrated around $\hat{y} \approx 200$.

Without carrying out the full computation to find the MAP configuration or the posterior distributions, it is immediately clear that the likely values of $\mathbf{x}$ are very close to $x = 500$. A good estimate for $f(x)$ should be centered near $x = 500$ with a relatively small variance. However, because the parents' posteriors method forms a mixture of $p(x|\hat{y})$, and every $p(x|\hat{y})$ has a huge variance, $f_{pp}(x)$ has a huge variance. In short, the method does not utilize information obtained from $\mathbf{x}$'s descendants as effectively as it could.

It should be noted, however, that the parents' posteriors method *does* utilize information from $\mathbf{x}$'s descendants — the information is not entirely overlooked. In the example, $Pr(\hat{y}|data) = Pr(\hat{y}|z = 500)$ is the weighting given to the mixture. This weighting does account for available evidence arising from the descendants of $\mathbf{x}$, and has the effect of helping to center $f(x)$ given the evidence. The primary drawback here is that the variance in $f(x)$ is excessively large when the connection to $\mathbf{x}$ from its parents is diffuse. This drawback is addressed by the method of Section 4.4.2.

A second problem of the parents' posteriors method is that the density of the discretization for the parents influences $f(x)$ in an undesirable fashion. Figure 4.5(a) shows a discrete distribution for $\hat{\mathbf{y}}$. $\mathbf{y}$ has been discretized to 91 discrete values, 90 of which are placed in close proximity around $y = 100$, and one of which is at a significantly larger value, $y = 200$, but which has a posterior probability 10 times greater than any of the other discrete values for $\hat{y}$. Despite this, 90% of the total probability mass is located near $y = 100$, so despite the fact that $y = 200$ appears very likely, it has a relatively minor contribution to $f(x)$. If, on the other hand, the discretization for $\hat{\mathbf{y}}$ is spread uniformly, as in Figure 4.5(b), the region around $y = 200$ predominantly determines $f(x)$. In short, it is not just the posterior distribution for $\mathbf{y}$ that weights the mixture for $f(x)$, the current discretization for $\mathbf{y}$ indirectly influences the mixture as well.

The iterative aspect of discretization has the potential to significantly alleviate this second drawback. If an effective re-discretization technique is performed on $\mathbf{y}$, the discretization in figure 4.5(a) should be changed to something closer to Figure 4.5(b), or even something more like Figure 4.5(c). Then, when $f(x)$ is estimated using the parents' posteriors method, the more likely values of $\mathbf{y}$ will indeed have the greatest contribution. Note, however, that the parents' posteriors method for reframing $\mathbf{y}$ will often not be effective for rediscretizing $\mathbf{y}$ in this fashion. The emphasis on more likely values can also be overdone in the other direction — i.e., the more likely values contributing more than they would using a uniform discretization.

This problem is less severe when we are searching for the MAP configuration.

Fundamentally, the parents' posteriors method aims at approximating

$$f(x) = \int p(y|data)p(x|y)dy \tag{4.2}$$

using only the discrete information available for the parents $\mathbf{y}$. One option for addressing the second drawback would be to employ some method for estimating the continuous parents' posterior, $p(y|data)$. For example, perhaps the above parents' posterior method, applied to the parents, would be one method for doing this. Any of the other methods discussed in this chapter might also be applied. Then $f(x)$ can be based directly on (4.2). The integral in (4.2), would have to be evaluated or approximated numerically whenever $f(x)$ or $F(x)$ is desired.

If the estimate for $p(y|data)$ is a good one, then using the continuous integral eliminates the bias created by nonuniform discretizations. However, if the parents' posterior is estimated using the discrete parents' posterior method, then the bias from nonuniform discretizations may exist in the estimate of $p(y|data)$, so this would serve only to push the problem back one level. It would do so at the expense of considerable computational overhead.

The recursion could be pushed all the way to the beginning of the propagation graph, but then the numerical evaluation of the integral would be comparable to solving the entire undecomposed problem directly. This would be a step in the wrong direction. If the estimate of $p(y|data)$ is based on the technique in Section 4.4.2, for example, it would not even be possible to push the recursion back to a point of termination in this fashion.

A second approach for dealing with this second drawback is to weight the components of the mixture uniformly (see Section 4.4.3) and to assume that the spacing of the discretization for y already reflects the probability density of y. This might be justified by the fact that the spacing of points were chosen on the previous iteration to reflect the density of f during that iteration. Assuming if the estimate f does not change significantly between iterations, this would also be a reasonable estimate of density for the current iteration, and thus, the spacing alone might be taken as an encoding of weighting. Experiments in Section 6.9 indicate that the uniform weighting deals with this phenomena in a fashion that does indeed translate to superior performance.

4.4.2 Using Markov Boundary's Posteriors

As discussed above, one drawback of the parents' posteriors method is that observed information for descendants of $\mathbf{x}$ does not influence the discretization of $\mathbf{x}$ as strongly as it should in some cases. The Markov boundary's posteriors method address this by directly using the posteriors from all neighbors — parents and children, or more generally the posteriors for all variables in $\mathbf{x}$'s Markov boundary, to form $f(x)$. The *Markov boundary* of $\mathbf{x}$ is the smallest set of variables in the dependency graph that renders $\mathbf{x}$ conditionally independent of all other variables in the graph. In an undirected graph, the Markov boundary consists of a variable's immediate neighbors. In a directed graph, the Markov boundary consists of $\mathbf{x}$'s parents, children and spouses, where a spouse is a variable that shares a common child ([Pearl, 1988, page 97]). Equivalently stated, the Markov boundary of a variable in a directed graph is the set of variables adjacent to the variable in the (undirected) moralized graph.

Figure 4.6 shows a portion of the dependency graph for the time-series segmentation problem (c.f. Figure 3.1). Variables t_{i-1}, s_{i-1}, s_i, and t_{i+1} are adjacent to t_i, thus forming the Markov boundary for t_i, and therefore the posteriors for these variables are to be used in forming $f(t_i)$. The Markov boundary method forms the mixture:

$$f_{mbp}(t_i) = \sum_{\substack{\hat{t}_{i-1},\hat{t}_i \\ s_{i-1},s_i}} Pr(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}|data)p(t_i|\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}, data) \tag{4.3}$$

The summation ranges over all discrete values of $\hat{t}_{i-1}$ and $\hat{t}_i$ according to the current discretization, and over all values of s_{i-1} and s_i (which are already discrete in the HSSMM). The weights for the mixture are

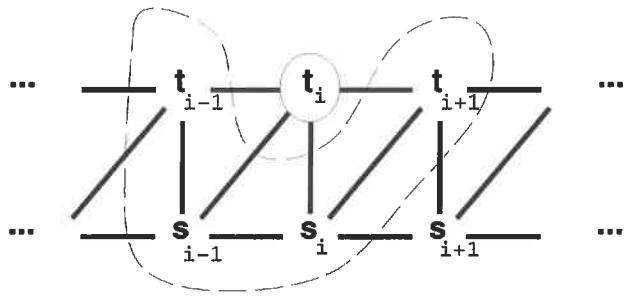


Figure 4.6: A portion of the dependency graph for the time-series segmentation problem. The Markov boundary of t_i consists of the variables immediately adjacent to t_i, in this case t_{i-1}, s_{i-1}, s_i, and t_{i+1}.

given by

$$Pr(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}|data) = \sum_{\hat{t}_i} \frac{Pr(\hat{t}_{i-1}, s_{i-1}, \hat{t}_i|data)Pr(s_{i-1}, \hat{t}_i, s_i|data)Pr(\hat{t}_i, s_i, \hat{t}_{i+1}|data)}{Pr(s_{i-1}, \hat{t}_i|data)Pr(\hat{t}_i, s_i|data)} \tag{4.4}$$

The probabilities appearing in the right hand side of (4.4) are all available as clique and separator potentials in the junction tree (see Figure 3.6 on page 51).

For the HSSMM, the mixture components are obtained as

$$p(t_i|\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}, data) = \frac{c_{s_{i-1}}(t_i - \hat{t}_{i-1})c_{s_i}(\hat{t}_{i+1} - t_i)d_{s_{i-1}}(data, \hat{t}_{i-1}, t_i)d_{s_i}(data, t_i, \hat{t}_{i+1})}{\int c_{s_{i-1}}(t_i - \hat{t}_{i-1})c_{s_i}(\hat{t}_{i+1} - t_i)d_{s_{i-1}}(data, \hat{t}_{i-1}, t_i)d_{s_i}(data, t_i, \hat{t}_{i+1})dt_i} \tag{4.5}$$

Sampling from f_{mbp}

To use f_{mbp}, it is necessary to draw random variates from f_{mbp} efficiently. Doing so is no obvious matter, but by combining all of the techniques for random variate generation discussed in Section 4.3.3, we can obtain an efficient algorithm for doing so.

The first step is to make use of the mixture form in (4.3). This requires drawing $(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1})$ jointly from the distribution $p(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}|data)$ given by (4.4). The junction-tree potentials make this a relatively easy task, since these potentials already reflect the influence of available data. The procedure, using the normalized sum potentials (Section 3.1.3) is:

1. Propagate sum potentials (Figure 3.10, on Page 57).
2. $(\hat{t}_{i-1}, s_{i-1}, \hat{t}_i) \sim \psi(\hat{t}_{i-1}, s_{i-1}, \hat{t}_i)$
3. $s_i \sim p(s_i)$ $\qquad p(s_i) = \frac{\psi(s_{i-1}, \hat{t}_i, s_i)}{\vartheta(s_{i-1}, \hat{t}_i)}$
4. $\hat{t}_{i+1} \sim p(\hat{t}_{i+1})$ $\qquad p(\hat{t}_{i+1}) = \frac{\psi(\hat{t}_i, s_i, \hat{t}_{i+1})}{\vartheta(\hat{t}_i, s_i)}$
5. Return $(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1})$

Then $(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}) \sim p(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}|data)$ as desired. Having chosen $(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1})$, t_i is drawn from $p(t_i|\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}, data)$ as given in (4.5). Next things get tricky.

One approach for drawing a sample from (4.5) is to use the product form of rejection sampling (see Section 4.3.3, Page 83). To do so, we simply draw t_i from $c_{s_{i-1}}(t_i - \hat{t}_{i-1})$, the waiting-time distribution, and use $h(t_i) = c_{s_i}(\hat{t}_{i+1} - t_i)d_{s_{i-1}}(data, \hat{t}_{i-1}, t_i)d_{s_i}(data, t_i, \hat{t}_{i+1})$ in the rejection criteria. It is also necessary to find some method for obtaining $\hat{h}$, the upper bound on $\sup_{t_i} h(t_i)$. Assuming $\hat{h}$ is obtained, the product form rejection sampling algorithm returns a random variate t_i from the desired distribution $p(t_i|\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}, data)$. Thus, $t_i \sim f_{mbp}$.

Unfortunately, the above use of product form rejection sampling is horribly inefficient in many important cases. For example, if $\hat{t}_{i-1}$ and $\hat{t}_{i+1}$ are far apart relative to $\sigma(c_{s_{i-1}})$ and $\sigma(c_{s_i})$, then almost any value that is likely to be generated by $c_{s_{i-1}}$ will be rejected by c_{s_i} with high probability. In the extremely unlikely event that a point is accepted by c_{s_i}, it must still be accepted by $d_{s_{i-1}}$ and d_{s_i}, so further rejections are likely, and if $\hat{h}$ is overestimated then even more unnecessary rejections occur, so that in total, a great many samples may be generated before one is actually accepted. As a result, the direct version of product form rejection sampling is not feasible for this problem.

An approach that does work well in this case is an adaptive version of the product form rejection sampling, except that instead of using $c_{s_{i-1}}$ as the generator, we use the product distribution $g(t_i) \propto c_{s_{i-1}}(t_i - \hat{t}_{i-1})c_{s_i}(\hat{t}_{i+1} - t_i)$ as the generator. The rejection function utilizes the shape recognizers, i.e., $h(t_i) = d_{s_{i-1}}(data, \hat{t}_{i-1}, t_i)d_{s_i}(data, t_i, \hat{t}_{i-1})$. Although $g(t_i)$ is in a product form, it would be unwise to use rejection sampling to draw from g since, as discussed above, product form rejection sampling becomes highly inefficient when $c_{s_{i-1}}$ and c_{s_i} become highly separated as can occur in the time-series segmentation application. Therefore, to draw from g other methods are utilized, discussed below.

The problem remains of determining $\hat{h}$. It is important not to be overly conservative with this estimate, otherwise efficiency is quickly sacrificed. On the other hand, if $\hat{h}$ is underestimated, the distribution from which t_i is drawn is distorted. However, in the present case, the distortion is not too evil, for as $\hat{h}$ is underestimated to a greater and greater extent, the distorted distribution approaches $g(t_i) = p(t_i|\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1})$. In other words, it simply pays less attention to the data between $\hat{t}_{i-1}$ and $\hat{t}_{i+1}$. Even in the most extreme case ($\hat{h} = 0$), this is a pretty good approximation for our purposes, especially when one recalls that f_{mbp} is already a subjective, heuristic estimate. If $\hat{h}$ is only barely underestimated, then such an approximation is more than satisfactory.

From these considerations, I have used an adaptive procedure for estimating $\hat{h}$. The approach inherently underestimates $\hat{h}$ (hence some distortion is always present), but hopefully not by much in the typical case. When a really tough case is encountered, i.e., when the data between $\hat{t}_{i-1}$ and $\hat{t}_{i+1}$ is in severe disagreement with transition probability expectations, the adaptive algorithm decays $\hat{h}$ until a sample is accepted. In this way, the algorithm ensures that a sample is drawn in a timely fashion by loosening the guarantee on what distribution it is drawn from. However, when possible, the sample will be drawn from something very close to f_{mbp}, and distortions to the distribution occur only to the extent that they are needed to ensure timeliness.

The adaptive algorithm is quite simple. To obtain an initial estimate of $\hat{h}$, a handful[1] of samples $\{x\}$ are drawn from g. The initial $\hat{h}$ is set to $\max_x h(x)$, where the max is taken over these initial samples. Then, a sample is generated as:

```
loop
    x ~ g
    u ~ U
    ĥ = γ max(ĥ, h(x))          ;γ = decay factor ≈ 0.99.
    until (uĥ ≤ γh(x))
```

The only operation that has not yet been described is the drawing of $x \sim g$. To do this, g is approximated by a piece-wise linear pdf, $\tilde{g}$, and the inversion method for random variate generation applied to $\tilde{g}$. Because $\tilde{g}$ is piece-wise linear, the inversion method is highly efficient.

The algorithm that I have used for constructing $\tilde{g}$ is shown in Figure 4.7. Once the segment boundaries are determined, the approximation is simply what one would expect. Specifically, the value of $\tilde{g}(x)$ at the point where two linear segments meet is set to be proportional to $g(x)$. The constraint that $\tilde{g}$ must be a pdf, and therefore have an area of 1, determines the constant of proportionality. The more notable aspect of the algorithm in Figure 4.7 is that the segment boundaries are not evenly spaced. Instead, the algorithm attempts to place more segments in areas where it is likely to be more critical for the approximation, where the heuristic method for determining this is spelled out in detail in Figure 4.7 with the selection of x_j. Other

[1] My implementation uses 10 initial samples.

Notation: Let $g_1(t_i) = c_{s_{i-1}}(t_i - \hat{t}_{i-1})$, $g_2(t_i) = c_{s_i}(\hat{t}_{i+1} - t_i)$, $g(t_i) \propto g_1(t_i)g_2(t_i)$.

G_1 and G_2 are the cdf's of g_1 and g_2.

$k = \#$ segments in piece-wise linear approximation, k is odd.

Initialization:

$$x'_j = \begin{cases} \hat{t}_{i-1} & \text{j=0} \\ G^{-1}\left(\frac{2j}{k-1}\right) & j = 1, .., \frac{k-1}{2} \\ G^{-1}\left(\frac{2j-k}{k-1}\right) & j = \frac{k+1}{2}, .., k-1 \\ \hat{t}_{i+1} & \text{j=k} \end{cases}$$

$x_j =$ sorted values x'_j.

$y_j = g_1(x_j)g_2(x_j), \quad j = 0, ..., k$

$A_0 = 0$

$$A_j = \frac{1}{2}(y_j + y_{j-1})(x_j - x_{j-1}) + A_{j-1}, \quad j = 1, .., k$$

Estimates: In all the following, j is such that $x_j \leq t \leq x_{j+1}$.

$$\tilde{g}(t) = \left[\frac{(t - x_j)(y_{i+1} - t_i)}{x_{i+1} - x_i} + y_i\right] / A_k$$

$$\tilde{G}(t) = \frac{1}{2}(y_{i+1} - y_i)(t - x_i) + A_i$$

$$\tilde{G}^{-1}(p) = \frac{2(p - A_i)}{y_{i+1} + y_i} + x_i$$

Random Variate Generation: Pick $u \sim \mathcal{U}$, return $\tilde{G}^{-1}(u)$.

Figure 4.7: Approximating g with a piece-wise linear pdf, $\tilde{g}$.

methods for selecting segment boundaries are obviously possible, but I have had no need to go beyond the simple scheme shown in the figure.

4.4.3 Alternative Weighing Regimes

The mixture in (4.3) that defines f_{mbp} consists of a set of components (which are continuous distributions) weighted by $Pr(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}|data)$. This weighting factor is the posterior marginal probability, also corresponding to the "sum potential" computed by propagation (see Section 3.1.3). This is not the only weighting scheme possible.

The reader should note that the definition of f_{mbp} given by (4.3) is heuristic. If the summation in (4.3) is replaced by an integration, an exact expression for $p(t_i|data, discretization)$ results, making it easy to overlook that turning the integration into a summation is only a heuristic move. It does not make that formula any more valid than another mixture with some other weighting scheme. Furthermore, the use of the discrete posterior as an estimate of f was also heuristic — f is *not* an estimate of the posterior, but instead an estimate that the value belongs to the optimum configuration. Using an approximation of the posterior for f is only a heuristic.

There are two other natural weighting schemes to consider. In each, we simply replace $Pr(\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}|data)$ with something else.

Since the weighting coefficients in (4.3) are the sum-potentials of propagation (i.e., those computed in 3.1.3), a natural alternative is to use the max-potentials (i.e., those computed in Section 3.2.4). Consider that the inference task studied here is that of finding the MAP configuration. It is not one of computing marginal distributions. Since the max-potentials directly reflects the goodness of a value relative to the optimum setting for all other variables, this more directly reflects the likelihood that a configuration is optimal than sum-potentials do. There is an added advantage as well: the step of propagating sum-potentials can be eliminated entirely. Max potentials must already be propagated to find the optimum configuration relative to the discretization, so these are already available.

Recall that all terms appearing on the right hand side of (4.4) are sum-potentials. The exact same quantities are also available for the max-potentials. The most straightforward implementation of max-potential weighting simply substitutes (normalized) max-potentials in (4.4) in place of the corresponding sum-potentials.

A second natural alternative weighting scheme to consider is to weight all components of (4.3) equally, i.e.,

$$f_{mbp}(t_i) = \sum_{\substack{\hat{t}_{i-1}, \hat{t}_i \\ s_{i-1}, s_i}} \frac{1}{\kappa} p(t_i|\hat{t}_{i-1}, s_{i-1}, s_i, \hat{t}_{i+1}, data)$$

where κ is the number of possible joint assignments to the Markov boundary. The motivation for this scheme is discussed in **Drawbacks** in Section 4.4.1 and illustrated by Figure 4.5. The idea here is that the spacing of discrete values already reflects the quality of a configuration. Possible values for neighboring values are concentrated more densely around more promising regions, so this density in effect already provides an appropriate weighting for the mixture. A similar uniform weighting is utilized in the algorithm of [Tanner and Wong, 1987], for example. The uniform weighting entirely eliminates the need to use the results of propagation to obtain f. However, the propagation of max-potentials must still be carried out since at each iteration the optimum relative to the current discretization must be identified.

4.5 The Initial Discretization: Growing the Model

The previous section outlines a process of iteratively refining discretizations based on previous choices of discretization. However, the problem remains of picking an initial discretization to bootstrap the whole process. The bootstrapping requires some special treatment.

As before, the initial discretization is chosen based on a subjective estimate, $f(x)$, using one of the techniques in Section 4.2. However, it is not possible to use $f_{mbp}(x)$ since this estimate is based on a previously chosen discretization. The estimate of $f_{pp}(x)$ is also based on a previously chosen discretization,

but only on a discretization of the parents of **x**. Thus, the variables in a directed acyclic graph can be discretized initially by growing the graph incrementally from parents to children. The same idea can be applied to an undirected graph simply by applying an artificial ordering to the variables. At each step, discrete posteriors are computed for the leaves, then $f_{pp}(x)$ is obtained for each variable **x** not yet appearing in the graph, but whose parents already appear. **x** can then be added using $f_{pp}(x)$ as the basis for discretization. For the first nodes with no parents, $f_{pp}(x)$ reduces to using the priors directly, i.e., $f_{pp}(x) = p(x)$.

In the case of the HSSMM, the growing process amounts to adding (t_i, s_i) at each iteration of the growing procedure. In time-series segmentation, there is no limit to the number of stages that can be added, so the growing procedure does not naturally terminate. However, if the last datum in the actual time-series data observed so far occurs at time T, there is a point where the probability of t_N being less than T is miniscule. Growing the chain beyond N stages would, at that point, provide little additional information about where transitions occur. Thus, pragmatically it is possible to terminate the growth even in the time-series segmentation application. Note that the actual number of variables added to the computational graph depends on the actual data. In a system that performs on-line segmentation, the chain may be grown incrementally in this fashion as new data arrives.

4.6 Control Structures

When dynamic iterative discretization is applied, several choices exist at every step of the iteration. An algorithm must decide whether to grow the graph or whether to rediscretize existing variables (or both). If variables are to be rediscretized, it must choose which variable(s) to rediscretize on this iteration. For each variable to be discretized, it must select the method for obtaining $f(x)$ (e.g., from those in Section 4.4) and the method for sampling from $f(x)$ (e.g., from those in Section 4.2). These choices need not necessary be the same for every variable in the graph. The portion of an iterative discretization algorithm that makes these choices is referred to as the control structure. An on-line system may also need to make decisions about handling incoming time-series data, for example, when to recompute, rediscretize, or grow based on the data received.

The control structure can be quite simple. One example of an iterative discretization algorithm is the following. At every step of the iteration, use the existing discrete posteriors to compute $f_{pp}(x)$ for every existing node and every node that can be added at this iteration. Then always grow the chain *and* rediscretize every variable according to $f_{pp}(x)$ using the area partitioning method. In this algorithm, the control structure is quite simply described: Always grow, always rediscretize every variable, and always use $f_{pp}(x)$ with area partitioning.

The richness of the space of possible dynamic iterative discretization algorithms in this framework is demonstrated by the following example of a more sophisticated control structure. First, decide whether a node remains to be added. In the time-series segmentation application, make this decision by comparing the time of the last data point to the time of the last transition in the optimal segmentation given the current discretized graph. If the optimal transition time occurs after the time of the final data point, do not grow the graph further, otherwise add a new transition node. If a node is added, rediscretize only the final three transition times, using a different $f(x)$ for each of them. For the added variable and for the leaves, use $f_{pp}(x)$ with uniform weighting (Section 4.4.3). For the penultimate variables (parents of a leaf) and the parents of penultimate variables, use f_{mbp}. If no node is added, rediscretize every variable using f_{mbp}.

A sophisticated control structure may be useful for addressing certain undesirable artifacts that arise from discretization; however, the richness of this space make it very difficult to study possible variations on control structures in anything close to an exhaustive manner. It should simply be recognized that variations on control structure is dimension distinguishing variations in possible algorithms.

4.7 Summary

Many forms of distributions on continuous variables cannot be propagated without a loss of information from exact methods. In some cases, the update of a local potential after a propagation step may not belong to the same parametric family as the original potential. Such a distribution on a graphical model may be

referred to as *non-conjugate*. Such models typically require the use of approximate techniques. In other cases, efficiency concerns alone may also motivate the use of an approximation technique.

Discretization is one such approach for dealing with non-conjugate cases or for obtaining approximation techniques to difficult problem instances. An exact solution to a discrete model may provide an approximate solution to the original model *if the discretization is good*. The challenge becomes one of picking a good discretization.

A good discretization should have a small number of possible values, since the number of values determines how hard it will be to solve the discrete model. When finding maximum a posteriori (MAP) configurations, as we are doing here, we would also ideally hope that the optimum configuration for the original model is included in the discretization, since if it is, the optimum answer for the discrete model will also be the optimum for the original. In practice, this latter property is not particularly useful since it is not operational. More pragmatically, it is important to allow for *non-uniform* discretizations, i.e., such that values are not necessarily spaced at equal intervals, since we hope to concentrate resolution where it matters or is most likely to help. Furthermore, discretizations should be dynamic, i.e., tailored to the problem instance (to the data observed so far, etc). These concerns make selecting a discretization essentially a chicken-and-egg problem, and therefore suggests the use of an iterative technique.

Iterative dynamic discretization, the technique described in this chapter, is based on the central idea that by solving a discrete model, we can learn something about what values are promising, and then we can use this knowledge to select a new and more informed discretization. The process can then be iterated.

The framework in this chapter has broken iterative dynamic discretization down into a few basic steps. Starting with an initial discretization, the discrete model is solved using exact methods (from Chapter 3). This information is utilized to form an estimate, f, in the form of a pdf for how promising the possible (continuous) values of a selected variable is. A variable is chosen for rediscretization, and the estimate for that variable guides the selection of new discrete values. Each of these steps can be instantiated in a number of different ways, and because of this, iterative dynamic discretization really refers to a class of algorithms. This bulk of this chapter has examined the possible instantiations for each step of the algorithm.

The following two chapters explore the properties and performance of iterative dynamic discretization. Chapter 5 examines some theoretical properties of the algorithm, while Chapter 6 explores the performance of the technique empirically, examining a variety of instantiations in order to understand the sources of power of the technique.

Chapter 5

Asymptotic Stability

Some results concerning the asymptotic behavior of iterative dynamic discretization are informative. For example, consider the following questions:

1. Does the solution produced by the algorithm in the limit depend on the initial discretization?

2. Will the algorithm always find an optimum segmentation if run long enough?

3. Can the limiting discretization(s) produced by the algorithm be characterized in a meaningful way?

4. Is there a relationship between the asymptotic behavior of the algorithm and the base distribution, p, over the space of possible configurations?

The first two of these questions are questions about the stability of the algorithm. They are answered definitively by the theoretical results in this chapter. It is shown that the same limiting behavior is obtained regardless of the starting discretization, and that the algorithm is Harris recurrent, implying that a segmentation arbitrarily close to the optimum will eventually be located. The algorithm approaches an unique ergodic distribution over the space of possible discretizations from any starting point.

The second two questions concern characterizations of the limiting behavior. Such characterizations have not been found and remain an open question. It is shown that for a two-variable model, a clean characterization is possible, but that this same characterization does not necessarily hold when there are three or more variables. The prospect of designing a new iterative dynamic discretization algorithm in which such a property holds is an intriguing possibility, but something that remains an open problem.

The main result is Theorem 7, which states that for any starting configuration, the iterative dynamic discretization algorithm approaches an unique limiting behavior in which it repeatedly visits all[1] possible discretizations, and that it visits these according to a unique limiting probability distribution, π. The main theorem states that it approaches this limiting distribution from any possible starting configuration.

The next section describes the assumptions behind the theoretical results, while the Section 5.2 contains the derivation of the Theorem, along with a gentle description of the rather advanced theory of Markov chains required for the result. Finally, Section 5.3 contains some discussion about the (open) problem of relating the asymptotic distribution to p. The treatment of iterative dynamic discretization in this chapter suggests another perspective: that iterative dynamic discretization is one way to combine Gibbs sampling with exact propagation methods, potentially harnessing the complementary strengths of each. Section 5.4 discusses this perspective.

[1] The mathematical notation in this section makes this more precise. Because the space of discretizations is continuous (uncountable), it is obviously not possible to visit all possible discretizations. The mathematically correct description is that it repeatedly visits all measurable sets of discretizations.

5.1 The Assumptions

Theoretical results require a certain precision regarding the assumptions used, variants of the algorithm allowed, etc. The more casual reader may be sufficiently satisfied with the informal description of the results given above, and may prefer to skip to Section 5.4 at this time. The slightly more sophisticated reader may wish to read this and the following subsection to understand the formal assumptions that the results depend on. The sections containing the proofs can easily be skipped over without loss of continuity.

First, I assume that each variable is discretized to m possible values. The fact that the number of values is the same for all variables is only used to simply notation, and of course, is not critical.

Second, I assume that one variable at a time is discretized at each step, and that either the variable to discretize is chosen at random or a repeating pattern of visits to all the variable is used. If variables are chosen at random, then every variable must have a positive probability of being chosen at any iteration. If they are chosen according to a repeating pattern, then every variable must occur in that pattern at least once. These are the same visitation assumptions usually assumed for Gibbs sampling[2].

When a given variable, $\mathbf{x}_i$, is discretized, I assume the random method is used to generate the m possible values. Specifically, a distribution $f(x_i)$ is obtained, and m values are chosen independently from $f(x_i)$. These become the m values of the discretization. For formal rigor, it is necessary to worry about the situation when two or more of the m values drawn are equal. If we have a bounded probability density on a continuous space (as with the time-series segmentation domain), duplicate draws occur with probability zero, so this case is of little concern. But to ensure that the results hold for finite or countable domains, and for infinite domains with impulses in the density function (provided they obey the other requirements below), it is of interest to treat this case carefully. One obvious way to handle this is to draw the m values from f without replacement, but the mathematics of this treatment are intractable. Instead, when two duplicate values are drawn, we take the set of possible values to be something less than m, and the other aspects of the iteration operate as they normally would if that were the number of actual values. Later in Section 5.3, a distribution $\hat{p}$ over the space of configurations is defined and involves randomly drawing a configuration from a discretization. With duplicate values possible, the algorithm is really visiting bags of configurations, and $\hat{p}$ is obtained by drawing $\mathbf{x}$ from one of these bags. In other words, a duplicate value doubles the odds of that value being drawn in that final step. Under this treatment of duplicate values, the results derived below hold. Again, for continuous spaces (with bounded densities) such as the one occuring with time-series segmentation, duplicate values occur with probability zero, so the need to worry about this does not arise.

Next, it is assumed that when $\mathbf{x}_i$ is discretized, the f used to generate m values is a mixture of conditional probabilities, where each component conditions on a possible value in the current discretization for *all* the other variables in the model (i.e., $\mathbf{x}_j, j \neq i$). In other words:

$$f(x_i) = \sum_{y \in Y} w_y p(x_i|y) \tag{5.1}$$

where $Y = X_1 \times ... \times X_{i-1} \times X_{i+1} \times ... \times X_n$ is the set of values in the current discretization for all variables other than x_i, and w_y is some weighting scheme placed on those values. Note that in a graphical probabilistic model, conditioning on $y \in Y$ amounts to conditioning on the variables in the Markov boundary. The results therefore cover variants of the f_{mbp} scheme, including arbitrary weightings, and not just weighting by "sum potentials". For example, the results hold for a uniform weighting or when the max potentials are used. Examples of the weighting scheme include weighting all possible values in Y equally, using the sum potentials as was done for the pure f_{mbp}, or using the "max potentials." So many variations of the iterative dynamic discretization algorithm are included under this assumption (and the results in this section hold for all of them), but for example, the use of f_{pp} is not covered by these results.

[2] It would be nice to prove the results for any sequence of one-variable-at-a-time discretizations where each variable is visited infinitely often, as is done in [Geman and Geman, 1984] for the case of Gibbs sampling on a finite sample space. However, this is not attempted here.

5.1.1 Positivity of p

A standard assumption in Gibbs sampling is that p (where p is the probability density of interest) is everywhere positive (see Section 3.4). This, coupled with the visitation schedule constraints, is enough to guarantee a unique ergodic distribution matching p. The positivity is used to guarantee p-irreducibility.

For the asymptotic results here, we require the same, i.e., that p have an everywhere positive density. However, positivity alone is not quite enough[3]. Here, a slightly stronger form of positivity of p is required.

Definition 2 *Denote the probability space as the cross product space $(\Omega, \mathcal{F})$. Suppose that for some ordering of the variables, there is a rectangular set $A = A_1 \times .. \times A_n \in \mathcal{F}$ with positive Lebesgue measure and there exists an $\epsilon > 0$ such that*

$$p(x_i | x_1..x_{i-1}x_{i+1}..x_n) \geq \epsilon \tag{5.2}$$

for all values $x_{j<i} \in A_{j<i}$ and $x_{j>i} \in \Omega_{j>i}$. When this holds, we say that A is a rock *for p, and if p is also an everywhere positive density, we say p is* rock-positive.

For intuition, one can think of A being a "rock-solid" positive set. If the density p is everywhere positive, then so is the conditional density, but this alone does not guarantee that the conditional density could not be made arbitrarily close to zero for some conditioning set. The existence of a "rock" (the set A) immediately reachable from all states with some nonnegligible probability, provides a way to ensure that the algorithm is recurrent.

It is possible for p to have a rock without being everywhere positive. Positivity is required for p-irreducibility, but the results generalize to distributions that are not positive everywhere provided that p-irreducibility can be ensured in some manner[4]. In the case of Gibbs sampling, [York, 1992] makes similar observations.

5.2 Convergence Results

Lemma 3 *Suppose p is rock-positive on n variables with rock A. Consider an iterative dynamic discretization algorithm where one step consists of (1) selecting a variable $\mathbf{x}_i$ to rediscretize at random from a fixed positive distribution over the n variables, (2) picking m values independently for $\mathbf{x}_i$ from a distribution f having the form of (5.1). Then there exists a $\delta > 0$ such that after n steps, all possible discrete configurations are contained in A with probability at least δ.*

Proof: Let $\gamma_i = \epsilon\mu(A_i)$ denote the probability of A_i, where μ is the standard Lebesgue measure over the space of possible discretizations, and let $\gamma = \min_i \gamma_i$. Since A_i has positive Lebesgue measure, $\gamma > 0$. Let $a(i)$ denote the probability that variable $\mathbf{x}_i$ is chosen for rediscretization on any given step. With probability $a(1)a(2)..a(n)$, the variables are chosen in order and all n variables rediscretized during the next n steps. When x_1 is rediscretized, with probability at least γ^m, all m samples drawn at random are all contained within A_1. (Actually, the probability is at least $m!\gamma^m$, since the order in which they are drawn is irrelevant, but by simply dropping the $m!$ we have something even more conservative, and perhaps easier to follow.) This is true for any w in (5.1). If this happens for $\mathbf{x}_i$, then when $\mathbf{x}_2$ is rediscretized, there is also a probability of at least γ^m that all points will be contained within A_2. Therefore, with probability $\delta = a(1)..a(n)\,(\gamma^m)^n$, all these events occur, and the resulting discrete points for all variables (and therefore all configurations) lie within A. This is a lower bound on the probability of all configurations occuring in A, and since $a(i) > 0$ and $\gamma > 0$, $\delta > 0$. □

Lemma 4 *Suppose p is rock-positive on n variables with rock A. Consider an iterative dynamic discretization algorithm where variables are rediscretized one-at-a-time according to a fixed repeating pattern of length $\ell > n$ that includes at least one visit to each variable. When a variable $\mathbf{x}_i$ is rediscretized, m values are*

[3] The degenerate cases for which positivity is not enough can occur only on infinite state spaces. For finite state spaces, the rock-positivity is automatically implied by the positivity of the probability distribution.

[4] p-irreducibility here refers to the p-irreducibility of Gibbs sampling.

chosen independently from a distribution f having the form of (5.1). Then there exists a $\delta > 0$ such that after ℓn steps, all possible discrete configurations are contained in A with probability at least δ.

Proof: Let γ be as in the proof for Lemma 3. Let z_i denote the number of times variable $\mathbf{x}_i$ occurs in the visitation pattern. Consider the event that all values for $\mathbf{x}_1$ are chosen to be within A on every visit to $\mathbf{x}_1$ during ℓn steps of the algorithm. Because A is a rock, this will occur with probability at least $(\gamma_1^m)^{z_1 n}$, since $\mathbf{x}_1$ is visited $z_1 n$ times during this period. If this event occurs, then the event that all values for $\mathbf{x}_2$ are chosen from within A_2 on the second through ℓ^{th} repetition of the pattern occurs with at least probability $(\gamma^m)^{z_2(n-1)}$, since during each of these iterations, all values of $\mathbf{x}_1$ are already guaranteed to be within A_1. Continuing for all variables, we have that all configurations lie within A at the end of ℓn iterations with probability at least

$$\delta = \sum_{i=1}^{n} (\gamma^m)^{z_i(n+1-i)}$$

Note that every term of this sum is positive, so $\delta > 0$. □

From this point on, I will refer to the iterative dynamic discretization algorithm as being either one of the two schemes outlined in Lemmas 3 or 4.

Some notation is called for. In what follows, we will deal extensively with the space of possible discretizations. Each member of this space is a bag $X = X_1 \times ... \times X_n$, where each X_i has m members, $X_i \subset \Omega_i$. In the proofs that follow, I will use s to denote a single discretization (i.e., s is a cross-product set of joint configurations), S to denote a set of discretizations, Θ to denote the set of all possible discretizations, and $\mathcal{B}$ to denote a σ-algebra on Θ. Furthermore, I assume that $\mathcal{B}$ is related to $\mathcal{F}$ (the σ-algebra over the space of possible configurations) in a rather obvious way. Recall that $\mathcal{F} = \sigma(\mathcal{F}_1 \times .. \times \mathcal{F}_n)$. Similarly, $\mathcal{B} = \sigma(\mathcal{B}_1 \times .. \times \mathcal{B}_n)$, where $\mathcal{B}_i \subset (\mathcal{F}_i)^m$, denotes (unordered) bags of size with elements from the space of the σ-algebra $\mathcal{F}_i$. In other words, $\mathcal{B}_i$ consists exactly of all elements of $(\mathcal{F}_i)^m$ except that the members of a set in $(\mathcal{F})^m$ are ordered, so that two sets in $(\mathcal{F}_i)^m$ that differ only on the ordering of their members are equated to the same set in $\mathcal{B}_i$.

An iterative dynamic discretization algorithm defines a Markov chain on the space of possible discretizations, $(\Theta, \mathcal{B})$. The next discretization is chosen stochastically based only on the current discretization, hence it is order-1 Markov. I use the symbol Φ to denote this Markov chain. In what follows, I utilize the theory of general space Markov chains extensively. The best reference on this topic is [Meyn and Tweedie, 1993]. For the version described in Lemma 3, Φ is a time-homogeneous Markov chain, meaning that the transition kernels are the same at every step; however, the version described in Lemma 3 is nonhomogeneous since the transition kernel is determined by what step of the pattern the algorithm is on. However, the *ℓ-skeleton* of Φ, i.e., the Markov chain obtained by looking only at every ℓ^{th} step, is time-homogeneous. Denote the ℓ-skeleton of Φ by Φ^ℓ, and for notational continuity, simply take $\ell = 1$ for the random visitation variant.

Denote the transition kernel of Φ by $\tau(s, S)$, where $s \in \Theta$ and $S \in \mathcal{B}$, and the transition kernel for Φ^ℓ by $\tau^\ell(s, S)$. Notice that in terms of this notation, Lemmas 3 and 4 simply state that there exists a nonnull set A and a $\delta > 0$ such that $\tau(s, A) \geq \delta$ from all $s \in \Theta$. The next proposition shows that these are in fact transition kernels in the standard sense.

Proposition 3

1. *For each $S \in \mathcal{B}$, $\tau(\cdot, S)$ is a nonnegative measurable function on Θ.*
2. *For each $s \in \Theta$, $\tau(s, \cdot)$ is a probability measure on $\mathcal{B}$.*

The same holds for τ^ℓ.

Proof: For any $s \in \Theta$ and $S \in \mathcal{B}$, $\tau(s, S)$ is simply the probability of choosing a bag of m points in S according to $p(\cdot|x)$ for some $x \in s$ (since f is a mixture with components $p(\cdot|x)$). This is clearly nonnegative and measurable for any x and since $p(\cdot|x)$ is measurable for any x, so is $\tau(s, \cdot)$. Finally, in general, when τ is a transition kernel, so is τ^ℓ. □

Let μ be a nontrivial measure on $\mathcal{B}$. A set $S \in \mathcal{B}$ is called a *μ-small set* when for some $n > 0$ $\tau^n(s, B) \geq \mu(B)$ for all $s \in S$ and $B \in \mathcal{B}$. S is called a *small set* when S is μ-small for some nontrivial measure μ [Meyn and Tweedie, 1993, Page 106]. Small sets (and a generalization termed *petite sets* in [Meyn and Tweedie, 1993], which are equivalent for our purposes) are utilized heavily in the theory of Markov chains on general state spaces. They create certain pseudo-atomic properties that are of great power on generalized state spaces. To gain an intuition about what a small set is, consider first the case where S contains a single element (if such a set is in $\mathcal{B}$). Obviously in this case $S = \{s\}$ would be a small set since the transition kernel leaving s would be a nontrivial measure by Proposition 3. However, if you take an arbitrary set S, it could be the case that for any target $B \in \mathcal{B}$, there is a source $s \in S$ that cannot reach B in n steps. Then $\mu(B)$ would have to everywhere zero. In this situation, S would not be small.

Lemma 5 *Suppose p is a rock-positive probability with rock A, and Φ is the Markov chain from the iterative dynamic discretization algorithm using p. Then A is a small set.*

Proof: Consider the nonnegative measure ν on $\mathcal{B}$ defined by setting $\nu(\bar{A}) = 0$, where $\bar{A}$ is the complement of the rock A, and then setting $\nu(S) = \int_S \epsilon^m d\mu = \epsilon^m \mu(S)$ when $S \subset A$ with μ being the standard Lebesgue measure over $\mathcal{B}$. Suppose that $s \in A$ and a new state will be chosen by iterative dynamic discretization. A variable $\mathbf{x}_i$ is chosen for rediscretization, and because $x_{j \neq i}$ is currently within the subrock $A_{j \neq i}$, the probability density from which new points are drawn is everywhere greater than ϵ within A_i. Therefore, the probability of m independently drawn points landing within a set $S \subset A$ is at least $\nu(S)$. Since this is true for any starting point $s \in A$, A is a ν-small set. □

Let $L(s, S)$ denote the probability that a discretization in S is reached in a finite number of steps from s. For a measure μ on $\mathcal{B}$, a Markov chain is said to be *μ-irreducible* when $L(s, S) > 0$ for any s whenever $\mu(S) > 0$ [Meyn and Tweedie, 1993, Page 87]. q

Lemma 6 *For any measure ϕ on $\mathcal{B}$, Φ and Φ^ℓ are ϕ-irreducible.*

Proof: From the definition of ϕ-irreducibility, it is obvious that whenever ϕ_1 is absolutely continuous with respect to ϕ_2 and Φ is ϕ_1-irreducible, then Φ is ϕ_2-irreducible. Since any measure in general is a restriction of some everywhere positive measure, it is only necessary to assume that ϕ is an everywhere positive measure on $\mathcal{B}$. Since all everywhere positive measures are mutually absolutely continuous, they are all equivalent for irreducibility.

Suppose Φ is currently at s. Consider a sequence of steps of the algorithm that visits (i.e., rediscretizes) every variable at least once. Some such sequence will be of finite length with probability 1 in the version where the variable to rediscretize is chosen at random, and some such sequence will be of length ℓ in the pattern-based version. Note that the end of such a sequence from Φ is also the end of such a sequence for Φ^ℓ, so the argument that follows holds for Φ^ℓ as well. Because p is everywhere positive by assumption, the conditional density $p(x_i | x_{j \neq i})$ is also everywhere positive. If we consider the last time $\mathbf{x}_i$ is rediscretized in the sequence, there is a positive probability of $x_i \in S_i$ for any $S_i \in \mathcal{B}_i$, since all m points are chosen independently from the everywhere positive conditional density. So at the end of the sequence, the resulting discretization has a positive probability of being in S for any $S \in \mathcal{B}$. Therefore, Φ is ϕ-irreducible. □

Some Markov chains are *periodic*, meaning that they are guaranteed to not be in certain states at certain evenly spaced time points. Periodicity requires some degree of determinism in the transition kernel. The exact definition of periodicity on an uncountable state space is needlessly complex for the present discussion, but can be found in [Meyn and Tweedie, 1993, Pages 116-8]. A chain that is not periodic is called *aperiodic*. When there exists a ν-small set, A, with $\nu(A) > 0$, then the Markov chain is said to be *strongly aperiodic* [Meyn and Tweedie, 1993, Page 118]. As expected, strong aperiodicity implies aperiodicity.

Proposition 4 *Φ and Φ^ℓ are strongly aperiodic.*

Proof: In the proof of Lemma 5, a measure ν is defined that assigns $\nu(A) = \epsilon^m \mu(A) > 0$ to the rock A, which by that lemma is a ν-small set. Thus, Φ is strongly aperiodic, and the same holds for Φ^ℓ. □

For the following lemma, recall from above that $L(s, A)$ is defined to be the probability that a discretization in A is reached in a finite number of steps from s. Following the notation in [Meyn and Tweedie, 1993], I will also write $E_s[\tau_A]$ to denote the expected number of steps under Φ to reach A from s, and $\sup_{s \in A} E_s[\tau_A]$ to denote the smallest upper bound on the expected time to return to A from A.

Lemma 7 *Let A be a rock.*

1. *For all $s \in \Theta$, $L(s, A) = 1$*
2. $\sup_{s \in A} E_s[\tau_A] < \infty$

Proof: From Lemmas 3 and 4, there exists a $\delta > 0$ such that from any $s \in \Theta$, A is reached within k steps for some finite $k > 0$ (k depends on which algorithm variation is used). Consider a different Markov chain, Φ', where on any step the chain transitions to A with probability δ. Since Φ' is everywhere less likely (or equally likely) to go A, the probability of getting to A in a finite time by Φ', $L'(s, A)$, cannot be greater than $L(s, A)$. A similar relation holds holds for the expected return time, $\sup_{s \in A} E'_s[\tau_A]$, except that we must remember that one step of Φ' is less efficient at reaching A than k steps of Φ. Therefore, $k \sup_{s \in A} E'[\tau_A] \geq \sup_{s \in A} E[\tau_A]$.

The probability of getting to A for the first time in exactly $i + 1$ steps under Φ' is the probability of missing A for the first i steps, and then transitioning to A on the $(i+1)^{th}$ step. Summing this over all finite length sequences, we obtain for any s

$$L'(s, A) = \delta \sum_{i=0}^{\infty} (1 - \delta)^i = \delta \frac{1}{\delta} = 1$$

Therefore, $1 \geq L(s, A) \geq L'(s, a) = 1$.

The expected first arrival time from a state s for Φ' is

$$E'_s[\tau_A] = \delta \sum_{i=1}^{\infty} i(1 - \delta)^{i-1}$$

so

$$E_s[\tau_A] \leq k E'_s[\tau_A] = \frac{\delta k}{1 - \delta} \sum_{i=1}^{\infty} i(1 - \delta)^i = \frac{\delta k}{1 - \delta} \frac{1 - \delta}{\delta^2} = \frac{k}{\delta}$$

for any $s \in \Theta$, and therefore for any $s \in A$. Thus, $\sup_{s \in A} E_s[\tau_A] \leq k/\delta < \infty$. □

The notion of irreducibility is a fairly weak version of stability, since it says only that all S are reachable from all s in a finite number of steps with nonzero probability. A stronger form of stability is called *recurrence*, which basically says that all S can be reached from any s in a finite number of steps with probability 1. While irreducibility says that the chain *can* visit all parts of the state space, recurrence says that the chain *will* (a.c.) visit all parts of the state space. On a finite state space, the concepts are equivalent, but on an infinite state space they are quite distinct.

More precisely, a Markov chain is said to be *recurrent* when there is a measure ϕ such that for every starting point $s \in \Theta$, $L(s, S) = 1$ and $E_s[\tau_S] < \infty$ whenever $\phi(S) > 0$.

Lemma 8 *Φ is recurrent*[5].

Proof: Theorem 8.3.6 of [Meyn and Tweedie, 1993] says that if Φ is ϕ-irreducible, A is petite[6], and $L(s, A) = 1$ for all $s \in A$, then Φ is recurrent. All these conditions have been established by the lemmas already given. □

[5] Φ is actually Harris recurrent, which is established during the proof of Theorem 7.

[6] Lemma 5 establishes that a rock A is small, so by [Meyn and Tweedie, 1993, Proposition 5.5.3], A is petite. By [Meyn and Tweedie, 1993, Theorem 5.5.7], petite sets and small sets are equivalent concepts when Φ is irreducible and aperiodic. For this reason, I have not included the definition of *petite sets* here.

A probability distribution π on $(\Theta, \mathcal{B})$ is said to be *stationary* for Φ when

$$\pi(S) = \int \tau(s, S)\pi(ds)$$

In other words, once the chain has reached an occupation distribution π, is stays there.

Theorem 7 *Iterative dynamic discretization asymptotically converges to a unique stationary probability distribution, π, over the space of possible discretizations, in the following sense. Let $P^k(s, S)$ denote the distribution over possible discretizations after k steps of IDD when started from the discretization s.*

- *For the algorithm variant in which the variable to discretize at each step is chosen at random, there is a unique distribution π such that for any $s \in \Theta$*

$$\sup_{S \in \mathcal{B}} |P^k(s, S) - \pi(S)| \longrightarrow 0$$

  as $k \longrightarrow \infty$.

- *For the algorithm variant in which the variable to discretize at each step is chosen according to a repeating ℓ-step pattern, the asymptotic distribution at each step of the pattern is unique, i.e., for any starting point $s \in \Theta$,*

$$\sup_{S \in \mathcal{B}} |P^{\ell k + j}(s, S) - \pi_j(S)| \longrightarrow 0$$

  as $k \longrightarrow \infty$ for $j = 1, .., \ell$.

Proof: First, simply consider convergence, as in the first case, to a single unique distribution, by considering the chain Φ^ℓ (in the second case, this corresponds to the distribution π_ℓ). Once the uniqueness of and convergence to π_ℓ is established, the uniqueness and convergence for the other π_j in the second condition is immediate.

Theorem 10.0.1 of [Meyn and Tweedie, 1993] states when Φ is recurrent, and when there exists a petite set A such that

$$\sup_{s \in A} E_s[\tau_A] < \infty$$

then Φ admits a unique invariant finite normalized measure π on $\mathcal{B}$. Because Φ is ϕ-irreducible by Lemma 6 and aperiodic by Lemma 4, and the rock A is small by Lemma 5, A is petite; recurrence is established by Lemma 8; and the additional requirement is established by Lemma 7. Therefore, the existence and uniqueness of π on Φ^ℓ is established.

The positivity of p ensures that τ^ℓ is absolutely continuous with respect to π. Corollary 1 of [Tierney, 1994] states that π-irreducibility for stationary π with absolute continuity implies *Harris recurrence*. *Harris recurrence* is a still stronger form of recurrence in which every $S \in \mathcal{B}$ is visited infinitely often with probability 1. Tierney's corollary thus establishes that Φ^ℓ is Harris recurrent. One can note that [Tierney, 1994, Theorem 1] establishes the current theorem for convergence in total variation distance. However, most readers will find convergence in maximum absolute deviation, as stated above in the theorem statement, easier to understand. This is established by the Aperiodic Ergodic Theorem, [Meyn and Tweedie, 1993, Theorem 13.0.1], which guarantees convergence in this sense for an aperiodic Harris recurrent chain with a unique invariant measure (π) when π is a probability distribution. □

As a note to help the reader concerning Theorem 7, in the case of a pattern-based algorithm, you can imagine treating an entire run through a pattern as if it were a single (mega-)step. Thus, the discretization would only be examined after each pass. This simplifies the statement of the second case to something like the first case, i.e., that the algorithm approaches a unique distribution π. The theorem describes what happens more generally if the process is examined at *any* step. For the analogous result for Gibbs sampling (Theorem 6 in Chapter 3 on Page 69), the distribution does not depend on what step of the pattern the process is stopped at. It is interesting to note that the same equivalence *is not* guaranteed by iterative dynamic discretization. Because of this, this more elaborate statement of Theorem 7 is informative.

5.3 Characterization of Asymptotic Behavior

Theorem 7 establishes that iterative dynamic discretization converges to some distribution over the space of possible discretizations. It does not, however, indicate what that distribution is, or properties it has, or how it relates to the distribution of interest, p. This characterization of the asymptotic behavior remains an open problem, one that I have not yet been able to solve.

When a model has exactly two variables, it is possible to characterize the asymptotic behavior in a simple and useful way. I introduce the *uniform projection* operation, which produces a distribution over the space of configurations from π (which is a distribution over the space of discretizations). It would be nice if this projected distribution was equal to p; however, this is not the case. I do show that in the two-variable case, the projected distribution agrees with p on marginals. In other words, iterative dynamic discretization can be viewed as visiting a collection of configurations at each step, and all these configurations have an asymptotic distribution $\tilde{p}$ that agrees with p on both marginal distributions. In this way, iterative dynamic discretization can be conveniently viewed as an amplified Gibbs sampling algorithm.

If one is interested in estimating marginal distributions, then sampling from a distribution that agrees with p on marginals is perfectly acceptable — the resulting marginal estimates will be the same.

Does $\tilde{p}$, the projected distribution, agree with p on marginals when there is more than two variables? It is shown (unfortunately) that this is not the case. Thus far, no simple characterization of $\hat{p}$ (or other relationship between π and p) has been discovered.

Finally, one should also recall that the pure Gibbs sampling algorithm results as a special case when $m = 1$. In this case, $\hat{p}$ is obviously equal to p.

Definition 3 *Let π be a distribution on $(\Theta, \mathcal{B})$ — the space of possible discretizations. The* uniform projection *of π onto $(\Omega, \mathcal{F})$ — the space of possible configurations — is the distribution $\hat{p}[\pi]$ on $(\Omega, \mathcal{F})$, where the probability $\hat{p}[\pi](X)$ is the probability that a configuration, x, drawn by the following procedure is in X:*

1. *$s \sim \pi$*

   (i.e., draw $s \in \Theta$ according to π.)

2. *$x \sim \mathcal{U}[s]$*

   (i.e., draw x uniformly from the configurations in s.)

As is customary with probability distribution notations, I will use the same symbol for marginal distributions. In other words, $\hat{p}_s^k(x_i)$ is the marginal of $\hat{p}_s^k(x)$ on $\mathbf{x}_i$.

Definition 4 *Two distributions on $(\Omega, \mathcal{F})$, $\Omega = \Omega_1 \times .. \times \Omega_n$, p and $\hat{p}$, are said to* agree on marginals *when $p(X_i) = \hat{p}(X_i)$ for all $X_i \in \mathcal{F}_i$ and all $i = 1, .., n$.*

Lemma 9 *For any p, there exists a distribution π on $(\Theta, \mathcal{B})$ whose projection agrees with p on marginals.*

Proof: Simply select a discretization by selecting m samples for each x_i according to $p(x_i)$ for all i. The resulting probability over possible discretizations, π, obviously agrees with p on marginals. □

Lemma 10 *Suppose iterative dynamic discretization is run on a model containing exactly two variables. If the projection of π to $(\Omega, \mathcal{F})$ ever agrees with p on marginals, then all projections for subsequent iterations of iterative dynamic discretization will also agree with p on marginals.*

Proof: Let $\mathbf{x}$ and $\mathbf{y}$ denote the variables in the model. Suppose $\tilde{p}$ agrees with p on marginals, and consider the projected discretization for the next discretization. Without loss of generality, suppose $\mathbf{x}$ is being discretized.

An individual point, x, is chosen according to (5.1). This is equivalent to choosing y according to w_y, then choosing x from $p(x|\mathbf{y} = y)$ [Devroye, 1986, Page 66].

During the projection, there are two possibilities. The values of x, y might be chosen so that y is the value used in 5.1 when this particular x set at the time the discretization of $\mathbf{x}$ was chosen. This occurs with (at least) $1/m$ probability, since there are (at most) m possible values for y. If so, $\tilde{p}(x, y) = p(x|y)\tilde{p}(y) = p(x, y)$. If this does not occur, then $\tilde{p}(x, y) = p(y) \int p(x|y')d(\tilde{p}(y') = p(y) \int p(x|y')d(py') = p(y)p(x)$, which again, agrees with p on marginals. □

Proposition 5 *Consider a model with exactly two variables. If π (or π_j) is the unique asymptotic distribution over the space of discretization reached by iterative dynamic discretization on this model (whose existence and uniqueness is guaranteed by Theorem 7), then the projection of π, $\hat{p}[\pi]$ agrees with p on marginals.*

Proof: Suppose iterative dynamic discretization is started using a configuration drawn from a π_0 with a projection that agrees with p on marginals. The existence of such a π_0 is established by Lemma 9. Then by Lemma 10, all subsequent projections of occupancy distributions for Φ will also agree with p on marginals, including the asymptotic distribution. (Note that uniform convergence on marginals follows from uniform convergence of P^k to π.) Since the asymptotic distribution is unique and reached from all initial starting points (as established by Theorem 7), the proof is complete. □

Proposition 6 *In general, for a model with three or more variables, and with $m \geq 2$, the asymptotic distribution π does not agree with p on marginals.*

Proof: I simply give an example where the asymptotic distribution does not agree on marginals. However, because it is extremely difficult to identify the asymptotic distribution explicitly, the proof that the asymptotic distribution for this example cannot agree with p on marginals is by contradiction, showing simply that the two conditions cannot both hold.

Let $\Omega_{\mathbf{X}} = \Omega_{\mathbf{Y}} = \Omega_{\mathbf{Z}} = [-1, 1]$, i.e., there are three variables, $\mathbf{x}, \mathbf{y}, \mathbf{z}$, that take on values between -1 and 1. Let the joint density be given by

$$p(x, y, z) = \begin{cases} 1/2 & x, y, z \geq 0 \\ 1/14 & \text{otherwise} \end{cases}$$

Suppose $m = 2$ and a uniform weighting scheme is used, and the variable to rediscretize is chosen randomly (the fully homogeneous case). Note that the distribution is totally symmetric among all three variables, and the same will hold for the asymptotic distribution. Also,

$$p(x|y, z) = \begin{cases} 7/8 & x, y, z \geq 0 \\ 1/8 & x < 0; y, z \geq 0 \\ 1/2 & \text{otherwise} \end{cases}$$

With uniform weighting, each point x in the rediscretization of $\mathbf{x}$ is picked by selecting $y, z \sim \hat{p}(y, z)$ and then picking a values $x \sim p(x|y, z)$. Here, $m = 2$ points are picked.

Assume that $\hat{p}$ agrees with p on marginals. Since $\hat{p}$ is a stationary distribution, $\hat{p}(x \geq 0) = p(x \geq 0)$ after a rediscretization, and

$$\hat{p}(x \geq 0) = \frac{7}{8}\hat{p}(y \geq 0, z \geq 0) + \frac{1}{2}(1 - \hat{p}(y \geq 0)) = p(x \geq 0) = \frac{7}{8}p(y \geq 0, z \geq 0) + \frac{1}{2}(1 - p(y \geq 0))$$

so $\hat{p}(y \geq 0, z \geq 0) = p(y \geq 0, z \geq 0)$. If this last equality did not hold, then $\hat{p}$ could not possibly be the asymptotic distribution. Since the model is totally symmetric for all three variables, $\hat{p}$ is the asymptotic distribution only if $\hat{p}(x \geq 0, y \geq 0) = p(x \geq 0, y \geq 0)$.

Consider $\hat{p}(x \geq 0, y \geq 0)$ just after x has been rediscretized. Suppose a configuration is drawn uniformly from the current discretization, and $x, y \geq 0$. This value for x had been chosen and placed in the discretization by sampling from $p(x|y', z')$ for some y', z'. It is possible that this $y' \geq 0$, in which case, the sampling distribution matches $p(x|y)$, but it is also possible that $x \geq 0$ was drawn even though $y' < 0$, and in this case, the distribution differs. Since there are only two values for y, $\hat{p}(y \geq 0) = p(y \geq 0) = 5/7$, this latter case occurs with probability $\frac{1}{2}\frac{2}{7} = \frac{1}{7}$, so

$$\hat{p}(x, y \geq 0) = \frac{6}{7}p(x, y \geq 0) + \frac{1}{7}p(x \geq 0)p(y \geq 0) = \frac{6}{7}\frac{4}{7} + \frac{1}{7}\frac{5}{7} = \frac{29}{49}$$

Keep in mind that this expression is based on the assumption that $\hat{p}$ agreed with p on marginals prior and after the discretization of x. Since $\hat{p}$ can agree on marginals and be stationary only if $\hat{p}(x, y \geq 0) = p(x, y \geq 0)$, we have a contradiction since $p(x, y \geq 0) = 4/7$ but $\hat{p}(x, y \geq 0) = 29/49$. Therefore, in this example, $\hat{p}$ cannot be both stationary and agree on marginals. □

An interesting area for future research is to attempt to modify the iterative dynamic discretization algorithm slightly in such a way so that the uniform projection of the asymptotic distribution is either equal to p or at least agrees with p on marginals. I believe that such an alteration is possible by appending a Metropolis-Hastings-style acceptance loop ([Metropolis *et al.*, 1953, Hastings, 1970, Smith and Roberts, 1993, Neal, 1993]) around a discretization step. The idea is that the current discretization step *proposes* a new discretization, s. A suitably chosen acceptance probability, $\alpha(s, s')$, is computed, based on the current discretization s and the proposed discretization s'. With probability $\alpha(s, s')$, the proposed discretization becomes the next discretization, otherwise the same discretization is repeated (in which case, there is no need to repeat a propagation). The acceptance probability has the effect of distorting the uniform projection distribution just enough so as to make it equal to p, or at least equal to a distribution that agrees with p on marginals (if both are possible, the latter may be possible with fewer rejections). This is similar to the ideas behind rejection sampling (Section 4.3.3). A direct application of the Metropolis-Hastings algorithm is not quite possible to accomplish this, but some minor variation of it may be.

5.4 Combining Stochastic Simulation & Exact Propagation

The perspective afforded by the analysis in this chapter suggests one reasonable way to view iterative dynamic discretization is as a Markov Chain Monte Carlo (MCMC) algorithm. However, MCMC algorithms typically visit *configurations* according to an ergodic Markov chain, while iterative dynamic discretization visits *discretizations* (i.e., collections of configurations) according to an ergodic Markov chain.

Gibbs sampling has been applied to optimize MAP problems. In these scenarios, a sequence of configurations are drawn from p using MCMC, and the one with the highest probability so far is remembered. A recurrent Markov chain visits all parts of the state space with probability 1, so this technique provides a certain guarantee that the true optimum will be found. Furthermore, since (asymptotically) higher probability configurations are more likely to be visited than lower probability configurations, one heuristically expects to visit good configurations fairly quickly. It is important to note that pure Gibbs sampling is typically not the best choice for MAP optimization problems — the related MCMC technique called *simulated annealing* ([Kirkpatrick *et al.*, 1983]) is usually considered superior for optimization problems, although Gibbs or related MCMC techniques are often applied in optimization problems (e.g., [Bielza *et al.*, 1996]). Simulated annealing uses the basic Gibbs sampling procedure, but during sampling the underlying distribution is slowly changed, starting with a very diffuse distribution with high mobility, and progressing to a distribution where the highest probability configurations are accentuated.

In any event, these suggest an interesting way to view the iterative dynamic discretization. It is a MCMC algorithm, but at each iteration a large number of configurations are considered. Within an iteration, exact propagation is used to find the optimum among this collection of configurations. In this way, iterative dynamic discretization provides a means for combining MCMC methods with exact propagation methods. MCMC methods allow the algorithm to visit the state space thoroughly (recurrently), while exact propagation techniques allow large collections of configurations to be analyzed in one step.

I believe this combination is useful for many types of problems, not just MAP optimization. For example, there may be ways to use this for computing expected values or even marginal estimates. However, so far I have been not been successful at a theoretical analysis that would make this claim credible.

In any event, in order for this combination of MCMC with exact propagation to be successful, it is necessary for the underlying graph structure to be amenable to propagation, at least when individual variables take on only a small number of values. In other words, the triangulated dependency graph must have a small number of variables in each clique. One might at first think that such a dependency graph is amenable to propagation techniques already, so that the application of approximation techniques is uncalled for; however, this is not always the case. What really determines the efficiency of exact propagation techniques is the number of possible joint values the variables within a clique can take on[7]. If one or more individual variables can take on a huge number of possible values, then the number of possible joint values in a clique can also be huge. It is in such a situation that this combination is appropriate. If the variables are continuous, then

[7]This applies, of course, to nonparametric potential representations. A conjugate parametric potential representation's efficiency might likewise be compared to the number of free parameters within a clique.

there are an infinite number of possible values no matter how many variables are within the clique. When the potentials involved are in a nonconjugate form (relative to the graphical structure), then clearly exact propagation is not automatically efficient.

Suppose one has a graph that does not have only a few variables per clique. One possibility is to conglomerate variables into "mega-variables," so that one has only a small number of variables once again. These mega-variables can then be iteratively sampled using iterative dynamic discretization. The conglomeration of variables can be readily based on the triangulation of a graph. Any two variables that appear in exactly the same set of cliques in the triangulated graph are conglomerated. From this, the combination of the methods is actually a very general technique.

The discussion here is intended simply to provide a useful perspective on the iterative dynamic discretization technique. Viewing it as a combination of MCMC and exact propagation appears to be a very legitimate perspective (one of many) and may provide some readers with a more thorough understanding of the algorithm. There are several aspects of this interpretation that I have not explored in this thesis.

5.5 Summary

This chapter has utilized mathematical tools to explore various aspects of the asymptotic behavior of a class of iterative dynamic discretization algorithms. In terms of Figure 4.1 on Page 78, the analysis assumes $f(x_i)$ is estimated in Step 6 by the Markov blanket posterior's method, but with any of the possible weighting schemes. For Step 7, it is assumed that a discretization is chosen from $f(x_i)$ using random sampling. The main result is that under mild positivity assumptions about the distribution of interest, the technique converges ergodically to a unique distribution over the space of possible discrete models, such that the limiting distribution obtained does not depend on the initial discretization. The recurrence of the technique ensures that a solution arbitrarily close to the global optimum MAP configuration will always be found if the algorithm is run sufficiently long.

Although the limiting behavior of iterative dynamic discretization does not depend on the initial discretization, a full and useful characterization of this behavior, and how it relates to the distribution of interest, remains an open question. A characterization along these lines would be required for using iterative dynamic discretization to compute marginal posterior inferences, and is therefore of great interest for future work.

It was only near the final stages of this research that I realized the close connection that iterative model construction techniques have (or can be designed to have) to Markov chain Monte Carlo techniques. This is a very fruitful connection to remember because it opens up formal means for understanding or characterizing the asymptotic characteristics of the algorithm's behavior. Keeping this in mind when inventing an iterative model construction algorithm can also help to ensure that the process is recurrent, i.e., that it does not permanently rule out some possible solution. These consideration also suggest that adding randomness to the model construction step can be of great benefit in an iterative framework, especially when it comes to asymptotic behavior.

Chapter 6

Empirical Evaluation

6.1 Scope and Rationale of Evaluation

At the current time, several aspects of the algorithm's behavior can only be examined empirically. While it might be nice to perform an exhaustive empirical exploration of the performance of all variants of iterative dynamic discretization over a wide spectrum of segmentation problems and problem instances, this space is much too broad to cover in anything close to an exhaustive manner. Perhaps even more importantly, it is not entirely clear that anything useful would be learned from such an exercise. Therefore, it is important to focus the empirical evaluation carefully to address specific questions.

To this end, there are two issues that I wish to address with the empirical study in this section:

1. Can iterative dynamic discretization substantially reduce the number of iterations required for convergence compared to Gibbs sampling?

2. How important are each of the algorithm's variations to overall performance?

In a sense, the first of these is an empirical existence proof. The case is made with a single example where iterative dynamic discretization significantly outperforms over Gibbs sampling. Finding such an example was not difficult.

A number of elaborations on the first question are possible as well. How often does iterative dynamic discretization significantly reduce the number of iterations required compared to Gibbs sampling? By how many iterations? Also, is iterative dynamic discretization more efficient than Gibbs sampling in terms of C.P.U. cycles? This latter question is relevant since iterative dynamic discretization requires more computation per iteration than Gibbs sampling. Unfortunately, it does not seem possible to give meaningful (empirical) answers to any of these questions. All of these questions are highly sensitive to the precise time-series models and data used. Efficiency further depends on the domain/model-specific property of how efficiently the basic sampling operation (picking x_i from $p(\mathbf{x}_i|\mathbf{x}_{j\neq i})$) can be performed, and on the efficiency of the implementation. The examples show that the number of iterations can be reduced very substantially, but conclusions beyond this should be reached with great caution.

The study of variations to the algorithm is too vast to take on in any comprehensive fashion, so some care must be taken. It is certainly not possible to compare every combination of variations in a formal empirical study, since the number of combinations considered would be quite substantial. There are also distinct limits to the number of different domain models (i.e., different HSSMMs) and number of distinct problem instances per model that can be reported. I have organized this part of the evaluation based on personal experience I have had experimenting with implementations of the algorithm and its graphical interface. Some of these experiences are described in Section 6.11, but a number of heuristic aspects of that experience are much more difficult to elucidate. It is possible to quickly experiment with several different models and several different (artificially generated) time-series using the graphical interface, and to quickly obtain an intuitive feeling for what works and what does not for certain models. Based on this experience, I have organized the empirical exploration in an incremental and systematic fashion, adding one feature at a time to the algorithm and exploring the impact of this over repeated runs. Thus, after reaching the final (superior) version, there is at

least some indication of the contribution that each little variation has on the end result. In order to make this incremental progression maximally meaningful, I perform the progression using a single problem instance. Thus, the performance of each variant can directly be compared to the performance of other variations. From my experiences, the example I have picked seems to be reasonably representative of other models and data that I have played with[1]. I believe that this format communicates more to the reader than a study across multiple different models and problem instances would. Although it is somewhat informative to see that one variation outperforms another, it is far more informative to understand the problem characteristics that allow one variation to outperform the other. Much of the deeper understanding is better obtained by looking at the specific phenomena and artifacts discussed in Section 6.11.

The model used for these runs is given in Appendix A.

6.2 Procedure and Evaluation

The time-series data used for these experiments was generated randomly according to the same model used to segment it. Therefore, the model is known to be a "correct" description[2] of the process that generated the data. Furthermore, because the data is synthetically generated, a *ground truth parse* (the times and states picked during the simulation) is available for comparison. Figure 6.1 shows the time-series data used for these experiments along with the ground truth parse.

Prior to each run, the program was supplied with the exact value for t_{20} in the ground truth parse. In other words, the time of the 20^{th} transition was given. In a real application, the value of t_{20} would never be available. However, for empirical evaluation supplying this value has a number of benefits. First of all, it ensures that relative evaluations are always comparing apples with apples. All parses compared always span the same time interval. Thus, differences in evaluations arising only as a result of a different amounts of data being covered cannot show up. Another advantage of supplying t_{20} is that the style of inference better matches that found in most applications of Bayesian networks, that is, where evidence typically occurs in the leaf nodes. t_{20} is essentially our leaf node in this application.

For all variations that follow, the propagation graph is initially grown to length 20, so that after growing stage completes, the propagation graph contains the variables $t_0, s_0, .., t_{20}, s_{20}$. During the growing stage, there is no rediscretization.

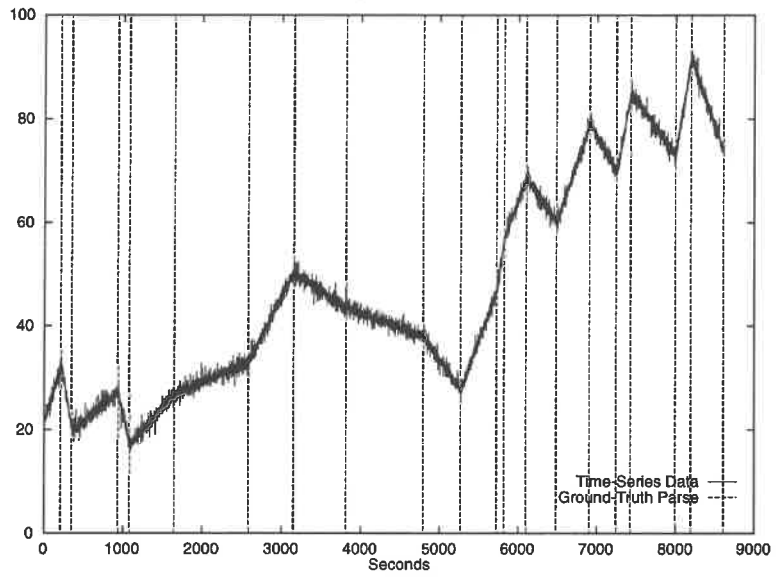
For all methods that rediscretize one variable per iteration, t_{19} is rediscretized first, followed by t_{18}, and so on down to t_1, and then the order repeated[3]. Some of the methods below rediscretize all variables in one iteration, and for them, the order has no effect.

At each iteration, the optimal configuration given the current discretization is extracted (using the propagation algorithm, Section 3.1.1, Figure 3.9 on Page 56). For (focussed) Gibbs sampling, this requires only an optimization over the state variables, since only one value exists for each transition time variable. For iterative discretization methods, the optimization is also over the m possible transition times identified by the discretization. This configuration, λ, is then evaluated by using Equation (2.3) on Page 35, but with $\alpha = 1$. Estimating the normalizing factor $\alpha = P(X)$ is infeasible. This λ is compared against the

[1] One exception to this has been that I have encountered many instances where the problems are considerably easier to solve than the one given here. A good example is where the transitions are deterministic or nearly so, and data has little noise. Some real Shuttle data fell in this category, and most of the iterative dynamic discretization methods performed very well to where it was very difficult to differentiate any significant performance differences between them. The problem instance used here was chosen to contain enough ambiguity so as to separate out the good variations from the bad.

[2] However, the HSSMM is only a partial specification of the total process since it is somewhat of a qualitative model. The simulator must make additional assumptions that the segmentation algorithm does not use. For example, shape recognizers look only at how linear a segment of data appears, while the simulator, even when creating a linear shaped signal, must pick absolute values for the end points of the segment. But, the simulated data is consistent here with all aspects of the more qualitative model that is used for parsing.

[3] The forward natural order and backward natural order generally have equal asymptotic convergence rate. This equivalence in rates is proven by [Roberts and Sahu, 1996] for a variety of models in which exact convergence rates be analytically derived.



Ground Truth Parse

i	s_i	t_i
0	*longRise*	0
1	*longFall*	217
2	*longRise*	358
3	*shortFall*	940
4	*longRise*	1086
5	*longRise*	1643
6	*longRise*	2584
7	*longFall*	3149
8	*longFall*	3819
9	*shortFall*	4795
10	*longRise*	5269
11	*longRise*	5725
12	*shortRise*	5822
13	*longFall*	6103
14	*shortRise*	6487
15	*longFall*	6912
16	*shortRise*	7248
17	*longFall*	7431
18	*longRise*	7996
19	*longFall*	8189
20	—	8610

Figure 6.1: Time-series data used for experiments and the ground truth parse (used to generate the data). In the table, s_i exists from t_i to t_{i+1}.

ground-truth parse, λ^*, yielding the following evaluation metric (known as a *Bayes factor*):

$$Eval(\lambda) = \ln \frac{P(\lambda|X)}{P(\lambda^*|X)} \tag{6.1}$$

This quantity does not require an evaluation of the normalization constant α since it appears in both $P(\lambda|X)$ and $P(\lambda^*|X)$ and thus it cancels out. A larger value for $Eval(\lambda)$ indicates a better segmentation, any segmentation with a negative evaluation is inferior to the ground truth parse, while any segmentation with a positive evaluation is superior to the ground truth parse.

Intuitively one might think the ground truth segmentation is the optimal parse, but this is not the case. The optimal segmentation, as found by the iterative dynamic discretization algorithm, appears in Figure 6.17 on Page 122 (which is where the variant of the algorithm that found that segmentation is discussed). Rare events occasionally occur in the ground truth parse, so that some other interpretation is actually superior. For example, $s_1 = longFall$ in the ground truth parse (Figure 6.1), so that the actual duration between t_1 and t_2 is chosen randomly during the simulation from $c_{longFall} = GammaDist[\alpha = 3, \beta = 200]$. Although this distribution has a mean of 600 seconds, there is also about 1 chance in 11 that the duration chosen is less than 210 seconds (the mean for $c_{shortFall}$). In fact, $t_2 - t_1 = 141$ seconds in the ground truth parse. Even though the data was generated between those times using the *longFall* state, the resulting data appears much more consistent with the *shortFall* state. Thus, the parse identical to the ground truth parse in Figure 6.1 except with $s_1 = shortFall$ is a better interpretation of the data than the ground truth parse. Also, the model specifies that a rising transition is followed by a falling transition with probability 0.8; however, due to the randomness, three rising transitions occasionally occur in succession. There is a high expectation in such a case that the middle segment will be falling, and when this is weighed against the labeling of what appears to be a rising signal as a falling signal, it sometimes is better to label it as rising. This happens at between t_5–t_6. The smaller the slope of the signal, or the fewer data points in that segment, the more likely it is for an expectation to dominate the apparent shape in the data. Also, depending on the relative strength of various expectations and the amount of noise in the data, it can be better to segment differently to fit certain noisy regions. The impact of expectations versus data is considered in Section 7.1. The same sort of phenomena occurs at several other transitions, making the optimal parse[4] for this data approximately 2e6 times better (according to Equation (2.3)) than λ, which the above metric assigns a score of about 14.7.

6.3 Gibbs Sampling

As an initial baseline, Gibbs sampling was run on the submodel $p(t_1, t_2, ..., t_{19})$. The value for t_0 and t_{20} were given. Since the state variables, $s_0, .., s_{20}$, each take on only four possible values, it was not necessary to sample these, so focussed Gibbs sampling (Section 3.4.1) could be employed. Since only the time variables are discretized by iterative dynamic discretization, this would seem to be the best basis for comparison with iterative dynamic discretization. Furthermore, in terms of iterations required[5], focussed Gibbs is more challenging to beat than pure Gibbs, so it provides a more substantial baseline.

Each iteration of Gibbs sampling results in a new single configuration. Figure 6.2 plots this evaluation as a function of iterations (for a single run), showing Gibbs sampling jumping around as it visits possible segmentations. Recall that asymptotically, Gibbs sampling converges to a distribution that visits possible segmentations according to $P(\lambda|X)$. (The performance of Gibbs sampling at any given iteration for this task can be taken to be the evaluation of the best configuration visited so far.) Figure 6.3 shows the performance over ten runs of Gibbs sampling on the example problem. In these graphs, the x-axis shows total iterations, where one total iteration corresponds to changing all 19 variables $t_{19}..t_1$. So the 1,000 total iterations shown on the graphs correspond to 19,000 steps of Gibbs sampling.

---

[4] Assuming that the best parse found during the course of all the experiments run with this data is very close to being optimal.

[5] In terms of C.P.U. cycles, Gibbs may outperform focussed Gibbs, since focussed Gibbs does more work per iteration.

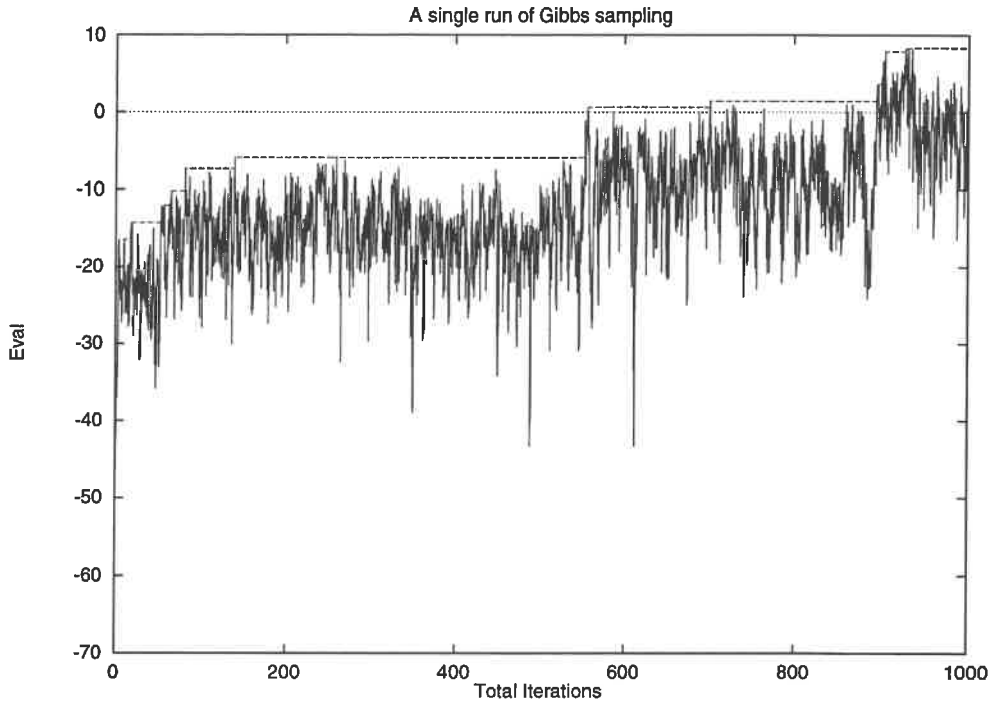


Figure 6.2: One run of Gibbs sampling. Y-axis shows the evaluation of the current discretization at each iteration.

Of the ten runs, the best segmentation found within the 1,000 total iterations (19,000 sampling steps) evaluated to $Eval(\lambda) = 10.12$. None of the 10 runs found a near optimum segmentation within the first 1,000 total iterations, although theory tells us that all runs are guaranteed to eventually get there if run long enough.

Of the ten runs, the final parse for the best and worst runs are shown in Figure 6.4. These are somewhat informative for understanding why the runs did not find the optimum in that amount of time. There are two effects that show up. First, each of the two final segmentations is in a qualitative maximum. In the first case, it has lumped the three consecutive rising segments from 5269 through 6103 into a single transition. In the second case, the three consecutive upward transitions from 1086 through 3149 have been lumped together. In both cases, the data in that region is a pretty good fit to a straight line. Any movement of a single boundary into one of these segments causes some other segment to have a very nonlinear shape. Several transition times have to move simultaneously to get out of this qualitative local maximum. The second phenomena is that many of the individual transition times appear not locally optimized. For example, in the first graph, a distinct transition occurs at 5269, but the parse shows the transition at about 5369. Many of the other transitions are similarly off just slightly. This occurs because at every iteration, the exact value of the transition time is chosen stochastically. For a parse to have all 19 transition times exactly on the best transition time, all random choices would have to land exactly on those points in the same (total) iteration. The ten runs represent 10,000 total iterations, and this combination simply never occurs in that amount of trials. Since the best point for a transition can be kept from one iteration to the next, iterative dynamic discretization provides a way to quickly overcome this drawback (see Sections 4.3.1 and 6.5). This second problem is even a little more subtle than this. Consider the transition that occurs in the data at 5269. Recall that $f_{mbp}(t_i)$ depends on the data between t_{i-1} and t_{i+1}, so one would expect a sharp probability peak in f_{mbp} around 5269, and therefore very high probability of selecting a transition very close to that value. However, the previous transition has also been placed slightly before the previous transition in the data in Figure 6.4, i.e., at 4584. It is actually a better fit to put two straight lines from 4584 to 4369 and from 4369 to 6103 than it is to put two straight lines from 4584 to 4269 and from 4269 to 6103. In reality, f_{mbp} actually has its maximum around 5369 rather than 5269. In other words, locally 5369 is an optimal

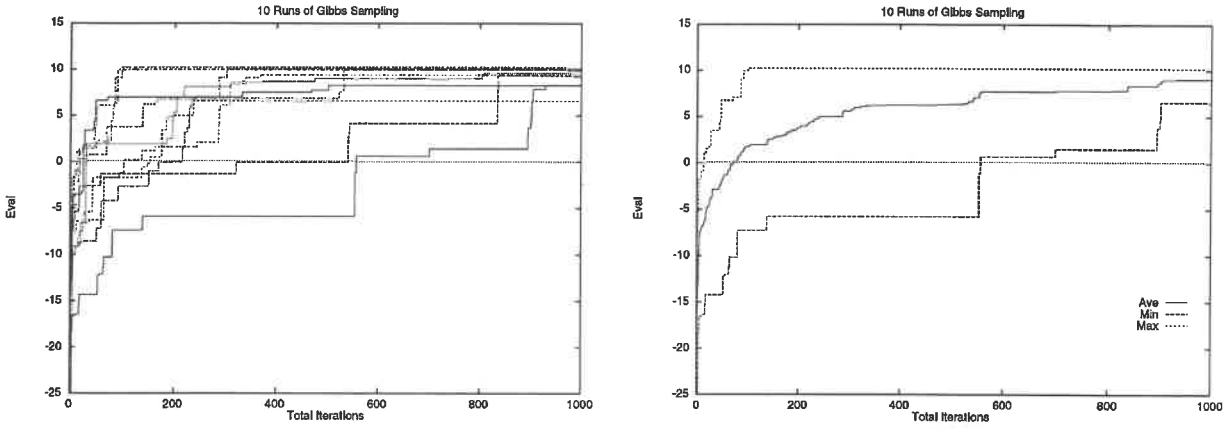


Figure 6.3: Performance of 10 individual runs of Gibbs sampling. The graph on the left shows performance of the individual runs, while the graph on the right shows the average (and minimum and maximum) of these at each iteration.

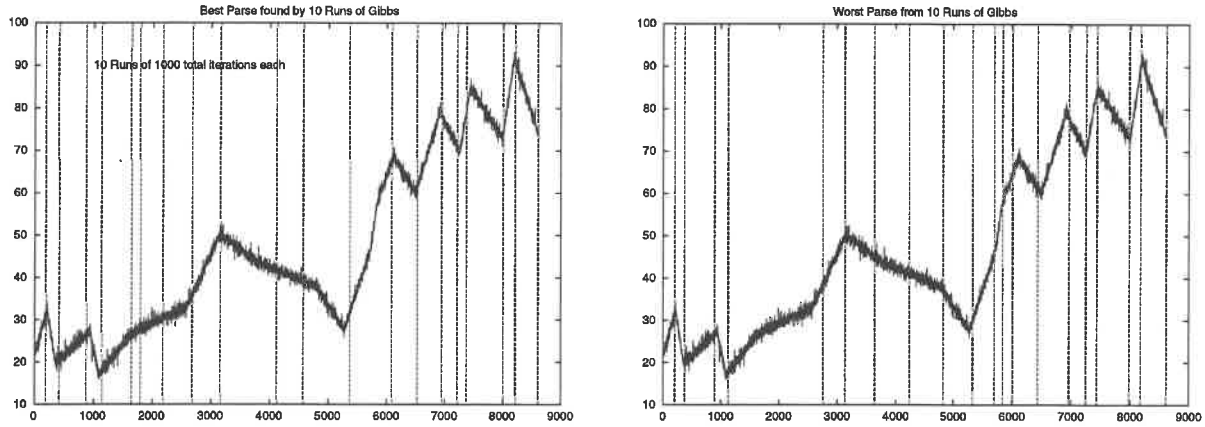


Figure 6.4: Best (left) and worst (right) parses returned by the 10 runs of Gibbs sampling. The parses evaluated to 10.1 and 6.5 respectively.

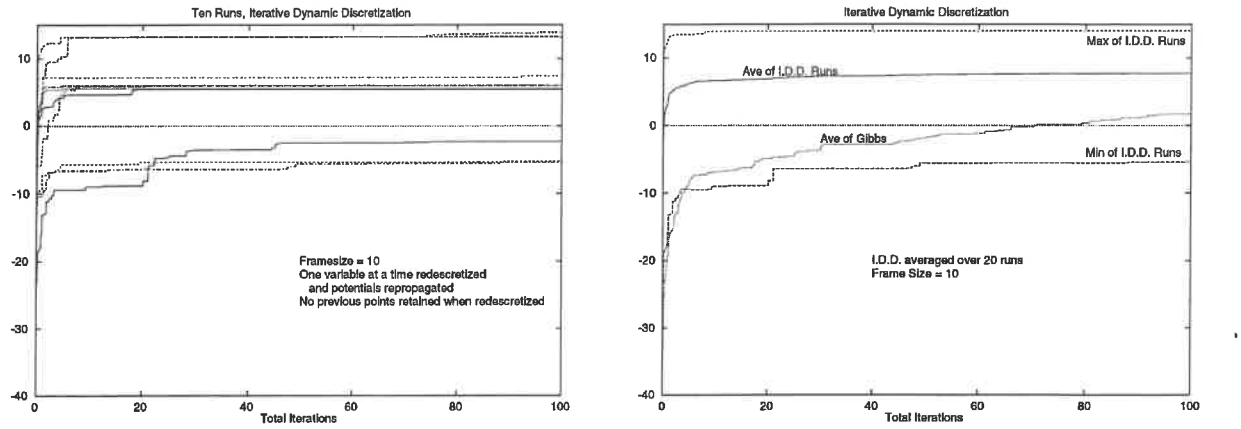


Figure 6.5: Left graph shows ten individuals runs of plain iterative dynamic discretization using $m = 10$ with no bells and whistles. No points from the previous discretization are kept. The right graph shows the averages over these and ten additional runs, along with the minimum and maximum evaluation at each iteration over the 20 runs. Also shown on the right is the average for Gibbs over the first 100 total iterations (from Figure 6.3).

transition, even though it does not land exactly on the obvious transition point.

These experiments demonstrate that Gibbs sampling gets trapped in locally immobile segmentations. These are similar to local optimums, expect that theory tells us that Gibbs sampling will eventually (with probability 1) escape from them, it is just that it can take a long time.

6.4 Pure Discretization

I now consider the first version of iterative dynamic discretization.

In these experiments, ten new discrete points are chosen for each time variable (i.e., $m = 10$) at each iteration. Unlike variations that follow, no points from the previous discretization are kept. The ten points are chosen using f_{mbp} (with the normal sum potential weighting), and like Gibbs sampling, one variable is reframed on each step. This requires a propagation after every step. However, once again I plot total iterations; hence, one total iteration corresponds to 19 steps, with each step consisting of a rediscretization of one variable and a full propagation on the discrete model. This is the natural starting point for comparison.

Because the later versions of iterative dynamic discretization converge very rapidly, I used the convention of running iterative dynamic discretization methods for a maximum of only 100 total iterations[6] (= 1900 reframing steps). Figure 6.5 shows the performance of twenty[7] runs for this pure version of iterative dynamic discretization (i.e., for only 1/10 the amount of time shown in the Gibbs sampling graph).

Although iterative dynamic discretization gets off to a much quicker start than Gibbs, what is extremely surprising is that in terms of the number of iterations, the performance of this variation is comparable to Gibbs sampling (after 100 iterations). Because it does much more work per iteration than Gibbs sampling, it is clearly *not* competitive.

Understanding this lack of improvement is informative. The bottom line is that the biggest determiner of success is mobility — the ability to escape locally immobile regions. Gibbs gets stuck because often pretty good interpretations are very immobile locally — moving from those to a better interpretation requires simultaneous unlikely movement of several transition points. The use of multiple values at each variable by iterative dynamic discretization both increases *and decreases* mobility. The increase in mobility comes from

[6] I did extended a few runs to 1,000 iterations and found no surprises.

[7] An extra ten runs (for a total of 20) to obtain the graph of the average performance because of indications that the random sampling during the first 10 runs was not representative of the average performance. The graph on the left of Figure 6.5 shows only the first ten runs due to the clutter that would result if all twenty were plotted.

using more points, and therefore being able to explore more territory within a single iteration. Exploring more territory results in greater mobility. Also, multiple points consistently allow the system to get off to a better start. The decrease in mobility is less intuitive.

How is it that multiple points can decrease mobility? This is a result of changing one variable at a time, and the inter-variable interactions that emerge as a result. To see this occurs, consider an example.

Suppose the system has found a parse that is locally immobile. In fact, it is so immobile that simultaneous changes to four transition times would be necessary to find a better parse. Just to be concrete, suppose transition times t_1, t_2, t_3, and t_4 must all be simultaneously changed to improve the segmentation.

The number of changes that must occur is only one aspect of mobility (or lack of it). The other is how likely each of these changes is to actually occur during sampling. However, the key here is that these odds are closely interrelated.

The current value of t_2 is very immobile given t_1 and t_3. Thus, the chances of it escaping to a region containing a better parse are fairly low as long as t_1 and t_3 have their current values. However, if Gibbs sampling suddenly, by chance, makes the necessary unlikely change to t_1, the mobility of t_2 may be suddenly increased. Much of its stability was a result of the previous setting for t_1, but now that t_1 has escaped, t_2's mobility is improved. Because of this, in Gibbs sampling, the joint mobility of $t_1, .., t_4$ can be can be considerably less than the combination of their individual mobilities given the current configuration. This phenomena is an indirect benefit arising from a certain lack of memory of Gibbs sampling.

The reduction in mobility for iterative dynamic discretization results because the use of multiple points discourages this beneficial lack of memory phenomena. In this case, t_2 may be very immobile given the current best discrete values for t_1 and t_3. Even though a few wild points may be chosen in the new discretization for t_1, it is quite likely that one or more new locally optimum discrete points will be chosen. The goodness of the few wild points cannot be appreciated until all four changes (t_1 through t_4) occur; however, because of those locally optimum points chosen for t_1, and because these are then weighted favorably, the mobility of t_2 is not improved. Thus, multiple points make it much more difficult for this synergistic (loss of memory) interaction to occur. It is in this way that multiple points can also serve to reduce mobility. This decrease in mobility is apparent from the flatness of the individual curves in Figure 6.5. These indicate a very quick adjustment to locally optimum configurations, with little mobility out of these after they are reached.

It may be this same phenomena that is responsible for the poor performance of (pure) *adaptive direction sampling* observed in [Gilks *et al.*, 1994]. That technique also utilized multiple points (in a rather different fashion) to control sampling.

The two effects seem to be comparable for this version of iterative dynamic discretization, hence the total iterations is comparable to that of Gibbs sampling. However, many of the variations that are possible with iterative dynamic discretization can both reduce this effect, and introduce additional sources of power that are not possible with Gibbs sampling.

6.5 Keeping The Best

The next variation considered is that of retaining the value from the optimum configuration (relative to the previous discretization) in the new discretization. There are three benefits from doing so:

1. The current configuration can only improve with additional iterations.

2. Local optimization is enhanced.

3. Numerical stabilities of certain algorithm variations are eliminated, therefore enabling those variations.

By keeping the best value, the optimum configuration from the previous discretization is still a configuration in the new discretization. Either it remains the optimum, or some other configuration beats it, in which case the optimum configuration improves.

It was previously noted that several of the transition times in Figure 6.4 do not even appear to be locally optimal. By (figuratively speaking) keeping one's thumb on the best value for t_i so far, local optimization is greatly enhanced, since there is no longer a reliance on luck of the draw for individual optimums to be drawn on the same iteration.

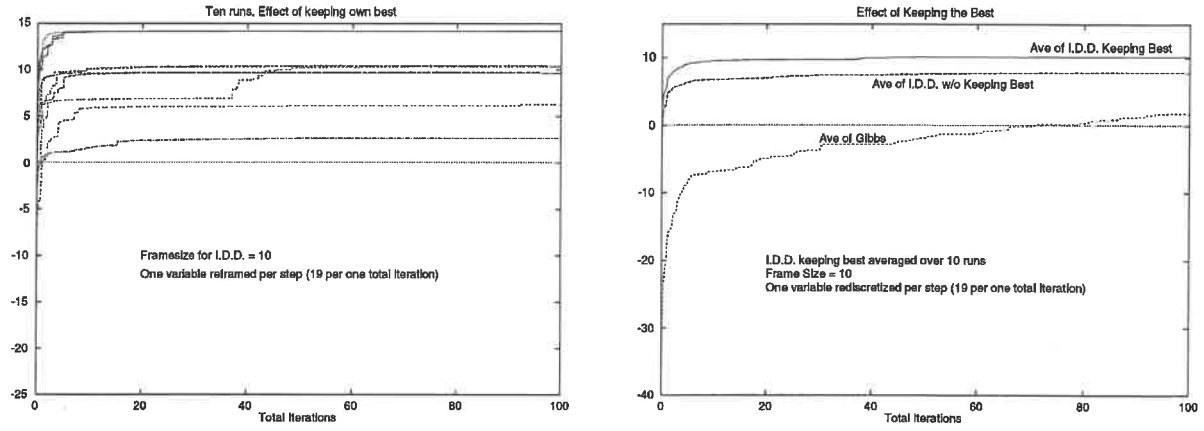


Figure 6.6: On the left is ten runs of iterative dynamic discretization using $m = 10$, rediscretizing one variable per step (all 19 per total iteration with a propagation between each step), and retaining the best value from the previous discretization. The average for these runs is shown on the right along with the previous variation (differing only in that the best value was not retained) and Gibbs.

However, perhaps the most important benefit of keeping the best configuration is that the potential numerical instability that can arise when simultaneously rediscretizing all variables is eliminated. The benefits and effects of simultaneous rediscretization is examined next in Section 6.6.

The performance of iterative dynamic discretization, while keeping the best value for a variable when that variable is rediscretized, is shown in Figure 6.6. Once again, variables are discretized one at a time, with a propagation performed between each rediscretization.

6.6 Simultaneous Rediscretization

It is also possible to rediscretize all variables (or a selected subset of variables) simultaneously. Previously, one variable at a time was rediscretized, and when the next variable was discretized, the sampling distribution was conditioned on the newest discretization. When variables are discretized simultaneously, all are sampled from distributions conditioned on the previous (total) iteration, so that the new values do not interact between variables.

When variables are discretized one-at-a-time, a propagation is necessary every time an individual variable is discretized. An significant advantage of rediscretizing simultaneously is to eliminate propagation steps — only one propagation is needed per total sweep through the variables. This effectively amortizes the cost of performing a propagation over all the variables.

When variables are discretized simultaneously, new values are chosen using less knowledge than with one-at-a-time discretization. For this reason, simultaneous discretization should slightly decrease convergence rates on a per iteration basis. However, since each iteration is more efficient, this tradeoff is often worthwhile.

One must take care to avoid numerical instabilities when rediscretizing all variables simultaneously. Since values are selected randomly according to f, there is normally nothing to guarantee that the resulting discretization contains a valid configuration. In the time-series segmentation task, only segmentations with $t_{i-1} < t_i$ are valid (all other get an evaluation of zero), but without extra precautions, simultaneous selection of new values for t_{i-1} and t_i could result in all new values of t_i occuring before all new values of t_{i-1}. The same cannot happen when variables are discretized one at a time since new values are drawn from f_{mbp}, a distribution conditioned on the neighbors' current values. The easiest (and very effective) way of avoiding this numerical instability is to always retain the value from the best configuration. This guarantees that at least one valid configuration will always be retained. The same trick cannot be utilized with Gibbs sampling, so simultaneous discretization is not variation that can be used with Gibbs sampling.

The impact of simultaneous discretization is seen in Figure 6.7. In this experiment, on each iteration 9 new values were simultaneously chosen for every variable ($t_1..t_{19}$), based on the values from the previous

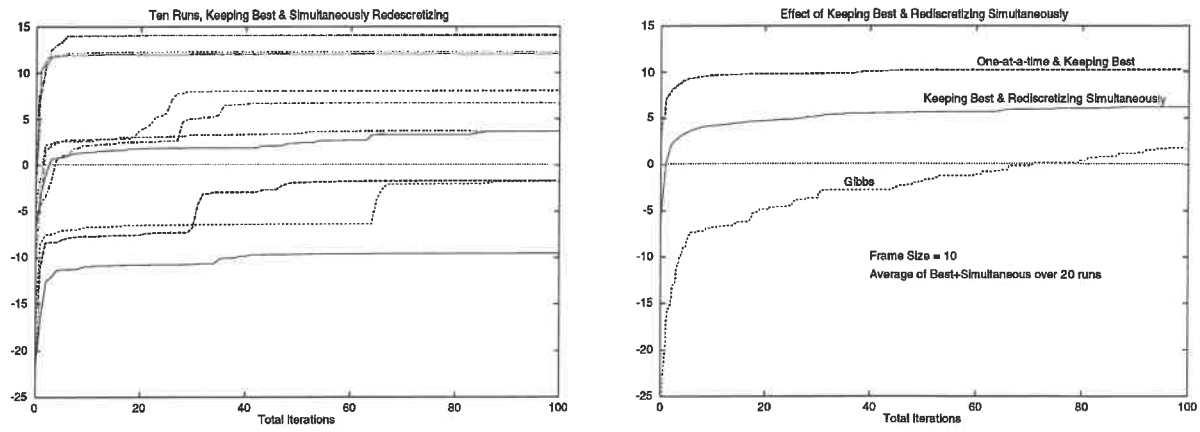


Figure 6.7: Plain iterative dynamic discretization using $m = 10$ and keeping the best configuration from the previous discretization and rediscretizing all time points at the same time.

iteration. Thus, the new values for t_1 had no influence on the new values for t_2, etc. Also, the value from the optimum configuration was retained, so that in total, each variable had 10 possible values. After all variables were discretized, potentials were propagated.

A slight improvement in mobility is apparent in the graph showing the individual runs (on the left of Figure 6.7). In that graph, sudden improvements occur after an apparent plateau has already been reached. This indicates that the system is breaking out of locally stable configurations and jumping to better ones. It appears that this is happening more frequently than was the case with one-at-a-time discretization. The effect here is relatively insignificant to the overall averages, but it is something that can be utilized advantageously with the next variation of the algorithm. Figure 6.7 confirms that there is a slight decrease in convergence rate from one-at-a-time discretization in terms of iterations, but that this difference is insignificant. In the graph, the simultaneous version eventually surpasses the one-at-a-time version. This may be due to the increased mobility, but there are not enough runs to conclude anything beyond the observation that the per-iteration penalty for simultaneous discretization is small. However, an experiment reported at the end of Section 6.7 found a somewhat larger difference between one-at-a-time and simultaneous discretization.

6.7 Keeping Neighbors' best

The above experiments with the first few variants of iterative dynamic discretization highlight the importance of mobility to convergence rates, and help to define the nature of mobility (or the lack of it) in this application. What often happens during the search is that good transition times are found, but they are assigned to the wrong time index. These tend to be create very immobile configurations because a whole series of transitions must slide over at the same time to reach an improved configuration (i.e., introducing or removing one additional transition between two existing transitions). This is closely related to decomposition of the HSSMM, specifically the fact that the t_i variables identify the time of the i^{th} transition. For example, whether t_8 occurs at $t = 3819$ depends critically on whether the obvious transition at $t = 3149$ is t_7, t_6 or otherwise.

Recognizing the nature of this source of immobility, an additional tweak to the discretization process can be introduced to add back (some of) this lost mobility. The idea is to include the best transition times for t_{i+1} and for t_{i-1} as possible transition times for t_i. This is what is meant by "keeping neighbors' best." If this is done for all points, then a channel is established by which all points can easily shift down by one, allowing a single new transition to be inserted or deleted between existing transitions without great bias against such shifts.

For example, if t_i is to be reframed with $m = 10$, we first identify the optimum configuration, then extract out the values of t_{i-1}, t_i, and t_{i+1} from that configuration. These become the first three possible values in the new discretization of t_i. Next, we select 7 more values using random sampling, for a total of

10 possible values.

If several variables shift over using the pathway created by retaining neighbors' best points in the new discretization, this has an effect similar to renumbering transition times. What used to be the fifth transition is now the sixth transition, etc.

An important benefit of iterative dynamic discretization (as compared to Gibbs sampling, for example) is that it is possible to set up such a channel to directly address the domain-specific sources of immobility. This same solution would not be possible with Gibbs sampling. But because we have multiple possible values in a discretization of a variable, there is the flexibility to set some of these aside to address problems like mobility. This seems to be one of the biggest advantages that iterative dynamic discretization has over competing techniques.

Unlike all other variations considered in this thesis, this variant (keeping neighbors' best) is *domain-dependent.* It relies on the fact that neighboring transition times are on the same scale, and a good value for t_i may be a good value for t_{i+1}. Relationships like these generally do not exist between adjacent variables in arbitrary graphical probabilistic networks. It is also a solution that is aimed directly at an artifact of this specific problem formulation. The formulation itself has a sort of deficiency, and this variant is designed specifically to address that deficiency. Because of this, this specific solution is of less general interest as the other methods considered. However, as a case study, it serves to illustrate where important sources of power lie (i.e., methods that carve channels of mobility through the space of configurations) and how these can sometimes be enabled via discretization. From this, there are important general lessons to be learned. Specifically, iterative model construction techniques must pay careful attention to barriers to mobility that the technique or formalism introduce, and it is wise for developers of such techniques to focus on how such barriers can be subverted.

The result of running iterative dynamic discretization while retaining the neighbors' best values is shown in Figure 6.8. Perhaps the most important property made evident by these graphs is the consistent ability to (rapidly) improve from the current state, which is evidenced by the sharp upswing in virtually all the individual runs after they had plateaued at a sub-optimal evaluation. This is a telltale sign that barriers to mobility have been removed.

Instead of simply retaining t_{i+1}'s best value as a possible value for t_i, another option is to retain more than one of a neighbors' best values. This could have added benefit in many situations. For example, the absence of a transition can often cause a single transition variable to do double duty. The single best transition time may be somewhere between two attractive transitions, so that the single transition makes somewhat of a compromise. Imagine, for example, that between the current best values for $t_7 = a$ and $t_9 = b$ lie two relatively obvious transitions — and with only t_8 to cover both, the best position for t_8 is between these two transitions. In this case, the two obvious transitions as well as their midpoint might all show up as favorable transition times, with their midpoint being the most favorable. In this situation, what is needed is a shift in the labeling of time points, so that two transitions occur between a and b. But if t_7 were to shift rightward, we would want it to find the leftmost obvious transition, not the midpoint. In other words, t_7 should borrow something other than its successor's best value.

Recognizing that there are many possible variations on this type of situation, it would be very difficult to invent a universal *ad hoc* rule for choosing which of the neighbors' values to keep. Instead, note that likely transition points will often be rated highly since they often result in good data fits, so the points rated highly by a neighbor have a good chance of being good points for a predecessor (especially after a shift). Thus, the easiest method is to simply keep more than one of the neighbor's best points. This cannot be taken to an extreme, however, since ever value of a neighbor that is retained means that one less point is available for sampling (for a given m). Consider also that if t_{i+1} is indeed doing double duty, setting its best value to somewhere between two good transition times as a compromise, then the two good transitions are likely to have a good evaluation, and it is likely that each is on opposite sides of the best transition. Thus, by retaining the top three of the neighbors' best points, there is a good chance of grabbing the most promising value. This rationale suggests that benefit may be obtained by keeping (at least) three of the neighbors' best. On the other hand, if three values from all neighbors were retained, this could severely cut into the possible values for t_i (in this case, there are two neighboring time values, so only three points would remain for random sampling with $m = 10$). A better compromise is to pick three points in a single direction. This still builds in some of the same flexibility for shifts in the other direction, but to a much lesser extent.

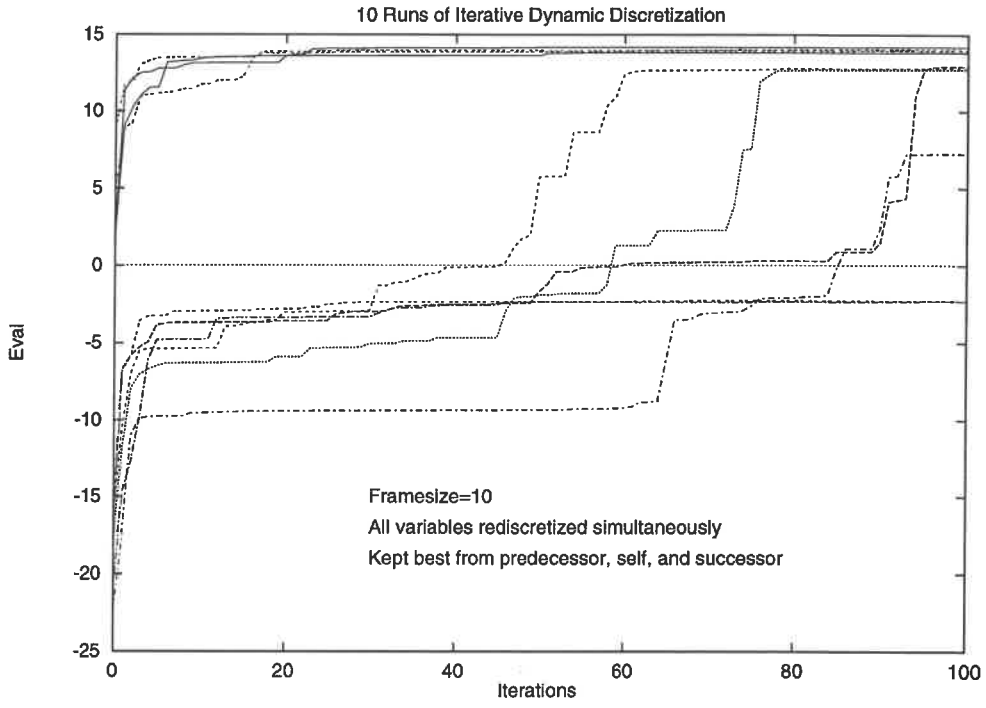


Figure 6.8: Ten individual runs of iterative dynamic discretization using $m = 10$, f_{mbp} weighted using sum potentials (i.e., the regular posteriors), and including the best current value from $\hat{t}_{i-1}$, $\hat{t}_i$ (i.e., own best), and t_{i+1}. (For average, see Figure 6.10.)

Since most biases present in the HSSMM formalism empirically tend to include extra transitions, it is most natural to retain three of the successor's best times and only one of the predecessor's best times, creating the greatest mobility for rightward shifts, and still leaving 5 values available for random sampling.

The result of retaining three of the successor's values is shown in Figure 6.9. The resulting algorithm is highly effective, with 10 out of 10 runs finding a nearly optimum segmentation (with evaluation of approximately 14.2 in all cases) in less than 60 iterations. The effects of mobility are evident in the individual traces, where plateaus are rare and very short lived. The tendency for a few runs to jump directly to the optimum right out of the starting gate is reduced slightly (compared to Figure 6.8) by the fact that the number of values being randomly sampled for each variable is slightly reduced, causing a single discretization to cover a little less local territory.

The experiments in Section 6.6 found that simultaneous discretization cause a slight reduction in performance on a per-iteration basis. But that was done without retaining the neighbors' best. Is it possible that keeping the neighbors' best might somehow interact synergistically with simultaneous rediscretization so that in concert they function better than either does alone? An additional experiment tested this and not only found this to not be the case, but found that the per-iteration penalty from simultaneous rediscretization was somewhat worse than in the previous experiment. The experiment keep the predecessor's, successor's and own (single) best point, with a total frame size of 10, but rediscretized one t_i at a time propagating potentials between each discretization. The result is shown in Figure 6.12. Again, a per-iteration comparison does not fully reflect the extra work required to propagate potentials at every iteration, so there is still some amount of tradeoff involved. Despite this, the performance of the one-at-a-time runs in Figure 6.12 are not quite as good as the runs obtained in Section 6.8 where simultaneous discretization is used yielding greater efficiency.

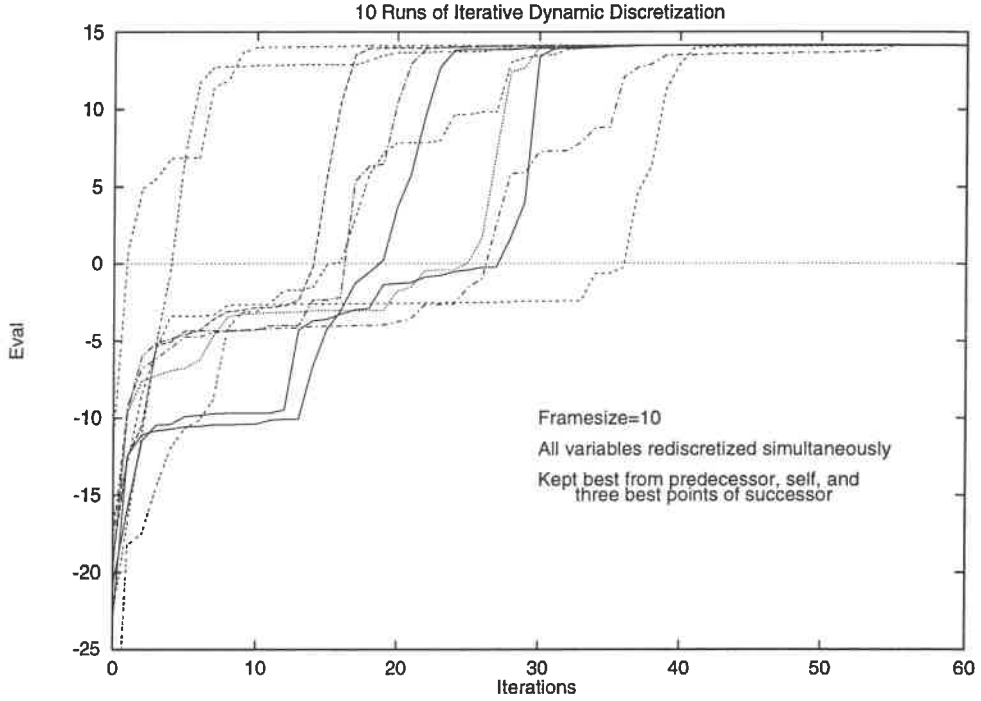


Figure 6.9: Ten individual runs of iterative dynamic discretization using $m = 10$, f_{mbp} weighted using sum potentials (i.e., the regular posteriors), and including the best current value from $\hat{t}_{i-1}$, $\hat{t}_i$ (i.e., own best), and the *three* best points from $\hat{t}_{i+1}$. All 10 runs had found the optimum before the 60^{th} iteration. (Average shown in Figure 6.10.)

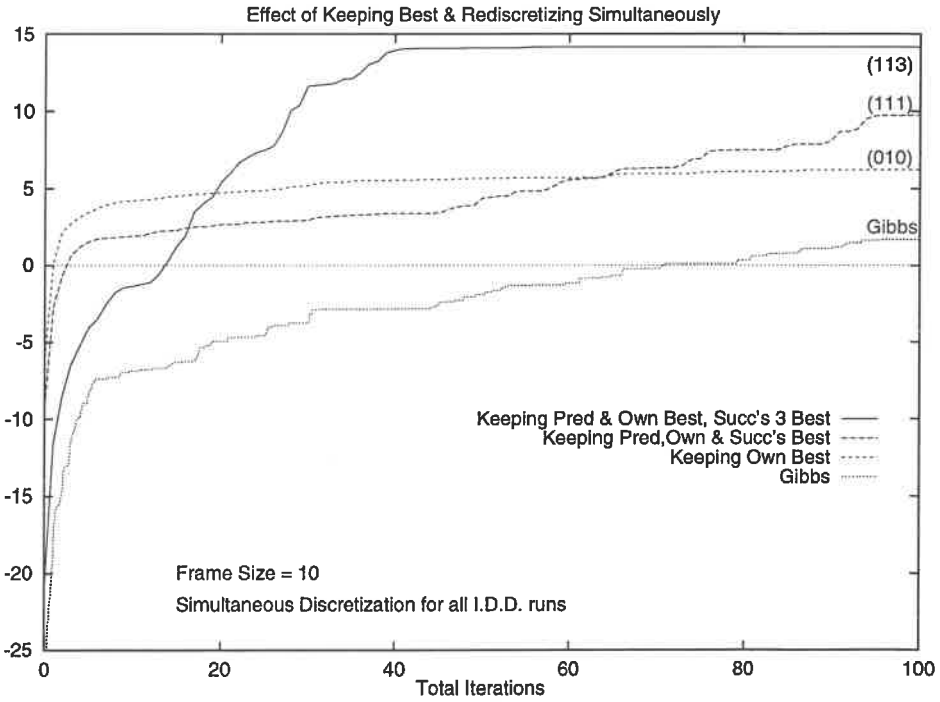
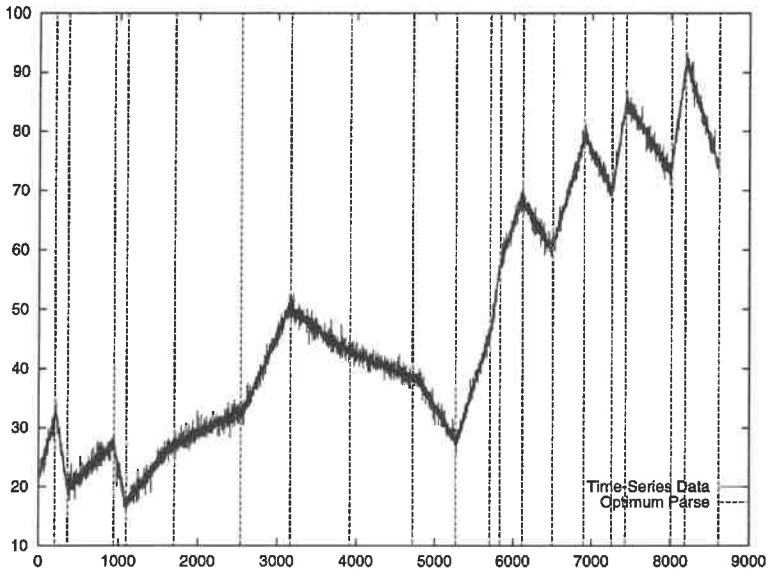


Figure 6.10: Averages for the runs that retain neighbors's best values during discretization.



Best Parse Found

i	s_i	t_i
0	*shortRise*	0
1	*shortFall*	204
2	*longRise*	365
3	*shortFall*	942
4	*longRise*	1098
5	*longFall*	1701
6	*longRise*	2537
7	*longFall*	3166
8	*longFall*	3919
9	*longFall*	4715
10	*longRise*	5263
11	*shortFall*	5704
12	*longRise*	5831
13	*longFall*	6121
14	*longRise*	6503
15	*longFall*	6903
16	*shortRise*	7252
17	*longFall*	7425
18	*shortRise*	8008
19	*longFall*	8181
20	(*shortRise*)	8610

Figure 6.11: This is the best segmentation found during the experiments in Section 6.7. It has an evaluation of 14.2.

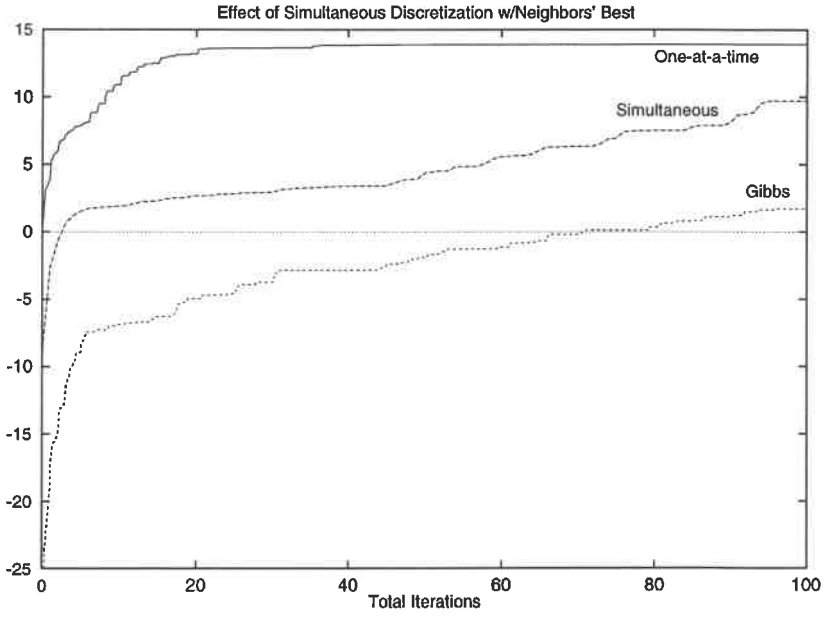
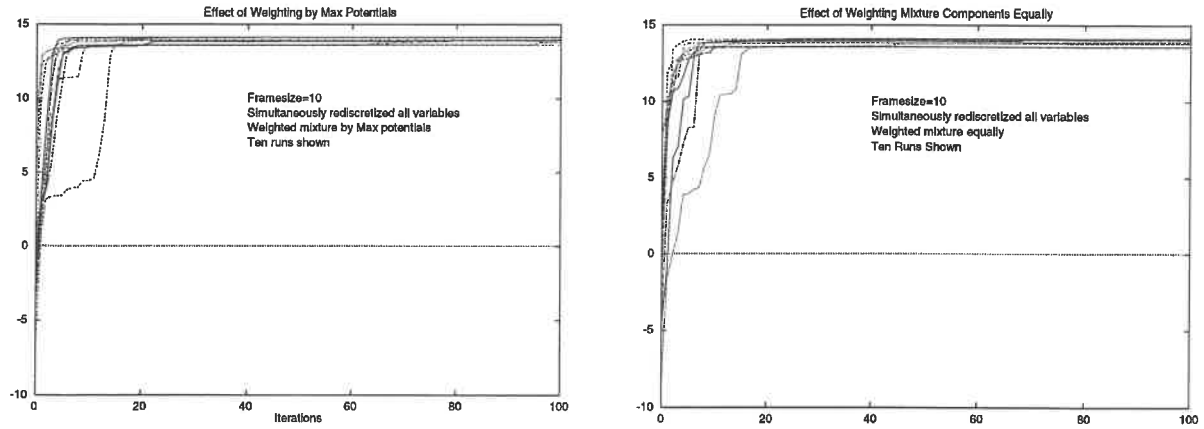


Figure 6.12: Effect of simultaneous rediscretization when neighbors' best are kept.



(a) (b)

Figure 6.13: Effect of using alternative weighting schemes (Section 4.4.3). (a) Using max potentials, (b) Using uniform weighting. These used $m = 10$, rediscretized all variables simultaneously, and retained the best value, the predecessor's best value, and the successor's three best values.

6.8 Alternative Weighting Schemes

Two final variations are obtained by using alternative weighting schemes, i.e., weighting the f_{mbp} mixture by so-called max-potentials or by weighting the components of the mixture equally. These variations are discussed in Section 4.4.3. Runs using these weighting schemes are shown in Figure 6.13. Averages are shown in Figure 6.14.

The experiments appear to indicate that this results in a very rapid initial convergence, quicker than with sum-potentials, but that the final fine adjustments takes slightly more time. In both cases, all 10 runs were very close to optimal within the first 20 iterations, but by the end of 100 iterations, many of the runs had not closed the remaining gap.

Throughly this research, I have noticed (informally and empirically) that it seems to be a general rule that variations can be more or less aggressive (opportunistic), and that greater opportunism generally results in faster convergence to local optimum solutions with a greater reduction in mobility out of those local optimums. The alternative weighting variants appear to largely violate this, displaying both a tendency to converge very rapidly and a tendency to avoid local optimums. However, the slowness at making the final adjustment may be an instance of this general tradeoff.

6.9 Frame Size

As a final variant considered, this subsection examines the impact of frame size (m) on the algorithm. Intuitively, we would expect a larger frame size to speed convergence. For instance, if the frame size were so large that the entire time scale of interest could be finely discretized to the nearest millisecond, the optimal configuration would be identified in the zeroth iteration. Furthermore, Gibbs sampling falls out as the degenerate case when $m = 1$, and the previous experiments have already demonstrated that maintaining multiple values (at least $m = 10$ values) can significantly improve convergence. What is not clear is exactly how convergence rate is impacted between the extremes.

First, even if larger frame sizes are better, they come at a significant computation price. The time and space complexity of the propagation algorithm scales as $O(m^2)$, so going from $m = 10$ to $m = 20$ quadruples the time for propagation at each iteration. Also, at this point there is no formal guarantee that larger frame sizes monotonically improve expected performance. One potential phenomena that could (and does) thwart monotonic improvement is discussed in the Drawbacks section of Section 4.4.1 and Figure 4.5.

The best observed performance from previous experiments seems to be from using max potentials, retaining a variable's best as well as its predecessor's best and its successor's three best. The experiments

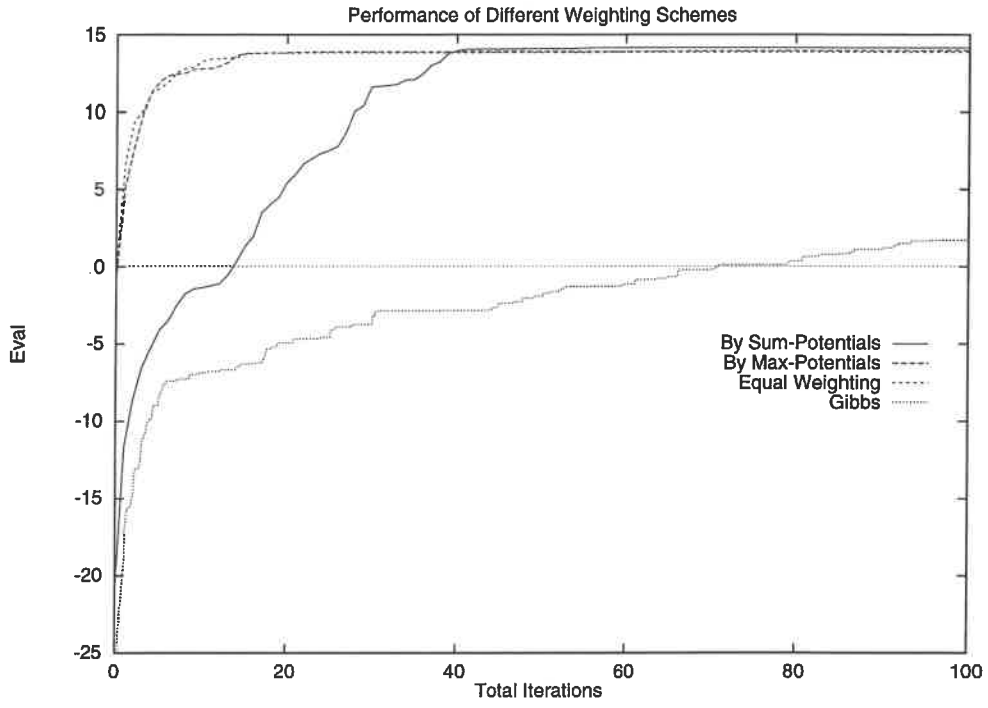


Figure 6.14: Average Performance using the various weighting schemes. (Individual runs shown in Figure 6.13).

here simply ran with these settings while varying frame size among $m = 10, 15, 20$, and 30. The results in average performance are shown in Figure 6.15.

This experiment turned up two surprises. First, convergence rates actually degraded slightly with increased frame sizes. It might be noted that larger frame sizes did get off to a slightly better start on the zeroth configuration ($m = 30$ started out with a zeroth interaction evaluation about 6 points higher than for $m = 10$), but just did not converge as rapidly. The phenomena of Figure 4.5 is to blame for this (discussed in Section 6.11.3). A good solution to this problem remains open.

The fact that larger m can decrease performance is both counter-intuitive and disturbing. The problem is discussed in detail on Page 86 and some further discussion is given in Section 6.11.3. Basically, it is caused by an interaction between consecutive iterations that causes the weighting of what appears promising to become distorted (roughly by counting the weighting twice). The weighting coefficients (max weighting in this case) are used to identify how promising discrete values for neighboring variables are. However, on the previous iteration, those values were selected based on how promising they looked at that time, so the spacing of those values also indirectly encodes the degree of promise. This has the effect of amplifying the attractiveness of those values already believed to be promising, and therefore decreases mobility despite the larger frame size.

This phenomena seems to indicate that a problem exists in the method by which $f(x)$ is estimated. The uniform weighting scheme would seem to eliminate this problem by relying only on the spacing of values for weighting. This is more-or-less confirmed by the experiment shown in Figure 6.16, in which the previous experiment was repeated using uniform weighting. The overall tendency in that graph is a slight improvement in performance with increasing m.

Although uniform weighting does seem to get around the double counting of importance that causes the decrease in performance with max potentials, it is not a perfect solution. Uniform weighting sacrifices the ability to quickly recognize a stray value that happens to stumble on very promising new value. The potentials can signal a highly promising value, even if it is a stray, while the uniform weighting scheme gives such a point only the same weight as every other point, decreasing the recognition of unexpected discoveries with larger frame sizes. This potentially hinders the algorithm's ability to explore. This leads me to believe

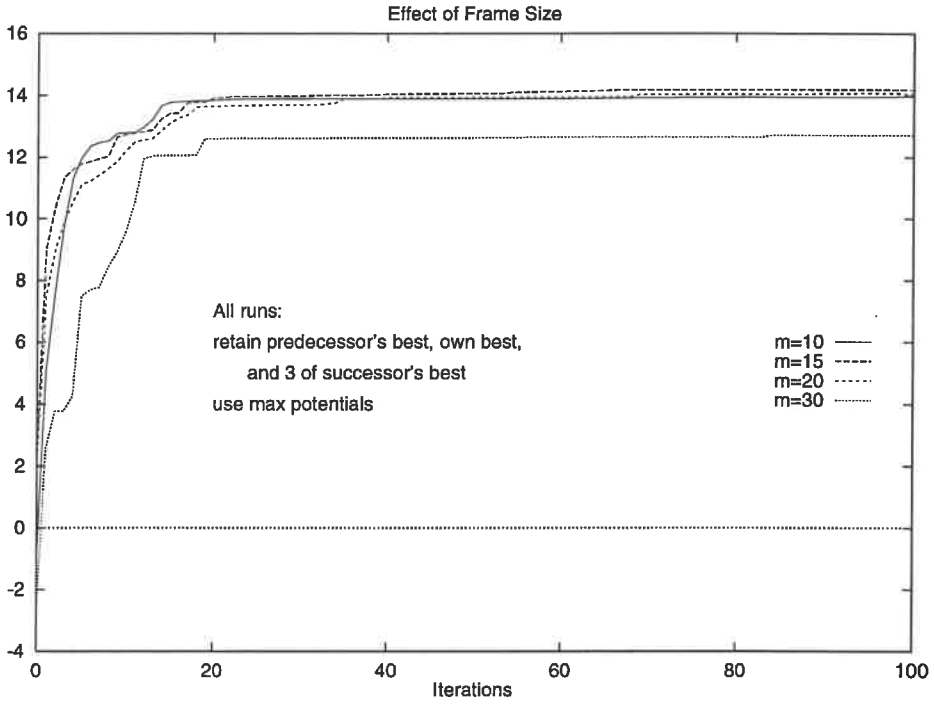


Figure 6.15: Effect of Frame Size.

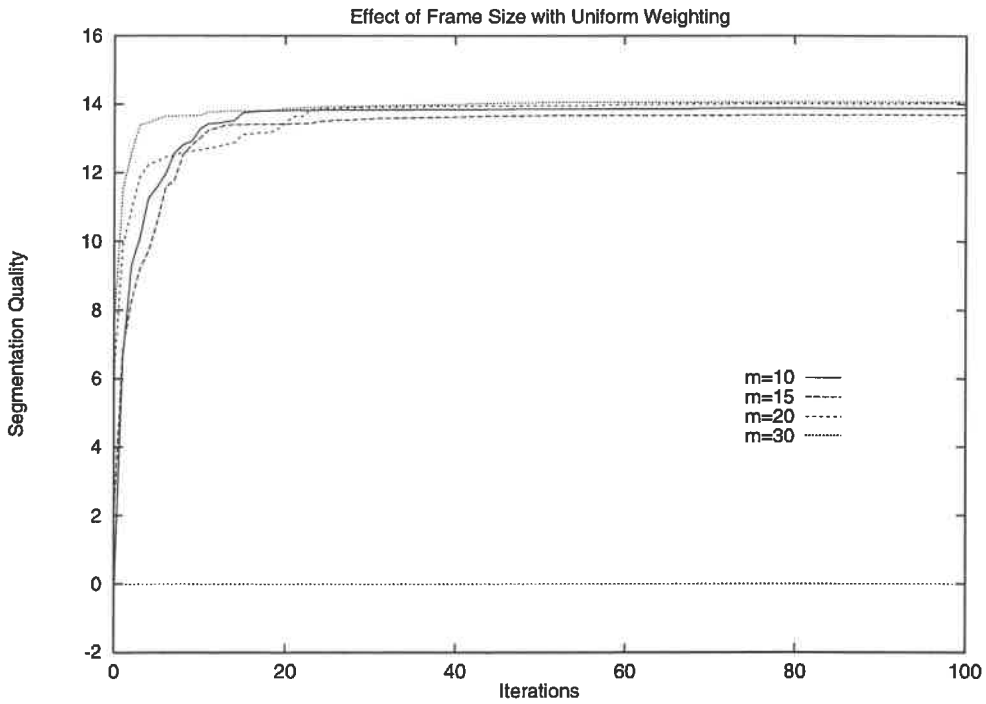
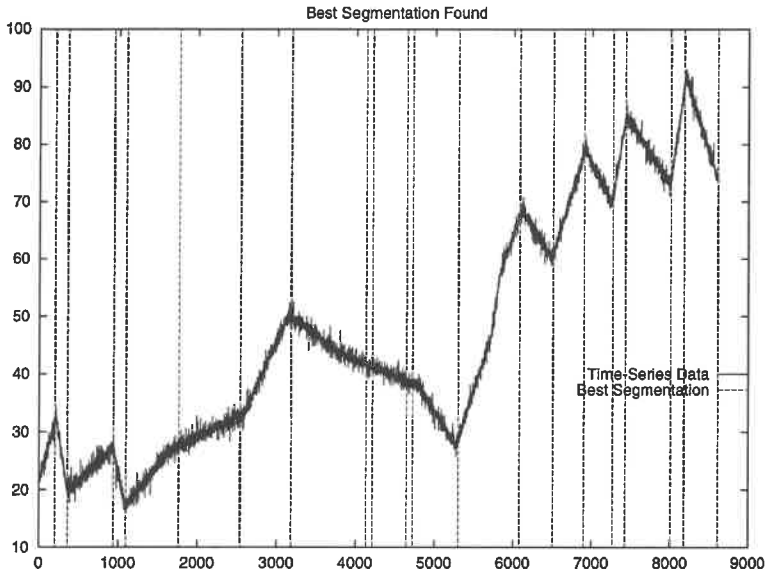


Figure 6.16: The effect of frame size when uniform weighting is used.



Optimal Parse?

i	s_i	t_i
0	*shortRise*	0
1	*shortFall*	204
2	*longRise*	365
3	*shortFall*	942
4	*longRise*	1099
5	*longFall*	1760
6	*longRise*	2533
7	*longFall*	3182
8	*shortRise*	4130
9	*longFall*	4212
10	*shortRise*	4647
11	*longFall*	4723
12	*longRise*	5299
13	*longFall*	6082
14	*longRise*	6508
15	*longFall*	6902
16	*shortRise*	7264
17	*longFall*	7424
18	*shortRise*	8005
19	*longFall*	8174
20	(*shortRise*)	8610

Figure 6.17: Best segmentation found during all experiments. It may be the optimal segmentation for this problem. It has an evaluation of 14.7.

that improvements on the method for estimating f could be made, even over the uniform weighting scheme. An improved method should somehow continue to use the computed weighting potentials while at the same time compensating somehow for the spacing of existing discrete values. To date, a reasonable method for accomplishing this has not been identified, so this challenge remains an important topic for future research.

A second surprise from the frame size experiments was the optimum segmentation found. The best segmentation found during all the experiments reported in the previous sections had an evaluation of 14.2. It agreed with the ground truth parse on the times of all transitions, but differed on the states at some of those transitions. Furthermore, it was consistently identified by the $m = 10$ algorithm. However, at least one of the runs for each of the larger frame sizes located an even better parse with an evaluation of almost 14.7 shown in Figure 6.17. Although the difference in evaluation is fairly small, I find it a little disturbing that the $m = 10$ variant consistently converged to the segmentation in Figure 6.11 without ever stumbling across the superior parse in Figure 6.17. Because of the many runs that had consistently found the same parse, I felt reasonably confident that Figure 6.11 was indeed optimal, until this set of experiments proved this not to be the case.

How is it that the true optimal could be consistently missed by so many runs, yet found several times by the runs with frame sizes greater than 10? This is an indication that the true optimum is very sharp, with nearby configurations being sub-optimal. Larger frame sizes are able to explore multiple possibilities at the same time, and are thus better able than small frame sizes are to stumble across local optimum with very small basins of attraction. The experiments suggest that this ability to effectively carry out multiple simultaneous searches is tremendously amplified by larger frame sizes.

6.10 Shuttle Data

This section briefly explores the performance of the iterative dynamic discretization algorithm on a time-series that was generated during a flight of the Space Shuttle. This time-series appeared in Chapter 2 on Page 28.

There are a couple reasons for including this experiment. First, it demonstrates the algorithm running on actual data produced by a real-world process. Second, it adds a second time-series to the experiments reported here (Chapter 7 adds a few additional permutations as well). Third, unlike the previous experiments, in the runs of this section I do not provide the algorithm with the time of the n^{th} transition. There were good reasons for providing the time of the 20^{th} transition in the previous experiments, e.g., to control for varying amounts of data coverage when comparing variants so that comparisons are meaningful. However, since such comparisons are not the focus here, and since an observation of that form would not normally be available in a time-series application, it is more interesting to run the experiments without the extra observation.

There is one very large deficiency in using non-synthetic data: we do not know the model that actually generated the data. This leaves open a very large degree of freedom, i.e., the specification of the model. This could, by itself, virtually determine the outcome of the experiment(s). As it is, the model must be subjectively assessed, and depending on what model is written down, very different outcomes could result. One way (partially) around this would be to learn the models from data, in which case an experiment like the one here would test both the learning algorithm's performance as well as the inference engine's performance. However, automatic learning of the models is beyond the scope of this thesis, and so a subjective assessment is necessary here. The model used here is listed in Appendix B. To obtain this model, I first read the description of the Shuttle's APU ([NSTS, 1988]) and gained an reasonable understanding of how the APU system functions. I then examined the time series for these two sensors over three different shuttle missions, and very quickly wrote down a description in the form of an HSSMM for what I believed I was seeing. In doing so, I made very crude estimates of the mean and standard deviations for transitions and for the "noise levels" on signal shapes, with a slight bias for over estimating uncertainty (to account for the imprecision of my estimates). Very little effort was spent on specifying the model, and the model was not refined further after any runs were performed. By its very nature of being a subjectively estimated model, the precise model used is inherently open to criticism; however, due to the lack of effort placed on obtaining the model, this criticism is somewhat minimized. A claim that the model has been designed specifically in a way that results in the desired performance would be unfounded. I believe the runs here are very representative of performance of the algorithm with reasonable models on this time series.

Figure 6.18 shows the rate of convergence for ten runs of the algorithm. For these runs, the estimation step used f_{mbp} with max potentials and a discretization frame size of ten was used. Discretization kept the predecessor's and own best values and three of the successor's best values, then picked the remaining five values in each frame using random sampling. Initially, the chain was grown to length ten (i.e., ten transitions). As before, the segmentation quality is measured using a log Bayes factor relative to a a parse I identified by hand. Since the data is non-synthetic, this simply serves the role of the ground truth parse and scales the segmentation quality numbers into a nice range. Figure 6.18 shows that in less than 15 iterations, all ten runs of the algorithm converge, with seven of the ten reaching almost total convergence in only three iterations. This is a pretty clear indication that this particular problem is considerably easier than the synthetic problem of the previous sections.

The final parse identified by these runs is shown in Figure 6.19. A few of the transition times appear to be slightly off from the obvious inflection points (e.g., at $t_3 = 12441$). This occurs because the signal shapes are modeled (Appendix B) as linear, while the actual shape is not linear. When these non-linear shapes are replaced by their best-fit lines, the transition times discovered by the algorithm result. Other than this slight discrepancy, the parse that was very reliably found is the segmentation we expect.

6.11 Phenomena and Artifacts

This section describes some of the phenomena that arises as a result of using a discretized model during the iterative model construction process. The insights in this section come from my experience examining

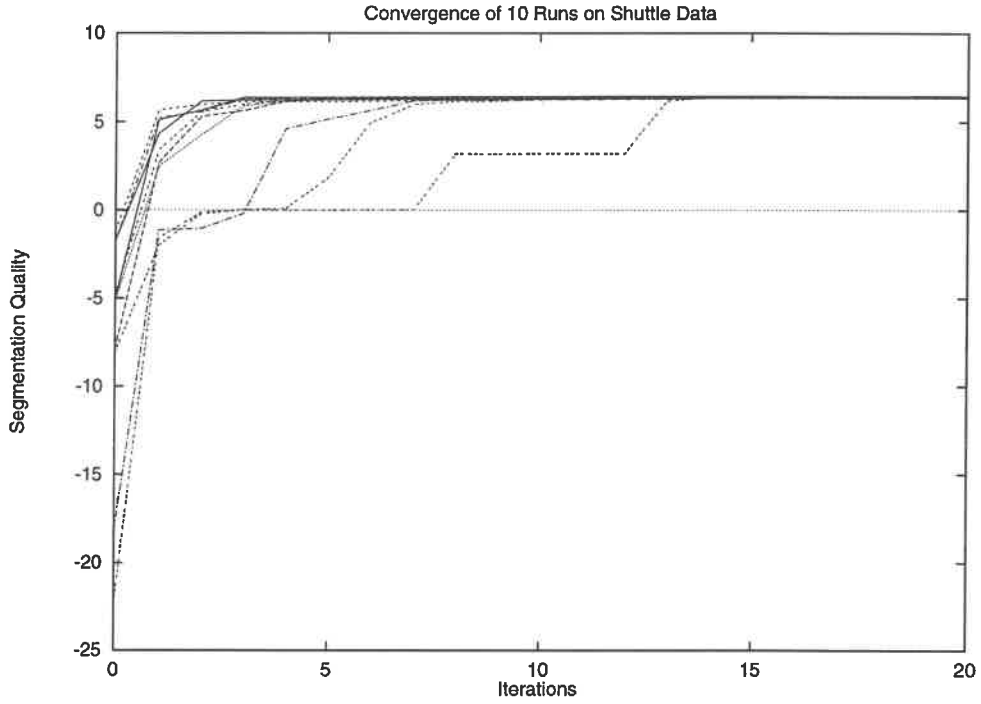


Figure 6.18: Convergence of iterative dynamic discretization on Shuttle Data.

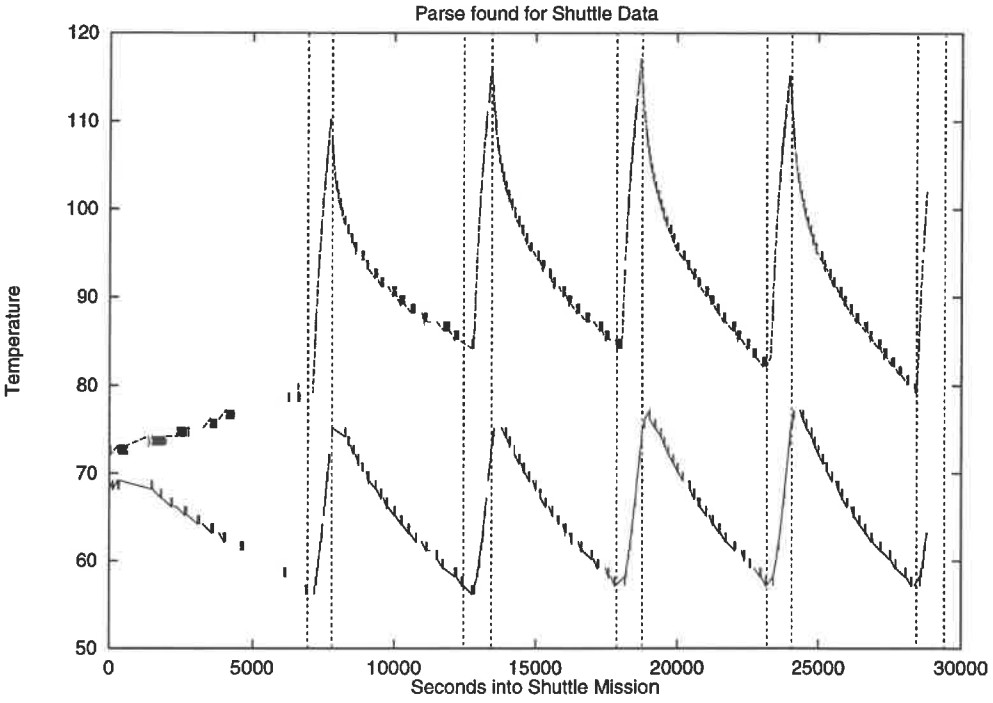
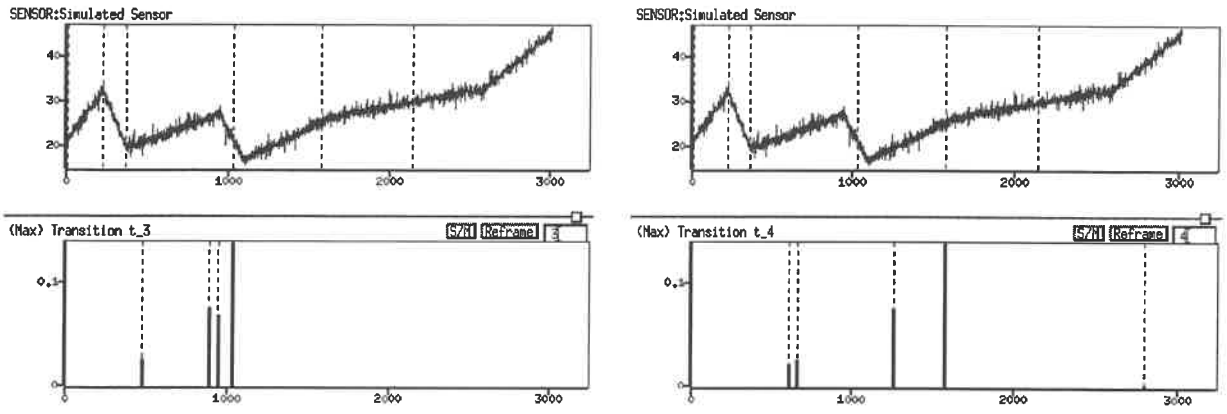


Figure 6.19: Segmentation found repeatedly by iterative dynamic discretization on Shuttle Data.



(a) (b)

Figure 6.20: Misaligned transition. The top panels show time series data and the best segmentation from those contained in the discrete model. The lower panels show the possible discrete values for t_3 and t_4. Because the discretization for t_4 (right) does not include the fourth inflection point in the data, t_3 (right) does not prefer the third inflection point in the data.

and debugging individual situations that have been observed to repeatedly occur when using the algorithm. Understanding these "artifacts" can be quite useful for making effective use of these and related methods, as well as for improving on the algorithm in the future. In addition, lessons from artifacts in this domain may transfer to other KBMC endeavors.

6.11.1 Misaligned Discrete Values

Many artifacts of discretization are an indirect result of a poor selection of possible values for some variable. If the best transition time is not included in the discretization for t_i, all other transition variables may be affected.

In Figure 6.20(b), the discretization for t_4 (appearing in the lower panel) does not include $t = 1086$, the time of the fourth transition in the data. Notice that the discretization for t_3 (lower panel of Figure 6.20(a)) does contain the time of the third transition; however, because of the "misalignment" in the discretization of t_4, the optimum value for t_3 is found to be something other than the actual time of the third transition.

In this example, $t_3 = 1022$ (between the third and fourth transitions in the data) results in a better segmentation than $t_3 = 940$ (the time of the third transition) given that t_4 can only be one of the five values in its current discretization. $t_3 = 940$ would produce a better data fit from t_2 to t_3, but would produce a much worse data fit between t_3 and t_4.

In some cases, this same problem can impact t_3 a second time if t_3 is rediscretized. Since $t_3 \approx 1022$ appears to be the most attractive value, a rediscretization of t_3 may cluster the new values around 1022, potentially eliminating $t_3 = 940$, the actual time of the third transition, from consideration (on that iteration). This, in turn, reinforces the belief that $t_4 = 1574$ (rather than $t_4 = 1086$, the actual time of the fourth transition), since $t_4 = 1574$ may indeed be optimal when $t_3 = 1022$. This result is that a bad framing can mislead the interpretation and a bad interpretation can mislead a discretization, quickly creating a locally stable (immobile) sub-optimal configuration.

Since this phenomena would seem to be inherent to the use of discretizations, methods (domain-dependent or otherwise) for increasing mobility are important for dealing with this artifact.

6.11.2 Growth Biases

Section 4.5 describes how a propagation graph is initially grown. During the initial growth phase, there is a tendency to insert more transitions than necessary (i.e., to space the transitions too closely together). There are several phenomena that create this bias, some of which are discussed in this section.

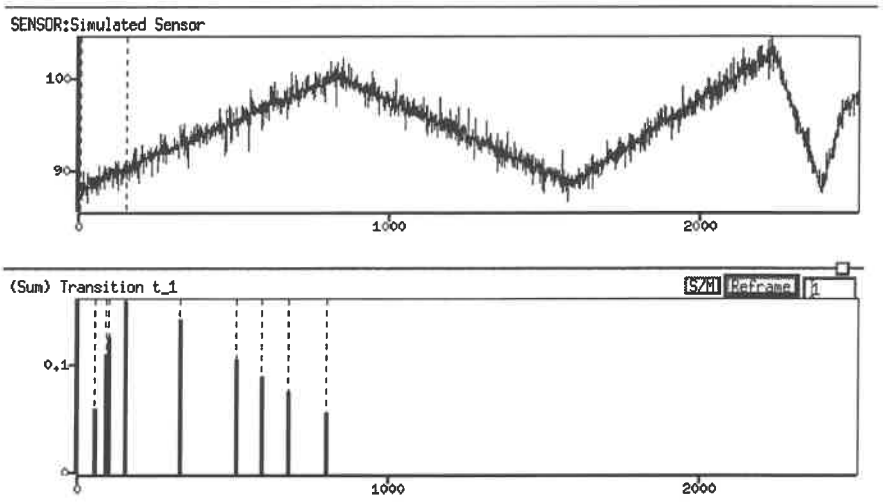


Figure 6.21: Two consecutive longer-than-average transitions. Bottom panel shows possible discrete times for t_1, with height of bar indicating discrete posterior probability for t_1 when t_1 is the horizon.

Limited Horizon

When a length-4 segmentation is evaluated, the evaluation criteria (Equation 2.3 on Page 35) includes the data fit from t_0 to t_4, but does not consider the state or data fit following t_4. The last transition time, t_4, serves as a horizon.

Times occuring before the "real" transition are often attractive values for the horizon variable. This is because there is a good data fit between t_{i-1} and t_i regardless of whether or not the shape changes at t_i. A change in the shape normally gets detected as a result of there being a good fit on either side of a proposed transition time (e.g., see Two-Window methods, Section 2.4.1, Page 40); however, the data fit for a horizon point only gets evaluated on one side.

Suppose two consecutive longer-than-average segments occur in the raw data (Figure 6.21). Even though a good initial framing is chosen for t_1 (i.e., one containing something close to $t_1^* = 824$ as a possible value), the mode of t_1's waiting-time distribution[8] — and not the value corresponding to the transition — appears most attractive. The data fit is virtually constant for all values before the transition in the data, so the potentials are almost exclusively determined by the waiting-time distributions.

When t_2 is added to the propagation graph, the mode-centered weighting of t_1 causes t_2's sampling distribution to be centered considerably earlier than where the actual transition for t_2 occurs (Figure 6.22). The result in this example is that t_2 gets assigned to the first transition in the data. In contrast, Figure 6.23 shows the discretization for t_2 that results when the precise time for t_1 is known with certainty. In this later case, the limited horizon effect does not influence t_2's discretization, and so the discretization is placed pretty much ideally.

Whenever a longer-than-expected transition occurs in the raw data, the limited-horizon bias kicks in, and since this does not occur in the opposite direction when quick transitions occur, the bias systematically causes extra transitions to be inserted during the growth stage.

Misaligned Values

When the problem of Section 6.11.1 occurs on the horizon of the propagation graph during the growth stage, an extra transition often gets inserted. This creates yet another bias for inserting too many transitions during the growing stage. To see why this occurs, consider Figure 6.24.

[8] It is actually the mode of the mixture of t_1's waiting-time distributions, where the mixture is weighted according to the initial state probabilities.

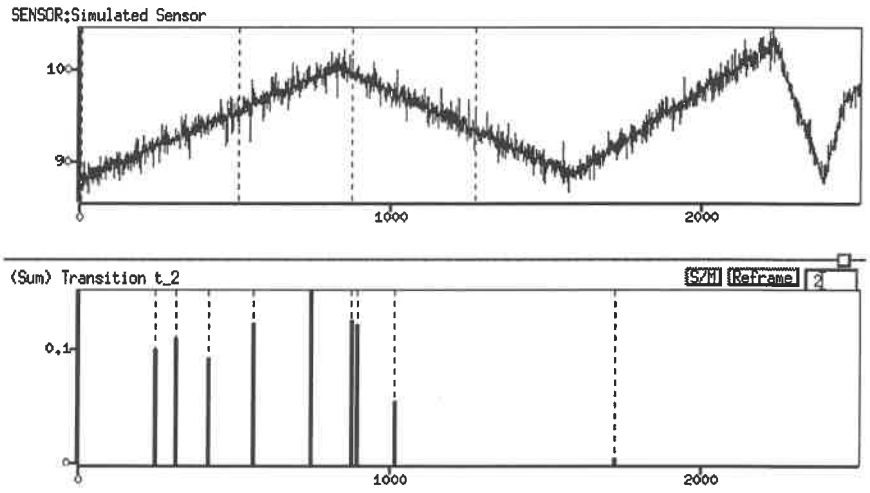


Figure 6.22: Impact of Limited Horizon on t_2's discretization.

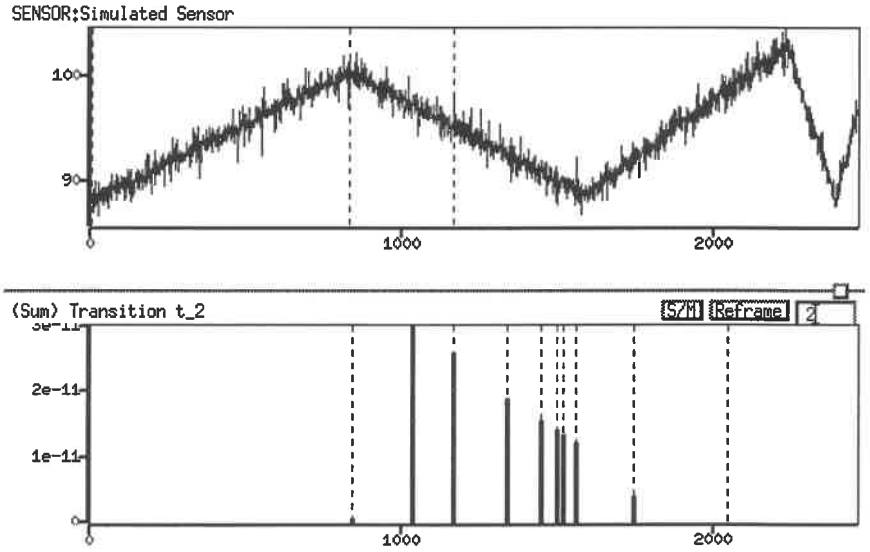


Figure 6.23: (Ideal) Placement of t_2's discretization, obtained when time of t_1 is known with certainty.

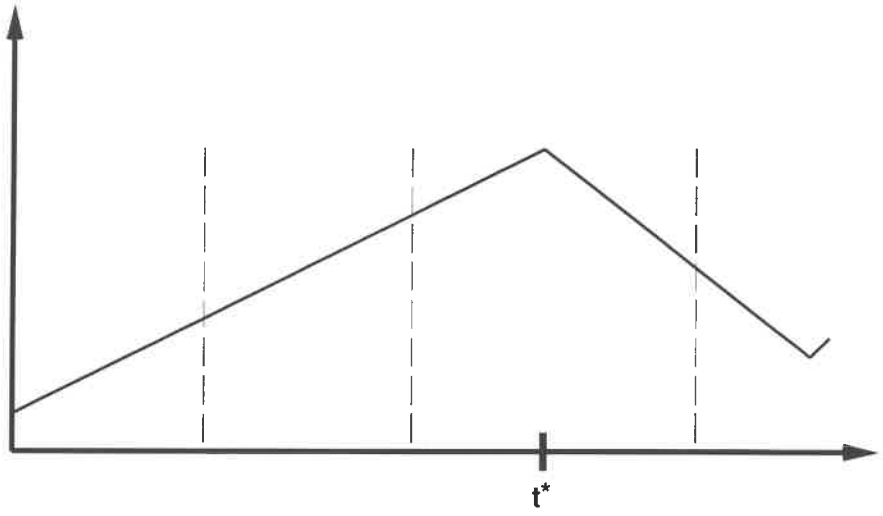


Figure 6.24: Misaligned discretizations on the horizon often results in an extra transition. Dashed vertical lines indicate discretization for t_1 ($m = 3$).

In Figure 6.24, t_1 has three possible values, but none of these values are perfectly aligned with the obvious transition (t^*) in the data. When a second transition is added, there are three choices for possible segmentations:

1. Both transitions occur after t^*, i.e., $t^* < t_1 < t_2$.
2. One transition occurs on each side of t^*, i.e., $t_1 < t^* < t_2$.
3. Both transitions occur before t^*, i.e., $t_1 < t_2 < t^*$.

In the first two cases, the best fit line must cut across both segments in the data, resulting in a poor data fit in either case. However, in the third case, the best fit line from t_0 to t_1, and the best fit line from t_1 to t_2 both coincide with the straight line in the data (from t_0 to t^*). Thus, in the third case, the data fit is very good. Note that this assumes the model allows for two consecutive rising segments. The same would not occur in a model with deterministic transitions in which a rising segment is always followed by a falling segment.

If the value t^* were included in the discretization of t_1, a very good data fit would occur with $t_2 > t^*$. Since the third case gets penalized somewhat by the waiting-time expectations, this would be the optimum configuration. However, due to the misalignment of the discretization for t_1, i.e., the fact that t^* is not a value in that discretization, the penalization from expectations may be less penalty than the bad data fit resulting from either of the first two cases.

Once again, this artifact of discretization creates a bias during the growth stage towards more rapid transitions.

Right Skewness

Waiting-time distributions are typically skewed to the right — i.e., with *mode* < *mean*. The reason for this is that the distributions cannot extend into $\Delta t < 0$, but usually do have a tail to the right. For example, any gamma distribution with $\alpha > 1$ is right skewed.

The result of right-skewness is that the majority of transitions last longer than the most likely duration. Thus, the maximum likelihood estimate of duration is in most cases an underestimate of the actual duration.

Max-potential weighting tends to center the discretization of a new stage at the expected distance from the *mode* of the previous horizon. Since the max-potential identifies the most likely transition time, there is an immediate bias towards short transitions. For other weighting schemes, right skewness interacts with the limited horizon effect to amplify the bias towards quicker transitions.

Local Stability of Grown Graph

The above biases that occur during the growing stage would not be problematic if iterative discretization could quickly correct the previous mistakes. However, all of these phenomena tend to identify locally stable initial discretizations. Once a bad choice is made during one step of the growing stage, stages added subsequently base their discretization on the previous bad choice. These make the bad decision look good locally since it fits well with the frames that follow. This tends to make the initial discretization locally stable and therefore difficult to improve upon using local or greedy optimization schemes. This makes the issues of mobility even more important.

Finally, it should be noted that all of the growth biases are ameliorated somewhat by using a uniform weighting scheme (see Section 4.4.3) for the horizon. Since the weighting communicates several of these artifacts to the discretization process for the added stage, ignoring weights altogether during growth helps to eliminate partially some of the biases.

6.11.3 Discretization Density

One undesirable artifact (which has already been discussed in Section 4.4.1 on Page 86, but is worth mentioning again) becomes quite significant when larger frame sizes are used with a non-uniform weighting scheme. The spacing of discrete values from the previous iteration can have a significant effect on the resulting estimate $f(x)$, particularly when m is large. In fact, this phenomena appears to be responsible for the decreased performance found in Section 6.9 when m is increased. The reader is referred to Page 86 for a full description of the phenomena, and to Section 6.9 for further discussion of its impact.

Loosing speaking, what happens is that the density of discrete points in the previous iteration indirectly reflects how promising possible values appeared during that previous iteration. The weighting coefficients (from the potentials) are then also introduced to determine importance. The two are somewhat redundant, and the resulting estimate becomes distorted as a result.

It was found in Section 6.9 that the uniform weighting scheme pretty much eliminates this artifact. On the other hand, (intuitively) the uniform weighting scheme would seem to ignore information that should be valuable. This presents somewhat of a dilemma. When we try to use weighting information intelligently, the discretization density artifact tends to create problems as m grows. This may be an aspect of the algorithm where future improvements in the estimation procedure may be possible with performance impact.

6.11.4 Local Stability

Throughout the previous sections, the issues of mobility and local stability (or immobility) are mentioned repeatedly. Because I have observed this phenomena to be such an important determiner of the success of iterative discretization algorithms, it deserves to be singled out and emphasized as an important phenomena.

Often discrete models are produced by the algorithm with the property that a rediscretization of any single variable will not result in an improved solution. Often it takes a simultaneous shift in several variables (often with each individual variable's required shift being somewhat unlikely to occur via random sampling) to yield an improved solution. It is in this sense that we say discretizations may be locally stable. This is a form (the predominate form) of immobility.

Many of the other artifacts discussed above are significant precisely because they tend to produce discretizations that are locally stable. In most of the cases, one of the above artifacts causes a preliminary commitment of some kind to be made. Choices later are made *based on that commitment.* Soon, even if the original choice was selected poorly, the subsequent choices artificially reinforce the commitment, making it locally stable since the subsequent choices all appear most appropriate for that original selection.

This phenomena makes locally stable discretizations very common. It makes it incredibly important to insert methods for increasing mobility (or decreasing local stability). This is why the technique of "keeping the neighbors' best" (Section 6.7) is so effective. It allows the algorithm to easily slide between what would otherwise be locally stable configurations.

The fact that the phenomena of local stability is so common is perhaps one of the most important lessons that this case study has for other iterative endeavors, including iterative KBMC methods. Solutions

for ameliorating this phenomena are likely to be varied and often domain-dependent, but it is extremely important to be prepared to address this problem from the outset of one's project.

6.12 Conclusions

This chapter explores the the empirical performance of iterative dynamic discretization and reveals a number of general lessons for related lines (or potential lines) of research.

Experiments demonstrate that the iterative dynamic discretization algorithm has significant potential to drastically decrease the number of iterations required as compared to Gibbs sampling.

A more careful examination of where this power comes from revealed a number of insights that are generally useful for any iterative model construction attempt. Foremost is that mobility (or lack of it) is a primary determiner of convergence rates. Mobility is essentially the ability to move from one qualitative class of solutions to another. This is, of course, well-recognized in the Markov chain Monte Carlo literature [MCMC, 1996], but before this chapter, the similar connection to model construction was anything but obvious. However, this case study has demonstrated that once the underlying reasons for mobility are identified, domain-specific methods for by-passing barriers to mobility can often be introduced. This was possible here (and not possible with Gibbs sampling, for example) due to the flexibility enabled by iterative dynamic discretization — in particular, because there are multiple possible values for each variable, there is room to use some of these to implement domain-specific channels to enhance mobility.

The iterative dynamic discretization algorithm is, so far, an algorithm developed and explored empirically. Although some theoretical aspects were considered briefly in Chapter 5, these say very little about the efficacy of the algorithm. The potential benefit of the approach has been demonstrated, but the algorithm remains ripe for future improvement and analysis.

Chapter 7

Time-Series Modeling Issues

The previous chapters present a formalism (the HSSMM) and algorithm for model-based time-series segmentation. What if a good model of the time series is not available? Does this mean the approach is useless? Or does the algorithm find reasonable segmentations even when the model is not very good or even slightly incorrect? How informative is a model of a time-series? Is the really important segmentation information mostly in the model or is it in the time-series data itself? How general is the HSSMM formalism? Are there important examples of time series that cannot be faithfully modeled by a HSSMM? Can a time-series model be learned form data? How? When would it be a good idea to do so? These are all examples of important issues that arise for any model-based time-series segmentation approach. They are all questions aimed at the modeling formalism rather than at the inference algorithms that use the knowledge and perform the search. This chapter briefly touches these issues. A thorough understanding of any one of these issues could form the basis of a thesis by itself, and consequently the discussion here is anything but complete. Nevertheless, it is still instructive to review these issues however briefly.

One common characteristic of Bayesian-style probabilistic models is that they support a range of knowledge from ignorance to informedness. In a model-based time-series segmentation task, this creates the flexibility to control the degree to which the segmentation process is driven by the data or driven by the model. This continuum is the primary topic of Section 7.1. Experiments in that section help to provide a very limited understanding of the robustness of the HSSMM formalism.

Section 7.2 discusses some important limitations of the HSSMM. The HSSMM utilizes a semi-Markov model. There are at least three common cases arising in practice that are not semi-Markov, and as a result, there is useful knowledge that cannot be captured within a HSSMM. Section 7.2.1 generalizes the HSSMM to the HS$\underline{\underline{G}}$SMM, based on a *generalized* semi-Markov model. The extension covers the interesting non-semi-Markov cases, but I do not study the HSGSMM further in this thesis. The discussion is intended to introduce a potentially useful time-series knowledge representation formalism that may be of interest in the future.

Finally, in Section 7.3, I provide some (unsatisfactorily brief) comments about the prospect of learning HSSMMs from raw time-series data. During the course of this research I have not attempted to develop or study learning algorithms for HSSMMs, and the ability to do so is in no way a prerequisite for the applicability of the methods in this thesis. Model learning is, nevertheless, an obvious issue that is frequently brought up during discussions of this work, and therefore is worthy of some discussion here.

7.1 The Model-Directed to Data-Directed Continuum

Examining historical stock market data over the past ten years (Figure 7.1), when did the S&P 500 last change from bear to bull? This time-series segmentation task is perhaps best analyzed in a very *data-directed* fashion — i.e., the raw time-series data, and not prior expectations, should dominate the determination of when the transition occurs.

It would be unusual for two consecutive heart beats to be separated by less then 300 milliseconds or by more than 2 seconds. In an ECG application, it is conceivable that a signal or signal to noise ratio may

Figure 7.1: The S & P 500 over the past 10 years.

become weak (e.g., from a poor sensor contact, insufficient amplifier gain, exogenous noise, etc.), in which case a segmentation system may still be required to identify transitions in the very weak signal. To do so effectively, it is wise to make good use of prior expectations. In this case, a model-based segmentation approach is warranted.

There is a continuum from data-based to model-based approaches to segmentation, distinguished essentially by the strength of prior expectations applied to the segmentation task. Weaker expectations result in a greater reliance on data. Consequently, this can also introduce a greater susceptibility to noise in the data. Stronger expectations create a greater reliance on prior knowledge, and hence, a greater chance of missing the unexpected. Between very strong and very weak expectations, there is a continuous range of possible balances between model- and data-based approaches.

This thesis describes a model-based time-series segmentation algorithm, and some readers may have the initial reaction that a model-based approach is inappropriate for their application because they have little predictive knowledge; however, a nice attribute of some model-based approaches (e.g., the one in this thesis[1]) is that it allows the user to traverse the data-to-model based continuum as needed.

In the HSSMM, strong expectations may exist in the form of low variance waiting-time distributions, low entropy transition probabilities, or highly-selective shape recognizers. It is possible to make an algorithm more data directed by increasing waiting-time distribution variances or transition entropies, or by decreasing the dynamic range of shape recognizer outputs.

It is of general interest to characterize the robustness of specific time-series models to possible alterations in information content. Models, almost by definition, are never exact, and so it is pragmatically important to recognize when a model inaccuracy will break a system, when excess ignorance will cause important transitions to pass unrecognized, or when excessively strong expectations may cause critical unexpected events to be ignored. While these are important considerations, they are also extremely difficult to characterize in a useful and general way.

One approach to studying robustness properties of the HSSMM formalism is to examine how the optimum parse of a fixed time series changes as various parameters of the model are altered to create stronger or weaker

[1] Generally, this is true of modeling formalisms that support the representation and manipulation of explicit degrees of ignorance, as is the case with probabilistic models.

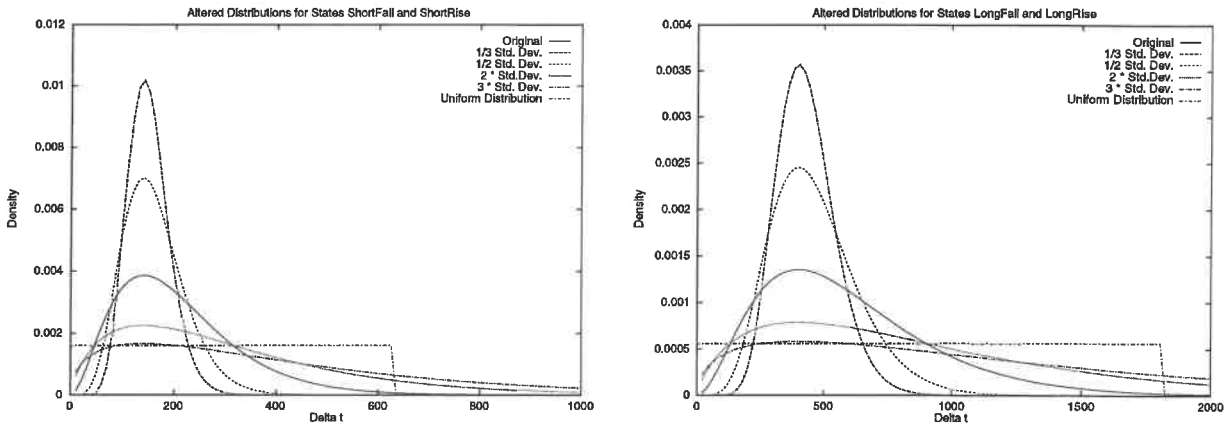


Figure 7.2: Waiting-time distributions for experiments. New models were obtained by varying the standard deviation of waiting-time distributions while keeping the same mode. A fifth model used a uniform distribution.

expectations. Some experiments along these lines follow.

7.1.1 Diffuse Waiting-Time Distributions

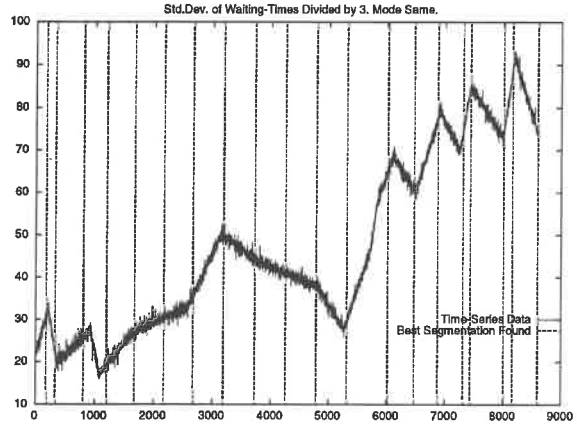
This section examines the effect that changing the variance of waiting-time distributions has on the optimal parse. First, a synthetic time series was generated. Here, I use the same time series used for the experiments in Chapter 6, which was generated from the model in Appendix A. A series of altered HSSMMs were then obtained by replacing the waiting-time distributions in the original model with various distributions. The objective is to compare how the optimum segmentation relative to the altered models differs from the optimum segmentation relative to the original model. This comparison gives an indication of how robust the optimum segmentation is to ignorance or over-specification in the waiting-time distributions.

As with the experiments in Chapter 6, I compared the optimum segmentations given the data and given t_{20}. This eliminates a number of troublesome phenomena that could cause some variations to be incomparable. However, one pragmatic problem exists: what is the optimum segmentation relative to these alternative models? Since an omnipotent oracle was not available for consultation, I used the iterative dynamic discretization algorithm — the most reliable variant from Chapter 6 — to search for the optimum. In each case, I ran at least ten trials each for at least 100 total iterations each and reported the best segmentation found during those runs. In some cases, I ran the algorithm considerably longer and for more runs until I was convinced that the resulting segmentation probably was the optimal one. Obviously, trust must be placed in the performance of the search algorithm, i.e., that it finds a nearly optimal segmentation; nevertheless, all the segmentations shown are considerably better (with respect to the corresponding altered model) than the optimal parse relative to the unaltered model.

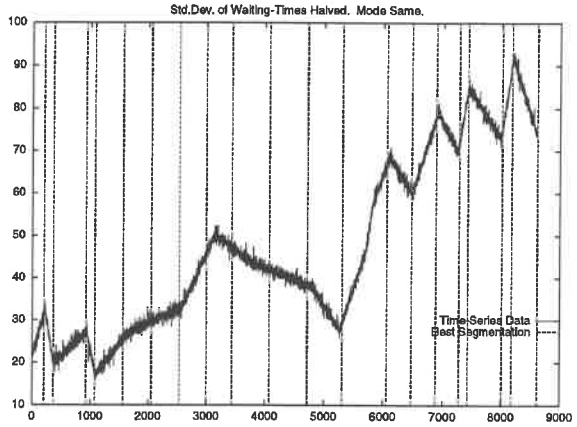
In the original model (Appendix A), the waiting-time distributions are gamma distributions. Four alternative models were obtained by replacing these with gamma distributions having a standard deviation of 1/3, 1/2, twice, and triple the original standard deviation while keeping the mode of the distribution the same. Keeping the mode the same means that the optimum segmentation given no data is not changed. A fifth alteration was obtained by replacing the gamma waiting-time distributions with a uniform distribution over $[0, 3 * \text{mean-of-original}]$. These five variations on waiting-time distributions are shown in Figure 7.2.

In cases (a) and (b) of Figure 7.3, the standard deviations of waiting-time distributions are reduced, resulting in stronger expectations, and therefore less reliance on the data. The expectations are strongest in (a). Many transitions are placed at points other than the inflection points in the data. This is because doing so results in durations much closer to the expected duration, and because of the reduced waiting-time variance, this becomes more important than getting a good fit on the data.

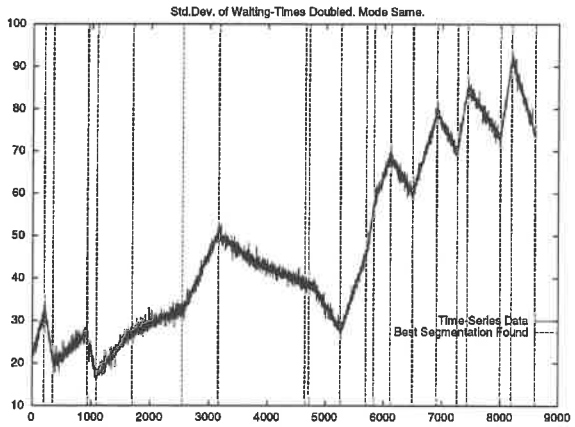
In cases (c) and (d) of Figure 7.3, the standard deviations of waiting-time distributions are increased, resulting in weaker expectations about transition times. In (d) the expectations are weakest. In these cases,



(a)

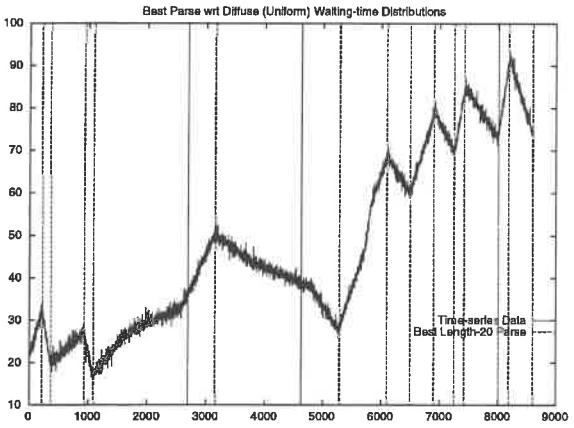


(b)



(c)

(d)



(e)

Evaluation of optimal parse of altered model relative to optimal parse of original model with respect to the altered model:

	Eval.
(a)	10.4
(b)	3.8
(c)	1.3
(d)	0.9
(e)	4.5

Figure 7.3: Optimal Parses for Altered Waiting-Time Distributions. (a)-(d) are for models where waiting-time distributions are replaced by gamma distributions with the same mode as the original, but with a standard deviation of (a) 1/3, (b) 1/2, (c) twice, and (d) three times the original standard deviation. (e) is for uniform waiting-time distributions.

small amounts of noise in the data have significant impact and are responsible for the more counter-intuitive transition times identified and omitted. There are two "tricks" that improve the fit when the expectations are weaker — a very short transitions can be placed over a few noisy data points to get a very good fit (since there is only a few data points to fit). A long transition can be placed over a long segment of hard-to-fit data, with a fairly poor fit, but with only a one-time penalty, leaving more transitions for short, nearly perfect fits. So, for example, instead of having 5 okay fits over a segment of data, it can instead use 1 long poor fit and 4 short great fits. As expectations become weaker, or as noise in the data increases, these tricks become more and more profitable.

Finally, Figure 7.3(e) is also a form of very weak expectations on waiting-time, and is in many ways comparable to (d). However, the expectations are of a slightly different form, coming from a uniform distribution, and the evaluation is perfectly happy using virtually instantaneous transitions to obtain perfect data fits. Because of the limited extent of the uniform distribution, it is not able to clump all the data between 3149 and 5269 into a single transition as was done by (d). In this way, the expectations for (e) are both stronger and weaker than those of (d), and so a slightly different segmentation results.

Shown on the bottom right of Figure 7.3 are the relative evaluations of the optimal parses for each model. These are the evaluations given by (6.1), except that the optimal parse for the original model is used in place of λ^*, and the evaluation is done with respect to the altered model. These numbers give an additional indication of how robust the models are to changes in expectations. For example, even though model (d), the one where waiting-time variances were multiplied by 9, the optimal parse for the original model is still rated favorably by the altered model, differing only by a factor of $e^{0.9}$. This is an indication that the model is reasonably robust to increases in waiting-time variances. On the other hand, the numbers indicate substantial changes to the evaluation for decreases in variance, indicating that it is far less robust to underestimates of waiting-time variances. There is one small anomaly in this table: the evaluation for (d) is less than for (c), even though (d) differs more from the original model than (c). Although not a proof, this is a plausible indication that the segmentation found for (d) might not be the optimal one.

7.1.2 Transition Probabilities

To examine the robustness to ignorance in transition probabilities, the transition probabilities of the model in Appendix A were all replaced with equal probability transitions (maximal ignorance). Iterative dynamic discretization was then run ($m = 10$ with max potentials, retaining predecessor's best, own best, and successor's three best) 10 times for 100 iterations. Eight of the 10 runs found the same segmentation (the remaining two found something only slightly inferior), shown in Figure 7.4 and scoring 2.4 above the best known parse for the original model (from Figure 6.17). The impact of ignorance in transition probabilities seems to be fairly small.

The opposite of excess ignorance is to have overly strong expectations about transitions. The impact of overly strong expectations about transitions is rather predictable and is shown in Figure 7.5. In this experiment, the transition probabilities in the model in Appendix A were altered such that each 0.1 probability was changed to a 0.05 and each 0.4 probability was changed to 0.45. The data was generated so that 0.8 of the transitions cause an inversion in the sign of the slope, however the model used to parse the data in the experiment assumes that 0.9 of the transitions should result in slope inversion. In addition, the uniform distribution on initial states was changed to $b_{shortRise} = 0.1$, $b_{shortFall} = 0.2$, $b_{longRise} = 0.3$, and $b_{longFall} = 0.4$. This essentially introduced expectations counter to what actually appears in the data; however, this change to initial distribution seems to have had absolutely no influence on the optimal transition. Thus, it is more highly biased to find transition points where the slope changes sign. This is exactly what happens in Figure 7.5.

7.1.3 Noisy Data

In this section experiments examine the effect of noisy data on the segmentation. Noise in the data is one aspect, another essentially equivalent concern is the effect of the strength of model's expectations about noise in the data. Overly weak expectations (i.e., the expectation that noise levels are greater than they really are) may reduce the power of the model, while overly strong expectations (i.e., the expectation that the data is cleaner than it really is) may cause a segmentation to be mislead by noise.

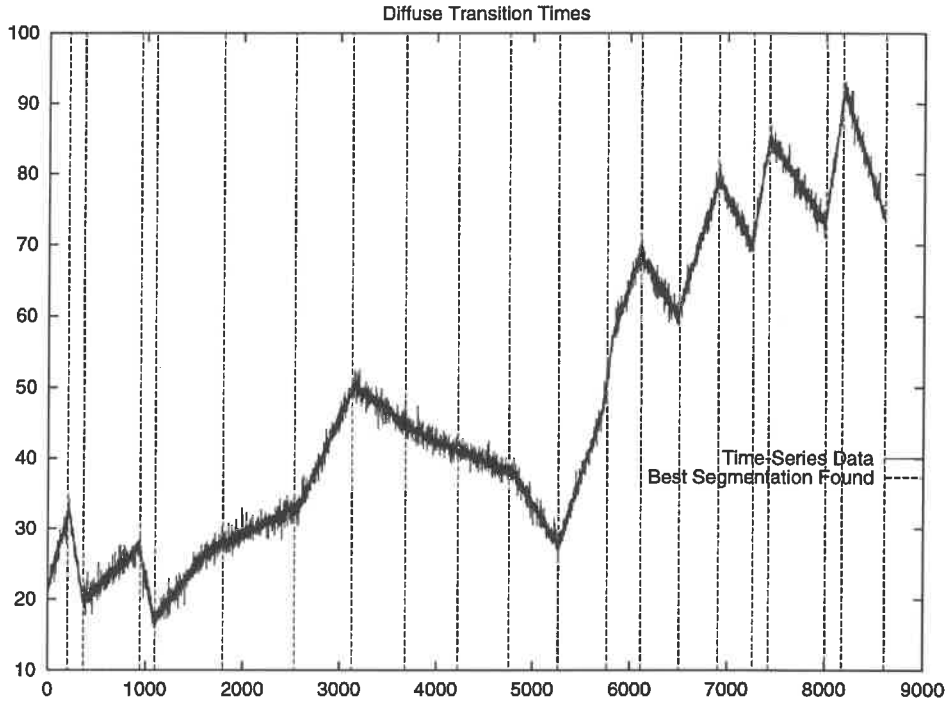


Figure 7.4: Effect of ignorance in transition probabilities.

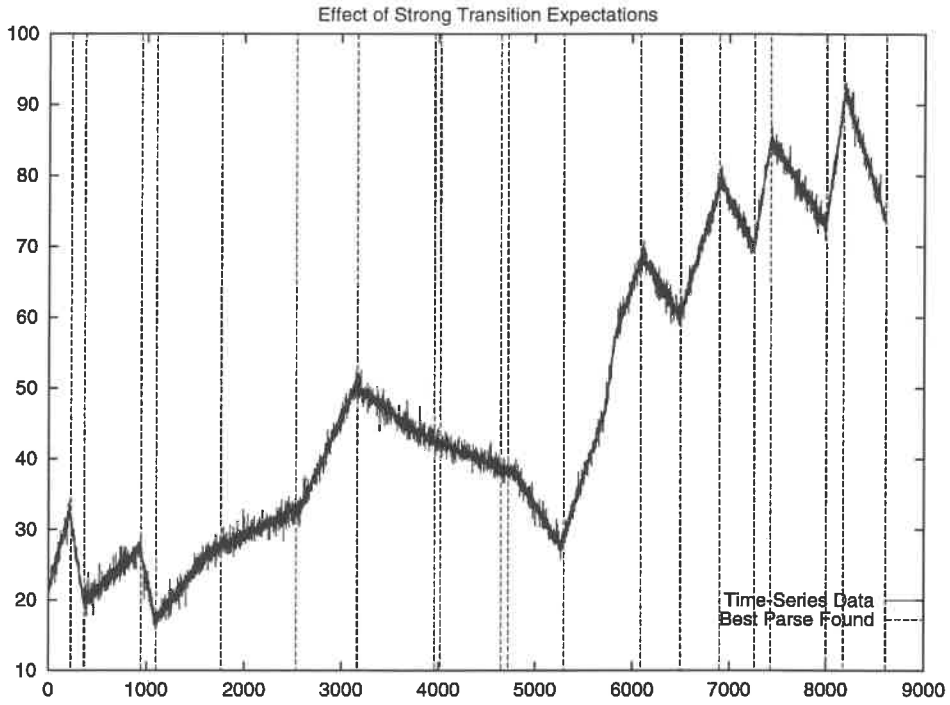


Figure 7.5: Effect of overly strong expectations about transition probabilities.

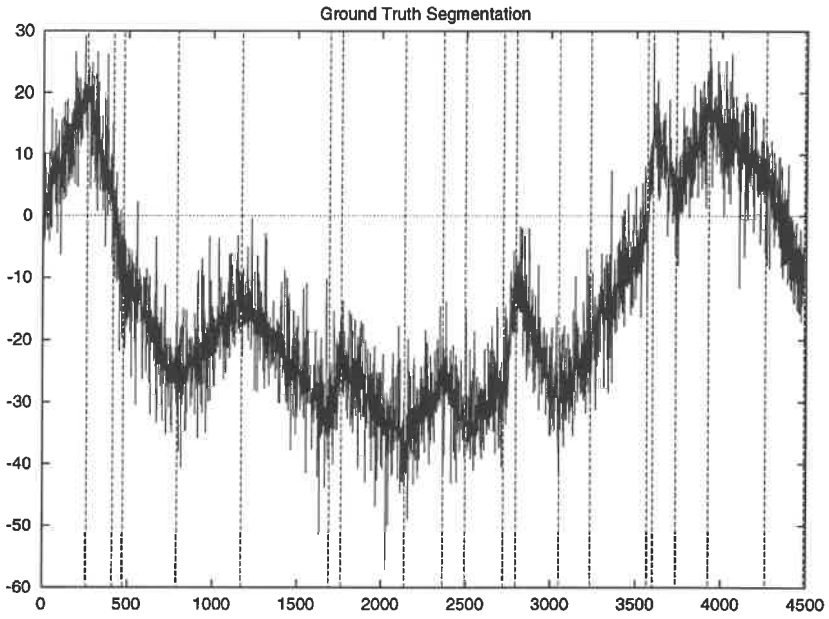


Figure 7.6: A time-series with ground truth transition times shown.

I examined these relationships by generating a new time-series using essentially the same model appearing in Appendix A, except for the modification that the noise levels on the sensor shapes were changed to a noise level of 3.0 (from 2.0) for `LINEAR_RISE` and `LINEAR_FALL`. The difference in signal-to-noise is noticeable in the resulting time-series, shown in Figure 7.6. Figure 7.6 also shows the ground truth segmentation used to generate the data shown.

The data in Figure 7.6 was segmented using four different models, each differing only on the noise level for the shape recognizers. The models used noise levels of (a) 2.0, (b) 3.0, (c) 6.0, and (d) 12.0. The resulting segmentations are shown in Figure 7.7. Each of these segmentations were obtained by running iterative dynamic discretization with a frame size of 10, max potentials, keeping the predecessor's and own best and successor's three best points, and rediscretizing simultaneously. Each experiment was repeated 10 times, and in all four cases all ten runs found approximately the same quality result. The best segmentation of the ten runs is shown in each case. In all cases, the segmentation shown evaluates considerably higher than the ground truth parse, the relative evaluations being: (a) 40.3, (b) 22.1, (c) 15.7, and (d) 14.5.

The result in Figure 7.7 (and the author's own experience with other examples) suggests that the resulting segmentation is quite robust to weakenings in expectations of noise levels, and is also very robust to noise provided that expectations are correspondingly weakened. For example, the segmentations in Figures 7.7(b-d) are all quite similar to the ground truth segmentation. The HSSMM is less robust to expectations that the data is cleaner than it actually is. This is demonstrated by the clustering of transitions around 1600 and the lumping of several transitions between 1700 and 2700 into a single segment in Figure 7.7(a).

7.1.4 Comments on Robustness

The above controlled experiments give some insight into the robustness of the HSSMM to various possible alterations in the model; however, some less formal comments may be even more informative. During the course of this research, I have have hand constructed several models for various time series, some artificial, some from real-world data sources. This has given me some hands-on experience, some feeling about how difficult it is to design an HSSMM for an application, and some idea of which parts of the model have the greatest sensitivity.

First, it seems to be far too simplistic to say that the HSSMM is, or is not, robust to imprecision in the parameters, or that it is highly sensitive to selected components of the model but not to others. My experience has been that the difficulty in designing a model is not in getting any single component right,

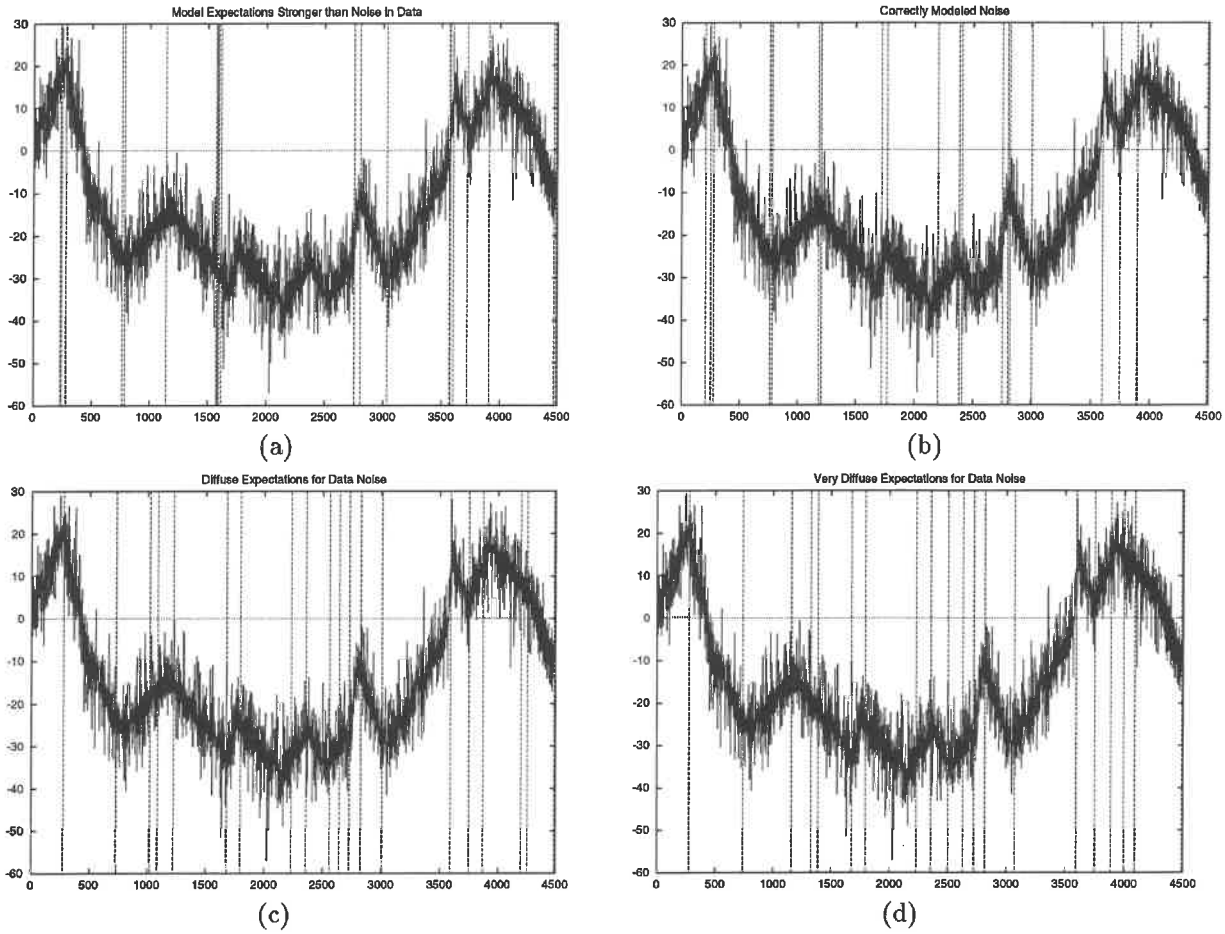


Figure 7.7: Effect of noise vs. expectations about noise on segmentation.

but instead the difficulty is in balancing the various components.

Each of the components — transition probabilities, waiting-time distributions, shape-recognizer noise tolerances, etc. — have a certain strength of expectation. The biggest problem seems to be in getting the balance of these expectations right. When the waiting-time expectations are weaker than the shape-recognizer expectations, then one ends up with something overly data-directed. This can happen even when both expectations are too strong or both are too weak. The thing that makes finding a satisfactory balance difficult is that there is no direct comparison between the expectations in each component. How does the noise tolerance of a shape recognizer relate to the standard deviation of a waiting-time distribution? This does not seem to be something that has a single application-independent answer.

In general, I did not find the task of achieving a satisfactory balance to be trivial. It was typical for me to guess at a model, then run the algorithm and notice either data-driven or model-driven effects dominating beyond what I felt was reasonable. I would then have to go back and adjust the parameters and run it again. I found that I typically had to repeat the process two or three times to get it to where I felt it was satisfactory. As a result, I do not make the claim that the HSSMM is highly robust to inaccuracies in model parameters. I cannot make the claim that by simply eye-balling available data and writing down reasonable guesses for the parameters that you will automatically get good results from the segmentation algorithm. Again, precise values for any single component do not seem particularly important, but getting the balance between components right so that the relative importance of data fit versus model expectations is reasonable is critical. This is, of course, a generalization, and in certain extreme cases may not be entirely true.

I did experiment with some Shuttle data in which the transition probabilities (for available data) was entirely deterministic. For example, certain time series oscillated between rising and falling. During some initial development and debugging of the algorithms, I used a deterministic transition model. This is the ultimate in expectations for that component. In this case, I found the results of the algorithm to be incredibly insensitive to variations in waiting-time distributions and shape recognizer noise variances. It seemed like the algorithm locked onto the perfect segmentation immediately for practically any reasonable model I gave to it (as long as I did not give it extreme (incorrect) models). I added extreme amounts of noise the data, used very diffuse waiting-time distributions, and still got good results, even with some of the weaker variants of the algorithm. In this case, the model seems to be highly robust, but at the same time, the segmentation task seems to be immensely easier. In fact, I found I could not use these examples for experimentation because it was very difficult to separate out the good techniques from the poor ones.

This example may generalize. It may very well be that when very extreme expectations in one component are warranted by an application, it may be that the sensitivity to precise values in other components is reduced. This is only a conjecture, but it does make some sense. For example, if durations were deterministic, it would only be a matter of comparing shapes and noise levels between known points in time to transition probabilities. In this case, the balance between transition probabilities and shape-recognizer noise levels is probably much less critical.

7.2 Limitations of the HSSMM

The HSSMM places little restriction on the form of shape recognizers or on the form of waiting-time distributions. As a result, the basic framework is quite general in many respects. However, there are several types of transition processes that cannot be faithfully modeled by a pure HSSMM. These typically arise because the transition process of interest is not semi-Markov. This section discusses some of the important limitations that arise in actual data that I have seen and introduces the Hidden Segmented *Generalized* Semi-Markov Model (HSGSMM), a generalization of the HSSMM, to describe these transition processes. This thesis does not explore the HSGSMM in any depth — it is discussed here solely as a lead to possible future research.

There are some obvious limitations of the HSSMM that I do not focus on here. For example, the HSSMM is clearly not appropriate as a model for slowly and continuously changing nonstationarity processes. Instead, the HSSMM is appropriate for systems with distinct transition points with a small number of distinct operating modes. Conceivably the number of reasonable operating modes could be extended by harnessing structure within the state space using the structural decomposition techniques of Chapter 3. Similar methods have been applied to Temporal Bayesian Networks (e.g., [Agogino and Ramamurthi, 1990, Kanazawa and Dean, 1989, Berzuini *et al.*, 1989, Kjaerulff, 1995a]), but this direction is not considered

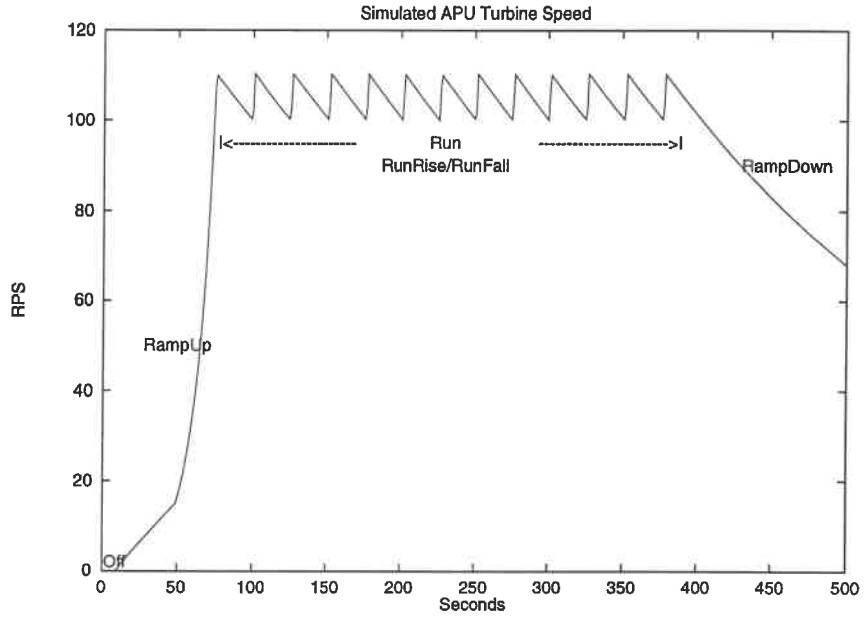


Figure 7.8: A hierarchical process.

further here.

I concentrate here on processes that are not semi-Markov. Recall that a semi-Markov process has the property that at the instant immediately following a transition, the current state always summarizes everything there is to know about the future evolution of the system. It is *semi*-Markov and not *Markov*, because this property need not hold at every point in time. Three important cases of non-semi-Markov processes occur commonly in practice.

Hierarchical Signal Shapes

Figure 7.8 shows the turbine speed as a function of time during a simulation of an auxiliary power unit based on an on-line description of the Space Shuttle's auxiliary power unit [NSTS, 1988]. A turbine generates the power output from the unit. When the unit is turned on, the turbine speed ramps up to a running mode, at which point a bang-bang controller keeps the rotational speed between two thresholds, resulting in a saw-tooth time-series. Finally, the unit is turned off and the speed ramps down.

At a coarse level of granularity, there are four major modes in Figure 7.8: OFF, RAMPUP, RUN, and RAMPDOWN. At this level, the time series appears semi-Markov.

At a finer level of granularity, the RUN mode can be further described as alternating between two modes: RUNFALL and RUNRISE. The alternation between these two sub-modes also appears semi-Markov, but the whole process, now consisting of the five modes OFF, RAMPUP, RUNFALL, RUNRISE, RAMPDOWN, is *not* semi-Markov. Consider, for example, the probability of transitioning next to the RAMPDOWN mode immediately following a transition to RUNRISE. Or consider the length of time until the *RampDown* is reached at this moment. Because the APU is typically run during the duration of takeoff, the total duration of the RUN mode is fairly regular, say (hypothetically) close to 20 minutes, so that the amount of time that the system has been only in the RUNFALL and RUNRISE modes is also relevant to future evolution of the system. When the system has been in the RUN mode for 19.9 minutes, the probability is high that a transition to RAMPDOWN is imminent, while after RUNning only 1 minute, the probably of a transition to *RampDown* anytime soon is low. Clearly this process is not semi-Markov.

The HSSMM models semi-Markov processes. As such it cannot faithfully model this situation (at least not as a five state process).

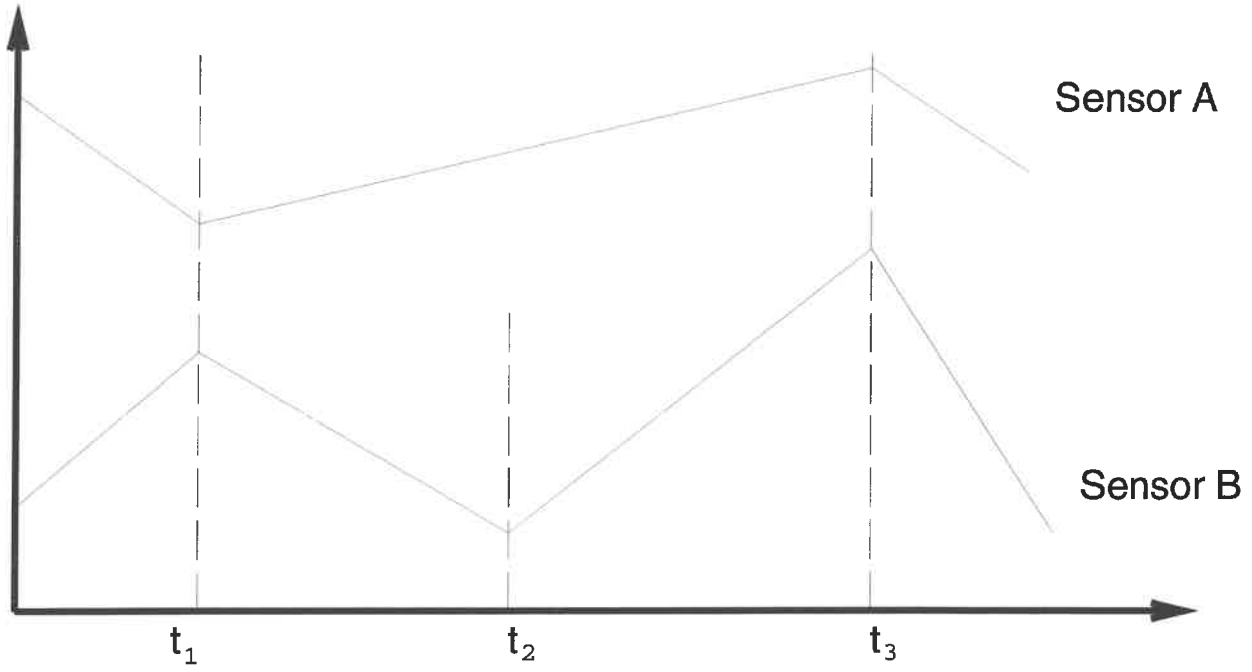


Figure 7.9: An event causes a transition in Sensor B at t_2 but does not effect Sensor A.

Transitions on a subset of sensors

A second limitation of the HSSMM arises from its simplistic treatment of multiple sensors. In Figure 7.9, transition t_2 affects Sensor B but does not influence Sensor A. Although Sensor A is linear on both sides of t_2, the fact that there is no bend in Sensor A at t_2 provides important information, which if modeled correctly, could aid in the inference process.

The HSSMM as defined in Chapter 2 assigns the same set of transitions to all sensors, thus it cannot model this situation properly.

The underlying transition process in Figure 7.9 could be semi-Markov or non-semi-Markov (such as with a hierarchical model when $t_3 - t_1$ is chosen according to a waiting-time distribution, and $t_2 - t_1$ is chosen separately). Even if the underlying transition process is semi-Markov, the signals shown in Figure 7.9 are not semi-Markov since immediately following the transition at t_2, knowing the state does not summarize all there is to know about the future — data for Sensor A between t_1 and t_2 adds additional information about the shape of Sensor A from t_2 to t_3. So this is again another example of a non-semi-Markov process.

This situation arises when certain events affect one sensor without affecting the other. Note that in Figure 7.9, certain events (at t_1 and t_3) affect both sensors. If this did not occur, there would be no problem since each sensor could simply be modeled and processed individually.

Interacting Processes

Figure 7.10 shows a certain fuel line pressure reading during a flight of the space shuttle. Once again, the time series shown in *not* semi-Markov. From Figure 7.10 alone, this may at first be nonobvious, but it becomes immediately evident after viewing the two related fuel line temperatures during the same time span shown in Figure 7.11. The two sensors in Figure 7.11 rise and fall independently, each in a semi-Markov fashion. The actual pressure plotted in Figure 7.10 occurs downstream on the fuel line from the temperatures in Figure 7.11, and is essentially a result of the average of the two (recall that pressure and temperature are proportionally related by Gay-Lussac's law or the more general $PV = nRT$).

There are basically four states in this system: $\{X_1 Rise, X_1 Fall\} \times \{X_2 Rise, X_2 Fall\}$. A transition in either individual process causes a transition in the joint process, but at the moment following a transition

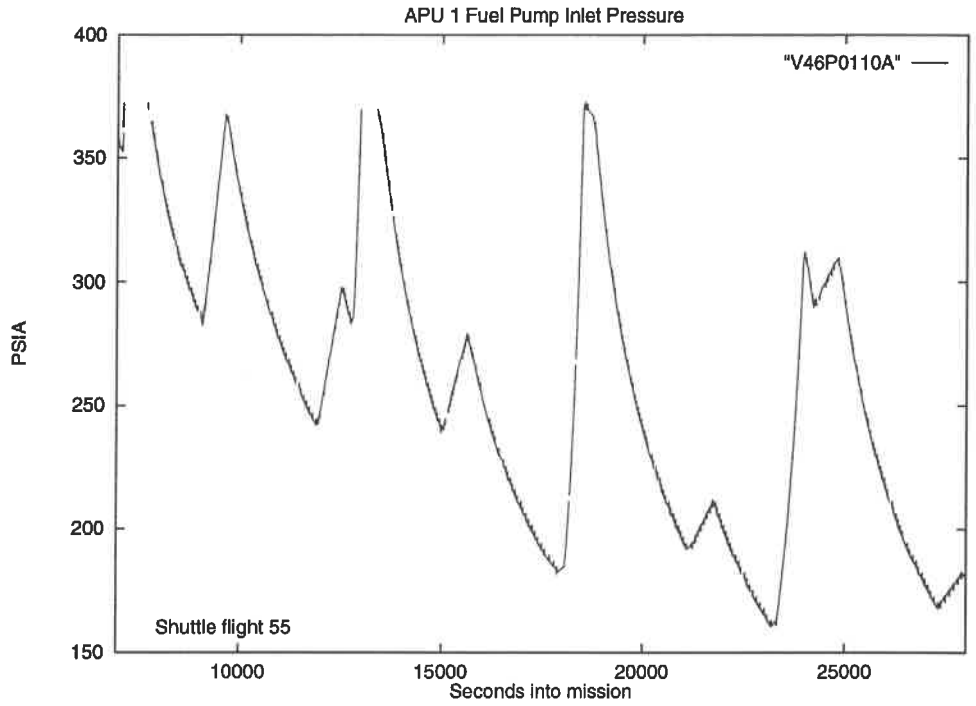


Figure 7.10: A non-semi-Markov process resulting from two interacting processes. (Space Shuttle Data).

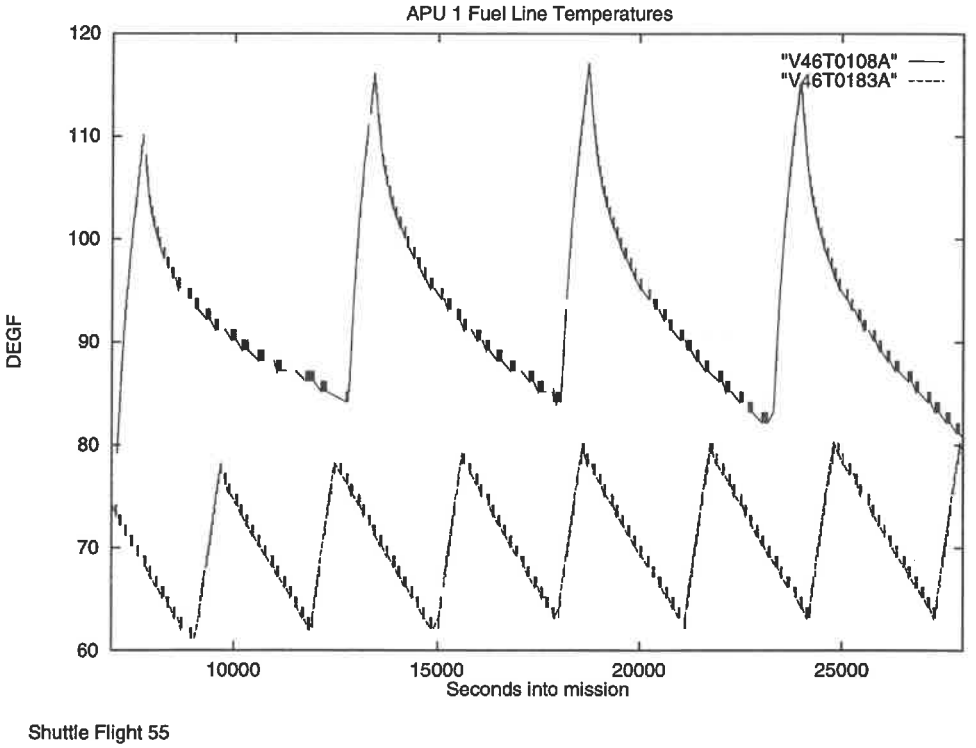


Figure 7.11: Two independent (semi-Markov) processes revealed by two temperature sensors. Each transition in Figure 7.10 occurs at the same time as a transition of one of these sensors.

in one process, the state does not capture duration information about the state of the other process. The joint process is therefore not semi-Markov.

In a big system like the Space Shuttle, there are often multiple independent (semi-Markov) processes operating simultaneously. When a single sensor is influenced by two or more of these processes, the signal from that sensor is, in general, not semi-Markov. Informally it can be said that the cross-product of two semi-Markov processes is not semi-Markov (except in a few degenerate cases). In contrast, the cross product of two Markov process *is* Markov.

Even though Figure 7.10 is not semi-Markov, it may still be quite reasonable to approximate it using a semi-Markov process (although the observation model would still requires a slight generalization). This is simply an instance of using a more diffuse model to move along the model-driven to data-driven continuum (Section 7.1). A thorough understanding of exactly how much is lost by such an approximation is an interesting topic but beyond the scope of this thesis. Regardless, it should still be realized that modeling signals more faithfully when the knowledge and justification to do so exists is not even an option in this case using a pure HSSMM.

7.2.1 The Generalized Semi-Markov Model

All of the above examples can be faithfully modeled using a Hidden Segmented *Generalized* Semi-Markov Model (HSGSMM). The semi-Markov transition model of the HSSMM is simply replaced by a *generalized semi-Markov transition model* ([Haas and Shedler, 1986, Haas and Shedler, 1996, Eakinos, 1991, Glasserman and Yao, 1992, Glasserman and Yao, 1996, Damerdji, 1996]). The observation model is adapted accordingly with a slight additional generalization.

Like a semi-Markov model, a generalized semi-Markov model operates on a finite state space with probabilistic transitions; however, the GSMM introduces the notion of multiple *clocks*. Clocks are a mathematical abstraction for modeling durations of various events within a process. In a 2-clock GSMM, both clocks are initially set to some positive value, and then each each begins decrementing towards zero. When one of the clocks reaches zero, a transition occurs. The transition probabilities depend on which clock reaches zero first, with the next state chosen according to these probabilities. A new value for the expired clock is chosen from a waiting-time distribution for that clock at the new state. Also, attached to the transition is an action to be applied to the unexpired clock. Standard actions are to (a) continue decrementing the clock from its current value, (b) reset it according to its waiting-time distribution at the new state, or (c) pause (disable) the clock. From this point, the whole cycle repeats — each (unpaused) clock decrementing until a transition is again triggered by one of them reaching zero, and so on.

The observation model is also generalized to take advantage of the greater generality in the transition model. Recall from Figures 7.9 and 7.11 that some sensors are not affected by certain transitions. Thus, the HSGSMM contains additional knowledge specifying which transitions affect which sensors. This can be accommodated by specifying for each sensor a partition of the state space. Only transitions across partition boundaries form transition boundaries for the given sensor. Note that the generalization to the observation model could equally well be applied to a pure HSSMM with a slight increase of representational power (for example, instances similar to that in Figure 7.9 might be handled with a standard HSSMM).

Putting all these together, an HSGSMM is described by the following components:

$S = \{s_1, .., s_n\}$	: The states.
$\Gamma = \{\gamma_1, .., \gamma_\ell\}$	: The clocks.
$a^{\gamma}_{s_i,s_j}$	: Transition Probabilities.
b_{s_0}	: Initial Occupancy Distribution.
$c^{\gamma}_{s_i}(\Delta t)$	: Waiting-time distribution for clock γ at state s_i.
$cl\_act(s_i, s_j, \gamma^*, \gamma) \in \{Continue, Reset, Pause\}$	: Action for clock γ when clock γ^* expires and causes a transition from s_i to s_j. By default, $cl\_act(\cdot, \cdot, \gamma^*, \gamma^*) = Reset$.
$\hat{S}_v$	: Partition of S for sensor v.
$sh(\hat{s}_i, v)$	: Shape of sensor v in $\hat{s}_i \in \hat{S}_v$.
$d_q(X^v, t_1, t_2)$	: Shape recognizers.

There is also an issue of how the clocks should be set initially. To avoid additional complexity, we can simply assume here that all clocks are initially set according to the waiting-time distribution in the initial

state. Another option would be to add initial clock distribution components to the HSGSMM definition.

7.2.2 Examples

All of the previous non-semi-Markov examples can be easily expressed as HSGSMMs. Each of the examples involves only two clocks, but in all cases the examples can be extended to where more than two clocks are necessary.

HSGSMM for Hierarchical Signal Shapes

In this example, Clock 1 controls transitions between OFF, RAMPUP, RUN, and RAMPDOWN, while Clock 2 controls transitions between RUNRISE and RUNFALL within the RUN mode. Below are the important components for this example.

$$
\begin{aligned}
a^{\gamma_2}_{\text{RUNRISE},\text{RUNFALL}} &= a^{\gamma_2}_{\text{RUNFALL},\text{RUNRISE}} = 1 \\
b_{\text{OFF}} &= 1 \\
a^{\gamma_1}_{\text{OFF},\text{RAMPUP}} &= a^{\gamma_1}_{\text{RAMPUP},\text{RUNFALL}} = a^{\gamma_1}_{\text{RUNRISE},\text{RAMPDOWN}} \\
&= a^{\gamma_1}_{\text{RUNFALL},\text{RAMPDOWN}} = a^{\gamma_1}_{\text{RAMPDOWN},\text{OFF}} = 1 \\
c^{\gamma_1}_{\text{RUNFALL}} &= \text{Distribution with very large mean} \\
c^{\gamma_1}_{\text{RAMPUP}} &= c^{\gamma_1}_{\text{RAMPDOWN}} = \text{Distribution(s) with medium-sized mean.} \\
c^{\gamma_2}_{\text{RUNRISE}} &= c^{\gamma_2}_{\text{RUNFALL}} = \text{Distribution(s) with small mean.} \\
cl\_act(\cdot, \text{OFF}, \cdot, \gamma_2) &= cl\_act(\cdot, \text{RAMPUP}, \cdot, \gamma_2) = cl\_act(\cdot, \text{RAMPDOWN}, \cdot, \gamma_2) = \mathit{Pause} \\
cl\_act(\cdot, \text{RUNFALL}, \gamma_1, \gamma_2) &= cl\_act(\cdot, \text{RUNRISE}, \gamma_1, \gamma_2) = \mathit{Continue} \\
cl\_act(\cdot, \cdot, \cdot, \gamma_1) &= \mathit{Continue}
\end{aligned}
$$

Because there is only one sensor, $\hat{S}_v = S$, and the shape recognizers correspond to the shapes seen in Figure 7.8.

The key for this model is that γ_1 continues counting down the entire time the process is in the RUNRISE and RUNFALL states, until eventually it expires and causes the transition to RAMPDOWN, at which time γ_2 is turned off (PAUSEd).

HSGSMM for Partially-Independent Sensors

The key to handling the situation depicted in Figure 7.9 lies with the modifications to the observation model. $\hat{S}_A$, the partition for Sensor A, is set to a strict partition of S, so that the transition at t_2 in Figure 7.9 is not a transition in $\hat{S}_A$. $\hat{S}_B$ may be S. With this, the transition at t_2 does not influence Sensor A.

HSGSMM for Interacting Processes

Although the most complicated of the examples, it may be the easiest example to map onto the HSGSMM framework. The state space is the cross product of the state spaces for the individual processes, and a separate clock is simply assigned for each process using the waiting times from the original processes. $cl\_act(\cdot, \cdot, \gamma^*, \gamma) = \text{CONTINUE}$ for all $\gamma^* \neq \gamma$. The sensors in Figure 7.11, if included in the model, would be assigned orthogonal partitions of $\hat{S}$, while the sensor in Figure 7.10 would use $\hat{S}_v = S$, i.e., responding to all transitions.

This example makes it clear that the cross product of ℓ HSSMM processes can in general be modeled by an ℓ-clock HSGSMM. Similarly, the cross product of an ℓ_1-clock HSGSMM with an ℓ_2-clock HSGSMM is at most an $(\ell_1 + \ell_2)$-clock HSGSMM.

7.3 Model Learning Issues

Designing good models for any substantial application domain is almost always a very tedious and time-consuming task. In addition, it can require substantial domain expertise to do well. For these reasons, in many situations it may be preferable to automatically learn HSSMM models from time-series data.

The most common method for learning Hidden Markov Models (HMMs) and Hidden Semi-Markov Models (HSMMs) is the EM-algorithm [Dempster *et al.*, 1977]. A good review of the method applied to HMMs is [Rabiner, 1989], for a review on HSMMs see [Guedon, 1992] (also for HSMMs, [Cook and Russell, 1986, Huang and Jack, 1989, Guedon and Cocozza-Thivent, 1990, Guedon, 1992]), and for HGSMMs see [Damerdji, 1996]. The EM-algorithm is a general technique for learning probabilistic models from data, and is fundamentally a search algorithm for finding a maximal likelihood model given the data[2]. The EM-algorithm is particularly well-suited for model refinement, where it can be used to improve an existing reasonably-good model.

The EM-algorithm involves to basic alternating steps: Estimation and Maximization[3]. The algorithm applies to models where certain quantities are not directly observable (in the case of the HSSMM, these are t_i and s_i, $i = 1, .., k$). During the estimation step, marginal probabilities for the unobservables given the data are computed based on the current model. These form estimations of the full state. During the maximization step, the maximum-likelihood model for the estimated full state is identified, and this new model replaces the previous model. The cycle then iterates. If both steps are done exactly, each iteration is guaranteed to improve the likelihood of the model given the data. The EM-algorithm is generally known to have very fast convergence to a locally optimum model.

First, it can be noted that all the machinery necessary to carry out the estimation step has been developed in this thesis. To perform the maximization step, it is necessary to adopt various restrictions on the HSSMM so that the necessary parameters can be identified. First, one must adopt a restricted parametric class for waiting-time distributions. For example, a convenient class is the class of gamma distributions. Having done this, the two parameters that define a gamma distribution, α, β, can be fit to the estimations (e.g., [Guedon and Cocozza-Thivent, 1990]). The other restriction that is necessary is on the form of shape recognizers. The most obvious way to enable the EM-learning approach is to specify a library of shape recognizers in advance. These may be parameterized as long as the parameters can be easily estimated given (weighted) examples of signals of that shape. With these restrictions, the EM-algorithm can be directly applied. All the apparatus for doing this has already been developed.

I have not attempted to automatically learn HSSMM models from time-series data. This is something that could be considered in future work. However, despite the fact that there is appeal in doing so, there are also situations where learning an HSSMM from data is not a wise approach. This research was initially motivated by a Space Shuttle monitoring application, and monitoring is a prime example where such a learning approach may not be appropriate. The problem is that in a monitoring application, available training data is usually not representative of the situations the model must handle in the field [Smyth, 1994]. Available Shuttle data from past missions is predominantly representative of *normal* behavior. If this were used to train a model, the resulting expectations would be very strong that behavior is normal, making it highly effective at tracking normal signals, but much less effective at handling novel anomalous cases. On the other hand, a human designed model can incorporate (heuristic) knowledge about where strong expectations are appropriate and where expectations should be weakened.

In summary, the machinery for a HSSMM (maximum likelihood) learning algorithm is in place, but this has not been attempted as part of the work for this thesis. Learning the HSSMM from available data appears largely inappropriate for the application that initially motivated this research, and these considerations may be something to consider for other potential applications as well.

---

[2] Although the EM-algorithm only guarantees a locally optimum solution, not the global optimum.

[3] The literature is inconsistent on exactly what EM stands for, with some authors calling it Expectation-Modification, or Expectation-Maximization, etc.

Chapter 8

Conclusion

Iterative dynamic discretization is a method for solving graphical probabilistic models by constructing a discretization of the original model, solving it exactly, and iterating to successively improve the solution. The central thesis of this research is that the information learned from solving a discrete model can be used effectively for guiding the selection of a new discretization. In so doing, the new discretization is more informed, thus forming the basis for an iterative algorithm, and providing a means for dynamically tailoring the discretization to the specific problem instance.

This topic has been explored using, as a motivating application, the time-series segmentation task. A basic framework was developed (Chapter 4) for utilizing the information obtained from solving a discrete model to select a new discretization. Evidence that this framework provides an effective means for selecting a new discretization, and for eventually converging to a good solution, has appeared in at least two forms. First, theoretical analyses (Chapter 5) have shown that with mild positive assumptions on the distribution of interest, the iterative algorithm is guaranteed to find a solution arbitrarily close to the best solution if run long enough. And second, actual runs of the algorithm (Chapter 6) show rapid convergence as a function of the number of iterations, indicating that the selection of new discretizations is being accomplished in an effective manner.

In pursuit of this thesis, this research has also developed a number technologies. In particular, this thesis has:

- Defined an expressive time-series modeling formalism, the HSSMM. Section 2.2
- Specified a (difficult) model-based time-series segmentation task as an optimization task. Section 2.3
- Used the theory and methods of graphical probabilistic models to decompose the original (enormous) optimization problem into a collection of much smaller optimization tasks. Section 3.1
- Developed and applied iterative dynamic discretization to handle real-valued variables and search the space of possible discretizations.

The time-series segmentation task has provided a challenging task for exploring computational techniques and issues surrounding the approximation of graphical probabilistic models. The centerpiece, iterative dynamic discretization, puts together many component technologies in a useful, interesting, and powerful fashion. The technique is applicable far beyond just time-series segmentation tasks, and can be applied to a wide class of graphical probabilistic inference problems.

8.1 Perspectives on this work

As the discussion in the introduction and throughout the thesis emphasizes, iterative dynamic discretization can be viewed from a number of different perspectives:

1. As a method for handling real-valued variables in Bayesian networks via discretization.

2. As a way to combine Monte Carlo Markov chain methods with exact propagation to leverage the individual strengths of each.

   - For MCMC, a method to speed up convergence by considering multiple configurations at each iteration and harnessing the ability to analyze this collection with exact methods.
   - For MCMC, flexibility to introduce domain-specific channels of mobility (e.g., keeping neighbor's best, Section 6.7).
   - For exact methods, a method dealing with nonconjugate distributions.
   - For exact methods, a method for dealing with very large (or infinite) sample spaces of an individual variable.

3. As an instance of iterative approximate Knowledge-Based Model Construction (KBMC). (Section 1.8.3) The central thesis of this work could be stated as: Information learned from solving a (simplified) constructed model can be used in an effective manner to construct a new simplified model. Discretization is simply one special case of model construction, and by supporting the thesis for this special case, the credibility of the endeavor for iterative KBMC in general has been increased, and perhaps some lessons and experience gained here may transfer to other forms of model construction.

8.2 Contributions & Results

Specific contributions of this thesis include:

1. Introduction of the Hidden Segmented Semi-Markov Model (HSSMM) as a useful language for expressing knowledge about time-series (Section 2.2). The novel part of this model is the segmented observation process. Related, but undeveloped by this thesis, is the introduction of the even more expressive Hidden Segmented Generalized Semi-Markov Model (HSGSMM), Section 7.2.1. Again, the segmented observation process is the novel contribution.

2. An approach to HSSMM-based segmentation, based on structural decomposition plus iterative dynamic discretization.

3. A general methodology of structural decomposition (Section 3.2).

   (a) With an emphasis on using graphical probabilistic models as a computational tool for solving problems arising in other formalisms, and for a tool that is useful in designing formalisms for specific applications.

   (b) A general axiomatization characterizing when structural decomposition can be applied. This includes two axiomatizations (the second being due to [Shenoy and Shafer, 1990]), elaborating the two different but closely related methods for propagation.

4. Focused Gibbs sampling. Although an instance of pure Gibbs sampling, with graphical probabilistic models focused Gibbs sampling provides yet another means for combining exact propagation with Monte Carlo methods.

5. A basic framework for studying iterative dynamic discretization and its possible variants (Section 4.2). This framework serves to relate possible approaches in a meaningful way.

6. Methods for estimating sampling distributions for iterative dynamic discretization, including an algorithm for sampling from f_{mbp} (Section 4.4.2).

7. A formal proof that iterative dynamic discretization is recurrent and ergodic (Chapter 5). Some characterization (both positive and negative) about the asymptotic behavior of iterative dynamic discretization (Section 5.3).

8. Experimental demonstration that iterative dynamic discretization can significantly speed up convergence (on a per-iteration basis) of MCMC methods (Chapter 6). Several of the most powerful variations of iterative dynamic discretization reduced the number of iterations required for convergence by at least two orders of magnitude as compared to (focused) Gibbs sampling.

9. A study of where the power of iterative dynamic discretization comes from. Two factors are especially important:

   (a) In a single iteration, a large number of configurations are analyzed with exact methods.

   (b) The added flexibility makes it possible to introduce domain-dependent "channels of mobility".

10. Section 7.1 provided some experimental results concerning the robustness of the HSSMM formalism. Experiments examined the impact of ignorance in waiting-time distributions and transition probabilities, and the impact of noise in the data. (This study was, however, very brief)

I have attempted to write this thesis in a manner that leaves the reader with a rich collection of tools and methods, and in which all algorithms presented are flexible and easy to modify or customize. I hope this allows the reader to come away with something more useful than just the precise algorithms used by and studied in this thesis. All the algorithms have been couched in terms of general methods so that they can easily be altered or changed to fit variations in assumptions or entirely different unforeseen applications.

8.3 The Generality of Iterative Dynamic Discretization

Iterative dynamic discretization is a general method for solving arbitrary graphical probabilistic networks with or without real-valued variables. Because it was applied in a very specific fashion in this thesis, this generality may not be entirely apparent, and so an elaboration on this point here is worthwhile.

The *general* procedure for applying iterative dynamic discretization to an arbitrary network is as follows:

1. Triangulate the network and extract a junction tree.

2. Identify the *clusters* of variables, such that each variable in a cluster appears in exactly the same nodes of the junction tree.

3. Treat each cluster as if it were a variable with a very large sample space. At each iteration, iterative dynamic discretization selects m joint values for the variables with a single cluster.

4. Once each cluster is assigned m distinct values, the junction tree is discrete and easily propagated to analyze the full discretization.

5. Iterate. The information obtained from analyzing the current discretization guides the next choice of discretization.

The identification of clusters bears resemblance to issues of blocking in Blocking Gibbs sampling (Sections 3.4.2 and 1.8.4). The method of selecting m discrete values for each cluster is similar to the method used to handle Gibbs frames in hybrid propagation [Dawid *et al.*, 1994, Kjaerulff, 1995b] (Section 1.8.4). However, hybrid propagation methods have not considered an iterative framework, and the precise use is somewhat different.

This thesis has, admittedly, not explored the generality of iterative dynamic discretization on arbitrary graphical structures. Its effectiveness when used in this fashion (as opposed to its use simply to discretize real-valued variables) is open for investigation. Note also that hybrid propagation has not, to date, been satisfactorily investigated empirically either, so the effectiveness of methods of this type in general graphical settings is yet to be seen.

8.4 Open Problems

For Iterative Dynamic Discretization

Perhaps the most immediate open problem specific to iterative dynamic discretization is to characterize the properties of its asymptotic behavior (see Section 5.3). In its current form, the algorithm *cannot* be applied directly to the problem of approximating marginal posterior distributions, or even to approximating expectations, since there is currently no guarantee in general that the uniform projection matches the distribution of interest in any meaningful way. This did not get in the way of applying it to the maximum a posteriori inference problem defined by the time-series segmentation application, but it is critical for more general uses of the technique.

It is possible that iterative dynamic discretization using uniform weighting results in a ergodic distribution whose posterior-weighted projection is the distribution of interest. Perhaps there is some other variation of iterative dynamic discretization and/or projection that has this or some other useful and easily characterized property. However, these characterizations remain to be proven.

It is also possible that some new variation on iterative dynamic discretization may have asymptotic properties that can be usefully characterized. One promising approach is to wrap a Metropolis-Hastings acceptance loop around the discretization step (discussed at the end of Section 5.3). There is reason to believe that, with the appropriate acceptance rule, this may result in a uniform projection equal to the distribution of interest, or agreeing on marginals with the distribution of interest.

Not only would results of this form expand the applicability of the method, they would also greatly clarify the basic method by formally identifying exactly what the algorithm is achieving as a MCMC method.

Regarding empirical evaluation of iterative dynamic discretization, it is fair to say that results in this thesis are barely the tip of the iceberg, and much further study in a wide variety of settings is certainly warranted.

For Time-Series Modeling

The HSSMM is a rich formalism for expressing domain knowledge. The formalism is general enough to handle virtually any arbitrary shape recognizer, yet during this research I did not have the time and resources to investigate this space much. There may be much to learn from exploring the use other shape recognizers.

I encountered several problems when using the HSSMM formalism. Many of these were solved during the course of the research, but some remain. I have noticed very poor/unintuitive results when waiting-time distributions within a model have variances that differ by more than an order of magnitude. Designing a HSSMM time-series model is not always trivial, and tools to help with this task (including learning algorithms) could help to expand the usefulness of the formalism.

Like Two-Window segmentation methods (Section 2.4.1), the HSSMM evaluates the plausibility of a proposed transition time based on the fit of the data on either side of the proposed time. In contrast, transition-recognition methods (Section 2.4.2) base this evaluation on features in the data, such as whether the point is an inflection point. I believe that transition-recognition methods are better for finding transition points that agree with human intuition. There is the possibility of introducing a transition recognition component into the HSSMM formalism as basically another term in the evaluation formula (2.3). This would probably add much flexibility for an engineer who wishes to design his model to yield optimum segmentations corresponding to his intuitions.

For Further Utilizing Time-Series Structure

Generalized Semi-Markov Models (Section 7.2.1) greatly extend the expressiveness of the HSSMM in useful ways. These extensions become even more important in very complex and high-dimensional systems where many separate but interacting processes are involved. Currently, little of the structure between different duration processes can be utilized. Even though two processes may operate independently (but influence a common sensor), it is not entirely clear how to harness this independence in the most effective fashion.

Related is the problem of utilizing structure within the state space of the HSSMM. This is a problem of interest also for temporal Bayesian networks and temporal influence diagrams, and progress in that area is relevant here.

These two areas, taken together, are perhaps the biggest determiner of scalability. The ability to handle large models of complex systems, for example, of the space shuttle and its thousands of sensors, hinges largely on good solutions to these problems. These are the types of problems that are unlikely to see a clean-cut solution, but are more likely to see improvements in technology over time.

Knowledge-Based Model Construction

Most existing works on KBMC consider knowledge-bases containing quantifiers. Although the time dimension in the HSSMM is in some sense quantified, this is at most a very specific case. Exploring the endeavor of iterative approximate KBMC in more classic knowledge bases is still of great interest.

8.5 Closing

Graphical probabilistic models provide a useful computational tool for obtaining algorithms for existing problems and for guiding the development of formalizations for specific tasks. Both exact and approximate solution techniques play important roles for these models, with complementary properties. Much of the future advancement in computational methods for solving these models will likely involve methods for utilizing combinations of exact and approximate methods. Iterative dynamic discretization provides one such data point for this enterprise.

Appendix A

A Model Used for Experiments

Model Specification

```
#State Definitions:

NumStates = 4
StateName[0] = "shortRise"
StateName[1] = "shortFall"
StateName[2] = "longRise"
StateName[3] = "longFall"

# Transition Definitions:

trans from state 0  {
  to state 0 with prob  0.1
  to state 1 with prob  0.4
  to state 2 with prob  0.1
  to state 3 with prob  0.4
}

trans from state 1  {
  to state 0 with prob  0.4
  to state 1 with prob  0.1
  to state 2 with prob  0.4
  to state 3 with prob  0.1
}

trans from state 2  {
  to state 0 with prob  0.1
  to state 1 with prob  0.4
  to state 2 with prob  0.1
  to state 3 with prob  0.4
}

trans from state 3  {
  to state 0 with prob  0.4
  to state 1 with prob  0.1
  to state 2 with prob  0.4
  to state 3 with prob  0.1
}
```

```
initially {
  state 0 with prob 0.25
  state 1 with prob 0.25
  state 2 with prob 0.25
  state 3 with prob 0.25
}

# Clock Time Distributions:

when clock 0 expires {
  at state 0 use gammadist(3,70)  # Fast rise Time
  at state 1 use gammadist(3,70)  # Fast fall Time
  at state 2 use gammadist(3,200) # Slow rise Time
  at state 3 use gammadist(3,200) # Slow fall Time
}

# Sensors and Shapes:

NumSensors = 1
Name[Sensor 0] = "SENSOR"

Description[Sensor 0] = "Simulated Sensor"

sensor 0 {
  has qualitative shape LINEAR_RISE(2.0) from state 0
  has qualitative shape LINEAR_FALL(2.0) from state 1
  has qualitative shape LINEAR_RISE(2.0) from state 2
  has qualitative shape LINEAR_FALL(2.0) from state 3
}
```

Transition Process

The model above is interpreted as follows.

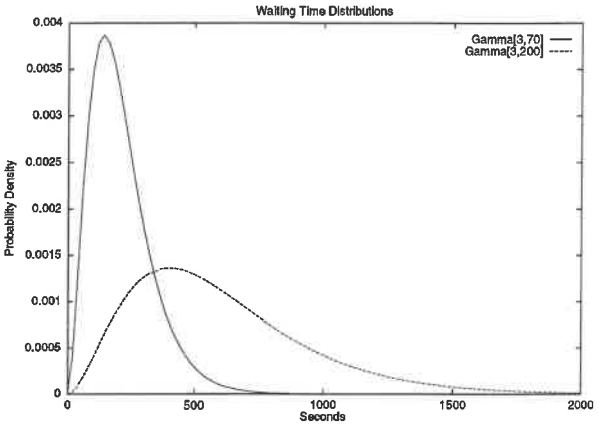
The model has four states: $\{shortRise, shortFall, longRise, longFall\}$.

Transition probabilities are given by the following matrix. Basically, there is a 0.8 chance of transitioning from a rising (falling) signal to a falling (rising) signal, and a 0.2 chance of transitioning from a rising (falling) signal to another rising (falling) signal (in which case the slope may change, but the sign of the slope does not change).

$$\left[a_{s_i,s_j}\right]_{i,j} = \begin{bmatrix} 0.1 & 0.4 & 0.1 & 0.4 \\ 0.4 & 0.1 & 0.4 & 0.1 \\ 0.1 & 0.4 & 0.1 & 0.4 \\ 0.4 & 0.1 & 0.4 & 0.1 \end{bmatrix}$$

Waiting time distributions are simply:

$$c_{shortRise}(\Delta t) = GammaDist[\alpha = 3, \beta = 70](\Delta t)$$
$$c_{shortFall}(\Delta t) = GammaDist[\alpha = 3, \beta = 70](\Delta t)$$
$$c_{longRise}(\Delta t) = GammaDist[\alpha = 3, \beta = 200](\Delta t)$$
$$c_{longFall}(\Delta t) = GammaDist[\alpha = 3, \beta = 200](\Delta t)$$



Sensors:

A single sensor was used for these experiments. The data for that sensor was in a file named `"SENSOR"`. This data was generated synthetically for the experiments using the same model above. How simulation was performed is discussed in Section A.1.

Observation Model

Two shape recognizers are used: `LINEAR_RISE`(ν) and `LINEAR_FALL`(ν). The argument, $\nu > 0$, specifies how much noise is to be tolerated. If ν is close to zero, then only very good (linear) fits with little noise evaluate to something significantly greater than zero. If ν is large, then noise and poor fits are tolerated to a greater extent.

These shape recognizers first determine the degree of linearity by fitting a straight line to the data that falls in the specified interval. A small-sample correction (discussed on Page 33) is made to the residual, and then the probability of observing at most this residual under the assumption that the residual distributes as an exponential distribution with standard deviation ν is computed. In other words, this is the likelihood of a residual equal to or less than what was actually observed. This likelihood serves as the degree of linearity.

Second, a judgement is made is to whether the slope is positive or negative. Since noise in the data could cause a positive slope in the best fit line even when the underlying shape is `LINEAR_FALL`, a degree of linearity is computed. If a slope is close to zero and very few data points are used in the fit, then there may be significant uncertainty in the sign of the underlying slope. On the other hand, if the best fit line has a steep slope and there is plenty of data used in that fit, then the slope of the underlying signal is very certain. There may be a lot of data, but the vast majority of it might be clumped in a small time interval. In this case there can still be a lot of uncertainty in the slope despite there being a lot of data since the line could easily pivot with little change to the residual.

Standard techniques for finding the best fit line also yield the variance of the estimates for slope and intercept ([Press *et al.*, 1992, Page 663]). This variance is low if there are many well-spread out data points and small if there is little data or if all data is clumped into a small time interval. With a normality assumption, this can be used to compute the likelihood that the underlying slope is positive or negative. If the variance estimate is small and the slope is not near zero, then the likelihood of the underlying slope being different from the fitted slope is very small. If the variance is large or the fitted slope is close to zero, then the likelihood is much larger (although it can never be more than 0.5). This likelihood (for the appropriate sign of slope) serves as slope detector.

The shape recognizer's evaluation is then the product of the degree of linearity with the likelihood that the underlying slope is falling or rising. The algorithms used (Chapter 3 and 4) do not require the estimate to be normalized, thus making this evaluation feasible.

A.1 Simulation

To obtain synthetic data from a known model, it necessary to simulate a HSSMM; however, the HSSMM only models signal shapes *qualitatively*, so that the precise time series is underspecified by the model. To generate synthetic data, it is necessary to fill in (artificially) the missing specifications.

The model does specify states, transition probabilities, and waiting-time distributions precisely. Thus, a sequence of transitions and states can be stochastically generated according to the probabilities specified by the HSSMM. After such a simulation, one is left with a sequence of states, transition times, and a label of the shape of each sensor in each segment.

For the model in this appendix, the shape in each segment is either `LINEAR_RISE(0.2)` or `LINEAR_FALL(0.2)`. These are underspecified since they do not indicate anything about what magnitude slope the signal has or anything about the absolute values of the end points of each segment. The simulation used to generate the data for the experiments picked an initial value for $y_v[t_0]$ from $\mathcal{U}[0, 100]$. From there, each $y_v[t_i]$, where t_i is the time of the t^{th} transition, was set to $y_v[t_{i-1}] + u$, $u \sim \mathcal{U}[5, 20]$, for `LINEAR_RISE` or $y_v[t_{i-1}] - u$, $u \sim \mathcal{U}[5, 20]$, for `LINEAR_FALL`. Then y_v specified a noiseless piecewise-linear signal. After this, the actual data was generated at each discrete 1 second interval by adding a zero-mean independent error to the ideal signal (given by y_v). The error was exponentially distributed in both directions with standard deviation given by ν ($\nu = 2.0$ in this model).

Appendix B

Model used for Shuttle Data

Model Specification

```
#State Definitions:

NumStates = 3
StateName[0] = "Startup"
StateName[1] = "Rise"
StateName[2] = "Fall"

# Transition Definitions:

NumClocks = 1
ClockName[0] = "Clock_0"

trans from state 0 {
  on clock 0 to state 1 with prob 0.5
  on clock 0 to state 2 with prob 0.5
}

trans from state 1 on clock 0 to state 2 with prob 1
trans from state 2 on clock 0 to state 1 with prob 1

initially {
  state 0 with prob 1
  state 1 with prob 0
  state 2 with prob 0
}

# Clock Time Distributions:

when clock 0 expires {
  at state 0 use uniformdist(5000,10000)
  at state 1 use gammadist(15,70)  # Fast rise Time
  at state 2 use gammadist(30,150) # Slow fall Time
}

# Sensors and Shapes:
```

```
NumSensors = 2
Name[Sensor 0] = "V46T0108A"
Name[Sensor 1] = "V46T0104A"

Description[Sensor 0] = "APU 1 Fuel Line Temp 1"
Description[Sensor 0] = "APU 1 Fuel Line Temp 2"

sensor 0 {
  depends on clock 0
  has qualitative shape LINEAR_RISE(2.0) from state 0
  has qualitative shape LINEAR_RISE(2.0) from state 1
  has qualitative shape LINEAR_FALL(2.0) from state 2
}

sensor 1 {
  depends on clock 0
  has qualitative shape LINEAR_FALL(2.0) from state 0
  has qualitative shape LINEAR_RISE(2.0) from state 1
  has qualitative shape LINEAR_FALL(2.0) from state 2
}
```

Bibliography

[Abraham, 1980] Bovas Abraham. Intervention analysis and multiple time series. *Biometrika*, 67(1):73–78, 1980.

[Agogino and Ramamurthi, 1990] A.M. Agogino and K. Ramamurthi. Real time influence diagrams for monitoring and controlling mechanical systems. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Nets, and Decision Analysis*, chapter 9, pages 199–228. John Wiley & Sons, 1990.

[Amit and Grenander, 1991] Y. Amit and U. Grenander. Comparing sweep strategies for stochastic relaxation. *Journal Multivariate Analysis*, 37:197–222, 1991.

[Andersen *et al.*, 1989] Stig K. Andersen, Kristian G. Olesen, Finn V. Jensen, and Frank Jensen. HUGIN — a shell for building Bayesian belief universes for expert systems. In *Eleventh International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1080–1085, San Mateo, California, 1989. Morgan Kaufmann.

[Andersen *et al.*, 1993] Heidi H. Andersen, Malene HØjbjerre, Dorte SØrensen, and Poul S. Eriksen. Linear and graphical models for the multivariate complex normal distribution. Technical Report R-93-2010, Deptartment of Mathematics and Computer Science, Aalborg University, Aalborg, Denmark, 1993.

[Andrews, 1993] D. Andrews. Tests for parameter instability and structural change with unknown change points. *Econometrica*, 61:821–856, 1993.

[Appel and Brandt, 1983] Ulrich Appel and Achim V. Brandt. Adaptive sequential segmentation of piecewise stationary time series. *Information Sciences*, 29(1):27–56, 1983.

[Arnborg *et al.*, 1987] Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284, 1987.

[Askar and Derin, 1981] Murat Askar and Haluk Derin. A recursive algorithm for the Bayes solution of the smoothing problem. *IEEE Transactions on Automatic Control*, AC 26:558–561, 1981.

[Azevedo-Filho and Shachter, 1994] Adriano Azevedo-Filho and Ross D. Shachter. Laplace's method approximations for probabilistic inference in belief networks with continuous variables. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 28–36, Seattle, WA, 1994. Morgan Kauffman.

[Bacchus and Grove, 1995] Fahiem Bacchus and Adam Grove. Graphical models for preference and utility. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, August 1995. Morgan Kauffman.

[Bacchus *et al.*, 1994] Fahiem Bacchus, Adam J. Grove, Joseph Y. Halpern, and Daphne Koller. Forming beliefs about a changing world. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI)*, volume 1, pages 222–229. MIT Press, 1994.

[Bacchus, 1993] Fahiem Bacchus. Using first-order probability logic for the construction of Bayesian networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 219–226. Morgan Kaufmann, 1993.

[Barnard, 1959] G. A. Barnard. Control charts and stochastic processes. *Journal of the Royal Statistical Society, Series B*, 21:239–271, 1959.

[Basseville and Benveniste, 1983] Michèle Basseville and Albert Benveniste. Sequential segmentation of non-stationary digital signals using spectral analysis. *Information Sciences*, 29(1):57–73, 1983.

[Basseville, 1980a] Michèlle Basseville. Introduction. In M. Basseville and A. Benveniste, editors, *Detection of Abrupt Changes in Signals and Dynamical Systems*, pages 169–215. Springer-Verlag, Berlin, 1980.

[Basseville, 1980b] Michèlle Basseville. The two-models approach for the on-line detection of changes in AR processes. In M. Basseville and A. Benveniste, editors, *Detection of Abrupt Changes in Signals and Dynamical Systems*, pages 169–215. Springer-Verlag, Berlin, 1980.

[Bauer and Hackl, 1978] Peter Bauer and Peter Hackl. The use of MOSUMS for quality control. *Technometrics*, 20:431–436, 1978.

[Baxter and Rosenthal, 1995] J. R. Baxter and Jeffrey S. Rosenthal. Rates of convergence for everywhere-positive Markov chains. *Statistics and Probability Letters*, 22:333–338, 1995.

[Becker and Geiger, 1994] Ann Becker and Dan Geiger. Approximation algorithms for the loop cutset problem. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 60–68. Morgan Kauffman, 1994.

[Bellman and Roth, 1969] Richard Bellman and Robert Roth. Curve fitting by segmented straight lines. *American Statistical Association Journal*, 64:1079–1084, 1969.

[Berzuini *et al.*, 1989] C. Berzuini, R. Bellazzi, and S. Quaglini. Temporal reasoning with probabilities. In *Proceedings of the 1989 Workshop on Uncertainty in Artificial Intelligence*, pages 14–21. Association for Uncertainty in Artificial Intelligence, July 1989.

[Besag *et al.*, 1995] Julian Besag, Peter Green, David Higdon, and Karrie Mengersen. Bayesian computation and stochastic systems (with discussion). *Statistical Science*, 10(1):3–66, February 1995.

[Besag, 1974] J. Besag. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36:192–236, 1974.

[Bickel and Doksum, 1977] Peter J. Bickel and Kjell A. Doksum. *Mathematical Statistics: Basic Ideas and Selected Topics.* Prentice Hall, Englewood Cliffs, New Jersey, 1977.

[Bielza *et al.*, 1996] Concha Bielza, Peter Müller, and David Ríos Insua. Monte Carlo methods for decision analysis with applications to influence diagrams. Madrid Technical University, Spain. Available at ftp.isds.duke.edu/pub/WorkingPapers/96-07.ps., June 1996.

[Billingsley, 1986] Patrick Billingsley. *Probability and Measure.* John Wiley & Sons, New York, second edition edition, 1986.

[Bodenstein and Praetorius, 1977] Günter Bodenstein and H. Michael Praetorius. Feature extraction from the electroencephalogram by adaptive segmentation. *Proceedings of the IEEE*, 65(5):642–652, May 1977.

[Box and Jenkins, 1976] George E. P. Box and Gwilym M. Jenkins. *Time Series Analysis: Forecasting and Control.* Holden-Day, San Francisco, California, second edition edition, 1976.

[Box and Tiao, 1975] G. E. P. Box and G. C. Tiao. Intervention analysis with applications to economic and environmental problems. *Journal of the American Statistical Association*, 70(349):70–79, March 1975.

[Box and Tiao, 1976] G. E. P. Box and G. C. Tiao. Comparison of forecast and actuality. *Applied Statistics*, 25(3):195–200, 1976.

[Brailovsky and Kempner, 1992] Victor L. Brailovsky and Yulia Kempner. Application of piece-wise regression to detecting internal structure of signal pattern recognition. *Pattern Recognition*, 25(11):1361–1370, 1992.

[Brailovsky, 1992] Victor L. Brailovsky. On vector piece-wise regression and clustering. In *Proceedings of the 11th IAPR International Conference on Pattern Recognition. Conference C: Image, Speech, and Signal Analysis*, volume 3, pages 83–87, The Hague, Netherlands, September 1992. IEEE Computer Society Press.

[Brandt, 1982] Achim V. Brandt. An entropy distance measure for segmentation and clustering of time series with application to EEG signals. In *Proceedings of the Sixth International Conference on Pattern Recognition*, volume 2, pages 981–984, Munich, 1982.

[Breese and Fertig, 1991] John S. Breese and Kenneth W. Fertig. Decision making with interval influence diagrams. *Uncertainty in Artificial Intelligence*, 6:467–478, 1991.

[Breese *et al.*, 1994] John S. Breese, Robert P. Goldman, and Michael P. Wellman. Introduction to the special section of knowledge-based construction of probabilistic and decision models. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(11):1577–1579, 1994.

[Breese, 1992] John S. Breese. Construction of belief and decision networks. *Computational Intelligence*, 8(4):624–647, 1992.

[Brooks and Roberts, 1995] S. Brooks and G. O. Roberts. Diagnosing convergence of Markov chain Monte Carlo algorithms. Technical Report TR-95-12, University of Cambridge, 1995.

[Brown *et al.*, 1975] R. L. Brown, J. Durbin, and J.M. Evans. Techniques for testing the constancy of regression relationships over time. *Journal of the Royal Statistical Society, Series B*, 37:149–163, 1975.

[Buchanan and Shortliffe, 1984] Bruce G. Buchanan and Edward H. Shortliffe. *Rule-based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, Mass., 1984.

[Buntine, 1994] Wray Buntine. Operations for learning with graphical models. *Journal of Artificial Intelligence Research*, 2:159–225, 1994.

[Cano *et al.*, 1991] José Cano, Miguel Delgado, and Serafín Moral. Propagation of uncertainty in dependence graphs. In R. Kruse and P. Siegel, editors, *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Uncertainty (ECSQAU).*, pages 42–47, Marseille, France, October 1991. Springer-Verlag.

[Casella and Robert, 1996] George Casella and Christian P. Robert. Rao-Blackwellisation of sampling schemes. *Biometrika*, 83:81–94, 1996.

[Castillo *et al.*, 1995a] Enrique Castillo, José Manuel Gutierrez, and Ali S. Hadi. Parametric structure of probabilities in Bayesian networks. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU)*, pages 89–98, Berlin, Germany, 1995. Springer-Verlag.

[Castillo *et al.*, 1995b] Enrique F. Castillo, Remco R. Bouckaert, and José M. Sarabia. Estimation in approximate Bayesian belief network inference. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, August 1995. Morgan Kauffman.

[Chang and Fung, 1991] Kuo-Chu Chang and Robert Fung. Symbolic probabilistic inference with continuous variables. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventh Conference*, pages 77–85. Morgan Kaufmann, 1991.

[Charniak, 1991] Eugene Charniak. Bayesian networks without tears. *AI Magazine*, 12(4):50–63, Winter 1991.

[Chavez and Cooper, 1990] R. Martin Chavez and Gregory F. Cooper. A randomized approximation algorithm for probabilistic inference in Bayesian belief networks. *Networks*, 20:661–685, 1990.

[Cheeseman, 1985] Peter Cheeseman. In defense of probability. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence (IJCAI)*, volume 2, pages 1002–1009, 1985.

[Chrisman, 1995] Lonnie Chrisman. Incremental conditioning of lower and upper probabilities. *International Journal of Approximate Reasoning*, 13(1):1–25, 1995.

[Chrisman, 1996] Lonnie Chrisman. Propagation of 2-monotone lower probabilities (a.k.a. Choquet capacities) on an undirected graph. In *Proceedings of the 12th Conference on Uncertainty in Artificial Intelligence*, 1996.

[Chu, 1995] Chia-Shang James Chu. Time series segmentation: A sliding window approach. *Information Sciences*, 85(1–3):147–173, 1995.

[Coles *et al.*, 1975] L. Stephen Coles, Alan M. Robb, Paul L. Sinclair, Michael H. Smith, and Ralph R. Sobek. Decision analysis for an experimental robot with unreliable sensors. In *Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, pages 749–757, 1975.

[Cook and Russell, 1986] A. E. Cook and M. J. Russell. Improved duration modelling techniques in hidden Markov models using series-parallel configuration of states. *Proceedings of the Institute of Acoustics Autumn Conference on Speech and Hearing*, 8(7):299–306, 1986.

[Cooper, 1987] Gregory F. Cooper. Probabilistic inference using belief networks is NP-hard. Technical Report KSL-87-27, Medical Computer Science Group, Stanford University, 1987.

[Cooper, 1990a] Gregory F. Cooper. Bayesian belief-network inference using recursive decomposition. Technical Report KSL-90-05, Knowledge Systems Laboratory, Medical Computer Science, Stanford University, Stanford, California, February 1990.

[Cooper, 1990b] Gregory F. Cooper. The computational complexity of probabilsitic inference using belief networks. *Artificial Intelligence*, 42:393–405, 1990.

[Cowles and Carlin, 1996] Mary Kathryn Cowles and Bradley P. Carlin. Markov chain Monte Carlo convergence diagnostics: A comparative review. *Journal of the American Statistical Association*, 1996. To Appear.

[Cox *et al.*, 1972] J. Cox, F. Nolle, and R. Arthur. Digital analysis of the electro-encephalogram, the blood presure wave, and the electrocardiogram. *Proceedings of the IEEE*, 60(10):1137–1164, 1972.

[Dagum and Chavez, 1993] Paul Dagum and R. Martin Chavez. Approximating probabilistic inference in Bayesian belief networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):246–255, March 1993.

[Dagum and Horvitz, 1993] Paul Dagum and Eric Horvitz. A Bayesian analysis of simulation algorithms for inference in belief networks. *Networks*, 23(5):499–516, August 1993.

[Dagum and Luby, 1993] Paul Dagum and Michael Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, March 1993.

[D'Ambrosio, 1990] Bruce D'Ambrosio. Symbolic probabilistic inference in belief nets. In *Proceedings of the Eighth National Conference on Artificial Intelligence (AAAI)*, pages 126–131. MIT Press, 1990.

[D'Ambrosio, 1994] Bruce D'Ambrosio. Symbolic probabilistic inference in large BN20 networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 128–135, Seattle, WA, 1994. Morgan Kauffman.

[Damerdji, 1996] H. Damerdji. Maximum likelihood estimation of generalized semi-Markov processes. *Discrete Event Dynamic Systems: Theory & Applications*, 6(1):73–104, 1996.

[Dannenberg, 1984] Roger Dannenberg. An on-line algorithm for real-time accompaniment. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 193–198, 1984.

[Darwiche, 1995] Adnan Darwiche. Conditioning methods for exact and approximate inference in causal networks. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, August 1995. Morgan Kauffman.

[Dawid and Lauritzen, 1993] A. P. Dawid and S. L. Lauritzen. Hyper Markov laws in the statistical analysis of decomposable graphical models. *The Annals of Statistics*, 21(3), September 1993.

[Dawid *et al.*, 1994] A. P. Dawid, Uffe Kjaerulff, and Steffen L. Lauritzen. Hyrbid propagation in junction trees. In *Proceedings of the Fifth International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems (IPMU)*, pages 965–971, Cité Internationale Universitaire, Paris, 1994.

[Dawid, 1979] A. P. Dawid. Conditional independence in statistical theory. *Journal Royal Statistical Society B*, 41(1):1–31, 1979.

[Dawid, 1980] A. P. Dawid. Conditional independence for statistical operations. *Annals of Statistics*, 8:598–617, 1980.

[Dawid, 1992] A. P. Dawid. Applications of a general propagation algorithm for probabilistic expert systems. *Statistics and Computing*, 2:25–36, 1992.

[de Dombal *et al.*, 1972] F. T. de Dombal, D. J. Leaper, J. R. Staniland, A. P. McCann, and J. C. Horrocks. Computer-aided diagnosis of acute abdominal pain. *British Medical Journal*, 2:9–13, 1972.

[Dean and Boddy, 1988] Thomas Dean and Mark Boddy. An analysis of time-dependent planning. In *Proceedings of Seventh National Conference on Artificial Intelligence (AAAI)*, volume 1, pages 49–54, St. Paul, MN, 1988.

[Dechter *et al.*, 1990] Rina Dechter, Avi Dechter, and Judea Pearl. Optimization in constraint networks. In R. M. Oliver and J. Q. Smith, editors, *Influence Diagrams, Belief Nets, and Decision Analysis*, chapter 18, pages 411–425. John Wiley & Sons, 1990.

[Dempster *et al.*, 1977] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1):1–38, 1977.

[Deshayes and Picard, 1986] Jean Deshayes and Dominique Picard. Off-line statistical analysis of change-point models using non parametric and likelihood methods. In M. Basseville and A. Benveniste, editors, *Detection of Abrupt Changes in Signals and Dynamical Systems*, pages 103–168. Springer-Verlag, Berlin, 1986.

[Devijver, 1985] P. A. Devijver. Baum's forward-backward algorithm revisited. *Pattern Recognition Letters*, 3:369–373, 1985.

[Devroye, 1986] Luc Devroye. *Non-Uniform Random Variate Generation*. Springer-Verlag, New York, 1986.

[Díez, to appear] F. J. Díez. Local conditioning in Bayesian networks. *Artificial Intelligence*, to appear.

[Djurič *et al.*, 1992] Petar M. Djurič, Steven M. Kay, and G. Fay Boudreaux-Bartels. Segmentation of nonstationary signals. In *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, volume 5, pages 161–164, San Francisco, California, March 1992.

[Doyle and Fayyad, 1991] Richard J. Doyle and Usama M. Fayyad. Sensor selection techniques in device monitoring. In *Proceedings of the Second Annual Conference on AI, Simulation and Planning in High Autonomy Systems: Integrating Qualitative and Quantitative System Knowledge*, Cocoa Beach, Florida, 1991. IEEE Comput. Society Press.

[Doyle *et al.*, 1993] Richard J. Doyle, Steve A. Chien, Usama M. Fayyad, and E. Jay Wyatt. Focused real-time systems monitoring based on multiple anomaly models. In *International Qualitative Reasoning Conference (QR-93)*, Eastsound, Washington, May 1993.

[Doyle, 1995] Richard J. Doyle. Determining the loci of anomalies using minimal causal models. In *International Joint Conference on Artificial Intelligence*, Montreal, Quebec, August 1995.

[Draper and Hanks, 1994] Denise L. Draper and Steve Hanks. Localized partial evaluation of belief networks. In L. de Mantaras and D. Poole, editors, *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 170–177. Morgan Kaufmann, July 1994.

[Draper, 1995] Denise L. Draper. Clustering without (thinking about) triangulation. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, August 1995. Morgan Kauffman.

[Driver and Morrell, 1995] Eric Driver and Darryl Morrell. Implementation of continuous Bayesian networks using sums of weighted Gaussians. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, August 1995. Morgan Kauffman.

[Druzdzel and Suermondt, 1994] Marek J. Druzdzel and Henri J. Suermondt. Relevance in probabilistic models: "backyards" in a "small world". In *Working notes of the AAAI-1994 Fall Symposium Series: Relevance*, pages 60–63, New Orleans, Louisiana, November 1994.

[Duda and Hart, 1973] Richard O. Duda and Peter E. Hart. *Pattern Classification and Scene Analysis.* Wiley & Sons, 1973.

[Duda *et al.*, 1976] Richard O. Duda, Peter E. Hart, and Nils J. Nilsson. Subjective Bayesian methods for rule-based inference systems. In *Proceedings of the National Computer Conference, American Federation of Information Processing Societies Conference (AFIPS)*, volume 45, pages 1075–1082, 1976.

[Dufour, 1982] Jean-Marie Dufour. Recursive stability analysis of linear regression relationships: An exploratory methodology. *Journal of Econometrics*, 19:31–76, 1982.

[Eakinos, 1991] D. Eakinos. Insensitivity of generalized semi-Markov processes evolving in a random environment. *Journal of the Operations Research Society*, 42(12):1111–1115, December 1991.

[Feldman and Sproull, 1977] Jerome A. Feldman and Robert F. Sproull. Decision theory and artificial intelligence ii: The hungry monkey. *Cognitive Science*, 1:158–192, 1977.

[Feldman and Yakimovsky, 1974] Jerome A. Feldman and Yoram Yakimovsky. Decision theory and artificial intelligence: I. a semantics-based region analyzer. *Artificial Intelligence*, 5:349–371, 1974.

[Ferreira, 1975] P. E. Ferreira. A Bayesian analysis of switching regression model: Known number of regimes. *Journal of the American Statistical Association*, 70:370–374, 1975.

[Fertig and Breese, 1993] Kenneth W. Fertig and John S. Breese. Probability intervals over influence diagrams. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):280–286, March 1993.

[Forney, 1973] G. D. Forney, Jr. The Viterbi algorithm. *Proceedings of the IEEE*, 61:268–278, 1973.

[Frydenberg, 1990] Morten Frydenberg. The chain graph Markov property. *Scandanavian Journal of Statistics*, 17:333–353, 1990.

[Fung and Chang, 1989] R. Fung and K. C. Chang. Weighting and integrating evidence for stochastic simulation in Bayesian networks. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, Windsor, Ontario, 1989. Morgan Kauffman.

[Geiger and Heckerman, 1994a] Dan Geiger and David Heckerman. A characterization of the Dirichlet distribution through global and local independence. Technical Report MSR-TR-94-16, Microsoft Research, Redmond, WA, 1994. Revised Feb 1995.

[Geiger and Heckerman, 1994b] Dan Geiger and David Heckerman. Learning Gaussian networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 235–243, Seattle, WA, 1994. Morgan Kauffman.

[Gelfand and Smith, 1990] Alan E. Gelfand and Adrian F. M. Smith. Sampling-based approaches to calculating marginal densities. *Journal of the American Statistical Association*, 85(410):398–409, June 1990.

[Gelfand, 1995] Alan E. Gelfand. Gibbs sampling. *Encyclopedia of Statistical Sciences*, 1995. to appear.

[Geman and Geman, 1984] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[Gilks *et al.*, 1994] W. R. Gilks, G. O. Roberts, and E. I. George. Adaptive direction sampling. *The Statistician*, 43(1):179–189, 1994.

[Ginsberg, 1969] A. S. Ginsberg. *Decision Analysis in Clinical Patient Management with an Application to the Pleural Effusion Problem.* PhD thesis, Stanford University, 1969.

[Glasserman and Yao, 1992] P. Glasserman and D. D. Yao. Structured buffer-allocation problems. *Mathematics of Operations Research*, 17(1):1–21, February 1992.

[Glasserman and Yao, 1996] P. Glasserman and D. D. Yao. Structured buffer-allocation problems. *Discrete Event Dynamic Systems: Theory & Applications*, 6(1):9–41, 1996.

[Glesner and Koller, 1995] Sabine Glesner and Daphne Koller. Constructing flexible dynamic belief networks from first-order probabilistic knowledge bases. In *Proceedings of the European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty (ECSQARU-95)*, Fribourg, Switzerland, July 1995.

[Goldman and Charniak, 1990] Robert P. Goldman and Eugene Charniak. Dynamic construction of belief networks. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pages 90–97, Cambridge, July 1990.

[Goldman and Charniak, 1993] Robert P. Goldman and Eugene Charniak. A language for construction of belief networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):196–208, March 1993.

[Golumbic, 1980] Martin Charles Golumbic. *Algorithmic Graph Theory and Perfect Graphs.* Academic Press, London, 1980.

[Good, 1961] I.J. Good. A causal calculus. *British Journal of the Philosophy of Science*, 11:305–318, 1961.

[Good, 1977] I. J. Good. Dynamic probability, computer chess, and the measurement of knowledge. In E. W. Elcock and D. Michie, editors, *Machine Intelligence*, pages 139–150. Wiley, New York, 1977.

[Gorry and Barnett, 1968] G. A. Gorry and G. O. Barnett. Experience with a model of sequential diagnosis. *Computers and Biomedical Research*, 1:490–507, 1968.

[Gorry *et al.*, 1973] G. A. Gorry, J. P. Kassirer, A. Essig, and W. B. Schwartz. Decision analysis as the basis for computer-aided management of acute renal failure. *American Journal of Medicine*, 55:473–484, 1973.

[Guedon and Cocozza-Thivent, 1990] Y. Guedon and C. Cocozza-Thivent. Explicit state occupancy modelling by hidden semi-Markov models: application of Derin's scheme. *Computer Speech and Language*, 4(2):167–192, April 1990.

[Guedon, 1992] Y. Guedon. Review of several stochastic speech unit models. *Computer Speech and Language*, 6(4):377–402, October 1992.

[Gupta *et al.*, 1987] V. N. Gupta, M. Lennig, and P. Mermelstein. Integration of acoustic information in a large vocabulary word recognizer. In *Proceedings of the ICASSP*, volume 2, pages 697–700, Dallas, TX, April 1987.

[Haas and Shedler, 1986] Peter J. Haas and G. S. Shedler. Regenerative generalized semi-markov processes. Technical Report IBMJ-5330, I.B.M., Yorktown Heights, 1986.

[Haas and Shedler, 1996] Peter J. Haas and G. S. Shedler. Estimation methods for passage times using one-dependent cycles. *Discrete Event Dynamic Systems: Theory & Applications*, 6(1):43–72, 1996.

[Haddawy *et al.*, 1995] Peter Haddawy, James W. Helwig, Liem Ngo, and Robert A. Krieger. Clinical simulation using context-sensitive temporal probability models. In *Proceedings of the 19th Annual Symposium on Computer Applications in Medical Care (SCAMC95)*, 1995.

[Haddawy, 1994] Peter Haddawy. Generating Bayesian networks from probability logic knowledge bases. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 1994.

[Hampshire and Waibel, 1990] John B. Hampshire, II and Alexander H. Waibel. A novel object function for improved phoneme recognition using time-delay neural networks. *IEEE Transactions on Neural Networks*, 1(2):219–228, June 1990.

[Hastings, 1970] W. K. Hastings. Monte carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

[Hawkins, 1987] D. L. Hawkins. A test for a change point in a parametric model based on a maximal Wald-type statistic. *Sankhya A*, 49(3):368–376, 1987.

[Heckerman and Horvitz, 1988] David E. Heckerman and Eric J. Horvitz. The myth of modularity in rule-based systems for reasoning with uncertainty. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence*, volume 2, pages 23–34. Elsevier Science Publishers B.V., North-Holland, 1988.

[Heckerman *et al.*, 1995] David Heckerman, A. Mamdani, and Michael Wellman. Real-world applications of Bayesian networks. *Communications of the ACM*, 38, 1995.

[Henrion *et al.*, 1991] Max Henrion, John S. Breese, and Eric J. Horvitz. Decision analysis and expert systems. *AI Magazine*, 1991. Or see [Horvitz *et al.*, 1988].

[Henrion, 1988] Max Henrion. Propagation of uncertainty by Bayesian networks by probabilistic logic sampling. In J. F. Lemmer and L. N. Kanal, editors, *Uncertainty in Artificial Intelligence 2*, pages 149–163. Elsevier/North-Holland, Amsterdam, London, New York, 1988.

[Henrion, 1991] Max Henrion. Search-based methods to bound diagnostic probabilities in very large belief nets. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventh Conference*, pages 142–150. Morgan Kaufmann, 1991.

[Hills and Smith, 1992] S. E. Hills and A. F. M. Smith. Parameterization issues in Bayesian inference (with discussion). In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 4*, pages 641–649. Oxford, Oxford University Press, 1992.

[Horvitz and Barry, 1995] Eric Horvitz and Matthew Barry. Display of information for time-critical decision making. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pages 296–305, Montreal, 1995. Morgan Kauffman.

[Horvitz and Heckerman, 1986] Eric J. Horvitz and David E. Heckerman. The inconsistent use of measures of certainty in artificial intelligence research. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*. Elsevier Science Publishers B.V., North-Holland, 1986.

[Horvitz *et al.*, 1988] Eric J. Horvitz, John S. Breese, , and Max Henrion. Decision theory in expert systems and artificial intelligence. *International Journal of Approximate Reasoning*, 2:247–302, 1988.

[Horvitz *et al.*, 1992] Eric Horvitz, Corinne Ruokangas, Sampath Srinivas, and Matthew Barry. A decision-theoretic approach to the display of information for time-critical decisions: The Vista project. In *Proceedings of SOAR-92*, NASA/Johnson Space Center, Huston, TX, August 1992.

[Howard and Matheson, 1984a] Ronald A. Howard and J. E. Matheson. Influence diagrams. In R. A. Howard and J. E. Matheson, editors, *Principles and Applications of Decision Analysis*, chapter 2. Strategic Decisions Group, Menlo Park, Ca, 1984.

[Howard and Matheson, 1984b] Ronald A. Howard and J. E. Matheson. *Readings on the Principles and Applications of Decision Analysis.* Strategic Decisions Group, Menlo Park, California, 1984.

[Howard, 1968] Ronald A. Howard. The foundations of decision analysis. *IEEE Transactions on Systems Science and Cybernetics*, 4:211–219, 1968. Reprinted in [Howard and Matheson, 1984b].

[Howard, 1970] Ronald A. Howard. Decision analysis: Perspectives on inference, decision, and experimentation. *Proceedings of the IEEE*, 58(5):632–643, 1970. Reprinted in [Howard and Matheson, 1984b].

[Howard, 1971] Ronald A. Howard. *Dynamic Probabilistic Systems: Volume II: SemiMarkov and Decision Processes.* Wiley, 1971.

[Hrycej, 1990] Tomas Hrycej. Gibbs sampling in Bayesian networks. *Artificial Intelligence*, 46(3):351–363, December 1990.

[Huang and Jack, 1989] X. D. Huang and M. A. Jack. Semi-continuous hidden Markov models for speech signals. *Computer Speech and Language*, 3:239–251, 1989.

[Inselman and Arsenal, 1968] E. H. Inselman and F. Arsenal. Tests for several regression equations. *Annals of Mathematical Statistics*, 39, 1968.

[Ishii *et al.*, 1979] Naohiro Ishii, Akira Iwata, and Nobuo Suzumura. Segmentation of non-stationary time series. *International Journal of Systems Science*, 10(8), 1979.

[Ishii *et al.*, 1980] Naohiro Ishii, Hideyuki Sugimoto, Akira Iwata, and Nobuo Suzumura. Computer classification of the EEG time series by Kullback information measure. *International Journal of Systems Science*, 11(6), 1980.

[Jacobs and Kiefer, 1973] Walter Jacobs and Maxine Kiefer. Robot decisions based on maximizing utility. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 402–411, 1973.

[Jacobs *et al.*, 1991] Robert A. Jacobs, Michael I. Jordan, Steven J. Nowlan, and Geoffrey E. Hinton. Adaptive mixture of local experts. *Neural Computation*, 3(1):79–87, 1991.

[Jensen and Andersen, 1990] Frank Jensen and Stig Kjaer Andersen. Approximations in Bayesian belief universes for knowledge-based systems. In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, 1990.

[Jensen and Jensen, 1994] Finn V. Jensen and Frank Jensen. Optimal junction trees. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, 1994.

[Jensen *et al.*, 1990a] Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, 4:269–282, 1990.

[Jensen *et al.*, 1990b] Finn Verner Jensen, Kristian G. Olesen, and Stig Kjaer Andersen. An algebra of Bayesian belief universes for knowledge-based systems. *Networks*, 20:637–659, 1990.

[Jensen *et al.*, 1994] Frank Jensen, Finn V. Jensen, and Søren L. Dittmer. From influence diagrams to junction trees. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, July 1994.

[Jensen *et al.*, 1995] Claus Skaanning Jensen, Uffe Kjaerulff, and Augustin Kong. Blocking gibbs sampling in very large probabilistic expert systems. *International Journal of Human-Computer Studies*, 42(6):647–666, June 1995.

[Jensen, 1993] F. V. Jensen. Introduction to Bayesian networks. Technical Report IR-93-2003, Dept. of Mathematics and Computer Science, Aalborg University, Aalborg, Denmark, 1993.

[Jensen, 1994] Frank Jensen. Implementation aspects of various propagation algorithms in Hugin. Technical Report R 94-2014, Aalborg University, Denmark, March 1994.

[Kanazawa and Dean, 1989] Keiji Kanazawa and Thomas Dean. A model for projection and action. In *Eleventh International Joint Conference on Artificial Intelligence (IJCAI)*, pages 985–990, 1989.

[Kelly, 1988] F. P. Kelly. Discussion comments on [Lauritzen and Spiegelhalter, 1988]. *Journal Royal Statistical Society, Series B*, 50(2):194–195, 1988.

[Kenley, 1986] C. Robert Kenley. Influence diagram models with continuous variables. Technical Report LMSC-DO67192, Astronautics Division, Lockheed Missles and Space Company, Sunnyvale, CA, June 1986.

[Kiiveri *et al.*, 1984] Harri Kiiveri, Terry P. Speed, and J.B. Carlin. Recursive causal models. *Journal of the Australian Mathematical Society A*, 36:30–52, 1984.

[Kim and Valtorta, 1995] Young-Gyun Kim and Marco Valtorta. On the detection of conflicts in diagnostic Bayesian networks using abstraction. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, Montreal, August 1995. Morgan Kauffman.

[Kirkpatrick *et al.*, 1983] S. Kirkpatrick, C. C. Gelatt, Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, May 1983.

[Kjaerulff, 1990] Uffe Kjaerulff. Triangulation of graphs — algorithms giving small total state space. Technical Report R-90-09, Dept. of Mathematics and Computer Science, Aalborg University, Aalborg, Denmark, 1990.

[Kjaerulff, 1993] Uffe Kjaerulff. Approximation of bayesian networks through edge removals. Technical Report IR-93-2007, Institute for Electronic Systems, Department of Mathematics and Computer Science, Aalborg, Denmark, August 1993.

[Kjaerulff, 1994] Uffe Kjaerulff. Reduction of computational complexity in bayesian networks through the removal of weak dependences. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, San Mateo, Calif., 1994. Morgan Kaufmann.

[Kjaerulff, 1995a] Uffe Kjaerulff. dHugin: A computational system for dynamic time-sliced bayesian networks. *International Journal of Forecasting*, 1995.

[Kjaerulff, 1995b] Uffe Kjaerulff. HUGS: Combining exact inference and Gibbs sampling in junction trees. In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, Montreal, 1995.

[Kohlmorgen *et al.*, 1994] J. Kohlmorgen, K.-R. Müller, and K. Pawelzik. Competing predictors segment and identify switching dynamics. In M. Marinaro and P. G. Morasso, editors, *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, volume 2, pages 1045–1048, Sorrento, Italy, May 1994. Springer-Verlag.

[Kotz *et al.*, 1982] Samuel Kotz, Norman Lloyd Johnson, and Campbell B. Read, editors. *Encyclopedia of Statistical Sciences (9 Volumes)*. John Wiley & Sons, New York, 1982.

[Krämer *et al.*, 1988] Walter Krämer, Werner Ploberger, , and Raimund Alt. Testing for structural change in dynamic models. *Econometrica*, 56(6):1355–1369, November 1988.

[Laskey and Lehner, 1994] Kathryn Blackmond Laskey and Paul E. Lehner. Metareasoning and the problem of small worlds. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(11):1643–1652, November 1994.

[Laskey, 1991] Kathrun Blackmond Laskey. Conflict and surprise: Heuristics for model revision. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventh Conference*, pages 197–204. Morgan Kaufmann, 1991.

[Laskey, 1993] Kathryn Blackmond Laskey. Sensitivity analysis for probability assessments in Bayesian networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 136–142. Morgan Kaufmann, 1993.

[Lauritzen and Spiegelhalter, 1988] Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their applications to expert systems. *Journal Royal Statistical Society, Series B*, 50(2):157–224, 1988. Reprinted in [Shafer and Pearl, 1990, Pages 415–448].

[Lauritzen and Wermuth, 1989] S. L. Lauritzen and N. Wermuth. Graphical models for associations between variables, some of which are qualitative and some quantitative. *The Annals of Statistics*, 17(1):31–57, 1989.

[Lauritzen *et al.*, 1984] S. L. Lauritzen, T. P. Speed, and K. Vijayan. Decomposable graphs and hypergraphs. *Journal of the Australian Mathematical Society (Series A)*, 36:12–29, 1984.

[Lauritzen *et al.*, 1990] S. L. Lauritzen, A. P. Dawid, B. N. Larsen, and H.-G. Leimer. Independence properties of directed Markov fields. *Networks*, 20:491–505, 1990.

[Lauritzen, 1982] S. L. Lauritzen. *Lectures on Contingency Tables.* University of Aalborg Press, Denmark, 2nd ed. edition, 1982.

[Lauritzen, 1992] Steffen L. Lauritzen. Propagation of probabilities, means and variances in mixed graphical association models. *Journal of the American Statistical Association*, 87(420):1098–1108, December 1992.

[Lee and Chou, 1989] An-Chen Lee and Jiing-Shyang Chou. Segmentation of piecewise stationary signals. *International Journal of Systems Science*, 20(10):1827–1842, 1989.

[Lee and Chou, 1990] An-Chen Lee and Jiing-Shyang Chou. Computer classification of the PCG waveform. *International Journal of Systems Science*, 21(3):593–609, 1990.

[Levinson *et al.*, 1983] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi. An introduction to the application of the theory of probabilistic functions of Markov process to automatic speech recognition. *Bell System Technical Journal*, 62(4):1035–1074, April 1983.

[Levinson, 1986] S. E. Levinson. Continuously variable duration hidden Markov models for automatic speech recognition. *Computer Speech and Language*, 1:29–45, 1986.

[Levy *et al.*, 1996] Bernard C. Levy, Albert Benveniste, and Ramine Nikoukah. High-level primitives for recursive maximum likelihood estimation. *IEEE Transactions on Automatic Control*, 1996.

[Li and D'Ambrosio, 1994] Zhaoyu Li and B. D'Ambrosio. Efficient inference in bayes networks as a combinatorial optimization problem. *International Journal of Approximate Reasoning*, 11(1):55–81, July 1994.

[Li and Gibson, 1996] Ta-Hsin Li and Jerry D. Gibson. Speech analysis and segmentation by parametric filtering. *IEEE Transactions on Speech and Audio Processing*, 4(3):203–213, May 1996.

[Lin and Teräsvirta, 1994] Chien-Fu Jeff Lin and Timo Teräsvirta. Testing the constancy of regression parameters against continuous structural change. *Journal of Econometrics*, 62:211–228, 1994.

[Liu *et al.*, 1994] Jun Liu, W. Wong, and A. Kong. Correlation structure of the Gibss sampler with applications to the comparisons of estimators and augmentation schemes. *Biometrika*, 81:27–40, 1994.

[Liu, 1991] Jun Liu. *Correlation Structure and Convergence Rate of the Gibbs Sampler.* PhD thesis, Department of Statistics, University of Chicago, June 1991.

[Lütkepohl, 1988] Helmut Lütkepohl. Prediction tests for structural stability. *Journal of Econometrics*, 39:267–296, 1988.

[Lütkepohl, 1989] Helmut Lütkepohl. Prediction tests for structural stability of multiple time series. *Journal of Business and Economic Statistics*, 7:129–135, 1989.

[Lütkepohl, 1993] Helmut Lütkepohl. *Introduction to Multiple Time Series Analysis.* Springer-Verlag, second edition edition, 1993.

[Maier, 1983] David Maier. *The Theory of Relational Databases.* Computer Science Press, 1983.

[Makhoul, 1975] John Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4):561–580, April 1975.

[Marroquin, 1995] Jose L. Marroquin. Measure fields for function approximation. *IEEE Transactions on Neural Networks*, 6(5):1081–1090, September 1995.

[Matzkevich and Abramson, 1993] Izhar Matzkevich and Bruce Abramson. Some complexity considerations inthe combination of belief networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 152–158. Morgan Kaufmann, 1993.

[McCarthy and Hayes, 1969] John McCarthy and Patrick Hayes. Some philosophical problems from the standpoint of artificial intelligence. In B. Meltzer and D. Michie, editors, *Machine Intelligence*, volume 4, pages 463–502. Edinburgh University Press, Edinburgh, U.K., 1969.

[McKeague and Wefelmeyer, 1995] Ian W. McKeague and Wolfgang Wefelmeyer. Markov chain Monte Carlo and Rao-blackwellization. Florida State University, December 1995.

[MCMC, 1996] MCMC preprint service. http://www.statslab.cam.ac.uk/ mcmc/html/index.html, 1996. (Repository of literature on Markov chain Monte Carlo methods).

[Metropolis *et al.*, 1953] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equations of state calculations by fast computing machine. *Journal of Chemical Physics*, 21:1087–1091, 1953.

[Meyn and Tweedie, 1993] Sean P. Meyn and Richard L. Tweedie. *Markov Chains and Stochastic Stability.* Springer-Verlag, London, 1993.

[Miller *et al.*, 1976] A. C. Miller, M. M. Merhofer, R. A. Howard, J. E. Matheson, and T. R. Rice. Development of automated aids for decision analysis. Technical report, SRI International, Menlo Park, California, 1976.

[Minsky and Selfridge, 1961] Marvin Minsky and O. G. Selfridge. Learning in random nets. In C. Cherry, editor, *Information Theory*, pages 335–347. Butterworths, London, 1961.

[Monti and Cooper, 1996] Stefano Monti and Gregory F. Cooper. Bounded recursive decomposition: A search-based method for belief network inference under limited resources. *International Journal of Approximate Reasoning*, 1996. to appear.

[Moody and Darken, 1989] John Moody and Christian J. Darken. Fast learning in networks of locally-tuned processing units. *Neural Computation*, 1(2):281–294, 1989.

[Moussouris, 1974] John Moussouris. Gibbs and Markov random systems with constraints. *Journal of Statistical Physics*, 10(1), 1974.

[Munson, 1971] John H. Munson. Robot planning, execution, and monitoring in an uncertain environment. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 338–349, 1971.

[Neal, 1993] Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept. of Computer Science, University of Toronto, 1993.

[Neal, 1995] Radford M. Neal. Suppressing random walks in Markov chain Monte Carlo using ordered overrelaxation. Technical Report 9508, Department of Computer Science, University of Toronto, 1995.

[Ngo and Haddawy, 1995a] Liem Ngo and Peter Haddawy. Answering queries from context-sensitive probabilistic knowledge bases. *Theoretical Computer Science: Special Issue on Uncertainty in Databases and Deductive Systems*, 1995. To Appear.

[Ngo and Haddawy, 1995b] Liem Ngo and Peter Haddawy. Probabilistic logic programming and Bayesian networks. In *Proceedings of the Asian Computing Science Conference*, Pathumthani, Thailand, December 1995.

[Ngo *et al.*, 1995] Liem Ngo, Peter Haddawy, and James Helwig. A theoretical framework for context-sensitive temporal probability model construction with application to plan projection. In *Proceedings of the Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, pages 419–426, Montreal, August 1995. Morgan Kauffman.

[Nicholson and Brady, 1994] Ann E. Nicholson and J. Michael Brady. Dynamic belief networks for discrete monitoring. *IEEE Transactions on Systems, Man and Cybernetics*, 24(11):1593–1610, November 1994.

[Nilsson, 1965] Nils J. Nilsson. *Learning Machines.* McGraw-Hill, New York, 1965.

[Noormohammadian and Oppel, 1993] M. Noormohammadian and U. G. Oppel. Examples of causal probabilistic expert systems. In *Symbolic and Quantitative Approaches to Reasoning and Uncertainty. European Conference ECSQARU Proceedings.*, pages 290–295, Berlin, Germany, 1993. Springer-Verlag.

[NSTS, 1988] NSTS 1988 news reference manual, 1988. `http://www.ksc.nasa.gov/shuttle/technology/sts-newsref/stsref-toc.html`.

[Olesen, 1993] Kristian G. Olesen. Causal probabilistic networks with both discrete and continuous variables. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(15):275–279, 1993.

[Ostendorf and Roukos, 1989] Mari Ostendorf and Salim Roukos. A stochastic segment model for phoneme-based continuous speech recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(12):1857–1869, December 1989.

[Page, 1954] E.S. Page. Continuous inspection schemes. *Biometrika*, 41:100–115, 1954.

[Pawelzik *et al.*, 1996] Klaus Pawelzik, Jens Kohlmorgen, and Klaus-Robert Müller. Annealed competition of experts for a segmented and classification of switching dynamics. *Neural Computation*, 8:340–356, 1996.

[Pearl, 1982] Judea Pearl. Reverend Bayes on inference engines: A distributed hierarchical approach. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pages 133–136, Pittsburgh, Pennsylvania, 1982. Morgan Kaufmann.

[Pearl, 1986a] Judea Pearl. A constraint-propagation approach to probabilistic reasoning. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 357–370. North Holland, Amsterdam, 1986.

[Pearl, 1986b] Judea Pearl. Fusion, propagation and structuring in belief networks. *Artificial Intelligence*, 29(3):241–288, 1986.

[Pearl, 1987] Judea Pearl. Evidential reasoning using stochastic simulation of causal models. *Artificial Intelligence*, 32(2):245–257, 1987.

[Pearl, 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, San Mateo, CA, 1988.

[Pearl, 1994] Judea Pearl. Three statistical puzzles. Technical Report R-217, U.C.L.A., February 1994.

[Pearl, 1995a] Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–709, December 1995.

[Pearl, 1995b] Judea Pearl. Causal inference from indirect experiments. *Artificial Intelligence in Medicine Journal*, 7(6):561–582, 1995.

[Pearl, 1996] Judea Pearl. Structural and probabilistic causality. In D.R. Shanks, K.J. Holyoak, and D.L. Medin, editors, *The Psychology of Learning and Motivation*, volume 34. Academic Press, San Diego, 1996. Also available as U.C.L.A. Computer Science Technical Report R-237, November 1995.

[Peirce, 1956] C. S. Peirce. The probability of induction. In *The World of Mathematics*, volume 2, pages 1341–1354. Simon and Shuster, New York, 1956.

[Peot and Shachter, 1991] Mark A. Peot and Ross D. Shachter. Fusion and propagation with multiple observations in belief networks. *Artificial Intelligence*, 48(3):299–318, 1991.

[Pesaran *et al.*, 1985] M. H. Pesaran, R. P. Smith, and J. S. Yeo. Testing for stability and predictive failure: A review. *The Manchester School of Economic and Social Studies*, 53:280–295, 1985.

[Ploberger and Krämer, 1992] Werner Ploberger and Walter Krämer. The CUSUM test with OLS residuals. *Econometrica*, 60(2):271–285, March 1992.

[Ploberger *et al.*, 1989] Werner Ploberger, Walter Krämer, and K. Kontrus. A new test for structural stability in the linear regression model. *Journal of Econometrics*, 40:307–318, 1989.

[Poh *et al.*, 1994] Kim Leng Poh, Michael R. Fehling, and Eric J. Horvitz. Dynamic construction and refinement of utility-based categorization models. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(11):1653–1663, November 1994.

[Poole, 1993a] David Poole. Probabilistic horn abduction and bayesian networks. *Artificial Intelligence*, 64(1):81–129, November 1993.

[Poole, 1993b] David Poole. The use of conflicts in searching Bayesian networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 359–367. Morgan Kaufmann, 1993.

[Pradhan *et al.*, 1995] Malcolm Pradhan, Max Henrion, Gregory Provan, Brendan Del Favero, and Kurt Huang. The sensitivity of belief networks to imprecise probabilities: An experimental investigation. Technical Report KSL-95-77, Institute for Decision Systems Research, Los Altos, California, November 1995. `ftp://ksl.stanford.edu/pub/KSL_Reports/KSL_95_77.ps`, submitted to Elsevier Preprints.

[Press *et al.*, 1992] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: The Art of Scientific Computing.* Cambridge University Press, second edition edition, 1992.

[Provan, 1993a] Gregory M. Provan. Dynamic network construction and updating techniques for the diagnosis of acute abdominal pain. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):299–307, March 1993.

[Provan, 1993b] Gregory M. Provan. Tradeoffs in constructing and evaluating temporal influence diagrams. In *Proceedings of the 9th Conference on Uncertainty in Artificial Intelligence*, pages 40–47, 1993.

[Provan, 1994] Gregory M. Provan. Tradeoffs in knowledge-based construction of probabilistic models. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(11):1580–1592, November 1994.

[Quandt, 1958] R. E. Quandt. The estimation of the paramters of a linear regression system obeying two separate regimes. *Journal of the American Statistical Association*, 53:873–880, 1958.

[Quandt, 1960] R. E. Quandt. Tests of the hypothesis that a linear regression system obeys two separate regimes. *Journal of the American Statistical Association*, 55:324–330, 1960.

[Rabiner and Sambur, 1976] Lawrence R. Rabiner and Marvin R. Sambur. Some preliminary experiments in the recognition of connected digits. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, ASSP-24(2), April 1976.

[Rabiner, 1989] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2), February 1989.

[Reddy *et al.*, 1973] D. R. Reddy, L. D. Erman, R. D. Fennell, and R. B Neely. The HEARSAY speech understanding system. In *Proc. Third International Joint Conference on Artificial Intelligence*, pages 185–193, Stanford, CA, 1973.

[Rissanen, 1978] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14:465–471, 1978.

[Rissanen, 1986] Jorma Rissanen. Stochastic complexity and modeling. *The Annals of Statistics*, 14(3):1080–1100, 1986.

[Roberts and Sahu, 1996] G. O. Roberts and S. K. Sahu. Updating schemes, correlation structure, blocking and parametrisation of the Gibbs sampler. to appear. Obtainable from [MCMC, 1996] until published., May 1996.

[Rojas-Guzmán and Kramer, 1993] Carlos Rojas-Guzmán and Mark A. Kramer. GALGO: A genetic ALGOrithm decision support tool for complex uncertain systems modeled with Bayesian belief networks. In *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 368–375. Morgan Kaufmann, 1993.

[Rosenthal, 1975] A. Rosenthal. A computer scientist looks at reliability computations. In Barlow, Fussell, and Singpurwalla, editors, *Reliability and Fault Tree Analysis*, pages 133–152. SIAM, Philadelphia, 1975.

[Rosenthal, 1995a] Jeffrey S. Rosenthal. Markov chain convergence: From finite to infinite, 1995. `ftp://ustat.toronto.edu/jeff/infinite.ps.Z`.

[Rosenthal, 1995b] Jeffrey S. Rosenthal. Minorization conditions and convergence rates for Markov chain Monte Carlo. *Journal of the Americal Statistical Society*, 90:558–566, 1995.

[Russell and Moore, 1985] M. J. Russell and R. K. Moore. Explicit modeling of state occupancy in hidden Markov models for automatic speech recognition. In *Proceedings of the ICASSP*, pages 5–8, Tampa, Florida, March 1985.

[Saffiotti and Emkehrer, 1994] Alessandro Saffiotti and Elisabeth Emkehrer. Inference-driven construction of valuation systems from first-order clauses. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(11):1611–1624, November 1994.

[Santos, 1994] Eugene Santos, Jr. A linear constraint satisfaction approach to cost-based abduction. *Artificial Intelligence*, 65(1):1–28, 1994.

[Saul and Jordan, 1995] Lawrence K. Saul and Michael I. Jordan. Exploiting tractable substructures in intractable networks. In *Advances of Neural Information Processing Systems: Proceedings of the 1995 Conference*, 1995.

[Saul *et al.*, 1996] Lawrence K. Saul, Tommi Jaakkola, and Michael I. Jordan. Mean field theory for sigmoid belief networks. *Journal of Artificial Intelligence Research*, 4:61–76, March 1996.

[Savage, 1954] Leonard J. Savage. *The Foundations of Statistics*. Wiley, New York, 1954. Second edition: Dover 1972.

[Sclove, 1983] Stanley L. Sclove. Time-series segmentation: A model and a method. *Information Sciences*, 29(1):7–25, 1983.

[Seewald, 1992] W. Seewald. Discussion of [Hills and Smith, 1992]. In J. M. Bernardo, J. O. Berger, A. P. Dawid, and A. F. M. Smith, editors, *Bayesian Statistics 4*, pages 241–243. Oxford, Oxford University Press, 1992.

[Segen and Sanderson, 1980] Jakub Segen and Arthur C. Sanderson. Detecting changes in a time series. *IEEE Transactions on Information Theory*, IT-26(2):249–255, March 1980.

[Sen, 1980] Pranab Kumar Sen. Asymptotic theory of some tests for a possible change in the regression slope occurring at unknown time point. *Zeitschrift fur Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 52:203–218, 1980.

[Sen, 1983] Pranab Kumar Sen. On some recursive residual rank tests for change-points. In M. Haseeb Rizvi, J. Rustagi, and D. Siegmund, editors, *Recent Advances in Statistics: Papers in Honor of Herman Chernoff on his Sixtieth Birthday*, pages 371–391. Academic Press, New York, 1983.

[Shachter and Kenley, 1989] Ross D. Shachter and C. Robert Kenley. Gaussian influence diagrams. *Management Science*, 35(5):527–550, 1989.

[Shachter and Peot, 1989] Ross D. Shachter and M. A. Peot. Simulation approaches to general probabilistic inference on belief networks. In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, Windsor, Ontario, 1989. Morgan Kauffman.

[Shachter *et al.*, 1990] Ross D. Shachter, Bruce D'Ambrosio, and Brendan A. Del Favero. Symbolic probabilistic inference in belief networks. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 126–131. MIT Press, 1990.

[Shachter *et al.*, 1991] Ross D. Shachter, Stig K. Anderson, and Peter Szolovits. The equivalence of exact methods for probabilistic inference on belief networks. Technical report, Dept. of Engineering-Economic Systems, Stanford University, Stanford, California, 1991.

[Shachter *et al.*, 1994] Ross D. Shachter, Stig K. Anderson, and Peter Szolovits. Global conditioning for probabilistic inference in belief networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pages 514–522, 1994.

[Shachter, 1986] Ross D. Shachter. Evaluating influence diagrams. *Operations Research*, 33(6), 1986.

[Shachter, 1990] Ross D. Shachter. Evidence absorption and propagation through evidence reversals. In M. Henrion, R. D. Shachter, L. N. Kanal, and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence 5*, pages 173–190. Elsevier Science, Amsterdam, 1990.

[Shachter, 1991] Ross D. Shachter. A graph-based inference method for conditional independence. In *Uncertainty in Artificial Intelligence: Proceedings of the Seventh Conference*, pages 353–360. Morgan Kaufmann, 1991.

[Shafer and Pearl, 1990] Glenn Shafer and Judea Pearl. Introduction. In *Readings in Uncertain Reasoning*, chapter Chapter 1, pages 1–6. Morgan Kaufmann, 1990.

[Shafer *et al.*, 1987] Glenn Shafer, Prakash P. Shenoy, and Khaled Mellouli. Propagating belief functions in qualitative Markov trees. *International Journal of Approximate Reasoning*, 1:349–400, 1987.

[Shenoy and Shafer, 1986] Prakash P. Shenoy and Glenn Shafer. Propagating belief functions using local computations. *IEEE Expert*, 1(3):43–52, 1986.

[Shenoy and Shafer, 1990] Prakash P. Shenoy and Glenn Shafer. Axioms for probability and belief-function propagation. In R.D. Shachter, T.S. Levitt, L.N. Kanal, and J.F. Lemmer, editors, *Uncertainty in Artificial Intelligence 4*. Elsevier Science Publishers B.V., North-Holland, 1990.

[Shenoy, 1992] Prakash P. Shenoy. Valuation-based systems for Bayesian decision analysis. *Operations Research*, 40(3):463–484, 1992.

[Shimony, 1994] Solomon Eyal Shimony. Finding MAPs for belief networks is NP-hard. *Artificial Intelligence*, 68(2):399–410, August 1994.

[Shortliffe and Buchanan, 1975] Edward H. Shortliffe and Bruce G. Buchanan. A model of inexact reasoning in medicine. *Mathematical Biosciences*, 23:351–379, 1975. Reprinted in [Buchanan and Shortliffe, 1984].

[Shortliffe, 1976] Edward H. Shortliffe. *Computer-based Medical Consultation: MYCIN*. Elsevier, New York, 1976.

[Simon, 1952a] Herbert A. Simon. Actions, consequences, and causal relations. *The Review of Economics and Statistics*, 34:305–314, 1952.

[Simon, 1952b] Herbert A. Simon. On the definition of the causal relation. *The Journal of Philosophy*, 49:517–528, 1952.

[Simon, 1955] Herbert A. Simon. A behavioral model of rational choice. *Quarterly Journal of Economics*, 69:99–118, 1955.

[Smith and Roberts, 1993] A. F. M. Smith and G. O. Roberts. Bayesian computation via the Gibbs sampler and related Markov chian Monte Carlo methods. *Journal of the Royal Statistical Society, Series B*, 55(1):3–23, 1993.

[Smyth *et al.*, 1996] Padhraic Smyth, David Heckerman, and Michael Jordan. Probabilistic independence networks for hidden Markov probability models. Technical Report A.I. Memo No. 1565, C.B.C.L. Memo No. 132, M.I.T., February 1996.

[Smyth, 1994] Padhraic Smyth. Markov monitoring with unknown states. *IEEE Journal on Selected Areas in Communications: Special Issue on Intelligent Signal Processing for Communications*, December 1994.

[Speed, 1979] Terry P. Speed. A note on nearest-neighbour Gibbs and Markov probabilities. *Sankhyā Series A*, 41:184–197, 1979.

[Spiegelhalter, 1986] David J. Spiegelhalter. Probabilistic reasoning in predictive expert systems. In L. N. Kanal and J. F. Lemmer, editors, *Uncertainty in Artificial Intelligence*, pages 47–68. North Holland, Amsterdam, 1986.

[Stockman *et al.*, 1976] George C. Stockman, L. Kanal, and M. C. Kyle. Structural pattern recognition of carotid pulse waves using a general waveform parsing system. *Communications of the ACM*, 19(12):688–695, 1976.

[Stockman, 1982] George C. Stockman. Waveform parsing systems. In P. R. Krishnaiah and L.N. Kanal, editors, *Handbook of Statistics, Vol. 2*, pages 527–548. North-Holland, 1982.

[Suermondt and Cooper, 1990] H. Jacques Suermondt and Gregory F. Cooper. Probabilistic inference in multiply connected networks using loop cutsets. *International Journal of Approximate Reasoning*, 4:283–306, 1990.

[Szolovits and Pauker, 1978] Peter Szolovits and Stephen G. Pauker. Categorical and probabilistic reasoning in medical diagnosis. *Artificial Intelligence*, 11:115–144, 1978.

[Tanner and Wong, 1987] Martin A. Tanner and Wing Hung Wong. The calculation of posterior distributions by data augmentation. *Journal of the American Statistical Association*, 82(398):528–540, June 1987.

[Tessem, 1992] Bjørnar Tessem. Interval probability propagation. *International Journal of Approximate Reasoning*, pages 95–120, 1992.

[Tierney, 1994] Luke Tierney. Markov chains for exploring posterior distributions. *The Annals of Statistics*, 22, 1994.

[Vercoe, 1984] Barry Vercoe. The synthetic performer in the context of live performances. In *Proceedings of the International Computer Music Conference (ICMC)*, pages 19–202, 1984.

[Verma and Pearl, 1993] T.S. Verma and J. Pearl. Deciding morality of graphs is NP-complete. In *Uncertainty in Artificial Intelligence: Proceedings of the Ninth Conference*, pages 391–397. Morgan Kaufmann, 1993.

[Waibel *et al.*, 1989] Alexander Waibel, Toshiyuki Hanazawa, Geoffrey Hinton, Kiyohiro Shikano, and Kevin J. Lang. Phoneme recognition using time-delay neural networks. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 37(3):328–339, March 1989.

[Wallace and Boulton, 1968] C. S. Wallace and D. M. Boulton. An information measure for classification. *Computer Journal*, 11(2):185–194, 1968.

[Warner *et al.*, 1961] H. R. Warner, A. F. Toronto, L. G. Veasy, and R. Stephenson. A mathematical approach to medical diagnosis: Application to congenital heart disease. *Journal of the American Medical Association*, 177:177–183, 1961.

[Weigend *et al.*, 1995] Andreas S. Weigend, Morgan Mangeas, and Ashok N. Srivastava. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, 6:373–399, 1995.

[Wellman and Liu, 1994] Michael P. Wellman and Chao-Lin Liu. State-space abstraction for anytime evaluation of probabilistic networks. In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, San Mateo, Calif., 1994. Morgan Kaufmann.

[Wellman *et al.*, 1992] Michael P. Wellman, John S. Breese, and Robert P. Goldman. From knowledge bases to decision models. *Knowledge Engineering Review*, 7(1):35–53, March 1992.

[Wellman, 1990] Michael P. Wellman. *Formulation of Tradeoffs in Planning Under Uncertainty.* Pitman and Morgan Kaufmann, 1990.

[Whittaker, 1990] Joe Whittaker. *Graphical Models in Applied Multivariate Statistics.* Wiley, 1990.

[Witkin, 1983] Andrew P. Witkin. Scale-space filtering. In *Proc. Eighth International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

[Wright, 1921] Sewall Wright. Correlation and causation. *Journal of Agricultural Research*, 20:557–585, 1921.

[Wright, 1934] Sewall Wright. The method of path coefficients. *Annals of Mathematical Statistics*, 5:161–215, 1934.

[Xiang *et al.*, 1992] Yang Xiang, David Poole, and Michael P. Beddoes. Exploring localization in Bayesian networks for large expert systems. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 344–351. Morgan Kaufmann, 1992.

[York, 1992] Jeremy York. Use of the Gibbs sampler in expert systems. *Artificial Intelligence*, 56:115–130, 1992.

[Zacks, 1983] S. Zacks. Survey of classical and Bayesian approaches to the change-point problem: Fixed sample and sequential procedures of testing and estimation. In M. Haseeb Rizvi, J. Rustagi, and D. Siegmund, editors, *Recent Advances in Statistics: Papers in Honor of Herman Chernoff on his Sixtieth Birthday*, pages 245–269. Academic Press, New York, 1983.

[Zhang and Poole, 1992] N. Lianwen Zhang and David Poole. Sidestepping the triangulation problem in Bayesian net computations. In *Uncertainty in Artificial Intelligence: Proceedings of the Eighth Conference*, pages 360–367. Morgan Kaufmann, 1992.