EXTENDING INFERENCE IN CONTINUOUS TIME BAYESIAN NETWORKS

by

Liessman Eric Sturlaugson

A dissertation proposal submitted in partial fulfillment
of the requirements for the degree

of

Doctor of Philosophy

in

Computer Science

MONTANA STATE UNIVERSITY
Bozeman, Montana

October, 2013

TABLE OF CONTENTS

TABLE OF CONTENTS – CONTINUED

ABSTRACT

The continuous time Bayesian network (CTBN) has been defined to enable reasoning about complex systems in continuous time by representing the system as a factored, finite-state, continuous-time Markov process. The dynamics of the CTBN are described by each node's conditional intensity matrices, determined by the states of the parents in the network.

As the CTBN is a relatively new model, many extensions that have been defined and researched with respect to Bayesian networks (BNs) have not yet been extended to CTBNs. The proposed research intends to address some of these.

First, we intend to formally prove several complexity results with respect to CTBNs. Specifically, it is known that exact inference in CTBN is NP-hard due to the use of a Bayesian network to set the initial states of the nodes. However, we propose to prove that exact inference in CTBNs is still NP-hard even when the initial states are fully observed. Furthermore, we suspect and intend to prove that approximate inference in CTBNs, as with Bayesian networks, is also NP-hard.

Second, we propose to formalize both uncertain and negative evidence in the context of CTBNs and extend existing inference algorithms to be able to support these new types of evidence.

Third, we show how methods for sensitivity analysis of Markov processes can be applied to the CTBN while exploiting the conditional independence structure of the network. This is done through what we call "node isolation," which approximates a nodes' unconditional intensity matrix, analogous to marginalization in a Bayesian network.

Lastly, we intend to investigate how and when the node isolation process might be used in approximate inference to increase efficiency without significantly decreasing accuracy. This prospectus reviews preliminary progress on these goals and outlines the direction of future research for the completion of this doctoral research.

CHAPTER 1
INTRODUCTION

This first chapter presents the motivation behind extending inference in continuous time Bayesian network and presents an overview of each of the subsequent chapters of this proposal.

1.1 Motivation

Temporal models serve an important role in many fields in which systems are observed to be changing through time. For example, doctors may use such models to track the likely progression of a disease and inform their prognosis. Similarly, maintenance engineers may use such models to track degradation of a system and predict its most likely time of failure. Security analysts may use temporal models to monitor systems and detect trends that indicate security anomalies, such as with intrusion detection systems. In all these cases, temporal models are used to reason about the most likely progression of change, given observations about the system, a process called inference.

However, reasoning about complex systems that are evolving in time is a difficult endeavor. As modeled systems become more complex, the computations necessary for inference become more difficult. The representation of time, whether discrete or continuous, is also a factor driving both the representational power of the model and the complexity of performing inference over the model. The continuous time Bayesian network (CTBN) is one such continuous-time model that attempts to simplify the representation of the system into interdependent subsystems, allowing modelers more control and flexibility over the complexity of the model.

The strength of a model lies in its ability to represent the system, represent observations about the system, and reason about the system given those observations. While several inference algorithms have been developed for CTBNs, there is still much room for improvement in the types of inference that the model can support and in the types of observations that the model can represent. This proposal lays out a number of valuable extensions to CTBN inference that make the model more useful, flexible, and applicable.

1.2 Overview

This section describes the organization of the remaining chapters of this proposal and gives a brief overview of the focus of each chapter.

In Chapter 2, we review the background work common to this entire prospectus. We start with a brief overview of types of temporal models. We then define discrete-state, continuous-time Markov chains, upon which CTBNs are built. Next, we formally define CTBNs and describe an example network. We then survey the literature describing the subsequent development of CTBNs, including inference algorithms and model learning algorithms, as well as CTBN extensions and application areas that have been researched to date. Each of the subsequent chapters focus on a different extension to the CTBN. As such, each chapter begins with additional background work relevant to its own proposed extension.

In Chapter 3, we start with a review of the theoretical work related to complexity of CTBNs. We propose to extend the theory of CTBNs with at least two new theorems, that exact inference in CTBNs is NP-hard and that approximate inference in CTBNs is also NP-hard. These proofs will draw on analogous complexity proofs for BNs, which we briefly review.

In Chapter 4, we start by reviewing the types of continuous-time evidence currently being used in CTBNs. The concepts of uncertain evidence and negative evidence, while defined for BNs, have not yet been extended to CTBNs. Furthermore, the temporal nature of the evidence leads to a rich variety of evidence that the CTBN can support. We propose to formalize and categorize these. The addition of uncertain and negative evidence also necessitates extensions to existing inference algorithms to be able to support these new types of evidence.

In Chapter 5, we extend sensitivity analysis to CTBNs. Whereas traditional inference in CTBNs calculates the probabilities given observations, sensitivity analysis measures how changes in the network parameters affect network performance. We show current work in applying perturbation realization, developed for sensitivity analysis of Markov processes, to the CTBN. For the CTBN, sensitivity analysis can be applied to different subnetworks individually through the technique we developed called node isolation.

In Chapter 6, we propose developing a new general approximate inference procedure for CTBNs that makes use of the node isolation technique. We present preliminary work on further developing the node isolation technique.

In Chapter 7, we lay out the dissertation plan, listing and describing the high-level goals necessary for completing this doctoral research.

In Chapter 8, we conclude with a survey of our past, present, and future publications related to this work and summarize the expected contributions resulting from this work.

CHAPTER 2
BACKGROUND WORK

This chapter presents the background work drawn on throughout the rest of the prospectus. While we focus on the CTBN, we start by a brief overview of other types of temporal models. We then focus on the background work necessary to understand CTBNs by reviewing Bayesian networks, dynamic Bayesian networks, and Markov processes. Then we formally define the CTBN, and finally we present an exhaustive survey of current literature for CTBN extensions and applications. Because each chapter focuses on different extensions to the CTBN, most chapters include additional background work relevant to their own topic.

2.1 Temporal Models

Many temporal models and their variations have been proposed for time-series analysis and forecasting. Early models focused on univariate signal processing. These include autoregressive integrated moving average (ARIMA) models, in which the output of a variable in the model maintains a linear dependence on the variable's previous values [1]. Filtering methods, such as Kalman filters [2, 3] and particle filters [4], can represent non-linear dynamical systems with continuous variables in continuous time and can be used for online tracking and prediction. Markov chains have been used extensively for modeling and as a means to reason about discrete-state, discrete-time stochastic processes, while Markov processes extend this idea to represent continuous-time Markov chains [5]. Hidden Markov models (HMMs) [6] and dynamic Bayesian networks (DBNs) [7] build upon Markov chains and are also used extensively for sequences of observations within in a Bayesian framework. Recurrent neural networks

extend the original neural network formulation to reason about sequences of model input and output [8]. A relatively recent temporal model, continuous time Bayesian networks, builds upon Markov processes and Bayesian networks to represent discrete variables in continuous time [9].

While these represent a survey of temporal models, each model is suited for different purposes and exhibits strengths and weaknesses. Some of the distinguishing features of the CTBN are that it is a multivariate, discrete-state, and continuous-time model. As with a BN, the graphical structure can provide an intuitive interpretation of interactions between variables, and the parameters are also relatively easily understood, consisting of expected times within states and state transition probabilities. Thus CTBNs can, and have been, constructed manually by domain experts. Lastly, according to the No Free Lunch theorems [10, 11], no particular model can be expected to dominate all other models for all problem domains. That is, while other temporal models will be superior to the CTBN on some problems, the CTBN will still be superior on others, and we expect at least some of these to be significant, real-world problems. Therefore, further research and development of the CTBN is a valuable contribution to the state-of-the-art in temporal modeling and reasoning.

2.2 Bayesian Networks

Bayesian networks are probabilistic graphical models that use nodes and arcs in a directed acyclic graph to represent a joint probability distribution over a set of variables [12].

Definition 1. Let $P(\mathbf{X})$ be a joint probability distribution over n variables $X_1, \ldots, X_n \in \mathbf{X}$. A Bayesian network $\mathcal{B}$ is a directed, acyclic graph in which each variable X_i is represented by a node in the graph. Let $\mathbf{Pa}(X_i)$ denote the parents of node X_i in the

graph. The graph representation of $\mathcal{B}$ factors the joint probability distribution as:

$$P(\mathbf{X}) = \prod_{i=1}^{n} P(X_i|\mathbf{Pa}(X_i)).$$

Without any factorization, the size of the full joint probability distribution is exponential in the number of variables and number of states for these variables. By factoring the joint probability distribution into only the relevant variable interactions, represented by the parent-child relationships in the network, the complexity of the network can be better managed.

2.3 Dynamic Bayesian Networks

The Bayesian network defined above is a static model. However, we can introduce the concept of time into the network by assigning discrete timesteps to the nodes to create a dynamic Bayesian network, a temporal version of a BN.

Definition 2. A dynamic Bayesian network (DBN) is a special type of Bayesian network that uses a series of connected time-slices, each of which contains a copy of a regular Bayesian network $\mathbf{X}_t$ indexed by time t. The probability distribution of a variable at a given time-slice can be conditionally dependent on states of that variable (or even other variables) throughout any number of previous timesteps. In first-order DBNs, the nodes in each time-slice are not conditionally dependent on any nodes further back than the immediately previous time-slice. Therefore, the joint probability distribution for a first-order DBN factors as:

$$P(\mathbf{X}_0, \ldots, \mathbf{X}_k) = P(\mathbf{X}_0) \prod_{t=0}^{k} P(\mathbf{X}_{t+1}|\mathbf{X}_t).$$

Spanning multiple time-slices, the DBN can include any evidence gathered throughout that time and use it to help reason about state probability distributions across different time-slices. Often, the conditional probability tables of the DBN can be defined compactly by defining a prior network $\mathbf{X}_0$ and a single temporal network $\mathbf{X}_t$. The temporal network $\mathbf{X}_t$ is then "unrolled" into $\mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_k$ for k time-slices.

The DBN model has been used in many application areas that have sequences of observations. Often these problems are concerned with performing classification only in the current timestep based on prior evidence. However, DBNs can also be unrolled further and forecast future states based on the evolving posterior probabilities. Some predictive tasks that the DBN has been used for include clinical prognostics [13, 14] and mechanical prognostics [15, 16, 17, 18].

2.4 Markov Processes

Although its name attempts to draw parallels between the conditional independence encoded by BNs, the CTBN is functionally a factored Markov process. There are variations and extensions of the Markov process model, but the CTBN model uses the model described here. We refer to a finite-state, continuous-time Markov chain as a Markov process. In a Markov process, a system is comprised of a discrete set of states, and the system probabilistically transitions between these states. The difference between a Markov process and a Markov chain is that each transition occurs after a real-valued, exponentially distributed sojourn time, which is the time it remains in a state before transitioning. The parameters determining the sojourn times and the transition probabilities are encoded in what is called an "intensity matrix." If the intensity matrix is constant throughout the lifetime of the system, we

refer to the Markov process as "homogeneous." Formally, we define a Markov process as follows.

Definition 3. A finite-state, continuous-time, homogeneous Markov process X with a state space of size n indexed by $\sigma = \{1, 2, \ldots, n\}$ is defined by an initial probability distribution P_X^0 over the n states and an $n \times n$ transition intensity matrix $\mathbf{A}_X$, in which each entry $a_{i,j} \geq 0$, $i \neq j$ gives the transition intensity of the process moving from state i to state j, and each entry $a_{i,i} \leq 0$ is the parameter for an exponential distribution, determining the sojourn times for the process to remain in state i.

With the diagonal entries constrained to be non-positive, the probability density function for the process remaining in state i is given by $\exp(a_{i,i}t)$, with t being the amount of time spent in state i, making the probability of remaining in a state decrease exponentially with respect to time. The expected sojourn time for state i is $1/|a_{i,i}|$. Each row is constrained to sum to zero, $\sum_j a_{i,j} = 0 \ \forall \ i$, meaning that the transition probabilities from state i can be calculated as $a_{i,j}/|a_{i,i}| \ \forall \ j, \ i \neq j$. Because the sojourn time uses the exponential distribution, which is "memory-less," the Markov process model exhibits the Markov property, namely, that all future states of the process are independent of all past states of the process given its present state.

Note that the model does not specify a particular time unit for the sojourn times. The modeler is responsible for choosing an appropriate time-scale (minutes, hours, days, etc.) for the specific application. The parameters are then set and interpreted accordingly.

Note also that a diagonal parameter can be set to 0, in which case the system never leaves the state once it reaches it. Such a state is called an absorbing state. If, however, at every point in time there is a non-zero probability of reaching every state from any other state at some time in the future, the Markov process is said to

be ergodic. Ergodicity plays a role in our approach to CTBN sensitivity analysis, as well as in questions of complexity of CTBN inference when constraints are placed on the parameters of the CTBN.

2.5 Continuous Time Bayesian Networks

The CTBN[1] was first introduced in [9] and then further developed in [20]. The model definition is as follows.

Definition 4. Let $\mathbf{X}$ be a set of Markov processes $\{X_1, X_2, \ldots, X_n\}$, where each process X_i has a finite number of discrete states. Formally, a continuous time Bayesian network $\mathcal{N}$ over $\mathbf{X}$ consists of two components. The first is an initial distribution denoted $P^0_{\mathbf{X}}$ over $\mathbf{X}$, which can be specified as a Bayesian network $\mathcal{B}$. This distribution $P^0_{\mathbf{X}}$ is only used for determining the initial state of the process. The second is a continuous-time transition model, which describes the evolution of the process from its initial distribution, specified as:

- A directed graph $\mathcal{G}$ with nodes $X_1, X_2, \ldots, X_n$, where $\mathbf{Pa}(X_i)$ denotes the parents of X_i in $\mathcal{G}$,
- A set of conditional intensity matrices (CIMs) $\mathbf{A}_{X|\mathbf{Pa}(X)}$ associated with X for each possible state instantiation of $\mathbf{Pa}(X)$.

Figure 2.1 shows an example CTBN from [9]. The initial distribution and intensity matrices for all the nodes can be found in [21]. Each child node has multiple intensity matrices, one for each combination of states of its parent nodes. For example, the

[1] Although identically named, the CTBN of [19] is *not* the same model as discussed in this prospectus.

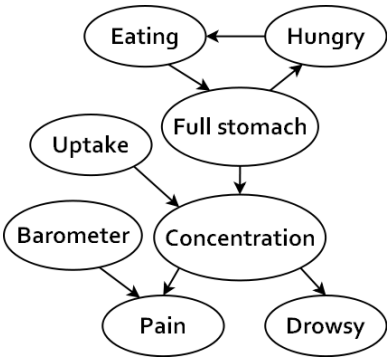

Figure 2.1: Example CTBN.

matrix denoted $C|u_1, f_0$ defines the dynamics of the node Concentration given that the state of Uptake is u_1 and that the state of Full stomach is f_0.

This model could be used to answer several interesting queries. For example, what is the expected proportion of time that the patient is in pain while drowsy? Or, given that the patient is initially in pain but that uptake occurred at time t_1 and that the patient finished eating at time t_2, what is the expected amount of time until the patient is not in pain? Or, given that the patient has been in pain from time t_2 to t_3, what is the expected number of transitions between the concentration levels that occurred during that time period?

2.5.1 DBNs vs. CTBNs

Note that, unlike a regular Bayesian network, cycles are allowed in $\mathcal{G}$, because the nodes represent subsystems of a single Markov process, and therefore only one state transition is allowed at a time for the whole CTBN. A cycle in a CTBN model would be analogous to a dynamic Bayesian network with variables X and Y where arcs such as $X_t \rightarrow Y_{t+1}$ and $Y_t \rightarrow X_{t+1}$ would be valid.

Similar to the Bayesian network, the conditional dependencies allow a more compact representation for the model. Like a Bayesian network, the local conditionally

dependent probability tables can be combined to form the full joint probability distribution. In the case of the continuous-time Bayesian network, this is called the full joint intensity matrix, which describes the evolution of the entire process. However, just as in the Bayesian network, in which the number of entries in the full joint probability distribution grows exponentially in the number of variables, so too the number of states in the full joint intensity matrix grows exponentially in the number of variables for the CTBN.

Despite these few similarities, the CTBN model is fundamentally different from the DBN model. Although the network topologies for both models encode conditional dependence, the models are differentiated by what the nodes represent. Whereas the nodes in a DBN are simple random variables, the nodes in a CTBN are conditional Markov processes. As a result, CTBNs can be queried about the state probabilities for any real-valued time. A DBN, unrolled for a discrete number of timesteps, can only be queried for state probabilities at these timesteps but not in-between adjacent timesteps. While the time interval between timesteps can be set with finer granularity, doing so multiplies the number of nodes needed to span the same amount of time as the original unrolled DBN and will never be fully continuous with respect to time.

2.5.2 CTBN Algorithms

The only exact inference algorithm that exists so far for CTBNs simply expands and works with the full joint intensity matrix, which is exponential in the number of nodes and the number of states [22]. However, this inference algorithm does not take advantage of the factored nature of the network. Thus, research has focused on approximate methods.

There have been a number of sample-based inference algorithms developed for CTBNs, such as importance sampling [22, 23, 24] and Gibbs sampling [25, 26, 27]. In

importance sampling, multiple samples are generated that are constrained to conform to the evidence. The particles are then weighted by their likelihood. Gibbs sampling, on the hand, takes a Markov Chain Monte Carlo (MCMC) approach. For each variable over each interval of evidence, the surrounding states (the node's parents, children, and children's parents) are held constant and a random walk is performed on the state of the node. The idea is that, on each interval of evidence, the distribution of the random walk will converge to the true distribution. Methods for expectation propagation [28, 29, 30] has also been developed. In this method, for each interval of evidence, the nodes employ a message-passing scheme with their neighbors. The idea is to pass approximate "marginals" which are unconditional intensity matrices, valid for that interval, until all of the nodes have a consistent distribution over that interval. Most recently, mean-field approximation [31, 32] have also been developed, which propagate the product of inhomogeneous Markov processes to approximate the distribution through a system of ordinary differential equations. There have also been approaches specifically for continuous-time particle filtering in CTBNs [33, 34] which sample forward in time, without having to be conditioned on upcoming evidence.

As a data-driven model, algorithms have been developed for model learning, both with learning the network structure [35], as well as learning and maturing the model parameters [36]. These define the sufficient statistics for the nodes of the CTBN and define the log-likelihood that can be used to measure how well the network structure and the parameters match the data. They also support learning in the presence of missing data, showing expectation maximization algorithms for CTBNs.

2.5.3 CTBN Applications and Extensions

CTBNs have found use in several applications. For example, CTBNs have been used for inferring users' presence, activity, and availability over time [37]; robot

monitoring [34]; modeling server farm failures [38]; modeling social network dynamics [39, 40]; modeling sensor networks [41]; building intrusion detection systems [42, 43, 44]; predicting the trajectory of moving objects [45]; and diagnosing cardiogenic heart failure and anticipating its likely evolution [46, 47]. The CTBN has also been extended to support decision-making, resulting in structured continuous-time Markov decision processes [48].

The CTBN model has also undergone several specializations and generalizations. The Generalized CTBN (GCTBN) of [49] combines the conditional probability tables of BNs and the conditional intensity matrices CTBNs, allowing nodes to be either what they call "delayed" variables or "immediate" variables. They show how inference can be performed when combining the conditional probabilities of the immediate nodes with the intensity matrices of the delayed nodes. The CTBN classifier (CTBNC) of [50, 51] is an instance of the GCTBN, adding a parent-less immediate class node, with marginal probabilities over the class label, for classifying a static object given continuous-time evidence about that object. The work of [52] changes the representation of the CTBN to be partition-based, using what they call conditional intensity trees and conditional intensity forests. This is analogous to representing the conditional probabilities in BNs as decision trees [53]. The Erlang-Coxian CTBN (EC-CTBN) of [54] replaces the exponential distribution of the sojourn times with Erlang-Coxian distributions. In [20], they had shown how combinations of nodes in the CTBN could represent Erlang-Coxian distributions [35], but the EC-CTBN replaces this with a single node and introduced a generalized conditional intensity matrix. Lastly, the asynchronous dynamic Bayesian network (ADBN) of [55] maintains a CTBN, converting the conditional intensity matrices to conditional probability tables to perform inference. The idea is that the parameters of the DBN will be populated from the continuous-time evidence for the CTBN, avoiding the assumption of

a uniform interval of time between all the timesteps of the DBN. After converting the conditional intensity matrices to conditional probability tables, inference can be performed over the DBN instead of the CTBN.

CHAPTER 3
COMPLEXITY OF INFERENCE

This chapter reviews the current state of complexity theory specific to the CTBN and poses several theorems for proof.

3.1 Background Work

Currently, little theoretical work has been done on the complexity of inference in CTBNs. Most of what is known about the complexity of CTBNs is derived from the fact that a Bayesian network is used to specify the initial distribution.

One complexity result specific to CTBNs arises from the difference between BN and CTBN structure learning. In structure learning, it is common to assign a scoring function to arcs in the network that quantifies how well the network topology matches the training data. Nodelman gives a polynomial-time algorithm for finding the highest-scoring set of k parents for a CTBN node [20]. The corresponding problem in a Bayesian network has been shown to be NP-hard, even for $k = 2$, due to the acyclic constraint of Bayesian networks [56]. Essentially, because cycles are allowed in CTBNs, each node can maximize its score independently. However, cycles introduce difficulties elsewhere, because the complete behavior of a node depends on all of its ancestors, which becomes the entire cycle at the very least.

Because the CTBN is relatively new, much of the complexity theory surrounding CTBNs has yet to be fully explored. Because we are focusing on CTBN inference, this proposal intends to expand the complexity theory of CTBN inference, under various assumptions and conditions. The work can build on the complexity results of BNs, which we now review.

It is known that exact inference in BNs is NP-hard via a reduction from 3SAT [57]. The 3SAT problem consists of a set of clauses $C = \{c_1, c_2, \ldots, c_m\}$ made up of a finite set U of n Boolean variables. Each clause contains a disjunction of three literals over U, for example, $c_3 = (u_2 \wedge \neg u_3 \wedge u_4)$. The 3SAT problem is determining whether there exists a truth assignment for U such that all the clauses in C are satisfied.

The 3SAT problem can be reduced to a Bayesian network decision problem of whether, for a $True(T)/False(F)$ node X in the network, $P(X = T) > 0$ or $P(Y = T) = 0$. We can represent any 3SAT instance by a Bayesian network as follows. For each Boolean variable u_i in U, we add a corresponding $True(T)/False(F)$ node U_i to the network such that $P(U_i = T) = \frac{1}{2}$ and $P(U_i = F) = \frac{1}{2}$. For each clause C_j, we add a corresponding $True(T)/False(F)$ node C_j to the network as a child of the three nodes corresponding to its three Boolean variables. Let w_j be the clause corresponding to the state of the three parents of C_j, and let $eval(w_j)$ be the truth function for this clause. The conditional probabilities of the node are

$$\begin{cases} P(C_j = T|w_j) = 1 & \text{if } eval(w_j) = T \\ 0 & \text{if } eval(w_j) = F \end{cases}$$

Finally, for each clause C_k, we add a $True(T)/False(F)$ node D_k. Each D_k is conditionally dependent on C_k and on D_{k-1} (except for D_1). The conditional probabilities for D_1 are $P(D_1 = T|C_1) = 1$ if and only if $C_1 = T$ and 0 otherwise. Similarly, the conditional probabilities for D_k $(k > 1)$ are $P(D_k = T|C_k, D_{k-1}) = 1$ if and only if $C_k = T$ and $D_{k-1} = T$ and 0 otherwise.

Essentially, the C and D nodes enforce the logical relations of the clauses using the representation of a Bayesian network. The probabilities of the U nodes allow for the choice of truth assignments to the Boolean variables. As such, the 3SAT instance

is satisfiable if and only if $P(D_m = T) > 0$, that is, if and only if there is a non-zero probability that some instantiation of the U nodes to T and F will cause all of the clauses to be satisfied. Thus, if an algorithm exists that is able to efficiently compute the exact probabilities in arbitrary Bayesian networks, the algorithm can efficiently decide whether $P(D_m = T) > 0$ for the specially constructed networks that can represent arbitrary instances of 3SAT.

Furthermore, it is known that even absolute and relative approximations in BNs is NP-hard [58]. We formally define these approximations as follows. Suppose we have a real value ϵ between 0 and 1, a BN with binary-valued nodes V, and two nodes X and Y in V instantiated to x and y, respectively.

Definition 5. A relative approximation is an estimate $0 \leq Z \leq 1$ such that

$$\frac{P(X = x|Y = y)}{(1 + \epsilon)} \leq Z \leq P(X = x|Y = y)(1 + \epsilon).$$

Definition 6. An absolute approximation is an estimate $0 \leq Z \leq 1$ such that

$$P(X = x|Y = y) - \epsilon \leq Z \leq P(X = x|Y = y) + \epsilon.$$

The proof of NP-hardness for relative approximation follows from the proof for exact inference. Satisfiability of the clause is determined whether $Z = 0$ or $Z > 0$, which is not influenced by the choice of ϵ. Thus, there is no constant-factor approximation for inference in BNs.

The proof of NP-hardness for absolute approximation starts with the reduction for exact inference as above, representing the variables and clauses with the same network and parameters. This time, one by one a truth assignment is set for each of the Boolean variable u_i, and the corresponding node U_i is removed from the network. The

truth assignment for u_i is determined by the higher probability of $P(U_i = T|D_m = T)$ and $P(U_i = F|D_m = T)$. However, if there exists an efficient approximate BN inference algorithm that can guarantee to be within $\epsilon = \frac{1}{2}$ of the exact probability on arbitrary Bayesian networks, this algorithm can be used to efficiently determine satisfying truth assignments to all Boolean variables of an arbitrary instance of 3SAT.

3.2 Complexity of Exact Inference in CTBNs

Because the initial distribution determines the evolution of a CTBN, it follows that exact inference in CTBNs is at least NP-hard. However, the process of propagating probabilities through time in a CTBN is different from calculating the probabilities in a BN. Thus, an interesting question becomes whether exact inference in a CTBN is still NP-hard, given the initial states. That is, is the complexity of CTBN inference a result of the computing the initial state or is the propagation of probabilities through time itself a difficult problem? The straightforward approach to inference expands the CTBN to the full joint intensity matrix, which is exponential in the number of nodes and the number of states of each node, and is thus intractable. However, can we expect to find an alternative approach to exact inference that avoids working with the full joint intensity matrix and computes the probabilities in polynomial time given the exact initial states? We hypothesize and propose to show that exact inference in CTBNs is at least NP-hard, even when given the exact initial states, following a similar reduction as the proof for BNs but replacing the conditional probability distributions with conditional intensity matrices.

3.3 Complexity of Approximate Inference in CTBNs

Because even approximate inference in BNs is NP-hard, it seems reasonable to suspect that a similar conclusion also holds for approximate inference in CTBNs. Here, however, it is not clear how the NP-hardness of the approximation of the initial distribution impacts the approximations of the probabilities propagated through time. In this case, we intend to show that approximate inference in CTBNs is also NP-hard, even when given the exact initial state. We can attempt similar reasoning to the proof regarding approximate inference in BNs—successively removing U nodes from the network as the approximation algorithm determines a truth assignment for each node. It is unclear at this point, however, how the approximation bound will carry over to CTBN inference or how the temporal aspect of the computed probabilities will need to dictate the truth assignments.

3.4 Preliminary Results

As preliminary work, we present a proof sketch that exact inference in CTBNs is at least NP-hard, even when given the exact initial states. The CTBN topology would match that of the BN for representing variables and clauses. The difference is that in our case we use conditional intensity matrices instead of the conditional probability tables of BNs. Figure 3.1 shows the CTBN for determining the satisfiability of the clause $(u_1 \vee u_2 \vee u_3) \wedge (\neg u_1 \vee \neg u_2 \vee u_3) \wedge (u_2 \vee \neg u_3 \vee u_4)$. The U nodes would start in a dummy third state S and, after a finite sojourn time, be absorbed into a $True(T)$ or $False(F)$ state. The C and D nodes would start in the F state. They would have a finite sojourn time before being absorbed into the T state if and only if their parents were in states corresponding to a satisfying assignment of their respective clauses;

otherwise they would remain in the F state. Let $D_m(t)$ be the state of D_m at time t. The 3SAT instance would be satisfiable if and only if $P(D_m(t) = T) > 0$ for some time $t > 0$.

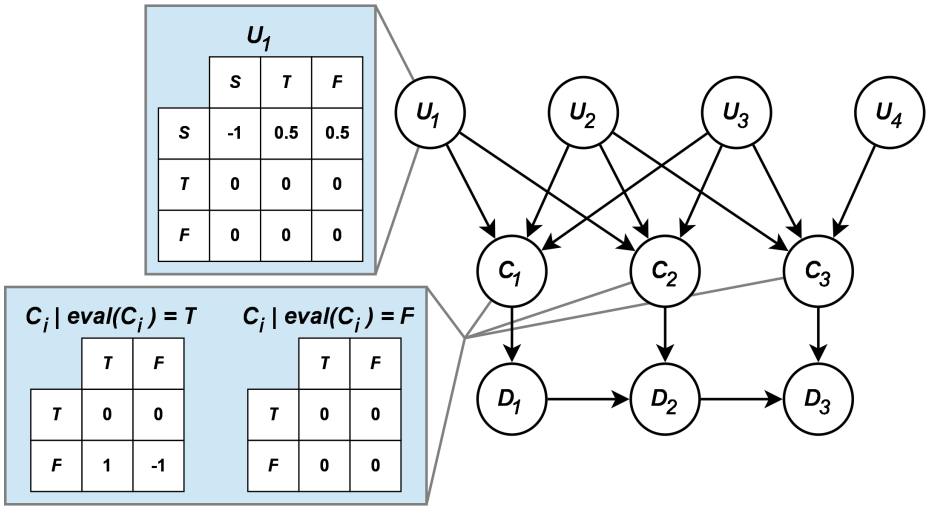

Figure 3.1: Example reduction from 3SAT to CTBN exact inference.

We note that our proposed proof relies on absorbing states in the nodes of the CTBN, setting $a_{i,i} = 0$ for some i. This is similar to allowing probabilities of 0 and 1 in conditional probability tables for BNs. Another question becomes whether we can avoid the NP-hardness by placing certain constraints on the parameters. The proofs in [58] include a variation in which they look at the complexity of inference in BNs when each probability p is constrained to be $l \leq p \leq u$ for some $l > 0$ and $u < 1$. Analogously for the CTBN, the question would be whether exact inference is NP-hard when making similar assumptions on the parameters, enforcing non-zero parameters and ensuring node ergodicity.

Proving the complexity of exact and approximate inference in CTBNs under different conditions will provide further insights into the complexity of working with these models and possibly suggest to modelers ways the complexity of specific models

can be better managed. It may serve as a step toward special-case CTBNs that can be proved to be tractable. Even though exact and approximate inference in BNs is known to be NP-hard, practical algorithms still exist that are able to perform inference efficiently for many networks. Similarly, inference in CTBNs is an ongoing, active, and fruitful area of research.

CHAPTER 4

CONTINUOUS-TIME EVIDENCE

Probabilistic models, such as Bayesian networks, provide a mathematically rigorous framework for reasoning under uncertainty. Given observations (evidence) about the system the network represents, the network can update the posterior probabilities of other states in light of that evidence. However, the representation of uncertainty in Bayesian networks can and has been extended further to allow for uncertainty in the evidence as well, such as with soft evidence or virtual evidence, in which there is uncertainty of the observations themselves. Uncertain evidence provides a generalization of evidence, in which the evidence also has degrees of uncertainty associated with it.

As a relatively new model, the CTBN currently only supports "crisp" evidence, in which all of the temporal state evidence is trusted with complete confidence. However, in many applications, such as fault prognosis in which we rely on instrument measurements, the evidence may contain noise or errors and can be trusted only to a certain degree. Uncertain evidence for CTBNs has not yet been defined, and algorithms have not been extended to allow for incorporating the uncertainty of temporal evidence. The inclusion of continuous time into the evidence gives rise to subtleties in negative evidence, as well. Therefore, we propose to define both uncertain and negative evidence and show how to support these definitions in the context of CTBN inference.

4.1 Background Work

This section briefly describes existing techniques for handling uncertain evidence in the context of Bayesian networks and describes why these do not carry over to the CTBN. In BNs, uncertain evidence is when the state of a node is only known with a given probability.

One approach for uncertain evidence in Bayesian networks is called virtual evidence, as described in [59]. To represent this type of uncertain evidence, an additional node is added as a child to an observation node. This child node then becomes the node that is observed, and the conditional probabilities of this child are set to match the strength of the evidence. In effect, virtual evidence sets up a ratio of likelihoods to represent the the confidence of an observation observing a particular state. Another approach is soft evidence, as differentiated from virtual evidence in [60, 61]. Soft evidence changes the marginal distribution of the evidence itself, using Jeffrey's rule as a generalization of conditioning on observed variables to condition on an observation of a probability distribution, in which the observed distribution holds the uncertainty of the observations.

While the name CTBN casts the model as a special type of Bayesian network, the underlying model is actually quite different from that of a Bayesian network. Thus, the methods for including uncertain evidence in Bayesian networks do not translate over to CTBNs. No literature for uncertain evidence has yet been found in the context of continuous-time Markov processes or CTBNs. In these cases, the evidence is temporal, containing state information over a real-valued period of time, which prohibits the uncertain evidence techniques of Bayesian networks (even dynamic Bayesian networks) from being applied directly. Nevertheless, definitions for

uncertain continuous-time evidence should be consistent with the extensive research in probabilistic inference with uncertain evidence [62].

4.2 Uncertain and Negative Evidence in CTBNs

First, we need a formal definition for uncertain evidence over continuous time. For contrast, we first review definitions for crisp evidence over continuous time. There are three types of crisp evidence currently in use by CTBNs. They are point, interval, and transition evidence.

Definition 7. Crisp point evidence in a CTBN is "at an instantaneous point in real-valued time t, the state of node X was observed to be in state x."

Definition 8. Crisp interval evidence in a CTBN is "on a real-valued interval of time $[t_0, t_1)$, the state of node X was observed to be in state x."

Definition 9. Lastly, crisp transition evidence in a CTBN is "at an instantaneous point in real-valued time t, the state of node X was observed to change from state x_i to state x_j."

Note that interval evidence is able to represent the other two types of evidence by setting the interval as Δt for some infinitesimal value. For transition evidence, this becomes two successive instances of infinitesimally short interval evidence: "on a real-valued interval of time $[t - \Delta t, t)$, the state of node X was observed to be in state x_i, while on a real-valued interval of time $[t, t + \Delta t)$, the state of node X was observed to be in state x_j."

We introduce definitions for uncertain interval evidence, temporally uncertain transition evidence, and negative interval evidence.

Definition 10. Uncertain interval evidence in a CTBN is "on a real-valued interval of time $[t_0, t_1)$ the state of node X is known to be in exactly one state over the entire interval, but the identity of that state is only known with some probability."

For example, from time t_0 to time t_1, it might be known that node X was in state x_0 with 75% probability and state x_1 with 25% probability. As mentioned above, uncertain interval evidence extends to uncertain evidence for points and transitions. Note that the uncertainty is in the identity of the state. However, we might also have uncertainty in the timing of an observation, such as the time of a transition.

Definition 11. Temporally uncertain transition evidence in a CTBN is "sometime during a real-valued interval of time $[t_0, t_1)$, the state of node X was observed to change from state x_i to state x_j.

Definition 12. Negative interval evidence in a CTBN is "on a real-valued interval of time $[t_0, t_1)$ the state of node X was observed to *not* be in state x."

Note that in CTBNs, uncertain evidence is different from negative evidence. Uncertain interval evidence, for example, says that the system was in different states with different probabilities, but whichever it was, the system stayed in that state over the whole interval. Negative evidence is saying something different. Negative interval evidence, for example, says that the system was never in a certain state over an interval. However, the system could have experienced multiple transitions between the other states over that interval. Furthermore, the two types are not mutually exclusive: we could have also uncertainty in our negative evidence. The proposed research intends to enumerate, formalize, and categorize the types of evidence, their uncertain/negative combinations, and extend CTBN inference algorithms to be able to support them.

4.3 Preliminary Results

In this section we present preliminary work on representing and reasoning with uncertain point and interval evidence. First, we examined and ruled out several approaches that attempted to incorporate uncertain evidence through modifications to the network topology, as with virtual evidence in BNs [59]. Because all the types of evidence are time-dependent and because nodes do not have a mechanism for incorporating specific times (as with timesteps in a DBN), we reason that the inference algorithm itself must be modified to handle uncertain and negative evidence.

The CTBN used in these preliminary experiments is shown in Figure 4.1. The prior probabilities of the states (when $t = 0$) are uniformly distributed for both nodes. The time units for this network are arbitrary, while the diagonal sojourn parameters are scaled to represent an order of magnitude difference in the conditional intensity matrix of the child node. Note that, without any evidence, the expected proportion of time spent in each state of A is $33.\overline{3}\%$, while the expected proportion of time in each state of B is 50%.

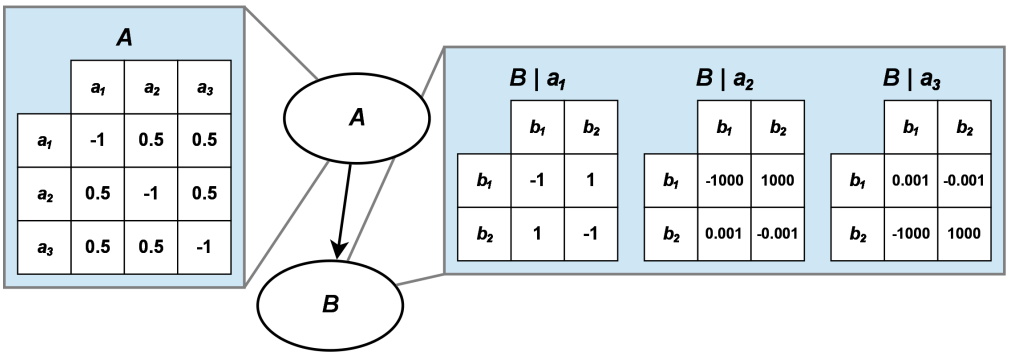

Figure 4.1: Example CTBN to demonstrate uncertain interval evidence.

We are currently developing our own CTBN inference engine, but for this preliminary work, we used the Continuous Time Bayesian Network Reasoning and Learning

Engine (CTBN-RLE) [21] to run the experiments. We chose importance sampling as the algorithm to extend, as the fastest sampling algorithm for approximate inference provided by the CTBN-RLE code. The importance sampling algorithm was modified so that, when the algorithm chooses the next transition of the sample, it checks to see if there is uncertain evidence set. If so, it samples from the uncertain evidence as a multinomial, instead of just setting the transition state. The weights for these different transitions will be updated accordingly, because they are weighted based on the future state sampled. Thus, the proportion of samples with different transitions will follow the distribution of the uncertain evidence.

Our time interval over which we queried the CTBN was $[0, 10)$. We added uncertain point evidence at $t = 5$, stating that A was in a_2 with 50% probability and a_3 with 50% (thus, 0% probability for a_1). We performed importance sampling with this evidence on the network in Figure 4.1 for ten thousand samples, querying the expected amount of time spent in each of the three states of A. The results are shown in Table 4.1.

Table 4.1: Expected amount of time spent in each state of A with modifications to importance sampling.

State	**Time**
a_1	2.854
a_2	3.574
a_3	3.572

As mentioned above, the expected amount of time in each state would have been $3.\overline{3}$ without any evidence. If the point evidence had been one state with complete certainty, the amount of time in that state would have been the highest and the other two would be lower and equal. As shown by the table, the results follow our intuition,

Table 4.2: Expected amount of time spent in each state of B.

State	**Time (20%/80%/0%)**	**Time (0%/100%/0%)**
b_1	3.915	3.742
b_2	6.085	6.258

that states a_2 and a_3 are both likely to have been visited longer than a_1, knowing that each had a 50% chance of being the visited state at $t = 5$.

Lastly, we used the importance sampling algorithm to calculate the expected time spent in each state of B over the time interval $[0, 10)$, but with the uncertain evidence that, over the time interval $[5, 6)$, A was in a_1 with 20% probability and in a_2 with 80% probability. The longer that A stays in a_2, the longer that B stays in b_2. We compare with the crisp evidence that A was in a_2 with 100% probability over $[5, 6)$. Again, we generated ten thousand samples. We see from Table 4.2 that, when the evidence is more sure, the expected amount of time is longer, as would be expected given the network.

Continuous-time systems allow for much greater variety in the types of evidence that would be useful to know about the system. The research presented in this chapter is preliminary work on incorporating uncertain evidence into CTBNs. The next step entails a more thorough treatment of uncertain evidence and further validation of the approach, as well as introducing negative evidence. Extending CTBNs to support uncertain and negative evidence represents a novel and most likely useful extension that makes the model more generalized and versatile.

CHAPTER 5
SENSITIVITY ANALYSIS

Sensitivity analysis studies how variations in the input to a model affect the model's output, and is useful in several contexts. In modeling, for example, sensitivity analysis can aid in model design, debugging, and maturation. It can also be used to measure the robustness of model inference, the extent to which noise in the input affects model output, or can be used to run hypothetical scenarios. Extending sensitivity analysis to CTBNs enables one to test how changes to the network parameters affect the expected cost/reward per unit time of the modeled system.

Research has been done on sensitivity analysis for many probabilistic networks, such as Bayesian networks [63, 64], Markov chains [65, 66, 67], Markov processes [68, 69, 70, 71, 72], and queuing networks [73, 74, 75]. To our knowledge, we present the first methods for sensitivity analysis researched specifically for the CTBN.

5.1 Background Work

Our approach to sensitivity analysis in CTBNs borrows from a sample-based method for sensitivity analysis in Markov processes called perturbation realization [68]. In addition to perturbation realization, there are at least two other approaches to performing sensitivity analysis on Markov processes, namely, the likelihood ratio method and the reduced augmented chain method.

The likelihood ratio (LR) method [76, 77, 78] takes the ratio of the likelihood of a sample path from the original process with the likelihood of one that incorporates small changes into the transition rates. After this ratio is simplified and differentiated, it can be estimated using the number of transitions between states and the state

sojourn times, as taken from a sample path. This estimate is then used with the performance function to estimate the expected performance measure derivative. The LR method faces an inherent trade-off between variance in the estimator and bias in the estimate of the steady-state probabilities depending on the length of the sample path that the estimator is given. One benefit of using LR, however, is that the process yields confidence intervals on the performance measure derivatives without extra effort.

The reduced augmented chain (RAC) method [71, 67] also uses a single sample path, but instead of simulating the nominal process by itself, it first creates a combined process of both the nominal and the perturbed process, representing a superposition of nominal and perturbed states. Depending on the number of states that have been perturbed, this could represent a substantial increase in the number of states that must be simulated. On the other hand, the method does not rely on knowing the intensity matrix values, working instead with direct observation of the nominal system, and can be used for online estimation of the steady-state probability sensitivities.

There is another important feature trade-off between the RAC method and perturbation realization. For perturbation realization, multiple perturbations can be tested for a given sample path while the performance function remains fixed (otherwise the performance potentials would need to be re-estimated). With the RAC method, the performance function can be varied for a given reduced augmented chain. After varying the parameter perturbations, however, a new reduced augmented chain must be created and sampled.

Note that LR and RAC could also be incorporated into the following approach. Even so, we chose perturbation realization for use in our preliminary work. Our goal is efficient sensitivity analysis of CTBNs, and perturbation realization's ability to

calculate and retain potentials for different subnetworks is directly applicable to that end.

We now give an overview of perturbation realization. The method relies on a single run of the system, referred to as a sample path and defined as follows.

Definition 13. A sample path $\mathcal{S}$ is a complete description of a single instance of a Markov process over an interval of time. It starts with the initial state of the process and records the time of each subsequent transition, as well as the resulting state after each transition.

Let $\boldsymbol{\pi} = (\pi_1, \pi_2, \ldots, \pi_n)$ denote the row vector representing the steady-state probabilities of each state of the Markov process, and let f, a mapping of the state space of the Markov process to the real numbers, be a performance function of the process. This function is used to calculate the performance measure, defined as the function's expected value,

$$\eta = \sum_i \pi_i f(i) = \boldsymbol{\pi}\mathbf{f}, \tag{5.1}$$

where $\mathbf{f} = (f(1), f(2), \ldots, f(n))^T$ is a column vector and each entry is a state's performance function value. By themselves, the transition intensities of a Markov process do not encode state values, only transition probabilities. The performance function f, on the other hand, allows us to attach cost/reward values to the states. The performance measure represents the expected cost/reward per unit time of running the process. Furthermore, the performance measure gives direction for performing sensitivity analysis, because now we can measure how changes to the intensity matrix affect performance. We note that this performance function is different from the reward model of the structured continuous-time Markov decision processes of [48], which attaches cost/reward values to the actions made in each state.

Perturbation realization calculates the column vector of state performance "potentials," $\mathbf{g} = (g_1, g_2, \ldots, g_n)^T$. These potentials represent the expected value of being in each state, which is related to but distinct from the performance values. While each state may have its own performance value, some states may have higher potentials than others because they have a higher probability of transitioning to other states with high performance values. This potential vector gives a way to calculate the derivative of the performance measure, as shown in [68]:

$$\frac{\partial \eta}{\partial \mathbf{Q}} = \boldsymbol{\pi} \mathbf{Q} \mathbf{g}. \tag{5.2}$$

The perturbation matrix $\mathbf{Q}$ of Equation (5.2) is supplied by the user, but the other quantities must be calculated in order to determine $\frac{\partial \eta}{\partial \mathbf{Q}}$. Algorithms for computing $\boldsymbol{\pi}$ and $\mathbf{g}$ based on a single sample path, summarized below, are provided in [69].

Let T_k be the kth transition epoch of X_t (the time of the kth transition), S_k be its kth sojourn time (the time it remains in the kth state), and X_k be its state after the kth transition. The indicator function $I_i(X_k)$ is 1 if $X_k = i$ for state i and 0 otherwise. Then the steady-state probability π_i and potential g_i of each state i can be calculated from a single sample path of $N \to \infty$ transitions, with probability one, as:

$$\pi_i = \lim_{N \to \infty} \frac{1}{T_N} \left\{ \sum_{k=0}^{N-1} I_i(X_k) S_k \right\} \text{ and} \tag{5.3}$$

$$g_i = \lim_{N \to \infty} \frac{\sum_{k=0}^{N} \left\{ I_i(X_k) \int_{T_k}^{T_k + T} f(X_t) dt \right\}}{\sum_{k=0}^{N} I_i(X_k)}. \tag{5.4}$$

Equation (5.3) sums the amount of time spent in a state and divides by the total running time of the process, giving the steady-state probability of that state, which

becomes the expected value within the limit. Equation (5.4) averages the accumulated performance value of starting in each state in the sample path, giving the potential for that state within the limit. The parameter T is a properly chosen positive value controlling the amount of time used to estimate the potentials. We found that our heuristic of setting T equal to the longest sojourn time in the sample path gave consistently accurate results across the different networks and sample paths. The benefit of using Equation (5.2) is that once $\boldsymbol{\pi}$ and $\mathbf{g}$ are computed from a single sample path, $\frac{\partial \eta}{\partial \mathbf{Q}}$ can be calculated for any number of user-defined $\mathbf{Q}$ matrices.

Note that closed-form solutions exist for calculating $\boldsymbol{\pi}$ without having to estimate it from the sample path using Equation (5.3) [5]. Furthermore, we found that the accuracy of the $\frac{\partial \eta}{\partial \mathbf{Q}}$ estimates were greatly improved, especially for large perturbations, when using the steady-state probabilities of the *perturbed* intensity matrix rather than the original intensity matrix. The closed-form solution for the perturbed steady-state probabilities is therefore found by calculating the normalized left eigenvector of $\mathbf{A}_\delta$ corresponding to the zero eigenvalue,

$$\boldsymbol{\pi}\mathbf{A}_\delta = \mathbf{0}. \tag{5.5}$$

5.2 Sensitivity Analysis of CTBNs

Because a CTBN represents a Markov process (although in compact, factored form), the sample path method of perturbation realization for Markov processes is a natural candidate for CTBN sensitivity analysis. A straightforward application of perturbation realization to CTBNs would result in simply flattening the entire network into one large Markov process, with the states being the Cartesian product of all the nodes' states. But this fails to take advantage of the factored nature of

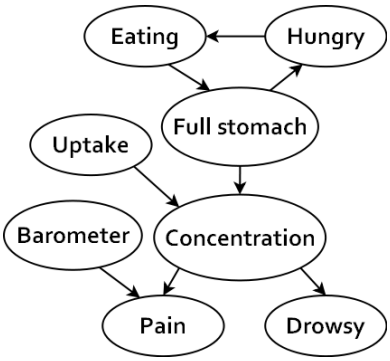

Figure 5.1: Example CTBN.

CTBNs, which are attempting to reduce complex state-spaces into more compact representations that model conditional dependencies instead of the full joint distributions. The naïve approach to sensitivity analysis requires working with the full joint intensity matrix, ignoring the very reason for having the factored representation in the first place. Generating sample paths becomes exponentially expensive, and perturbation realization becomes infeasible. The factored nature of the networks enables sensitivity analysis to work on smaller subnetworks. These subnetworks can be sampled, the performance potentials calculated, and multiple $\mathbf{Q}$ matrices tested–all separately.

For a CTBN sensitivity analysis example, consider the example network from Chapter 2, shown again in Figure 5.1. Suppose that a performance function is attached to the two states of the *Pain* node. Instead of amalgamating all of the nodes into one large Markov process, we would like to divide the system into smaller subnetworks. Smaller subnetworks limit the state-space size of any one subnetwork and allows for more tractable evaluation of Equation 5.2 for calculating the change in performance for given perturbations. Our method for creating subnetworks, as with perturbation realization, follows a sample path approach.

For a node with parents in the network, the node will have conditional intensity matrices for every possible state instantiation of its parents. We want to "isolate" the node in the network, which entails estimating the node's single unconditional intensity matrix, thereby removing the conditional dependence on its parents. Node isolation is analogous to marginalizing out variables in a Bayesian network.

We now present a method for node isolation. It approximates the unconditional intensity matrix from a sample path, aggregating the average sojourn times and transition counts between subsystems. While combinations of nodes in a CTBN are able to model complex distributions, such as mixtures of exponentials, this sampling approach approximates the behavior of a subsystem with a single node. While some accuracy is lost from this approximation, in many cases, the exponential distribution is sufficient to approximate the behavior of the subsystem while reducing the network complexity. Even in the simplest case, in which a two-state node can be removed from the network, this still reduces the state-space over the entire network in half.

The following algorithm shows the pseudocode for how node isolation can be accomplished for an arbitrary node in the network. The algorithm accepts a sample path and the node to be isolated and returns the unconditional intensity matrix of the isolated node.

Algorithm IsolateNode($\mathcal{S}$, X // Estimate unconditional intensity matrix of CTBN node

Input: $\mathcal{S}$ as sample path, X as node in $\mathcal{S}$ to isolate
Output: $\mathbf{A}_X$ as unconditional intensity matrix of X with entries $a_{i,j}$
$\mathbf{A}_X \leftarrow \mathbf{0}$, $\mathbf{v} \leftarrow \mathbf{0}$
$\langle X(t), t\rangle \leftarrow$ initial state of X in $\mathcal{S}$
$i \leftarrow X(t)$
for each transition $\langle X(t), t'\rangle$ of X in $\mathcal{S}$
    $j \leftarrow X(t)$
    $t \leftarrow t'$
    $a_{i,i} \leftarrow a_{i,i} + (t' - t)$
    $a_{i,j} \leftarrow a_{i,j} + 1$

11: $v_i \leftarrow v_i + 1$
12: $i \leftarrow X(t)$
13: **end for**
14: **for each** state i of X
15: $a_{i,i} \leftarrow a_{i,i}$ / v_i
16: $a_{i,i} \leftarrow -(1/a_{i,i})$
17: $z \leftarrow 0$
18: **for each** state $j \neq i$ of X
19: $z \leftarrow z + a_{i,j}$
20: **end for**
21: **for each** state $j \neq i$ of X
22: $a_{i,j} \leftarrow a_{i,j} \cdot (|a_{i,i}|/z)$
23: **end for**
24: **end for**
25: **return** $\mathbf{A}_X$

First, lines 3-5 of the algorithm initialize the variables. Although the states of ancestor nodes may be changing throughout the sample path, lines 6-13 are concerned *only* with state changes of the node to isolate. Specifically, line 9 calculates the total amount of time spent in each state of the node, line 10 counts how many times the system has transitioned between each state of the node, and line 11 counts the total number of transitions that have occurred between states of the node. Next, lines 14-24 transform these statistics into an unconditional intensity matrix. Line 15 calculates the average sojourn time (amount of time spent per visit) for each state of the node, which is taken as the expected sojourn time. Therefore, line 16 takes the negative reciprocal for use in the exponentially decreasing probability function. The relative number of transitions from a state to the remaining states represent the transition probabilities, which lines 17-23 normalize in accordance with the intensity matrix row constraints.

Using perturbation realization and the method for isolating subnetworks, we can describe a generalized method for performing sensitivity analysis on CTBNs. The primary obstacle to overcome is that the state-space size is exponential in the num-

ber and cardinality of the CTBN nodes. For complex CTBNs, even relatively long sample paths over the entire network never reach all possible states. Because of these unvisited states, perturbation realization has no information on the state's performance potential and the value returned from Equation (5.2) becomes inaccurate at best. Thus, being able to create and analyze sample paths from smaller subnetworks, and then being able to combine the results, becomes essential.

The following algorithm shows the pseudocode for performing sensitivity analysis. The algorithm accepts a CTBN and a set of $\mathbf{Q}$ matrices for the nodes of the CTBN that represent changes to the nodes' intensity matrices. The algorithm returns an estimate of the change in performance per unit time given the perturbations in the intensity matrices. It calls five helper methods in addition to the IsolateNode algorithm. CollapseCycles($\mathcal{G}$) detects cycles in $\mathcal{G}$, amalgamates each cycle's intensity matrices, and replaces each cycle with a single node. PerturbCTBN($\mathcal{C}$, $\mathbf{Q}$) applies the perturbations $\mathbf{Q}$ to the intensity matrices of the CTBN $\mathcal{C}$. AmalgamateIntensityMatrices($\mathcal{C}$) expands the full joint intensity matrix of the CTBN $\mathcal{C}$, which is used by the algorithm on a subnetwork consisting of a node Y and its parents. CalculateSteadyStateProbabilities($\mathcal{C}$) calculates the steady-state probabilities $\boldsymbol{\pi}$ from the full joint intensity matrix of $\mathcal{C}$ by calculating the normalized eigenvector of corresponding with the zero eigenvalue as per Equation (5.5). CalculatePotentialVector($\mathcal{S}$) parses the sample path $\mathcal{S}$ and calculates the potential vector $\mathbf{g}$ as per Equation (5.4).

Algorithm EstimateDerivative($\mathcal{C}$, $\mathbf{Q}$)
1: **Input:** $\mathcal{C}$ as CTBN, $\mathbf{Q}$ as set of Q matrices for the nodes of $\mathcal{C}$
2: **Output:** $\partial\eta/\partial\mathbf{Q}$ as performance measure derivative
3: $\mathcal{G}' \leftarrow$ CollapseCycles($\mathcal{G}$)
4: $\partial\eta/\partial\mathbf{Q} \leftarrow 0$
5: **repeat** until termination

6: $\mathbf{X} \leftarrow \bigcup_{X \in \mathcal{G}'} X$ where $\mathbf{Pa}(X) = nil$
7: **if** $\mathbf{X} = nil$ **then** terminate
8: $\mathbf{Y} \leftarrow \bigcup_{X \in \mathbf{X}} \mathbf{Ch}(X)$ where $\mathbf{Pa}(\mathbf{Ch}(X)) \subseteq \mathbf{X}$
9: **for** $Y \in \mathbf{Y}$ **do**
10: $\mathcal{C}_Y \leftarrow Y \cup \mathbf{Pa}(Y)$
11: $\mathcal{S} \leftarrow \text{Sample}(\mathcal{C}_Y)$
12: $Y' \leftarrow \text{IsolateNode}(\mathcal{S}, Y)$
13: $\mathcal{C}'_Y \leftarrow \text{PerturbCTBN}(\mathcal{C}_Y, \mathbf{Q}_Y \cup \mathbf{Q}_{\mathbf{Pa}(Y)})$
14: $\mathcal{S}' \leftarrow \text{Sample}(\mathcal{C}'_Y)$
15: $Y'' \leftarrow \text{IsolateNode}(\mathcal{S}', Y)$
16: $\mathbf{Q}_Y \leftarrow \mathbf{Q}_Y + (\mathbf{A}_{Y''} - \mathbf{A}_{Y'})$
17: $\boldsymbol{\pi} \leftarrow \text{CalculateSteadyStateProbabilities}(\mathcal{C}'_Y)$
18: $\mathbf{Q} \leftarrow \text{AmalgamateIntensityMatrices}(\mathbf{Q}_Y \cup \mathbf{Q}_{\mathbf{Pa}(Y)})$
19: $\mathbf{g} \leftarrow \text{CalculatePotentialVector}(\mathcal{S})$
20: $\partial\eta/\partial\mathbf{Q} \leftarrow \partial\eta/\partial\mathbf{Q} + \boldsymbol{\pi}\mathbf{Q}\mathbf{g}$
21: **for** $X \in \mathbf{Pa}(Y)$ **do**
22: remove edge (X, Y) in $\mathcal{G}'$
23: **end for**
24: replace Y with Y' in $\mathcal{G}'$
25: $\mathbf{f}_Y \leftarrow 0$
26: **end for**
27: **end repeat**
28: **return** $\partial\eta/\partial\mathbf{Q}$

First, line 3 collapses the cycles of the CTBN. This is necessary before the top-down isolation begins, because node isolation requires a sample path with all of the node's ancestors and every node in a cycle is an ancestor to every other node in the cycle. This turns the network into a directed acyclic graph. Line 4 initializes the performance measure derivative estimate, which will be incrementally updated during the top-down isolation process. Lines 5-27 repeat until the condition of line 7 is satisfied, that is, when every node has been isolated. Line 6 finds the roots of the network, those without any parents. Line 8 finds all the immediate children of root nodes, i.e., all the second-level nodes. These will be isolated and become the new root nodes. Lines 9-26 iterate over the children. Line 10 creates a subnetwork of a

second-level node and its parents. Line 11 creates a sample path for the child node, while line 12 uses that sample path to isolate it. Line 13 applies any perturbations to the subnetwork's conditional intensity matrices, and lines 14-15 isolate the perturbed child node. Line 16 adds the difference between the unconditional intensity matrices to the child perturbations. Line 17 calculates the steady-state probabilities of the child node and its parents. Line 18 calculates the $\mathbf{Q}$ matrix for the child node and its parents. Line 19 calculates the potential vector from the sample path. Line 20 calculates the performance measure derivative for the subnetwork consisting of the child and its parents. Lines 21-24 finishes the isolation of the child, removing the arcs from its parents and replacing the conditional intensity matrices with the single unconditional intensity matrix calculated in line 12. Line 25 removes the performance function for the child node, so that the performance of the node is not over-counted when moving to lower layers. After reaching the leaves of the CTBN, the aggregated performance measure derivative is returned in line 28.

The EstimateDerivative algorithm avoids handling the whole CTBN at once. Sample paths only have to be created for a node and its parents. Likewise, the largest full joint intensity matrix that must be expanded is only over a node and its parents. Thus, the complexity of the algorithm is driven by the size of the cycles that had to be collapsed into single nodes and the number of parents of each node (reminiscent of tree-width in clique tree inference on Bayesian networks). However, the algorithm is able to take advantage of the network factorization and only deal with smaller subnetworks at any one time.

Table 5.1: Sensitivity Analysis Results

Network	Node count	State count	State count of largest subnetwork	Magnitude of perturbation	Sample count	Rel. error
Synthetic	5	72	18	$\times 10$	100K	$< 1\%$
CAS [79]	19	6.6M	160	$\times 200$	1M	$< 1\%$
Milling [80]	7	1458	81	$\times 10$	100K	$< 1\%$

5.3 Preliminary Results

We demonstrated the algorithms for CTBN sensitivity analysis on three networks. The first is a small, synthetic network in which the exact answers can be computed easily using Equation 5.1. The second is a dynamic fault tree for a cardiac assist system (CAS) encoded as a CTBN. The third is a model of a milling machine as learned from run-to-failure data. The performance function for each of these was set as a unit performance measure on the default state of the lowest node in the network. Thus, any perturbations to the parameters of nodes higher in the network would have an effect on the node's performance measure.

For each network, the performance measure derivative is calculated using our algorithms for sensitivity analysis and compared to the ground truth, taken as either the exact solution for the synthetic network or estimated from brute-force sampling over the whole network for the two real-world networks. We measure relative error between our algorithm and the ground truth.

The results are shown in Table 5.1 for the three networks. The table shows the number of nodes and the total number of distinct states. This is the size of the state-space from which the brute-force approach is sampling and dictates the size of the network's full joint intensity matrix. Using node isolation, our sensitivity analysis is able to restrict its calculations to one subnetwork at a time. The size of the state-space

for the largest of these subnetworks is also given by the table. The magnitude of the perturbations are the factor by which the original parameters were perturbed. The number of samples generated were for the brute-force approach and each instance of node isolation. They are not the total number of samples generated, but the number of transitions that were generated in the target node (either the node to isolate or the node with the performance measure).

In each instance, we see that our approach for sensitivity analysis is able to take advantage of the factored nature of the network, focusing on only smaller portions of the state-space at any one time. In each case, the relative error was converging to either the exact solution for the synthetic network or the estimates from brute-force sampling.

In these networks, the dynamics of the failure node was slower than its ancestors, because failure was a relatively rare occurrence compared to the full life-cycle of the system. Therefore, when generating the sample paths, transitions in the ancestors were sampled far more frequently than transitions in the failure node. This slowed down the node isolation process, which could take up to several hours to sample sufficient transitions to isolate a single node. The next section, although focused on general inference in CTBNs, begins to address this deficiency by introducing a closed-form solution that approximates the unconditional intensity matrix without resorting to sample paths.

CHAPTER 6

NODE ISOLATION FOR APPROXIMATE INFERENCE

In this chapter, we propose exploring how the node isolation concept introduced in the previous chapter may be used in a new CTBN inference algorithm. We describe preliminary work in further developing the node isolation process.

6.1 New Approximate Inference Algorithm

The sensitivity analysis of the previous chapter used sample paths of the system under nominal and perturbed conditions to compute the expected performance of the system. However, it was not meant to be able to incorporate observations about the system through time. Because of this, the node isolation technique was only concerned with calculating a single unconditional intensity matrix describing the dynamics of a single node for all time throughout the process.

To be able to perform general inference in CTBNs, on the other hand, we must be able to incorporate continuous-time evidence. In sampling algorithms, for example, the sampling process changes based on previous, current, and upcoming evidence. Incorporating evidence is anticipated to be the greatest challenge in leveraging our node isolation technique, as it is currently unclear how node isolation and evidence must interact. Many of the inference algorithms described in Section 2.5.2 divide time into segments, corresponding to intervals during which the evidence remains constant. It is anticipated that our inference algorithm will also do this, possibly calculating a new unconditional intensity matrix for each segment conforming to the evidence over that segment.

6.2 Preliminary Results

The following two sections show further work with the node isolation process. First, we define a closed-form solution for node isolation. Note that, while a closed-form solution, it is still an approximation due to the fact that it is approximating the behavior of multiple conditional intensity matrices with a single unconditional intensity matrix. Second, we show preliminary work in addressing node isolation with cycles. Originally, all ancestors were required to isolate a node, whereas we propose a technique that uses only a node and its immediate parents in the case of cycles.

6.2.1 Closed-Form Node Isolation

In this subsection, we describe a closed-form solution for CTBN node isolation. First, we create a single parent of m states, amalgamating the node and the its ancestors as a single supernode. The states of the amalgamated supernode are represented as (p_k, c_l), meaning the parent p is in state k while the child c is in state l. We find the full joint intensity matrix of the parent with m states and the child of n states, as shown in Figure 6.1.

The intensity submatrix $\boldsymbol{A}_i$ along the diagonal of $\boldsymbol{A}$ denotes the dynamics of subsystem i when the state of the child is held constant. The intensity submatrix $\boldsymbol{A}_{i,j}$ is a diagonal matrix showing the transition probabilities from the states of subsystem i to the states of subsystem j. Each of these matrices is diagonal because two nodes of a CTBN cannot transition simultaneously, and an off-diagonal entry in the $\boldsymbol{A}_{i,j}$ would represent a transition in both p and c. Thus, when the state of the child changes, the state of the parent must be held constant.

$$
\boldsymbol{A} = \begin{array}{c} \\ (p_1,c_1) \\ \vdots \\ (p_m,c_1) \\ \\ (p_1,c_n) \\ \vdots \\ (p_m,c_n) \end{array}
\begin{array}{c|c|c}
(p_1,c_1)\ldots(p_m,c_1) & \cdots & (p_1,c_n)\ldots(p_m,c_n) \\
\hline
A_1 & \cdots & A_{1,n} \\
\hline
\vdots & \ddots & \vdots \\
\hline
A_{n,1} & \cdots & A_n
\end{array}
$$

Figure 6.1: The amalgamated intensity matrix for a node and its parent as a block matrix.

To find the unconditional intensity matrix of the child, we replace each submatrix $\boldsymbol{A}_i$ with the mean sojourn time in subsystem i and replace each submatrix $\boldsymbol{A}_{i,j}$ with the probability of transitioning from subsystem i to subsystem j.

The first thing we need is the stationary distribution of the embedded Markov chain of the entire system. The off-diagonal entries of $\boldsymbol{A}$ are normalized by the diagonal to get the transition probabilities and the diagonals are zeroed out. The stationary distribution of this embedded Markov chain can then be found using a technique called first step analysis, which breaks down the problem by analyzing the possible transitions on the first step and then applying the Markov property [5]. The system of equations for the stationary distribution of the embedded Markov chain is:

$$
\begin{array}{lcccccccc}
P_{1,1} & = & & & \frac{(p_{1,2}c_1)}{|(p_{1,1}c_1)|}P_{1,2} & + & \cdots & + & \frac{(p_{1,m}c_1)}{|(p_{1,1}c_1)|}P_{1,m} \\
 & & & & \vdots & & & & \\
P_{1,n} & = & \frac{(p_{2,1}c_i)}{|(p_{2,2}c_i)|}P_{2,1} & + & \frac{1}{|(p_{2,2}c_i)|} & + & \cdots & + & \frac{(p_{2,m}c_i)}{|(p_{2,2}c_i)|}T_m \\
P_{2,1} & = & \frac{(p_{2,1}c_i)}{|(p_{2,2}c_i)|}T_1 & + & \frac{1}{|(p_{2,2}c_i)|} & + & \cdots & + & \frac{(p_{2,m}c_i)}{|(p_{2,2}c_i)|}T_m \\
 & & & & \vdots & & & & \\
P_{m,n} & = & \frac{(p_{m,1}c_i)}{|(p_{m,m}c_i)|}T_1 & + & \frac{(p_{m,2}c_i)}{|(p_{m,m}c_i)|}T_2 & + & \cdots & + & \frac{1}{|(p_{m,m}c_i)|} \\
1 & = & P_{1,1} & + & P_{1,2} & + & \cdots & + & P_{m,n}
\end{array}
$$

where $P_{i,j}$ is the steady-state probability the parent being in state i and the child being in state j. For the mean sojourn time in the subsystem, we start by examining the diagonal matrices $\boldsymbol{A}_i$:

$$\boldsymbol{A_i} = \begin{pmatrix} (p_{1,1}, c_i) & (p_{1,2}, c_i) & \cdots & (p_{1,m}, c_i) \\ (p_{2,1}, c_i) & (p_{2,2}, c_i) & \cdots & (p_{2,m}, c_i) \\ \vdots & \vdots & \ddots & \vdots \\ (p_{m,1}, c_i) & (p_{m,2}, c_i) & \cdots & (p_{m,m}, c_i) \end{pmatrix}$$

We can compute the expected time to leave the subsystem also using first step analysis. The system of equations for expected time T_j to leave subsystem i starting from each state j in the subsystem is:

$$\begin{array}{rcccccccc} T_1 & = & \frac{1}{|(p_{1,1},c_i)|} & + & \frac{(p_{1,2},c_i)}{|(p_{1,1},c_i)|}T_2 & + & \cdots & + & \frac{(p_{1,m},c_i)}{|(p_{1,1},c_i)|}T_m \\ T_2 & = & \frac{(p_{2,1},c_i)}{|(p_{2,2},c_i)|}T_1 & + & \frac{1}{|(p_{2,2},c_i)|} & + & \cdots & + & \frac{(p_{2,m},c_i)}{|(p_{2,2},c_i)|}T_m \\ & & & & \vdots & & & & \\ T_m & = & \frac{(p_{m,1},c_i)}{|(p_{m,m},c_i)|}T_1 & + & \frac{(p_{m,2},c_i)}{|(p_{m,m},c_i)|}T_2 & + & \cdots & + & \frac{1}{|(p_{m,m},c_i)|} \end{array}$$

This gives the mean time to leave the subsystem starting from each state. Now we need the entry distribution into the subsystem, which we find by weighting the transition probability into each state of the subsystem by the originating state's stationary distribution. Each of the other subsystems can transition once into each state of the subsystem, as shown by the off-diagonal matrices $\boldsymbol{A}_{i,j}$:

$$\boldsymbol{A_{i,j}} = \begin{pmatrix} (p_0, c_{i,j}) & 0 & \cdots & 0 \\ 0 & (p_1, c_{i,j}) & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & (p_m, c_{i,j}) \end{pmatrix}$$

The expected sojourn time in subsystem j is:

$$c_j = \frac{\sum_{i=1}^{n} \left(\frac{(p_k, c_{i,j})}{|(p_{k,k}, c_i)|} P_{k,i} T_k \right)}{\sum_{i=1}^{n} \left(\frac{(p_k, c_{i,j})}{|(p_{k,k}, c_i)|} P_{k,i} \right)}$$

The transition probability between subsystems i and j is:

$$c_{i,j} = \sum_{i=1}^{n} \left(\frac{(p_k, c_{i,j})}{|(p_{k,k}, c_i)|} P_{k,i} \right)$$

Having the expected sojourn time in each subsystem and the transition probabilities betweens subsystems, we can now populate the unconditional intensity matrix $\boldsymbol{A}'$ for the child:

$$\boldsymbol{A}' = \begin{pmatrix} -1/c_1 & c_{1,2}/z_1 & \cdots & c_{1,n}/z_1 \\ c_{2,1}/z_2 & -1/c_2 & \cdots & c_{2,n}/z_2 \\ \vdots & \vdots & \ddots & \vdots \\ c_{n,1}/z_n & c_{n,2}/z_n & \cdots & -1/c_n \end{pmatrix},$$

where z_i is a normalizer to ensure each row sums to zero. The complexity of node isolation is in solving the systems of equations, which is driven by the number of ancestors and the number of states.

Compared to the sample-based IsolateNode algorithm from the previous chapter, this closed-form solution performed substantially faster in all preliminary experiments. Furthermore, as the number of samples increased, the unconditional intensity matrix returned by the IsolateNode algorithm converged to the unconditional intensity matrix calculated by this closed-form solution, which further validates this technique.

6.2.2 Node Isolation in Cycles

One weakness of node isolation is that it requires all of the ancestors of the node to isolate. If the network is a directed acyclic graph (DAG), then we can isolate each layer in succession, and the complexity of isolation depends on the number of immediate parents to each node. However, cycles are allowed in CTBNs. When a cycle is introduced, every node in the cycle must be included to isolate any node in the cycle, because every node in the cycle is an ancestor of every other node in the cycle.

In this section, we propose an iterative step to our node isolation algorithms that avoid dealing with the entire cycle all at once. The idea is that, given a cycle, we start at an arbitrary node in the cycle and temporarily remove the incoming arc. Previously, the node had a set of conditional intensity matrices, whereas now we need to replace it with one unconditional intensity matrix. However, this unconditional intensity matrix depends on the dynamics of the parent that was just removed. Nevertheless, we simply use an unconditional intensity matrix that is the average of its conditional intensity matrices.

Now, we isolate its child node, finding its unconditional intensity matrix. Depending on the actual parameters, this unconditional intensity matrix of the isolated child may be a poor approximation, because of how we constructed the unconditional intensity matrix of its parent. However, it is likely a better approximation of the unconditional intensity matrix than that of its parent. This process repeats, isolating nodes around the cycle, back to the originally chosen node. Now, this node is the child, and its conditional intensity matrices are used for its isolation. This process continues to loop around the cycle. We hypothesize and intend to prove that, given

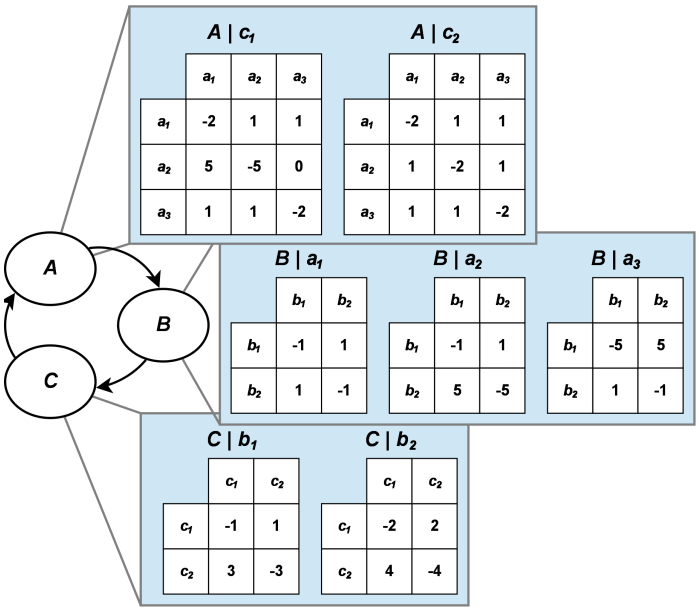

Figure 6.2: Example CTBN to demonstrate iterative isolation in cycle.

enough iterations, each of the unconditional intensity matrices of the nodes in the cycle will converge.

We demonstrate this process on the simple, three-node network given in Figure 6.2. We used the algorithm for finding the closed-form solution from the previous section on the whole network to isolate node A. This was taken as the ground truth. We compared this unconditional intensity matrix $\mathbf{A}$ with the unconditional intensity matrix $\mathbf{A}'$ calculated from the iterative process described above. Error was calculated as

$$\sum_{i=1}^{3}\sum_{j=1}^{3}\frac{|\mathbf{A}_{i,j}-\mathbf{A}'_{i,j}|}{|\mathbf{A}_{i,j}|},$$

which is the sum of the relative errors between entries in the two matrices. After three iterations around the cycle, the process converged and the cumulative relative error for node A was 1.4%.

This experiment demonstrates the feasibility of the approach. More experiments are needed to analyze how the approach handles larger cycles with more states and more varied parameters and how these may affect accuracy. We also need to examine how this works when there are other parents and children, even multiple overlapping cycles. However, we can see the potential benefit of the approach even with this single, small cycle. The iterative approach works with at most half of the state-space at any one time. As cycles grow longer, the ability to work with only a single node and its immediate parents becomes even more beneficial.

CHAPTER 7

DISSERTATION PLAN

The following lays out the high-level goals for completing this doctoral research.

1. Continue writing the dissertation, incorporating feedback from the comprehensive presentation.

2. Write formal proofs that exact and approximate inference in CTBNs is at least NP-hard, even when given the initial state, under ergodic and non-ergodic conditions. Determine an appropriate venue and submit paper for publication.

3. Formalize and characterize types of continuous-time evidence (point, interval, and transition) when the evidence is uncertain and/or negative. Extend existing inference algorithms to be able to support all enumerated types of evidence. Finishing writing paper and submit to the Journal of Artificial Intelligence Research (JAIR).

4. Continue to evaluate the approach for sensitivity analysis in CTBNs, analyzing the conditions under which the node isolation technique is able to maintain sufficient accuracy. Respond to feedback on paper for sensitivity analysis of CTBNs currently under review for IEEE Transactions on Reliability. Write paper on CTBN performance functions, their applications, and their structural representation in CTBNs and most likely submit to the Florida Artificial Intelligence Research Society (FLAIRS).

5. Formulate a new approach to general inference in CTBNs that uses the node isolation technique and that is able to support all enumerated types of evidence. Determine an appropriate venue and submit paper for publication.

6. Demonstrate each of these extensions on real-world networks, showing a diversity of applications for which the extensions are beneficial.

7. Finish writing the dissertation.

8. Present and defend the dissertation in April, 2014.

CHAPTER 8
CONCLUSION

In this proposal, we have described four major topics for extending CTBN inference and our preliminary work in meeting those goals. Furthermore, we have enumerated the high-level tasks necessary for completing this work. We conclude with a list of papers already published, already submitted, in progress, and for future submission, along with a summary of the contributions that this work represents once published.

8.1 Publishing Plan

The following papers have already been published, although they are not all directly related to the research presented here. Some of the concepts overlap, however, such as temporal models and their applications, along with uncertain and fuzzy evidence.

- "A standards-based approach to gray-scale health assessment using fuzzy fault trees" [81]
- "Principal component analysis preprocessing with Bayesian networks for battery capacity estimation" [82]
- "Extending high-dimensional indexing techniques Pyramid and iMinMax (θ): lessons learned" [83]
- "Diagnostic Bayesian networks with fuzzy evidence" [84]
- "Implementing AI-ESTATE with prognostic extensions in Java" [85]

We submitted "Sensitivity Analysis of Continuous Time Bayesian Network Reliability Models" to IEEE Transactions on Reliability in July, 2013 and are awaiting feedback.

We are currently working on the paper formulating uncertain and negative evidence in CTBNs and extending CTBN inference to support these new types of evidence.

The following papers are planned for future submission:

- A paper proving that exact and approximate inference in CTBNs is at least NP-hard even when given the initial states, under ergodic and non-ergodic conditions.
- A paper on presenting various applications for CTBN performance functions, as well as describing a structural representation for performance functions in CTBNs.
- A paper describing a new approximate inference procedure that uses the closed-form approach to node isolation.

8.2 Contributions

The research proposed will contribute a number of important and valuable advancements to inference in CTBNs.

First, the formal complexity proofs will provide greater insight into the problem of propagating probabilities through time. They will further motivate the development of more efficient approximate inference techniques. They may also provide a step toward further results that show provably tractable exact or approximate inference

for certain special cases of CTBNs. They will inform modelers of important considerations when balancing the complexity of the model with tractable inference. To date, however, almost all of the complexity results relevant to CTBNs are derived from the NP-hardness of the Bayesian network used for the initial distribution, instead of considering the process of propagating probabilities through time as a separate problem.

Second, the extension of inference to handle uncertain and negative will make the model more powerful and versatile, allowing for further representation and reasoning under uncertainty. It allows the inference algorithms to incorporate knowledge about the noisiness and robustness of the evidence. It allows for more varied types of evidence, such as when the true state is unknown over an interval of time, but some states can be ruled out. For these types of models, it is important to know not only the most probable answer, but to also know the confidence with which we can trust the most probable answer. The introduction of uncertain and negative evidence into the CTBN is an important extensions that has not yet been formulated.

Third, we presented the first algorithms for sensitivity analysis specific to CTBNs. Sensitivity analysis is another form of inference, one that queries the behavior the model, given changes in the model parameters rather than traditional observations. It is useful for measuring the robustness of the model and detecting which parameters most influence network performance. We have shown how to take advantage of the factored nature of the CTBN model to be able to more efficiently perform sensitivity analysis.

Fourth, we believe that our node isolation technique can serve as the core of a general CTBN inference algorithm that can be potentially very efficient. While several CTBN inference algorithms have now been developed, each inference algorithm has its strengths and weaknesses, breaking down under certain conditions. The introduction

of another inference algorithm gives the model user more options for performing inference for the user's specific model. Furthermore, many of these inference algorithms have already undergone refinement, and the node isolation technique may serve as a further refinement.

As a newer model than the BN, the CTBN has not yet undergone the extensive development enjoyed by the BN. However, many of the extensions defined for the BN model are similarly valuable for the CTBN model. Therefore, we have proposed a wide range of extensions to the CTBN model that starts to bridge this gap.

REFERENCES CITED

[1] Neil A Gershenfeld. *The Nature of Mathematical Modeling*. Cambridge University Press, 1999.

[2] G. Welch. An introduction to Kalman filtering. *Technical Report, Department of Computer Sc. and Engg., Univ. of North Carolina at Chapel Hill*, 2002.

[3] James D Hamilton. State-space models. 1993.

[4] M Sanjeev Arulampalam, Simon Maskell, Neil Gordon, Tim Clapp. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking. *IEEE Transactions on Signal Processing*, 50(2):174–188, 2002.

[5] Howard M. Taylor, Samuel Karlin. *An Introduction to Stochastic Modeling*. Academic press, 1998.

[6] Lawrence Rabiner, B Juang. An introduction to hidden Markov models. *ASSP Magazine, IEEE*, 3(1):4–16, 1986.

[7] Kevin Patrick Murphy. *Dynamic Bayesian networks: representation, inference and learning*. Doctoral Dissertation, University of California, 2002.

[8] Jerome T Connor, R Douglas Martin, LE Atlas. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254, 1994.

[9] U. Nodelman, C. Shelton, D. Koller. Continuous time Bayesian networks. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, pages 378–387, 2002.

[10] David H Wolpert. The lack of a priori distinctions between learning algorithms. *Neural computation*, 8(7):1341–1390, 1996.

[11] David H Wolpert, William G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 1997.

[12] D. Koller, N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009.

[13] Marcel A.J. van Gerven, Babs G. Taal, Peter J.F. Lucas. Dynamic Bayesian networks as prognostic models for clinical patient management. *Journal of Biomedical Informatics*, 41(4):515–529, 2008.

[14] K.P. Exarchos, G. Rigas, Y. Goletsis, D.I. Fotiadis. Towards building a dynamic Bayesian network for monitoring oral cancer progression using time-course gene expression data. *10th IEEE International Conference on Information Technology and Applications in Biomedicine (ITAB), 2010*, pages 1 –4, nov. 2010.

[15] F. Camci, R.B. Chinnam. Dynamic Bayesian networks for machine diagnostics: hierarchical hidden Markov models vs. competitive learning. *Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN), 2005*, Volume 3, pages 1752–1757 vol. 3, July-4 Aug. 2005.

[16] Alexandre Muller, Marie-Christine Suhner, Benot Iung. Formalisation of a new prognosis model for supporting proactive maintenance implementation on industrial system. *Reliability Engineering & System Safety*, 93(2):234 – 253, 2008.

[17] Ming Dong, Zhi-bo Yang. Dynamic Bayesian network based prognosis in machining processes. *Journal of Shanghai Jiaotong University (Science)*, 13:318–322, 2008.

[18] Kamal Medjaher, Jean-Yves Moya, Noureddine Zerhouni. Failure prognostic by using dynamic Bayesian networks. *Dependable Control of Discrete Systems*, 1:291–296, 2009.

[19] Hichem Boudali, Joanne Bechta Dugan. A continuous-time Bayesian network reliability modeling, and analysis framework. *IEEE Transactions on Reliability*, 55(1):86–97, 2006.

[20] Uri Nodelman. *Continuous Time Bayesian Networks*. Doctoral Dissertation, Stanford University, Stanford, California, 2007.

[21] Christian R. Shelton, Yu Fan, William Lam, Joon Lee, Jing Xu. Continuous time Bayesian network reasoning and learning engine. *Journal of Machine Learning Research*, 11(Mar):1137–1140, 2010.

[22] Yu Fan, Jing Xu, Christian R Shelton. Importance sampling for continuous time Bayesian networks. *The Journal of Machine Learning Research*, 99:2115–2140, 2010.

[23] Y. Fan, C.R. Shelton. Sampling for approximate inference in continuous time Bayesian networks. *Tenth International Symposium on Artificial Intelligence and Mathematics*, 2008.

[24] Jeremy C Weiss, Sriraam Natarajan, C David Page. Learning when to reject an importance sample. 2013.

[25] Tal El-Hay, Nir Friedman, Raz Kupferman. Gibbs sampling in factorized continuous-time Markov processes. *Proceedings of the Twenty-Fourth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pages 169–178, Corvallis, Oregon, 2008. AUAI Press.

[26] Vinayak Rao, Yee Whye Teh. Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. *arXiv preprint arXiv:1202.3760*, 2012.

[27] Vinayak Rao, Yee Whye Yeh. Fast MCMC sampling for Markov jump processes and extensions. *arXiv preprint arXiv:1208.4818*, 2012.

[28] Uri Nodelman, Daphne Koller, Christian Shelton. Expectation propagation for continuous time Bayesian networks. *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)*, pages 431–440, Arlington, Virginia, 2005. AUAI Press.

[29] Tal El-Hay, Ido Cohn, Nir Friedman, Raz Kupferman. Continuous-time belief propagation. 2010.

[30] Suchi Saria, Uri Nodelman, Daphne Koller. Reasoning at the right time granularity. *arXiv preprint arXiv:1206.5260*, 2012.

[31] Ido Cohn. *Mean Field Variational Approximations in Continuous-Time Markov Processes*. Doctoral Dissertation, The Hebrew University, 2009.

[32] Ido Cohn, Tal El-Hay, Nir Friedman, Raz Kupferman. Mean field variational approximation for continuous-time Bayesian networks. *Proceedings of the Twenty-Fifth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-09)*, pages 91–100, Corvallis, Oregon, 2009. AUAI Press.

[33] E Busra Celikkaya, Christian R Shelton, William Lam. Factored filtering of continuous-time systems. *arXiv preprint arXiv:1202.3703*, 2012.

[34] B. Ng, A. Pfeffer, R. Dearden. Continuous time particle filtering. *International Joint Conference on Artificial Intelligence*, Volume 19, page 1360, 2005.

[35] Uri Nodelman, Christian R Shelton, Daphne Koller. Expectation maximization and complex duration distributions for continuous time Bayesian networks. *arXiv preprint arXiv:1207.1402*, 2012.

[36] Dongyu Shi, Jinyuan You. Update rules for parameter estimation in continuous time Bayesian network. *PRICAI 2006: Trends in Artificial Intelligence*, pages 140–149. Springer, 2006.

[37] Uri Nodelman, Eric Horvitz. Continuous time Bayesian networks for inferring users' presence and activities with extensions for modeling and evaluation. Technical Report, 2003.

[38] R. Herbrich, T. Graepel, B. Murphy. Structure from failure. *Proceedings of the 2nd USENIX workshop on Tackling computer systems problems with machine learning techniques*, pages 1–6. USENIX Association, 2007.

[39] Yu Fan, Christian Shelton. Learning continuous-time social network dynamics. *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 161–168, 2009.

[40] Yu Fan. *Continuous Time Bayesian Network Approximate Inference and Social Network Applications*. Doctoral Dissertation, University of California, 2009.

[41] Dongyu Shi, Xinhuai Tang, Jinyuan You. An intelligent system based on adaptive CTBN for uncertainty reasoning in sensor networks. *Intelligent Automation & Soft Computing*, 16(3):337–351, 2010.

[42] Jing Xu, Christian Shelton. Intrusion detection using continuous time Bayesian networks. *J. Artif. Int. Res.*, 39(1):745–774, September 2010.

[43] Jing Xu. *A Continuous Time Bayesian Network Approach for Intrusion Detection*. Doctoral Dissertation, University of California, 2010.

[44] Jing Xu, Christian Shelton. Continuous Time Bayesian Networks for Host Level Network Intrusion Detection. Walter Daelemans, Bart Goethals, Katharina Morik, editors, *Machine Learning and Knowledge Discovery in Databases*, Volume 5212 series *Lecture Notes in Computer Science*, pages 613–627. Springer Berlin / Heidelberg, 2008.

[45] Shaojie Qiao, Changjie Tang, Huidong Jin, Teng Long, Shucheng Dai, Yungchang Ku, Michael Chau. PutMode: prediction of uncertain trajectories in moving objects databases. *Applied Intelligence*, 33(3):370–386, 2010.

[46] E. Gatti, D. Luciani, F. Stella. A continuous time Bayesian network model for cardiogenic heart failure. *Flexible Services and Manufacturing Journal*, pages 1–20, 2011.

[47] Elena Gatti. *Graphical models for continuous time inference and decision making*. Doctoral Dissertation, Università degli Studi di Milano-Bicocca, 2011.

[48] K. F. Kan, C. R. Shelton. Solving structured continuous-time Markov decision processes. *Proceedings of Tenth International Symposium on Artificial Intelligence and Mathematics*, 2008.

[49] Luigi Portinale, Daniele Codetta-Raiteri. Generalizing continuous time Bayesian networks with immediate nodes.

[50] F Stella, Y Amer. Continuous time Bayesian network classifiers. *Journal of biomedical informatics*, 2012.

[51] Daniele Codecasa, Fabio Stella. Conditional log-likelihood for continuous time Bayesian network classifiers.

[52] Jeremy Weiss, Sriraam Natarajan, David Page. Multiplicative forests for continuous-time processes. *Advances in Neural Information Processing Systems 25*, pages 467–475, 2012.

[53] Nir Friedman, Moises Goldszmidt. Learning Bayesian networks with local structure. *Learning in graphical models*, pages 421–459. Springer, 1998.

[54] Karthik Gopalratnam, Henry Kautz, Daniel S Weld. Extending continuous time Bayesian networks. *Proceedings of the National Conference on Artificial Intelligence*, Volume 20, page 981. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2005.

[55] Avi Pfeffer, Terry Tai. Asynchronous dynamic Bayesian networks. *arXiv preprint arXiv:1207.1398*, 2012.

[56] David Maxwell Chickering. Learning Bayesian networks is NP-complete. *Learning from Data*, pages 121–130. Springer, 1996.

[57] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial intelligence*, 42(2):393–405, 1990.

[58] Paul Dagum, Michael Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial intelligence*, 60(1):141–153, 1993.

[59] H. Chan, A. Darwiche. On the revision of probabilistic beliefs using uncertain evidence. *Artificial Intelligence*, 163(1):67–90, 2005.

[60] R. Pan, Y. Peng, Z. Ding. Belief update in Bayesian networks using uncertain evidence. *18th IEEE International Conference on Tools with Artificial Intelligence, 2006.*, pages 441–444. IEEE, 2006.

[61] J. Bilmes. On virtual evidence and soft evidence in Bayesian networks. *Dept. of EE, U. of Washington, Tech. Rep. UWEETR-2004-0016*, 2004.

[62] F.J. Groen, A. Mosleh. Foundations of probabilistic inference with uncertain evidence. *International Journal of Approximate Reasoning*, 39(1):49–83, 2005.

[63] E. Castillo, J. Gutierrez, A. Hadi. Sensitivity analysis in discrete Bayesian networks. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 27(4):412–423, Jul 1997.

[64] Hei Chan, Adnan Darwiche. Sensitivity analysis in Bayesian networks: From single to multiple parameters. *Proceedings of the Twentieth Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pages 67–75, Arlington, Virginia, 2004. AUAI Press.

[65] Xi-Ren Cao. Potential based sensitivity analysis of Markov chains. *Proceedings of the 35th IEEE Decision and Control*, Volume 1, pages 568–569, December 1996.

[66] Xi-Ren Cao, Xue-Ming Yuan, Li Qiu. A single sample path-based performance sensitivity formula for Markov chains. *IEEE Transactions on Automatic Control*, 41(12):1814–1817, December 1996.

[67] C.G. Cassandras, S.G. Strickland. On-line sensitivity analysis of Markov chains. *IEEE Transactions on Automatic Control*, 34(1):76–86, January 1989.

[68] Xi-Ren Cao, Han-Fu Chen. Perturbation realization, potentials, and sensitivity analysis of Markov processes. *IEEE Transactions on Automatic Control*, 42(10):1382–1393, Oct 1997.

[69] Xi-Ren Cao, Yat-Wah Wan. Algorithms for sensitivity analysis of Markov systems through potentials and perturbation realization. *IEEE Transactions on Control Systems Technology*, 6(4):482–494, July 1998.

[70] C. Cassandras, S. Strickland. Observable augmented systems for sensitivity analysis of Markov and semi-Markov processes. *IEEE Transactions on Automatic Control*, 34(10):1026–1037, October 1989.

[71] Christos Cassandras, Stephen Strickland. An "augmented chain" approach for on-line sensitivity analysis of Markov process. *26th IEEE Conference on Decision and Control*, Volume 26, pages 1873–1878, December 1987.

[72] Zikuan Liu, Fengsheng Tu. Single sample path-based sensitivity analysis of Markov processes using uniformization. *IEEE Transactions on Automatic Control*, 44(4):872–875, April 1999.

[73] Liyi Dai, Yu-Chi Ho. Structural infinitesimal perturbation analysis (SIPA) for derivative estimation of discrete-event dynamic systems. *IEEE Transactions on Automatic Control*, 40(7):1154–1166, July 1995.

[74] Yu-Chi Ho, Jian-Qiang Hu. An infinitesimal perturbation analysis algorithm for a multiclass G/G/1 queue. *Operations Research Letters*, 9(1):35–44, 1990.

[75] Li Xia, Xi-Ren Cao. Relationship between perturbation realization factors with queueing models and Markov models. *IEEE Transactions on Automatic Control*, 51(10):1699–1704, October 2006.

[76] Martin Reiman, Alan Weiss. Sensitivity analysis via likelihood ratios. *Proceedings of the 18th conference on Winter simulation*, WSC '86, pages 285–289, 1986.

[77] Marvin K. Nakayama, Ambuj Goyal, Peter W. Glynn. Likelihood ratio sensitivity analysis for Markovian models of highly dependable systems. *Operations Research*, 42(1):pp. 137–157.

[78] Paul Glasserman. Derivative estimates from simulation of continuous-time Markov chains. *Oper. Res.*, 40:292–308, March 1992.

[79] D. Cao. *Novel models and algorithms for systems reliability modeling and optimization*. Doctoral Dissertation, Wayne State University, 2011.

[80] A. Agogino, K. Goebel. Mill Data Set. *BEST lab, UC Berkeley. NASA Ames Prognostics Data Repository, [http://ti.arc.nasa.gov/project/prognostic-data-repository], NASA Ames, Moffett Field, CA*, 2007.

[81] Patrick J Donnelly, Liessman E Sturlaugson, John W Sheppard. A standards-based approach to gray-scale health assessment using fuzzy fault trees. *AUTOTESTCON, 2012*, pages 174–181. IEEE, 2012.

[82] Liessman E Sturlaugson, John W Sheppard. Principal component analysis preprocessing with Bayesian networks for battery capacity estimation. *2013 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 98–101. IEEE, 2013.

[83] Karthik Ganesan Pillai, Liessman Sturlaugson, Juan M Banda, Rafal A Angryk. Extending high-dimensional indexing techniques Pyramid and iMinMax (θ): Lessons learned.

[84] Nicholas Ryhajlo, Liessman Sturlaugson, John W Sheppard. Diagnostic Bayesian networks with fuzzy evidence. *AUTOTESTCON, 2013*.

[85] Liessman Sturlaugson, Nathan Fortier, Patrick J Donnelly, John W Sheppard. Implementing AI-ESTATE with prognostic extensions in Java. *AUTOTESTCON, 2013*.