The Long-Run Behavior of Continuous Time Bayesian Networks

Liessman Sturlaugson and **John W. Sheppard**
Department of Computer Science
Montana State University
Bozeman, MT 59717
listurlaugson@gmail.com, john.sheppard@cs.montana.edu

Abstract

The continuous time Bayesian network (CTBN) is a temporal model consisting of interdependent continuous time Markov chains (Markov processes). One common analysis performed on Markov processes is determining their long-run behavior, such as their stationary distributions. While the CTBN can be transformed into a single Markov process of all nodes' state combinations, the size is exponential in the number of nodes, making traditional long-run analysis intractable. To address this, we show how to perform "long-run" node marginalization that removes a node's conditional dependence while preserving its long-run behavior. This allows long-run analysis of CTBNs to be performed in a top-down process without dealing with the entire network all at once.

1 INTRODUCTION

Many problems in artificial intelligence require reasoning about complex systems. One important and challenging type of system is one that changes in time. Temporal modeling and reasoning present additional challenges in representing the system's dynamics while efficiently and accurately inferring the system's behavior through time. Continuous time Bayesian networks (CTBNs) were introduced by Nodelman et al. (2002) as a temporal model capable of representing and reasoning about finite- and discrete-state systems without uniformly discretizing time, as found with dynamic Bayesian networks (Murphy, 2002). CTBNs have since been applied in a wide variety of temporal domains, from medical prognosis (Gatti et al., 2011; Gatti, 2011) to network security (Xu & Shelton, 2008, 2010) and reliability modeling (Herbrich et al., 2007; Cao, 2011; Sturlaugson & Sheppard, 2015).

While a variety of algorithms exist for querying probabilities of nodes in a CTBN at a specific time given temporal evidence, another useful type of query is to analyze a network's long-run behavior, i.e., the stationary distributions of a CTBN's nodes. None of the previous CTBN inference algorithms were specifically designed to solve this problem. This paper presents the first inference algorithms for efficiently computing the stationary distribution of nodes in a CTBN.

The paper is organized as follows. Section 2 provides the background for the rest of the paper. Section 3 gives the theory and algorithms for computing stationary distributions in CTBNs. In Section 4, we demonstrate the algorithms on three CTBNs. Section 5 contains the conclusion and future work.

2 BACKGROUND

In this section, we begin by describing Markov processes and their long-run behavior. We then introduce the CTBN and discuss how combinations of nodes can be viewed as Markov processes.

2.1 MARKOV PROCESSES

Although its name draws on the parallels between the conditional independence encoded by Bayesian networks, the CTBN is functionally a factored Markov process. Therefore, we start with background on Markov processes.

2.1.1 Definition

There are variations and extensions of the Markov process model, but the CTBN model uses the model described in this section. We refer to a finite-state, continuous-time Markov chain as a Markov process. In a Markov process, a system comprises a discrete set of states, and the system transitions probabilistically 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 2.1 (Markov Process)**.** *A finite-state, continuous-time, homogeneous Markov process X with a state space of size n is defined by an initial probability distribution P_X^0 over the n states and an $n \times n$ transition intensity matrix*

$$\mathbf{Q}_X = \begin{pmatrix} -q_{1,1}^X & q_{1,2}^X & \cdots & q_{1,n}^X \\ q_{2,1}^X & -q_{2,2}^X & \cdots & q_{2,n}^X \\ \vdots & \vdots & \ddots & \vdots \\ q_{n,1}^X & q_{n,2}^X & \cdots & -q_{n,n}^X \end{pmatrix}$$

in which each entry $q_{i,j}^X \geq 0$ for $i \neq j$ gives the transition rate of the process moving from state i to state j, and each entry $q_{i,i}^X = \sum_j q_{i,j}^X$ is the parameter for an exponential distribution, determining the sojourn times for the process to remain in state i. For notational shorthand, the size of the state-space of X will be denoted as $|X|$.

The value $q_{i,i}^X$ gives the rate at which the system leaves state x_i, while the value $q_{i,j}^X$ gives the rate at which the system transitions from state x_i to state x_j. Let $X(t)$ denote the state of X at time t. For $i \neq j$, we have that

$$\lim_{h \to 0^+} \frac{P(X(t+h) = x_j | X(t) = x_i)}{h} = q_{i,j}^X,$$

while $q_{i,i}^X = \sum_{j \neq i} q_{i,j}^X$. With the diagonal entries constrained to be non-positive, the probability density function for the process remaining in state i is given by $q_{i,i}^X \exp(-q_{i,i}^X t)$, 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 $1/\left|q_{i,i}^X\right|$. The transition probabilities from state i to state j can be calculated as $\theta_{i,j}^X = q_{i,j}^X / q_{i,i}^X$. 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. In other words, for $0 < s < t < \infty$,

$$P(X(t+h)|X(t), X(s)) = P(X(t+h)|X(t)).$$

Rather than looking at the Markov process as a whole, we can also consider subsets of states, which we refer to as a subsystem. Formally, a subsystem S defines the behavior of a subset of states of a full Markov process X. The intensity matrix $\mathbf{Q}_S$ of the subsystem S is formed from the entries of $\mathbf{Q}_X$ that correspond to the states in S.

2.1.2 Stationary Distributions of Markov Processes

A common analysis of Markov processes is to consider their long-run behavior. In particular, we can consider the stationary distribution $\boldsymbol{\pi}_X = \{\pi_1^X, \ldots, \pi_n^X\}$ of the process, where

$$\pi_i^X = \lim_{t \to \infty} P(X(t) = i).$$

Assuming $\boldsymbol{Q}_X$ is non-singular, we can compute the stationary distribution by setting up the system of equations

$$\boldsymbol{\pi}_X = \boldsymbol{Q}_X \boldsymbol{\pi}_X \tag{1}$$

with the added constraint $\sum_{i=1}^n \pi_i^X = 1$ (Taylor & Karlin, 1998). The complexity of solving for $\boldsymbol{\pi}_X$ is determined by n, the number of states in X.

The state convergence properties of a Markov process can be analyzed from the stationary distribution. As an application used in the paper, this could be the expected long-term availability of a system. Changes to the model could be tested to optimize the reliability of the system (e.g, what is the minimum reliability of each component to guarantee the target reliability of the entire system). Stationary distributions of Markov processes have been used to analyze long-term trends in meteorology, economics, sociology, biology, immunology–just to name a few. As a factored Markov process, the CTBN can be used wherever a Markov process is applicable, while allowing the model to become more powerful and flexible through its factored representation.

2.2 CONTINUOUS TIME BAYESIAN NETWORKS

The CTBN was first introduced in Nodelman et al. (2002) and then further developed in Nodelman (2007) as a continuous-time probabilistic graphical model.

2.2.1 Definition

The motivation behind CTBNs is to factor a Markov process in much the same way that a Bayesian network factors a joint probability distribution. Instead of conditional probabilities, the CTBN uses conditional Markov processes. The CTBN is defined formally as follows.

Definition 2.2 (Continuous Time Bayesian Network)**.** *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 $\mathcal{N} = \langle \mathcal{B}, \mathcal{G} \rangle$ over $\mathbf{X}$ consists of two components. The first is an initial distribution denoted $P_{\mathbf{X}}^0$ over $\mathbf{X}$ specified as a Bayesian network $\mathcal{B}$. This distribution $P_{\mathbf{X}}^0$ is only used for determining the initial state of the process. The second is a continuous-time transition model $\mathcal{G}$, which describes the evolution of the process from its initial distribution. $\mathcal{G}$ is represented as a directed graph with nodes $X_1, X_2, \ldots, X_n$. Let $\mathbf{Pa}(X)$ denote the set of parents of X in $\mathcal{G}$, and let $\mathbf{Ch}(X)$ denote the set of children of X in $\mathcal{G}$. Let $\mathbf{pa}_X$ denote the set of all combinations of state instantiations to $\mathbf{Pa}(X)$, and let $\langle pa_X \rangle \in \mathbf{pa}_X$. A set of* *conditional intensity matrices (CIMs), denoted $\mathbf{Q}_{X|\mathbf{Pa}(X)}$, is associated with each $X \in \mathbf{X}$ and comprises matrices $\mathbf{Q}_{X|\langle pa_X \rangle}$ $\forall \langle pa_X \rangle \in \mathbf{pa}_X$*

For example, suppose we have a two-node CTBN with dependencies as $A \leftrightarrows B$. Nodes A and B each have two states, with conditional intensity matrices as follows.

$$\mathbf{Q}_{A|b_0} = \begin{array}{c|cc} & a_0 & a_1 \\ a_0 & -1 & 1 \\ a_1 & 2 & -2 \end{array}, \; \mathbf{Q}_{A|b_1} = \begin{array}{c|cc} & a_0 & a_1 \\ a_0 & -3 & 3 \\ a_1 & 4 & -4 \end{array}$$

$$\mathbf{Q}_{B|a_0} = \begin{array}{c|cc} & b_0 & b_1 \\ b_0 & -5 & 5 \\ b_1 & 6 & -6 \end{array}, \; \mathbf{Q}_{B|a_1} = \begin{array}{c|cc} & b_0 & b_1 \\ b_0 & -7 & 7 \\ b_1 & 8 & -8 \end{array}$$

Nodes A and B are two distinct but interdependent subsystems of a larger Markov process.

2.2.2 Amalgamation

While we have shown that the CTBN is able to represent a Markov process as a set of interdependent subsystems, it is also useful to show how the subsystems of a CTBN can be merged together into "supernodes" containing the dynamics of multiple subsystems (Nodelman et al., 2002).

First we introduce additional notation for specific state instantiations and sets of state instantiations. Let $\langle pa_{X\setminus Y} \rangle$ denote the state instantiation $\langle pa_X \rangle$ excluding any state of Y (this changes $\langle pa_X \rangle$ only if Y is a parent of X). Then $\mathbf{Q}_{X|\langle pa_{X\setminus Y}\rangle,Y}$ is the set of conditional intensity matrices that are dependent on the state instantiation $\langle pa_{X\setminus Y} \rangle$ and each state of Y,

$$\mathbf{Q}_{X|\langle pa_{X\setminus Y}\rangle,Y} = \{\mathbf{Q}_{X|\langle pa_{X\setminus Y}\rangle,y} | y \in Y\}.$$

Let $\mathbf{pa}_{X\setminus Y}$ denote the set of all combinations of state instantiations to $\mathbf{Pa}(X)$ excluding any state of Y (again, this changes $\mathbf{pa}_X$ only if Y is a parent of X).

The process involves combining sets of conditional intensity matrices from two different nodes, $\mathbf{Q}_{X|\langle pa_{X\setminus Y}\rangle,Y}$ and $\mathbf{Q}_{Y|\langle pa_{Y\setminus X}\rangle,X}$, and forming a new conditional intensity matrix $\mathbf{Q}_{XY|\langle pa_{XY}\rangle}$, where $\langle pa_{XY} \rangle = \langle pa_{X\setminus Y} \rangle \cup \langle pa_{Y\setminus X} \rangle$. That is, the state instantiations for the parents of X and Y are combined, excluding the state instantiations for X and Y. The state instantiations for X and Y are excluded from $\langle pa_{XY} \rangle$ because $\mathbf{Q}_{XY|\langle pa_{XY}\rangle}$ will be defined over all state combinations of X and Y.

Let $q^X_{i,j}$ be entry i, j of $\mathbf{Q}_{X|\langle pa_X\rangle,y_k}$, and let $q^Y_{k,l}$ be entry k, l of $\mathbf{Q}_{Y|\langle pa_Y\rangle,x_i}$. The combined CIM $\mathbf{Q}_{XY|\langle pa_{XY}\rangle}$ is the matrix defined over the states (x_i, y_k), with the entries populated as follows.

Algorithm 1 Amalgamate two nodes of a CTBN.

```
Amalgamate(X, Y)
 1: $\mathbf{Q}_{XY|\mathbf{Pa}(XY)} \leftarrow \emptyset$
 2: for $\langle pa_{X\setminus Y}\rangle \in \mathbf{pa}_{X\setminus Y}; \langle pa_{Y\setminus X}\rangle \in \mathbf{pa}_{Y\setminus X}$
 3:     $\mathbf{Q}_{XY} \leftarrow \mathbf{0}$
 4:     for $i, j = 1, \ldots, |X|; l, k = 1, \ldots, |Y|$
 5:         $\mathbf{Q}_X \leftarrow \mathbf{Q}_{X|\langle pa_{X\setminus Y}\rangle,x_i}$
 6:         $\mathbf{Q}_Y \leftarrow \mathbf{Q}_{Y|\langle pa_{Y\setminus X}\rangle,y_k}$
 7:         if $i = j \wedge k = l$
 8:             $q^{XY}_{(i,j),(k,l)} \leftarrow q^X_{i,j} + q^Y_{k,l}$
 9:         else if $i = j \wedge k \neq l$
10:             $q^{XY}_{(i,j),(k,l)} \leftarrow q^Y_{k,l}$
11:         else if $i \neq j \wedge k = l$
12:             $q^{XY}_{(i,j),(k,l)} \leftarrow q^X_{i,j}$
13:         end if
14:     end for
15:     $\mathbf{Q}_{XY|\langle pa_{XY}\rangle} \leftarrow \mathbf{Q}_{XY}$
16:     $\mathbf{Q}_{XY|\mathbf{Pa}(XY)} \leftarrow \mathbf{Q}_{XY|\mathbf{Pa}(XY)} \cup \{\mathbf{Q}_{XY|\langle pa_{XY}\rangle}\}$
17: end for
18: return $\mathbf{Q}_{XY|\mathbf{Pa}(XY)}$
```

$$q^{XY}_{(i,j),(k,l)} = \begin{cases} q^X_{i,j} & \text{if } i \neq j \text{ and } k = l \\ q^Y_{k,l} & \text{if } i = j \text{ and } k \neq l \\ q^X_{i,j} + q^Y_{k,l} & \text{if } i = j \text{ and } k = l \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

The CIM $\mathbf{Q}_{XY|\langle pa_{XY}\rangle}$ defines the simultaneous dynamics of X and Y, given that their parents are in states $\langle pa_{XY} \rangle$. Thus, the state-space of XY is the Cartesian product of the states of X and Y, making $\mathbf{Q}_{XY|\langle pa_{XY}\rangle}$ an $|X||Y| \times |X||Y|$ matrix.

Definition 2.3 (Amalgamation)**.** *Amalgamation takes two nodes X and Y and replaces them with node XY, having the set of conditional intensity matrices $\mathbf{Q}_{XY|\mathbf{Pa}(XY)}$ as formed by combining $\mathbf{Q}_{X|\langle pa_{X\setminus Y}\rangle,Y}$ and $\mathbf{Q}_{Y|\langle pa_{Y\setminus X}\rangle,X}$ $\forall \langle pa_{X\setminus Y}\rangle \in \mathbf{pa}_{X\setminus Y}$ and $\forall \langle pa_{Y\setminus X}\rangle \in \mathbf{pa}_{Y\setminus X}$ according to Equation 2. Amalgamation can be viewed as a multiplication operation over sets of conditional intensity matrices and is denoted $\mathbf{Q}_{XY|\mathbf{Pa}(XY)} = \mathbf{Q}_{X|\mathbf{Pa}(X)} \times \mathbf{Q}_{Y|\mathbf{Pa}(Y)}$.*

Amalgamation takes two nodes and combines all of their CIMs, producing a set of CIMs that are conditioned on the combined parent states of X and Y. Thus, the set $\mathbf{Q}_{XY|\mathbf{Pa}(XY)}$ contains $\prod_{Z\in\mathbf{Pa}(XY)} |Z|$ conditional intensity matrices.

Algorithm 1 shows the pseudocode for amalgamating two nodes of a CTBN. Line 1 initializes the empty set of conditional intensity matrices for the amalgamated node. Lines 2-18 iterate over all combinations of parent state instantiations of X and Y, excluding the state of X and Y. Line 3 initializes the conditional intensity matrix to be populated. Lines 4-14 iterate over the state combinations of X

and Y. Lines 5-6 assign the conditional intensity matrices to temporary variables for simpler notation. Lines 7-13 populate the parameters of the conditional intensity matrix initialized in line 3 per Equation 2. Lines 15-17 add the conditional intensity matrix to the set of conditional intensity matrices of the amalgamated node, which is returned in Line 19.

Definition 2.4 (Full Joint Intensity Matrix)**.** *The full joint intensity matrix of a CTBN is the matrix resulting from amalgamating all nodes of the CTBN,*

$$\mathbf{Q} = \prod_{X \in \mathcal{N}} \mathbf{Q}_{X|\mathbf{Pa}(X)}.$$

The size of $\mathbf{Q}$ *is* $n \times n$*, where* $n = \prod_{X \in \mathcal{N}} |X|$*.*

For example, the full joint intensity matrix from the CTBN in Section 2.2.1 is as follows.

$$\mathbf{Q}_{AB} =$$

$$\begin{array}{c} \\ (a_0, b_0) \\ (a_0, b_1) \\ (a_1, b_0) \\ (a_1, b_1) \end{array} \begin{array}{c} \begin{array}{cccc} (a_0, b_0) & (a_0, b_1) & (a_1, b_0) & (a_1, b_1) \end{array} \\ \left(\begin{array}{cccc} -6 & 5 & 1 & 0 \\ 6 & -9 & 0 & 3 \\ 2 & 0 & -9 & 7 \\ 0 & 4 & 8 & -12 \end{array} \right) \end{array}$$

3 NODE MARGINALIZATION IN THE LIMIT

Now we want to address the problem of computing stationary distributions for nodes in a CTBN $\mathcal{N}$. Formally, we want to compute $\boldsymbol{\pi}_X$ for $X \in \mathbf{X}$. We can calculate the stationary distribution for a node X having no parents in $\mathcal{N}$ by simply using the approach of Equation 1 on the single intensity matrix of X (provided that X is irreducible). For nodes with dependencies (which is the point of the CTBN model), each node's stationary distribution depends on all ancestors in the network. In the worst case, a node could have all other nodes as ancestors (one of our experiments is a case of this). But as we have shown, the number of equations is exponential in the size of the network when working with the full joint intensity matrix directly.

We need to perform node marginalization so as to remove a node's dependence on its parents. This will allow us to contain the problem to individual subnetworks and not the entire network. Marginalization methods for CTBNs have been developed in the past, most notably expectation propagation (Nodelman et al., 2005) and belief propagation (El-Hay et al., 2010). Both of these methods approximate a node's unconditional intensity matrix; however, each of the unconditional intensity matrices by these methods are computed for a specific interval of constant evidence. They are not intended to describe the dynamics of a node as $t \to \infty$, which is what we need if we are to compute stationary distributions. The remainder of this section develops a novel CTBN node marginalization method that computes a long-run unconditional intensity matrix.

3.1 THEORY

The key to computing the stationary distributions of nodes in the CTBN is to compute stationary distributions of subsystems of a Markov process. This allows us to work with subsets of nodes, instead of dealing with $\mathbf{Q}$ all at once. Let S be the starting subsystem and D be the destination subsystem. We now want to compute the rate at which S transitions to D in the limit. Formally, we want a tractable way to evaluate

$$q^X_{S,D} = \lim_{t \to \infty} \lim_{h \to 0^+} \frac{P(X(t+h) \in D | X(t) \in S)}{h}. \quad (3)$$

Theorem 3.1. *For a Markov process with disjoint subsystems S and D and $q^X_{S,D}$ as defined above, then*

$$q^X_{S,D} = \frac{1}{Z} \sum_{i \in S} \pi^X_i \sum_{j \in D} q^X_{i,j}$$

where $Z = \sum_{i \in S} \pi^X_i$.

Proof. Because S is a set of multiple states when conditioning on $X(t) \in S$, we must weight each state i in S by the probability of being in state i at time t and renormalize.

$$\begin{aligned}
\lim_{t \to \infty} \lim_{h \to 0^+} \frac{P(X(t+h) \in D | X(t) \in S)}{h} &= \\
\lim_{t \to \infty} \lim_{h \to 0^+} \frac{1}{\sum_{i \in S} P(X(t) = i)} \sum_{i \in S} P(X(t) = i) &\times \\
\sum_{j \in D} \frac{P(X(t+h) = j | X(t) = i)}{h} &= \\
\lim_{t \to \infty} \frac{1}{\sum_{i \in S} P(X(t) = i)} \sum_{i \in S} P(X(t) = i) &\times \\
\sum_{j \in D} \lim_{h \to 0^+} \frac{P(X(t+h) = j | X(t) = i)}{h} &= \\
\lim_{t \to \infty} \frac{1}{\sum_{i \in S} P(X(t) = i)} \sum_{i \in S} P(X(t) = i) \sum_{j \in D} q^X_{ij} &= \\
\frac{1}{\sum_{i \in S} \pi^X_i} \sum_{i \in S} \pi^X_i \sum_{j \in D} q^X_{ij} &= \\
\frac{1}{Z} \sum_{i \in S} \pi^X_i \sum_{j \in D} q^X_{ij} &
\end{aligned}$$

□

Corollary 3.2. *Suppose that a Markov process X comprises n disjoint subsystems $\{S_1, \ldots, S_n\}$. Then $q^X_{S_i,S_i} = \sum_{j=1, j \neq i}^{n} q^X_{S_i,S_j}$.*

Algorithm 2 Compute long-run unconditional intensity matrix of a node.

```
MarginalizeNode($Fam_X$)
1: for $i = 1, \dots, |X|$
2:     for $j = 1, \dots, |X|$ s.t. $j \neq i$
3:         $Z \leftarrow \sum_{k \in S_i} \pi_k^X$
4:         $q_{i,j}^X \leftarrow \frac{1}{Z} \sum_{k \in S_i} \pi_k^X \sum_{l \in D_j} q_{k,l}^{Fam_X}$
5:     end for
6:     $q_{i,i}^X = \sum_{j=1, j \neq i}^{|X|} q_{i,j}^X$
7: end for
8: return $\mathbf{Q}_X$
```

Proof. This follows from the definition of a Markov process, which constrains $q_{i,i}^X = \sum_{j \neq i} q_{i,j}^X$. In other words, if we know the rate at which the process leaves one subsystem and enters every other subsystem, we also know the rate with which the process leaves the subsystem. □

3.2 ALGORITHMS

Now we apply this idea of computing long-run transition rates for Markov process subsystems to amalgamated nodes in a CTBN. In this case, we take the subsystems to be the states of a child node in an amalgamation of the child and its parents. After computing the long-run transition rates for the subsystems, we can construct an unconditional intensity matrix for the child node.

Algorithm 2 computes a long-run unconditional intensity matrix for node X from a set of amalgamated nodes Fam_X that includes X and all parents of X. The variables i and j iterate over the rows and columns, respectively, for the unconditional intensity matrix of X. Line 3 computes the normalization constant. The sets S_i and D_j are the states in Fam_X that include state i and state j of X, respectively. Line 4 computes the long-run transition rate of node X from state i to state j, according to Theorem 3.1. Line 6 computes the long-run sojourn rate of state i of node X, according to Corollary 3.2. Line 8 returns the long-run unconditional intensity matrix of X that are populated by entries $q_{i,j}^X$. The complexity of Algorithm 2 is dominated by the computation of the stationary distribution. Assuming $\mathbf{Q}_X$ is non-singular, its stationary distribution can be computed in $O(n^3)$.

Now that we can compute a long-run unconditional intensity matrix for a node, we can break the dependence of the child on its parents. The unconditional intensity matrix computed for the child will already incorporate the stationary distribution of the parents. We can repeat the process in a top-down fashion through the network, computing the stationary distributions of every node in the network without having to deal with the entire network all at once. Algorithm 3 calculates the long-run unconditional intensity matrices for all of the nodes in a CTBN.

Algorithm 3 Compute long-run unconditional intensity matrices of a CTBN.

```
MarginalizeCTBN($\mathcal{G}$)
1: $\mathcal{G}' \leftarrow$ CollapseCycles($\mathcal{G}$)
2: repeat until termination
3:     $\mathbf{L}_1 \leftarrow \bigcup_{X \in \mathcal{G}'} X$ s.t. $\mathbf{Pa}(X) = null$
4:     $\mathbf{L}_2 \leftarrow \bigcup_{X \in \mathbf{L}_1} \mathbf{Ch}(X)$ s.t. $\mathbf{Pa}(\mathbf{Ch}(X)) \subseteq \mathbf{L}_1$
5:     if $\mathbf{L}_2 = null$ then terminate
6:     for $X \in \mathbf{L}_2$
7:         $Fam_X \leftarrow X$
8:         for $Y \in \mathbf{Pa}(X)$
9:             $Fam_X \leftarrow$ Amalgamate($Fam_X, Y$)
10:        end for
11:        $\mathbf{Q}_{X'} \leftarrow$ MarginalizeNode($Fam_X$)
12:        for $Y \in \mathbf{Pa}(X)$
13:            remove edge $(Y, X)$ from $\mathcal{G}'$
14:        end for
15:        $\mathbf{Q}_X \leftarrow \mathbf{Q}_{X'}$
16:    end for
17: end repeat
```

We need to turn $\mathcal{G}$ into a directed acyclic graph (DAG) from which we can divide the graph into top-to-bottom levels. The behavior of a node depends on all of its ancestors, thus to create a DAG we need to amalgamate all the nodes of each cycle. In the next section we will show an approximation step that avoids collapsing cycles when the cycles themselves are too large for their amalgamation to be tractable.

In Algorithm 3, line 1 collapses the cycles in the $\mathcal{G}$ by amalgamating all of the nodes in each cycle. Lines 2-17 iterate over the levels of the DAG. Lines 3-4 find all of the 2nd-level nodes, i.e., nodes with no other ancestors than their immediate parents. If there are no more 2nd-level nodes, then all of the nodes have been marginalized (no nodes have parents), and line 5 terminates the loop. Lines 6-16 iterate over all of the 2nd-level nodes. Lines 7-10 amalgamate each 2nd-level node with all of its parents. Line 11 computes the long-run unconditional intensity matrix for each 2nd-level node. Lines 12-14 remove the dependency of the node on its parents, and line 15 updates the node's set of intensity matrices with the single intensity matrix from line 11. At the conclusion of the algorithm, the nodes of $\mathcal{N}$ are individual unconditional Markov processes. Note that the stationary distributions are computed along the way by Algorithm 2. The complexity of the algorithm is dominated by the maximum number of parents of any node (analogous to tree-width in Bayesian network inference).

3.3 APPROXIMATION FOR CYCLES

One difficulty of node marginalization is that the dynamics of a node depend on all of its ancestors. If the network is a directed acyclic graph (DAG), then we can marginalize each level 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 marginalize any node in the cycle, because every node in the cycle is an ancestor of every other node in the cycle.

We can address this complication by adding an iterative step to our long-run node marginalization algorithm that avoids dealing with the entire cycle all at once. First, we identify the cycles and all of the nodes they comprise. Let $\mathbf{X}_C$ denote the set of arbitrarily chosen nodes that cover all of the cycles (it is possible that a single node could cover multiple cycles). The set $\mathbf{X}_C$ covers a cycle when at least one node in $\mathbf{X}_C$ is part of the cycle.

For each $X \in \mathbf{X}_C$, we temporarily remove all incoming arcs. Previously, the node had a set of conditional intensity matrices, whereas now we need to replace it with one unconditional intensity matrix. While this unconditional intensity matrix depends on the dynamics of the parents that were just removed, we simply use an unconditional intensity matrix that is the average of the node's conditional intensity matrices. Formally, for each $X \in \mathbf{X}_C$, we remove the incoming arcs to X and estimate an initial unconditional intensity matrix for X as

$$\widehat{\mathbf{Q}}_X \leftarrow \frac{1}{|\mathbf{Q}_{X|\mathbf{Pa}(X)}|} \sum_{\mathbf{Q}_{X|\langle pa_X \rangle} \in \mathbf{Q}_{X|\mathbf{Pa}(X)}} \mathbf{Q}_{X|\langle pa_X \rangle}.$$

Once we have done this for every cycle, the graph becomes a DAG, and we run the *MarginalizeCTBN* algorithm as before.

Depending on the actual parameters, the resulting unconditional intensity matrices could be a poor approximation, because of how we estimated the unconditional intensity matrix of the nodes in $\mathbf{X}_C$. Now we can improve on our estimates for $\widehat{\mathbf{Q}}_X$ because, after the first iteration, we have an unconditional intensity matrix for every immediate parent of X. So we add back all of the incoming nodes of each $X \in \mathbf{X}_C$ and call *MarginalizeNode* on each of these nodes. This results in updated estimates for each $\widehat{\mathbf{Q}}_X$ which now take into account an estimate of the dynamics of the parents of each X. Now we have the original DAG with updated unconditional intensity matrices for each $X \in \mathbf{X}_C$. We can call *MarginalizeCTBN* again.

This process continues to loop around the cycles until convergence. This process is analogous to loopy belief propagation in cyclic graphs, such as in Markov random fields and in Bayesian networks in which the acyclic constraint has been relaxed (Koller & Friedman, 2009). The long-run unconditional intensity matrices are approximations in this case, because we have never viewed the cycle as a whole. On the other hand, if the cycles have too many nodes, we have kept the problem tractable.

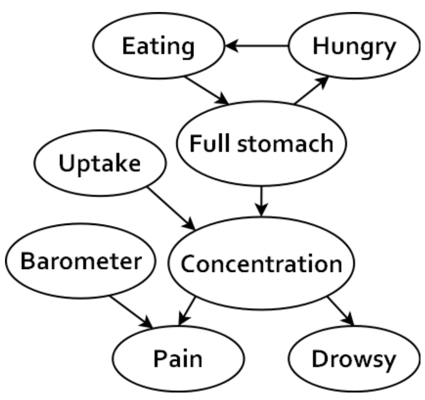

Figure 1: Drug effect network.

4 EXPERIMENTS

We demonstrate the long-run marginalization methods on three networks—two synthetic networks and one real-world network. The two synthetic networks are small enough that we can compute the stationary distributions from the full joint intensity matrix. For the real-world network, we can approximate the stationary distributions by forward sampling the CTBN long enough into the future such that the samples will have converged to the stationary distribution.

4.1 DRUG EFFECT NETWORK

First, we used the drug effect network from Nodelman et al. (2002) as shown in Figure 1. The network is a toy model that shows the interaction of several variables on a patient's pain and drowsiness.

We collapse the *Hungry* $\rightarrow$ *Eating* $\rightarrow$ *Full Stomach* cycle and marginalize *Concentration*. The unconditional intensity matrix of *Concentration* is

$$\mathbf{Q}_{Concentration} \approx \begin{pmatrix} -0.02 & 0.01 & 0.01 \\ 0.25 & -0.26 & 0.01 \\ 0.01 & 0.50 & -0.51 \end{pmatrix}.$$

We then marginalize *Pain* and *Drowsy*, which yields,

$$\mathbf{Q}_{Pain} \approx \begin{pmatrix} -0.56 & 0.56 \\ 0.28 & -0.28 \end{pmatrix}$$

and

$$\mathbf{Q}_{Drowsy} \approx \begin{pmatrix} -0.18 & 0.18 \\ 0.46 & -0.46 \end{pmatrix}.$$

The calculated stationary distributions from both the marginalized nodes and the full joint intensity matrix are

$$\begin{aligned}\pi_{\textit{pain}} &\approx 0.336\\ \pi_{\textit{pain-free}} &\approx 0.664\\ \pi_{\textit{drowsy}} &\approx 0.713\\ \pi_{\textit{non-drowsy}} &\approx 0.287.\end{aligned}$$

However, using the full joint intensity matrix (brute-force method) required solving a system of 864 equations. Through our long-run node marginalization method, we needed to solve systems of 72, 12, and 4 equations. In other words, the complexity has been reduced by approximately a factor of 10.

4.2 RING NETWORK

This second experiment tests our long-run marginalization method on cyclic networks of varying length. For a ring network of size n, we construct the network by adding n three-state (s_0, s_1, s_2) nodes and connecting them as follows:

$$X_1 \to X_2 \to \cdots \to X_n \to X_1.$$

Let each $q^k_{i,j}$ be an independent sample from a uniform distribution over the interval $(0, 1)$. The conditional intensity matrices for the nodes are defined as follows (for ease of definition, X_0 and X_n denote the same node):

$$\begin{aligned}\mathbf{Q}_{X_k|X_{k-1}=s_0} &= \begin{pmatrix}-q^k_{1,1} & q^k_{1,1} & 0\\ 0 & -q^k_{1,2} & q^k_{1,2}\\ q^k_{1,3} & 0 & -q^k_{1,3}\end{pmatrix},\\ \mathbf{Q}_{X_k|X_{k-1}=s_1} &= \begin{pmatrix}-q^k_{2,1} & \frac{q^k_{2,1}}{2} & \frac{q^k_{2,1}}{2}\\ \frac{q^k_{2,2}}{2} & -q^k_{2,2} & \frac{q^k_{2,2}}{2}\\ \frac{q^k_{2,3}}{2} & \frac{q^k_{2,3}}{2} & -q^k_{2,3}\end{pmatrix},\\ \mathbf{Q}_{X_k|X_{k-1}=s_2} &= \begin{pmatrix}-q^k_{3,1} & 0 & q^k_{3,1}\\ q^k_{3,2} & -q^k_{3,2} & 0\\ 0 & q^k_{3,3} & -q^k_{3,3}\end{pmatrix}.\end{aligned}$$

We vary the length of the cycle from 3 nodes to 8 nodes and, for each cycle length, apply the approximate node marginalization method described in Section 3.3 and compare the accuracy to the results from the brute-force method. (Note that in this case the brute-force and exact node marginalization methods are identical, because the whole network is a cycle.) Because each network is generated with random parameters, we run a total of 100 trials for each cycle length and average the results. We keep track of the average number of iterations for the stationary distribution estimates to converge, and we compute the average KL-divergence of the stationary distributions results using the full joint intensity matrix from the results using the iterative node marginalization method. These results are shown in Table 1.

Table 1: Results for the ring networks.

Cycle Length	Avg. Iterations to Converge	Average KL-Divergence
3	12.1	3.1E-4
4	10.1	5.5E-5
5	8.2	2.1E-5
6	7.6	1.7E-5
7	6.8	1.9E-5
8	6.0	1.4E-5

At least for these randomly generated networks, the iterative node marginalization method maintained accurate estimates of the stationary distributions, and the number of iterations to converge tended to decrease as the cycle grew. Notice that the error decreases as the cycle length increases. For these networks, at least, the dynamics of a node are most influenced by the immediate parent. The second-most influential node is the parent's parent, and so on. As the length of the cycle increases, the influence of the "arc that completes the cycle" (whichever arc one chooses this to be) exerts less influence on the dynamics of the cycle as a whole. Therefore, temporarily removing an arc has a decreasing impact as the cycle becomes larger.

For a 3-node cycle, amalgamating the whole cycle will most likely still be a tractable approach. For longer cycle lengths, on the other hand, the applicability of the iterative node marginalization method becomes more critical. In our setup, every node added to the cycle triples the size of the full joint intensity matrices and hence the system of equations to solve.

4.3 CARDIAC ASSIST SYSTEM

Third, we compared the inference methods on a larger, real-world network. We used the model for a cardiac assist system (CAS), presented by Cao (2011), which is broadly used in the literature and based on a real-world system (Boudali et al., 2007; Portinale et al., 2010). Cao (2011) shows how the CTBN is able to encode Dynamic Fault Trees (DFTs), which are reliability models that use Boolean logic to combine series of lower-level failure events while preserving failure sequence information (Dugan et al., 1992). The intensity matrices of the CTBN are used to represent the gates available in the DFT, including AND, OR, warm spare (WSP), sequence enforcing (SEQ), probabilistic dependency (PDEP), and priority AND (PAND). Our model for this experiment is the DFT for the CAS system represented as a CTBN. Of the various repair policies evaluated by Cao (2011), we use the repair rate of $\mu = 0.1$ (10 hours) for all components.

Figure 2 shows the network, while Table 2 gives the node names. In this model, we are interested in the stationary

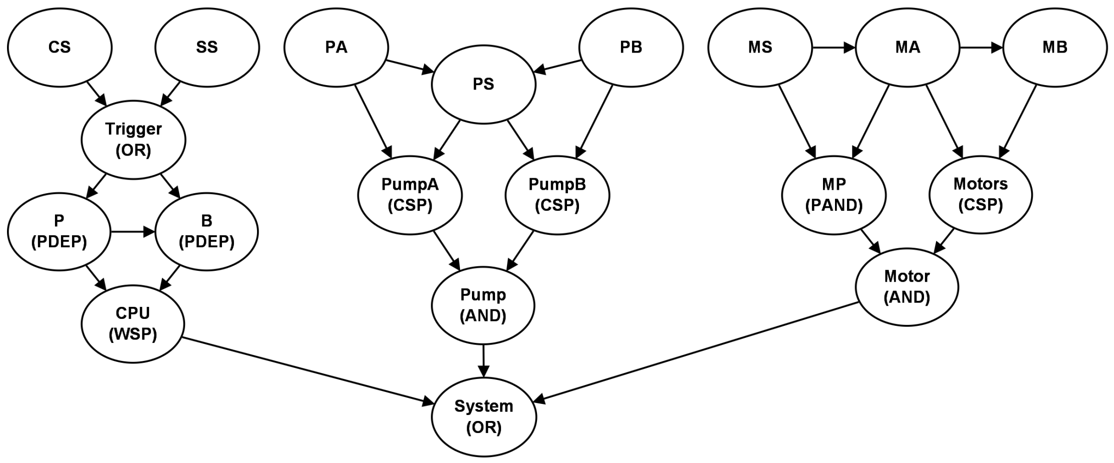

Figure 2: Cardiac assist system model.

distribution of the *System* node, i.e., in the long run, what proportion of the time will the *System* be operational? This corresponds to the operational availability of the subsystem.

Notice that the node in which we are most interested is a descendant of every other node in the network. Using our node marginalization method, we can compute the stationary distribution of the *System* node as

$$\begin{aligned}\pi_{\textit{system-up}} &\approx 0.942\\ \pi_{\textit{system-down}} &\approx 0.058.\end{aligned}$$

If we had attempted to work with the full joint intensity matrix directly, we would be faced with solving a system of over 6.6 million equations. In other words, the brute-force method is intractable for this real-world network. Instead, our node marginalization method divided the network into a total of 14 subnetworks, with the two largest representing systems of only 20 equations.

Instead of trying to solve the system of equations for this large network, we can approximate the stationary distribution by forward sampling the CTBN and observing the convergence of the state probabilities. Because this is an approximation method, we ran multiple trials to quantify the average behavior of the approximation. We ran 100 trials and averaged the results. For each trial, we sampled the network so that we had 10K transitions for the *System* node. Because the dynamics of *System* depend on every other node, we had to sample transitions from all other nodes as well. By the time we had generated 10K transitions for *System*, we had generated more than 100K transitions on average for the other nodes. Our approximation of the stationary distribution of *System* from 10K transitions still resulted in a KL-divergence of 7.2E-3 on average. On the other hand, our node marginalization method computed the exact answer over 3 times faster on average.

Note that our node marginalization method produces another useful output. Because the method computes unconditional intensity matrices along the way, we can observe not only the stationary distributions of different nodes but their long-run expected sojourn times as well. For example, looking at the diagonal entries of the unconditional intensity matrices of *System*, we see that, in the long-run, the mean time between failures (MTBF) for the system is about 153 hours and the mean time to repair (MTTR) is about 9.37 hours. Which of the three subsystems contributes the most to these values? Because of the top-down marginalization process, we have already calculated the same values for each of three subsystems, summarized in Table 3.

We have identified that the *CPU* subsystem contributes the most to *System* failure, while the *Motor* subsystem, due to

Table 2: CAS component names.

Abbreviation	Name	Subsystem
P	primary CPU	CPU
B	warm spare CPU	CPU
CS	cross switch	CPU
SS	system supervision	CPU
MA	primary motor	Motor
MB	cold spare motor	Motor
MS	switching component	Motor
PA	pump A	Pump
PB	pump B	Pump
PS	cold shared pump	Pump

Table 3: Expected sojourn times for CAS subsystems.

Subsystem	MTBF (hrs)	MTTR (hrs)
CPU	154	9.38
Pump	60K	5.00
Motor	410M	6.50

its high reliability rates and its redundancy, very rarely contributes to *System* failures. The *Pump* subsystem is identified as the fastest to be repaired. From long-run analysis on both the stationary distributions and the expected sojourn times, we have efficiently identified and quantified the unreliability of the *CPU* subsystem for efforts to make the CAS more robust and reliable.

5 DISCUSSION

The experiments demonstrate the capability of the exact and approximate node marginalzation methods developed in this paper. We started with two synthetic networks that were small enough to compute the stationary distributions using the traditional approach when viewing a CTBN as a Markov process via its full joint intensity matrix. With the drug effect network, we showed that the exact method computes the same values with a fraction of the computational complexity. With the cyclic network, the brute-force and exact methods become indistinguishable. We showed how an iterative variation of the exact node marginalization method can effectively approximate the stationary distributions of nodes in cycles without handling the entire cycle all at once. Lastly, we applied the node approximation method to a non-trivial real-world network. In this case, working with the full joint intensity matrix (the tradition, brute-force approach) is intractable. We compare our exact node marginalization method to an approximate method based on forward sampling. For this experiment, our node marginalization method is both more efficient and yields the exact answer instead of an approximation.

Our methods assume that the stationary distributions of individual subnetworks can be computed efficiently. Specifically, solving Equation 1 requires the matrix to be nonsingular (i.e., that the Markov process be irreducible). For some CTBNs, this may not be the case, and some subsystems of the process may not be irreducible. Our exact method breaks down in this case. However, note that the traditional approach also breaks down, because it is also based on Equation 1. As long as there exists a method to compute π for a subnetwork (or at least approximate π), this vector can be used in our top-down and/or iterative node marginalization methods.

6 CONCLUSION

We have shown how to compute stationary distributions and long-run expected sojourn times for CTBNs tractably without working directly with the full joint intensity matrix. For CTBNs with long cycles, we have shown an iterative marginalization method that can be used to approximate the long-run behavior. To demonstrate the methods, we tested on three networks of varying complexity and showed the advantage of using our marginalization methods. Future work involves analyzing the behavior of the iterative marginalization method, including research into network topologies and parameters that could make the approximation poor, as well as analyzing convergence properties such as proof of convergence.

References

Boudali, H., Crouzen, P., & Stoelinga, M. (2007). Dynamic fault tree analysis using input/output interactive Markov chains. In *37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks* (pp. 708–717).

Cao, D. (2011). *Novel models and algorithms for systems reliability modeling and optimization.* Wayne State University.

Dugan, J., Bavuso, S., & Boyd, M. (1992). Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, *41*(3), 363–377.

El-Hay, T., Cohn, I., Friedman, N., & Kupferman, R. (2010). Continuous-time belief propagation. In *Proceedings of the 27th International Conference on Machine Learning (ICML).*

Gatti, E. (2011). *Graphical models for continuous time inference and decision making.* Università degli Studi di Milano-Bicocca.

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

Herbrich, R., Graepel, T., & Murphy, B. (2007). Structure from failure. In *Proceedings of the 2nd USENIX workshop on tackling computer systems problems with machine learning techniques* (pp. 1–6).

Koller, D., & Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques.* MIT Press.

Murphy, K. (2002). *Dynamic Bayesian networks: representation, inference and learning.* University of California.

Nodelman, U. (2007). *Continuous time Bayesian networks.* Stanford University.

Nodelman, U., Koller, D., & Shelton, C. (2005). Expectation propagation for continuous time Bayesian networks. In *Proceedings of the Twenty-First Conference Annual Conference on Uncertainty in Artificial Intelligence (UAI-05)* (pp. 431–440). Arlington, Virginia: AUAI Press.

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

Portinale, L., Raiteri, D., & Montani, S. (2010). Supporting reliability engineers in exploiting the power of dynamic Bayesian networks. *International Journal of Approximate Reasoning (IJAR)*, *51*(2), 179–195.

Sturlaugson, L., & Sheppard, J. W. (2015). Sensitivity analysis of continuous time Bayesian network reliability models. *SIAM/ASA Journal on Uncertainty Quantification*, *3*(1), 346–369.

Taylor, H., & Karlin, S. (1998). *An Introduction to Stochastic Modeling*. Academic Press.

Xu, J., & Shelton, C. (2008). Continuous Time Bayesian Networks for Host Level Network Intrusion Detection. In W. Daelemans, B. Goethals, & K. Morik (Eds.), *Machine Learning and Knowledge Discovery in Databases* (Vol. 5212, pp. 613–627). Springer Berlin / Heidelberg.

Xu, J., & Shelton, C. (2010, September). Intrusion detection using continuous time Bayesian networks. *Journal of Artificial Intelligence Research (JAIR)*, *39*(1), 745–774.