UNIVERSITY OF CALIFORNIA
RIVERSIDE

Multivariate Continuous-Time Models: Approximate Inference Algorithms and Medical Informatics Applications

A Dissertation submitted in partial satisfaction
of the requirements for the degree of

Doctor of Philosophy

in

Computer Science

by

Emine Busra Celikkaya

March 2016

Dissertation Committee:

Dr. Christian R. Shelton, Chairperson
Dr. Eamonn Keogh
Dr. Stefano Lonardi
Dr. Vagelis Hristidis

The Dissertation of Emine Busra Celikkaya is approved:

_____

_____

_____

_____

Committee Chairperson

University of California, Riverside

Acknowledgments

I would like to begin by thanking my advisor, Dr. Christian R. Shelton, for his mentoring, support, and extreme patience over the years. It was a privilege to be a part of his research group. His dedication to research and teaching, his knowledge, his brilliant mind and character will always be an inspiration.

I would like to thank my committee members, Dr. Eamonn Keogh, Dr. Stefano Lonardi, and Dr. Vagelis Hristidis for their time and helpful comments, and Dr. Mart Molle and Dr. Ertem Tuncel for being in my committee in the past. I would also like to thank the Children's Hospital Los Angeles VPICU group, especially Dr. Robinder G. Khemani, Dr. Randall C. Wetzel, Dave Kale, and Dr. Patrick A. Ross, for the chance to work on a medical problem with them. My sincere thanks also goes to to Dr. Ibrahim Korpeoglu who encouraged and helped me to pursue this program.

I feel lucky to have known the RLAIR group members, past and present: Teddy Yap, Yu Fan, Antony Lam, Jing Xu, Joon Lee, William Lam, Kevin Horan, Juan Casse, Zhen Qin, Dave Gomboc, Mike Izbicki, Sepideh Azarnoosh, Matthew Zarachoff, Kazi Islam, and Amir Feghahati.

I thank all the wonderful friends I made during my time here: Amy Ricks, Makbule and Abdurrahman Koksal, Ece Gelal, Mustafa Y. Arslan, Priscilla Tang, Gloria Chatzopoulou, Pamela Bhattacharya, Jacqui Gilchrist, Eli H. Brewer, Selda Ors, Ali C. Cirik, Doruk Sart, Onur Turkcu, M. Esat Belviranli, Yuan Hao, Bilson Campana, Zeliha and Cenk Kucukyumuk, and many more.

I thank Muzaffer Akbay for all the love, all the laughs, and the countless times he was there for me.

Lastly, I thank my parents, Reyhan and Hasan Celikkaya, for everything. I could not have made it here without their endless support and understanding.

Parts of Chapter 6 first appeared in Algorithms to Estimate PaCO2 and pH Using Noninvasive Parameters for Children with Hypoxemic Respiratory Failure, Respiratory Care, 2014, 59(8): 1248–1257. Co-authors Dr. Robinder G. Khemani and Dr. Christian R. Shelton directed and supervised the research.

To my late grandmothers Yurdagul and Emine,

and my late grandfathers R. Tosun and Ismail.

ABSTRACT OF THE DISSERTATION

Multivariate Continuous-Time Models: Approximate Inference Algorithms and Medical Informatics Applications

by

Emine Busra Celikkaya

Doctor of Philosophy, Graduate Program in Computer Science
University of California, Riverside, March 2016
Dr. Christian R. Shelton, Chairperson

Temporal modeling of real-life systems, such as social networks, financial markets and medical decision-support systems, is important to understand them better, and make predictions. Temporal data of these systems have irregular time granularity; therefore continuous-time models are a natural fit. In this thesis, we focus on one of the building blocks in this process, statistical inference. Additionally, we apply continuous-time models to a medical informatics application.

In our inference algorithms, we focus on continuous-time Markov processes (CTMPs). Answering queries about the components as a CTMP evolves involves inferring the probability distributions of the state of the system at query time points. When the number of components is large, exact inference becomes intractable since the state space is exponentially large in the number of components. Structured representations provide a framework to apply inference methods in an efficient way. Such representations usually also discretize time. However, choosing the right time-width is challenging since the observations are not synchronized among components and there might be large intervals without any observations. Therefore, our inference algorithms use the structured representation of continuous-time Bayesian networks (CTBNs), but they can also be applied to

other continuous-time representations. A CTBN provides a compact representation using local dependencies. Unfortunately, exploiting the structure in the dynamics does not alleviate the need to represent the full joint space, and exact inference in CTBNs is intractable.

Our approximate inference computations concentrate on the key calculation for a CTMP, the matrix exponential. We use two different expansion of the matrix exponential to derive different approximation algorithms. Our first algorithm keeps the solution in the factorized state space by using uniformization. It is the first non-sampling method to have bounded error. Also, it has better experimental results than the previous methods. Our second algorithm is built upon the sum of time-ordered products. It combines the advantages of deterministic and sampling methods as it is deterministic and anytime. It converges to the true distribution in the limit of infinite computation time, and it is not random. Random methods such as sampling methods can lead to instability when used inside parameter estimation algorithms. We show that it performs as well as or better than the current best sampling approaches on benchmark problems.

Our last work is an application of a multivariate Gaussian process (MGP) to a medical informatics problem. We estimate the blood gas values of a patient during mechanical ventilation in a pediatric intensive care unit. Frequent blood gas values allow more responsive care which can reduce the duration of ventilation and risk of lung injury. Estimating these values from non-invasive measurements can reduce the number of invasive blood tests, which are challenging in children. We estimate them by using previous values of all variables, and current values of all non-invasive variables. We develop an MGP model because the variables are naturally continuous. Our results show promising prediction accuracies, which could be used to automate the ventilation process.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Everyday a great amount of temporal data is generated and stored in various areas of life. Examples include a social network with people forming and breaking relationships in an asynchronous manner; a computer network with computers receiving and sending packets; a patient in an intensive care unit with many procedures and drugs applied and vitals read. Unfortunately, most of this data is not used to its maximum potential. If we could utilize it to obtain meaningful information, we could answer many questions and even increase life quality, as one of the leading areas in temporal data generation is healthcare. However, the biggest obstacle in extracting meaningful information is the computational difficulty, since real-life dynamic systems that generate this data are usually complicated and involve many components.

Modeling such dynamic systems and reasoning about them is very important to understand them better and make predictions that help with the many related tasks. The modeling of dynamic systems includes selecting the model to use, learning the properties of the system and dynamics from the observations, and performing inference to answer queries and make predictions given the observations. Inference methods are the key component in this setting; they are necessary

for parameter estimation, reasoning, and prediction.

Answering queries about the components as the system evolves involves inferring the probability distributions of the state of the system in time. When the number of components is large, exact inference becomes intractable since the state space is exponentially large in the number of state variables. Structured representations of the system dynamics provide a framework to apply inference methods in an efficient way. For discrete time systems, a dynamic Bayesian network (DBN) [Dean and Kanazawa, 1989] is one such example. However, discretizing time when modeling a real-life system has many disadvantages, and it introduces a level of approximation. Choosing the right resolution to discretize the time is especially challenging since the observations are not synchronized among components and there might be large intervals without any observations. There might be components that evolve at different rates making the computation inefficient. For example, a patient's respiratory rate might change more rapidly than their temperature.

By contrast, a continuous-time Bayesian network (CTBN) [Nodelman et al., 2002] provides a representation for continuous-time systems where temporal dynamics are explicit. The dynamics are specified by decomposing the system into components and describing each component's dynamics as a function of its parents in a directed graph. This representation can answer queries for when specific events happen in the system. However, exploiting the structure in the dynamics does not alleviate the need to represent the state distribution of the joint space, which is exponential in the number of components. Therefore, approximate inference methods are used for such systems. In this work we study approximate inference methods aimed for CTBNs, but they can also be applied to other representations.

Apart from CTBNs, Petri nets also have the potential to represent a CTMP in a compact way [Kartson et al., 1994]. A different approach is to use symbolic encodings for the rate matrix

of CTMP and for approximate probability distribution vectors, as in matrix diagram data structures [Ciardo and Miner, 2005]. In our work, we choose to work on CTBNs as prior work in posterior estimation of non-steady state properties has focused on CTBNs.

In this thesis, we focus on the inference task and its applications for continuous-time models. Especially of interest are continuous-time Markovian models and Gaussian process models both of which have a wide applicability. We work on continuous-time Markov processes (CTMP) with discrete state spaces for our inference algorithms in Chapters 4 and 5. For modeling the patient monitor system in our medical informatics application in Chapter 6, which have continuous variables, we turn to a multivariate Gaussian process.

1.1 Contributions

We developed two inference methods for Markovian models. The first one is a filtering algorithm with deterministic approximation (Chapter 4). It uses uniformization with the CTBN factored representation to keep the state distribution in factored state space. We provide a theoretical analysis of its error bound, and show that it does not increase asymptotically.

The second inference method we developed is both deterministic and also can be stopped anytime to obtain a valid solution (Chapter 5). It fills the need for an anytime algorithm that converges to the true result without being random. This is especially useful in parameter estimation algorithms such as expectation-maximization [Dempster et al., 1977]. In this method, we use CTBN representation with the time-ordered products, and we can calculate any expectation of the marginal state distributions.

Lastly, in Chapter 6, we apply continuous-time models to medical informatics. The setting

of our problem necessitates using continuous valued variables in our model. Therefore, we use a multivariate Gaussian process instead of a discrete state Markovian model. Our method estimates the blood gas values of pediatric patients with lung disease, during the time they are mechanically ventilated in the PICU. The blood gas values are required for optimum ventilation management, but they involve invasive blood tests. Frequent blood tests are challenging for patients in PICU, therefore our algorithm's estimates might enable better and shorter mechanical ventilation.

Chapter 2

Continuous-Time Markov Processes

A continuous-time Markov process (CTMP) is a continuous-time stochastic process with discrete state space. CTMPs are used as building blocks for models in many different areas. Historically, they are used for performance evaluation and dependability analysis of fault tolerant computer and communication systems, particularly in the area of queueing networks [Reibman and Trivedi, 1988]. They are also used in model checking literature to formalize the probabilities over paths in a system [Aziz et al., 2000]. Additionally, they are widely applied to biological systems [Sandmann and Wolf, 2006], phylogenetics [Mateiu and Rannala, 2006], and financial systems [Duffie and Glynn, 2004].

In general, a dynamic system that is assumed to have the Markov property is modeled by giving assignments to the components of the system. The joint assignment to all components is the state of the system, and it evolves according to system dynamics. Our work in this thesis falls under transient analysis, in which usually of interest is to predict the state of the system at a given time. More specifically, given the state distribution at an initial time point, we want to compute the distribution at a future point by using the rate matrix of the Markov process. The restricting

factor for real-life problems is that the rate matrix cannot be represented explicitly, as the number of states is exponentially large in the number of components. Apart from the state explosion problem, characteristics of the rate matrix (such as stiffness) are also important since they affect the accuracy and stability of the methods.

In this chapter we present an overview of unstructured homogeneous CTMP models and numerical methods for transient analysis since some of these methods constitute the basis of our methods in the Chapters 4 and 5.

Prior work on unstructured CTMP models include a Gibbs sampling method for queueing networks [Sutton and Jordan, 2008], a Markov chain Monte Carlo sampler based on uniformization [Rodrigue et al., 2008], another sampler based on uniformization [Hobolth and Stone, 2009], and a particle-based Monte Carlo sampling approach [Hajiaghayi et al., 2014].

2.1 Representation

Let $X = \{X(t), t \geq 0\}$ be a CTMP model with finite state space $\mathcal{S}$ with cardinality n. Then for all $t \geq 0$, $h \geq 0$, $t > s \geq 0$, and $i, j \in \mathcal{S}$,

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

We limit our discussion to homogeneous CTMPs in which the rate matrix does not vary with time. In this case, the state transition rates of X are captured with an n-by-n rate matrix (or intensity

matrix)

$$Q = \begin{bmatrix} -q_1 & q_{12} & \cdots & q_{1n} \\ q_{21} & -q_2 & \cdots & q_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n1} & q_{n2} & \cdots & -q_n \end{bmatrix},$$

where $q_{ij} \geq 0$, $i \neq j$ is the rate of transitioning from state i to state j. Diagonal elements of the rate matrix are the negative row sums, that is, $q_i = \sum_{i \neq j} q_{ij}$ and q_i represents the rate of the system leaving state i. Specifically,

$$q_{ij} = \lim_{h \to 0} \frac{P(X(t+h) = j | X(t) = i)}{h} \text{ , and}$$
$$q_i = \lim_{h \to 0} \frac{1 - P(X(t+h) = i | X(t) = i)}{h} .$$

If $X(0) = i$, then the time until the transition to another state (or holding time) is exponentially distributed with parameter q_i. The probability density function and the cumulative probability distribution of staying in state i for an amount of time t is respectively

$$f(t) = q_i e^{-q_i t} \text{ , } t \geq 0 \text{ , and}$$
$$F(t) = 1 - e^{-q_i t} \text{ , } t \geq 0 .$$

Correspondingly, the expected time of transitioning from i is $\frac{1}{q_i}$, and given there is a transition, the probability of transitioning from i to j is $\frac{q_{ij}}{q_i}$.

The distribution of the state of the system at time t is represented with a row vector

$p(t) = [P_X^1(t), P_X^2(t), \ldots, P_X^n(t)]$, where $P_X^i(t) = Prob(X(t) = i)$ denotes the probability of being at state i at time t. Given such a CTMP model that is defined by a rate matrix Q, and an initial distribution vector $p(0)$, we want to compute the state distribution at t.

The transient solution of a CTMP with a finite state space satisfies the forward Chapman-Kolmogorov systems of equations [Stewart, 1994]:

$$\frac{dp(t)}{dt} = p(t)Q. \tag{2.1}$$

Therefore, when we are given the initial state distribution of the system as a row vector $p(0)$, we can obtain the solution as a row vector:

$$p(t) = p(0)e^{Qt}\,. \tag{2.2}$$

In this computation, e^{Qt} is the *matrix exponential* of matrix Q, which is the main calculation in transient analysis. The common representation for matrix exponential is given by the Taylor series expansion

$$e^{Qt} = \sum_{k=0}^{\infty} \frac{(Qt)^k}{k!}\,. \tag{2.3}$$

Example 1 *We will use an example CTMP to illustrate calculating $p(t) = p(0)e^{Qt}$. The example CTMP is a 4-state system described by the initial distribution $p(0) = \begin{bmatrix} 0.4 & 0.1 & 0.2 & 0.3 \end{bmatrix}$, and*

the rate matrix

$$Q = \begin{bmatrix} -3 & 2 & 1 & 0 \\ 4 & -5 & 0 & 1 \\ 3 & 0 & -7 & 4 \\ 0 & 3 & 3 & -6 \end{bmatrix}.$$

Then, marginal distribution at $t = 0.5$ *is* $p(t) = p(0)e^{Q0.5}$,

$$p(t) = p(0) \begin{bmatrix} 0.52 & 0.26 & 0.11 & 0.10 \\ 0.49 & 0.30 & 0.10 & 0.11 \\ 0.41 & 0.24 & 0.17 & 0.18 \\ 0.38 & 0.27 & 0.16 & 0.2 \end{bmatrix} = \begin{bmatrix} 0.45 & 0.26 & 0.14 & 0.15 \end{bmatrix} .$$

2.2 Matrix Exponential Calculations

Exponentiation of the rate matrix is an essential computation in CTMP models, solving ordinary differential equations, and control theory. Therefore many numerical methods have been proposed to solve it efficiently and accurately.

From here on, we will denote the initial distribution as p and the distribution at time t as p' for simplicity. There are two main approaches to compute p'. The first way is to calculate e^{Qt} explicitly by forming the full Q matrix, and then, to multiply it with the initial distribution p. We denote methods that take this approach as *explicit matrix exponential methods*. These methods are not suitable for large matrices such as the ones that occur frequently in CTMPs because of memory requirements and accuracy concerns.

The second approach is the direct computation of a vector times matrix exponential, pe^{Qt}, without explicitly computing e^{Qt}. This approach is more appropriate for large and sparse matrices. The matrix exponential is usually full even when Q is sparse, therefore compact representation is not possible. Another reason this approach is convenient is that computing pe^{Qt} directly is usually more numerically accurate. We call such methods *implicit matrix exponential methods*.

In the next section, we give an overview of the explicit and implicit methods, before going into detailed descriptions of prominent implicit matrix exponential methods. Some of these implicit methods are the bases for our works in Chapter 4 and Chapter 5.

2.2.1 Explicit Methods

Moler and Loan [2003] present a survey of these methods, and they conclude that all of those are "dubious" because of stability and accuracy issues. These methods are not applicable to all classes of problems without certain considerations. However, three approaches, which are scaling and squaring, matrix decomposition, and customized ordinary differential equation (ODE) methods, are deemed to be the best candidates. In this section we briefly describe the scaling and squaring and matrix decomposition methods. We leave the discussion of the ODE methods to Section 2.4.

Other methods that are mentioned include polynomials and splitting based methods, but their applicability is very limited. Polynomial based methods use characteristic polynomial of Q, but they first have to compute the characteristic polynomial. They are prone to roundoff errors and they are not time efficient. The splitting methods approximate $e^{Qt} = e^{(A+B)t} \approx e^{At}e^{Bt}$. However, this technique requires matrices A and B with special constraints. Hence, its applicability is limited to only certain settings.

Scaling and Squaring with Rational Function Approximations. The scaling and squaring method is based on the property $e^{Qt} = (e^{(Q/2)t})^2$. In this method, first Q is scaled by a factor of $\frac{1}{m}$ where $m = 2^s$ and s is a positive integer, and the resulting matrix is exponentiated. Then, it is squared repeatedly to reach to $(e^{(Q/m)t})^m$. One disadvantage of this approach is the computational ($\mathcal{O}(n^3)$) and memory ($\mathcal{O}(n^2)$) requirements, when Q is n-by-n [Stewart, 1994]. Moreover, repeatedly squaring the initial matrix exponential may lead to growing roundoff errors, especially if $m \gg 1$.

For computation of the matrix exponential $e^{(Q/m)t}$, different methods can be used. However the ones that are efficient and stable around the origin are favored in order to minimize roundoff errors. Rational approximation methods such as Padé and Chebyshev approximations are preferred in this case. Particularly, the Padé approximation is widely used for dense matrices, and a variant of this method is implemented in MATLAB's `expm` function.

Matrix Decomposition. There are various matrix decomposition methods such as eigenvector decomposition, Jordan canonical form, Schur decomposition and block diagonal method. They involve factorization or decomposition of the matrix into the form: $Q = VDV^{-1}$. The challenge for these methods is to find a V that is not defective to maximize accuracy, while ensuring that D is close to diagonal. The block diagonal methods aim to find a balance between those two important tasks. Although it is accurate, it involves very expensive operations.

Among the methods mentioned above, Schur decomposition has attracted much attention and is very accurate. However, it is expensive and also it can cause underflow when the diagonal elements are nearly equal [Philippe and Sidje, 1995].

2.2.2 Implicit Methods

Explicit computation of the matrix exponential can only be applied to small matrices. Due to memory requirements and accuracy concerns, especially when the matrix is large and sparse, it is more convenient to compute pe^{Qt} directly. In the next sections we give detailed explanations of the main methods that take this approach. Of those, ordinary differential equations and uniformization methods form the basis of our algorithm in Chapter 4, and time ordered product representation forms the basis of Chapter 5.

2.3 Arnoldi Based Krylov Projection

Krylov projection methods are based on the approximation of the m^{th} element of the Taylor series expansion of pe^{Qt} to Krylov subspace [Saad, 1992, Philippe and Sidje, 1995]. This approach has the advantage of reducing the dimension of the problem to m, which usually takes values between 30 and 50 while the order of Q could be thousands or more.

For a square matrix Q of size n-by-n and a vector p of size n, the Krylov subspace of dimension m is the subspace spanned by the following vectors.

$$K_m(p, Q) \equiv span\{p,\, pQ,\, pQ^2,\, \ldots,\, pQ^{m-1}\}.$$

The Krylov subspace includes all the vectors that are a product of polynomials of p and Q up to degree $m-1$. When we consider the Taylor series expansion of pe^{Qt}, we see that if we truncate this expansion at $m-1^{th}$ element, we obtain a vector that is a product of p and an $m-1$ degree polynomial of Q, which is an element in $K_m(p, Q)$. Krylov subspace methods exploit to find the

element of $K_m(p, Q)$ that is closest to pe^{Qt} (in Euclidean distance) which might capture more than the first m elements.

The basis for Krylov subspace is computed by a projection method such as the Arnoldi algorithm (since it constructs an orthonormal basis) [Philippe and Sidje, 1995]. The Arnoldi projection constructs the basis and an upper triangular Hessenberg matrix H_m, which is the projection of Q to the Krylov subspace. H_m is upper triangular, and also small and dense. These properties make the computation of e^{H_m} easier and allows application of explicit matrix exponential methods. Therefore, $e^{(H_m)t}$ is computed using one of these methods, and projected back using the orthonormal basis. Here, the choice of the explicit matrix exponential method is an additional parameter of the algorithm that affects the performance.

Moreover, the accuracy of the matrix exponential computation depends on the norm of the matrix, and therefore none of the numerical methods multiply the matrix by time t directly. Instead, an iterative method is used, where time t is broken into smaller time steps. So, $e^{(H_m)t}$ is computed iteratively using a fixed stepsize which affects roundoff errors and time efficiency. For this reason, stepsize selection is also important for the algorithm.

2.4 Ordinary Differential Equation Solvers

An equation of type $y'(t) = f(t, y)$ is called a first order ODE. We see that forward Chapman-Kolmogorov equations (Equation 2.1) are systems of first order ODEs; hence, given the differential equation $y'(t) = Qt$ and an initial condition such as $y(t = 0) = p$, the unique solution, $y(t) = p' = pe^{Qt}$, can be found using general purpose ODE solvers. These methods discretize the solution interval to small step sizes, denoted as h. Then, starting from an initial value, they compute

all the points in this interval iteratively.

Let y_{i+1} denote $y(t_{i+1})$, the value at the next step t_{i+1}. Methods which use only the preceding step, y_i, to find y_{i+1} are called single-step methods. Methods that use a set of previous solutions y_j, for $j = i-1, i-2, \ldots i-k$; are called multi-step methods. The accuracy and stability of both methods can be improved using adaptive step sizes. In general, single-step methods are numerically more stable than multi-step methods for sufficiently small h.

The most commonly used explicit single-step methods are Runge-Kutta formulas [Press et al., 1992]. In this set of methods, y_{i+1} is calculated by first finding the derivatives, $f(t, y)$, on intermediate points between t_i and $t_i + h$. Then, the derivative at (t_i, y_i) is determined by a linear combination of these derivatives. Lastly, y_{i+1} is obtained using the computed derivative at this point. Previous derivatives are used to determine the next intermediate point in the same iteration to improve accuracy.

These formulas adapt the step size to the current error, thereby allowing for quick progress at times of slow system change. Among these, the Runge-Kutta-Fehlberg (RKF) variations are favored as they are accurate to fifth-order.

2.5 Uniformization

Uniformization method is a mathematical transformation of the matrix exponential. It transforms the Taylor series of pe^{Qt} (Equation 2.3) so that calculation of the sum becomes more accurate and stable. Calculating this series in the original form is prone to roundoff errors. Terms of the Taylor series include successive powers of Q. Since the rate matrix Q has negative diagonal elements, successive terms have alternating signs. This causes numerical problems when the

successive terms are added to the sum.

For this reason, the rate matrix Q of the CTMP is decomposed into a stochastic matrix

$$Q = \alpha(M - I), \tag{2.4}$$

where $|M|_1 = 1$ and $0 \leq m_{ij} \leq 1$. Here, $\alpha \geq \max_i |q_i|$ and it is called the *uniformization rate*. Ideally α should be as small as possible as it represents the rate of the process sampling the intervals. Calculating the series using M is numerically more stable because all the terms are now positive. Additionally, since $|M|_1 = 1$, roundoff errors are not expected to grow.

Example 2 *Uniformization on the Q matrix given in Example 1 results in $\alpha = 7$ and therefore $M = Q/7 + I$ is*

$$M = \begin{bmatrix} -0.43 & 0.29 & 0.14 & 0 \\ 0.57 & -0.71 & 0 & 0.14 \\ 0.43 & 0 & -1 & 0.57 \\ 0 & 0.43 & 0.43 & -0.86 \end{bmatrix} + \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} 0.57 & 0.29 & 0.14 & 0 \\ 0.57 & 0.29 & 0 & 0.14 \\ 0.43 & 0 & 0 & 0.57 \\ 0 & 0.43 & 0.43 & 0.14 \end{bmatrix}.$$

The resulting approximation is truncated after a finite number of terms, at the l^{th} term.

$$\hat{p}' = \sum_{k=0}^{l} e^{-\alpha t} \frac{(\alpha t)^k}{k!} p M^k \, . \tag{2.5}$$

Since the series is truncated, an error is introduced by omitting the rest of the terms. The value of l is chosen so that the error between $\hat{p}'$ and p' is bounded according to the desired error tolerance ϵ.

$$\begin{aligned} \|p' - \hat{p}'\|_1 &\leq \left\| \sum_{k=l+1}^{\infty} e^{-\alpha t} \frac{(\alpha t)^k}{k!} \right\|_1 \\ &\leq 1 - \sum_{k=0}^{l} e^{-\alpha t} \frac{(\alpha t)^k}{k!} \leq \epsilon_{tol} \, . \end{aligned} \tag{2.6}$$

The computation of $\hat{p}'$ might cause underflows for large t values because it involves computing $e^{-\alpha t}$. Therefore in the implementation of the algorithm, $\hat{p}'$ is first scaled down as

$$\hat{p}' \approx p(e^{\alpha\pi} e^{\pi M})^m, \tag{2.7}$$

where $m = t/\lceil \alpha t/\theta \rceil$ and $\pi = t/m$. So, the value of αt is ensured to never exceed the desired parameter, θ. The resulting algorithm will compute $e^{-\alpha\pi} \frac{(\alpha\pi)^k}{k!} p M^k$ in ml iterations according to step size π. In each iteration, a matrix vector product,

$$v^k = (\alpha\pi) v^{k-1} M \, , \tag{2.8}$$

is computed to find the intermediate result of $\hat{p}'$, which is the main calculation.

Uniformization can also be seen as a transformation of a continuous-time Markovian system into a discrete-time one. However, it does not correspond to constructing either the embedded

Markov chain of the continuous-time process, nor to time-slicing the system at regular intervals. It is equivalent to sampling the intervals between *potential* state changes from an Poisson process with a fixed rate α and then sampling a suitable Markov chain just at these time points (with stochastic matrix M), so that the resulting distribution over trajectories matches the original continuous-time Markov system. Note that while the continuous-time process never has a "self-transition" explicitly, M does have non-zero diagonal elements (corresponding to those states whose rates of leaving are not maximal).

Specifically, M can be seen as the rate matrix of a discrete-time Markov process (DTMP), which is equivalent to the CTMP embedded inside a Poisson process with a distribution function [Stewart, 1994]

$$P\{N(t) = k\} = \frac{(\alpha t)^k}{k!} e^{-\alpha t}, \text{ for } k \geq 0 \text{ and } t \geq 0.$$

So, the probability that the DTMP process has k transitions in time interval t is distributed as a Poisson random variable with expected value αt. The DTMP process has the same state space as the CTMP, and it has the same probability of transitioning from state i to j ($i \neq j$) as the CTMP, that is, [de Souza e Silva and Gail, 2000]

$$\frac{q_{ij}/\alpha}{q_i/\alpha} = \frac{q_{ij}}{q_i}.$$

2.5.1 Sparse Uniformization

The main drawback of uniformization is that the number of iterations tend to be large. This is problematic especially when α or t is large since l increases significantly in that case. Sparse uniformization does not modify the number of iterations, but rather decreases the computational

intensity of the vector-matrix product computed in each iteration. During the multiplication, states with low probabilities in the state distribution vector are discarded. As a result, an approximate vector-matrix product is calculated in each iteration. In step k, rather than Equation 2.8, $v^k(M+E_k)$ is computed ($\alpha\pi$ coefficient in Equation 2.8 is omitted for convenience). Sidje et al. [2007] denote that the local errors (E_k) provide a good means to keep track of the global error. They prove that such an approximation introduces a global error which grows linearly with local error in each iteration.

More formally, the multiplication at any iteration, $vM = v(Q/\alpha + I)$, can be computed as

$$vM = \frac{1}{\alpha}\begin{bmatrix} v_1 & v_2 & \cdots & v_n \end{bmatrix}\begin{bmatrix} — & \mathbf{q}_1 & — \\ — & \mathbf{q}_2 & — \\ & \vdots & \\ — & \mathbf{q}_n & — \end{bmatrix} + v = \sum_{j=1}^{n} \frac{v_j}{\alpha}\mathbf{q}_j, \tag{2.9}$$

where $\mathbf{q}_j$, $j = \{1, \ldots, n\}$ are the row vectors of the rate matrix Q. The strategy for the approximation is to compute $\frac{v_j}{\alpha}\mathbf{q}_j$ only to a sufficient accuracy, so that

$$vM \approx \sum_{j \in Supp_\epsilon(v)} \frac{v_j}{\alpha}\mathbf{q}_j, \tag{2.10}$$

where the ϵ-support of v is defined as

$$Supp_\epsilon(v) = \{1 \leq j \leq n | v_j > \epsilon\}. \tag{2.11}$$

This approximation is based on the assumption that when the number of states is very large, the majority of the states in the state distribution vector will be very small or close to zero

(since they sum to 1). According to Sidje et al. [2007], this is possible since a large subset of the states in vector v might not be reachable at every point in time. Hence, states with very low probability will be omitted in the multiplication with respect to the threshold parameter ϵ_{sup}.

Let $\tilde{p}'$ denote the state distribution vector after all the iterations computed as described above. Sidje et al. [2007] show that $\|\hat{p}' - \tilde{p}'\|_1$ is bounded, in order to find the total error of the sparse uniformizaton method. Since we know the error caused by $\hat{p}'$ (Equation 2.6), the total error bound is

$$\|\hat{p}' - \tilde{p}'\|_1 \leq 2\epsilon \sum_{k=1}^{l} (n - |Supp_{\epsilon} v^k|) \leq 2nl\epsilon_{sup} \, . \tag{2.12}$$

If the rate matrix is very large and the states that are dominant at an instance are comparably very small, this approximation will lead to efficient computation. However, this assumption will not hold in all classes of problems.

2.6 Time-Ordered Products

A different representation for matrix exponential comes from decomposing the rate matrix as $Q = A + B$. Note that the commutative property does not hold for the matrix exponential unless the matrices commute. Therefore, $e^{Qt} = e^{(A+B)t}$ is treated as a perturbation of e^{At} in the direction

of Bt [Najfeld and Havel, 1995]. This representation is denoted as

$$\begin{aligned} e^{(A+B)t} &= e^{At} \\ &+ \int_0^t e^{As} B e^{A(t-s)} ds \\ &+ \int_0^t \left(\int_0^s e^{Ar} B e^{A(s-r)} dr \right) B e^{A(t-s)} ds \\ &+ \int_0^t \left[\int_0^s \left(\int_0^r e^{A\tau} B e^{A(r-\tau)} d\tau \right) B e^{A(s-r)} dr \right] B e^{A(t-s)} ds \\ &+ \dots \end{aligned} \tag{2.13}$$

The expansion can be written in a compact form as sum of recursive functions,

$$e^{(A+B)t} = \sum_{l=0}^{\infty} F^l(t) \tag{2.14}$$

where

$$\begin{aligned} F^0(t) &= e^{At} \\ F^1(t) &= \int_0^t F^0(s) B e^{A(t-s)} ds \\ &\dots \\ F^l(t) &= \int_0^t F^{l-1}(s) B e^{A(t-s)]} ds \\ &\dots \end{aligned} \tag{2.15}$$

The sum of all terms in F^l describes a system that evolves according to A instead of Q, except for l time points. At these points, which can be positioned anywhere, the correction $B = Q - A$ is applied. If $A = 0$, then each level is one term in the Taylor expansion of Q.

This series was first explored in quantum field theory [Dyson, 1949], and is called a series

of time-ordered products (TOP), or sometimes a path-ordered exponential. In stochastic processes, Mjolsness and Yosiphon [2006] called it a time-ordered product expansion and used it to guide a sampling algorithm. In Chapter 5, we employ this expansion to derive our deterministic method, Tree of Time-Ordered Products (TTOP).

2.7 Summary

In this chapter, we gave background information on unstructured CTMP models and matrix exponential methods to compute the transient solution. In the next chapter we will explain a structured representation model for CTMPs, continuous-time Bayesian networks (CTBNs).

Chapter 3

Continuous-Time Bayesian Networks

A continuous-time Bayesian network (CTBN) [Nodelman et al., 2002] is a structured representation for a homogeneous CTMP. Most of the time, evolution of one component of a system directly depends on only a subset of other components. A graphical representation of CTBN dynamics of a drug system can be seen in Figure 3.1 [Nodelman et al., 2002]. In this network, we can see that drowsiness changes depending directly on the changes of drug concentration in blood. However, given the change in the concentration, drowsiness do not depend directly on the changes on the uptake of the drug. As seen in this example, a dynamic system's joint state space of all components can be factored into the state space of local dependencies. CTBNs adopt the graphical structure of Bayesian networks to represent the continuous-time dynamics with local components. This factorization leads to a compact representation of the joint system.

Dynamic Bayesian Networks (DBNs) apply this factorization to temporal systems that are assumed to evolve in regular intervals (or time slices). DBNs enable efficient computations for systems where a natural discretization of time is possible. Time discretization might be inefficient

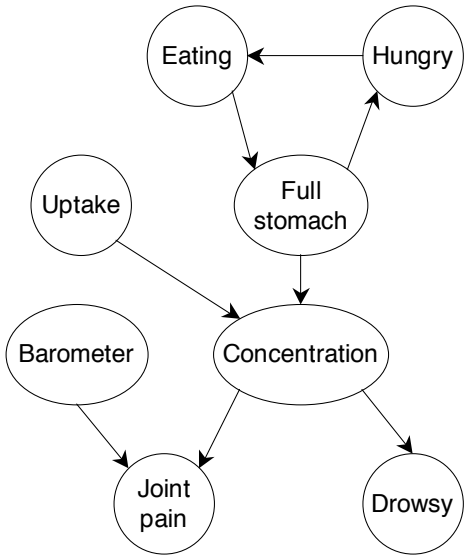


Figure 3.1: CTBN model of drug effect network.

in the case when components evolve in different intervals, since the time slice has to be the smallest interval. In most real-life systems, components do not evolve in regular intervals, so it is already difficult to find a adequate time slice. Moreover, DBNs do not model time explicitly, which makes it hard to answer queries related to the time when a specific event happens in a DBN.

In CTBNs, the structured representation of temporal dynamics is extended to continuous time. When time is modeled explicitly, the system can be propagated from one event to the next without having to propagate through smaller regular intervals. This leads to efficient computations. The factored graphic representation also enables us to exploit the structure in state space in our algorithms for continuous-time systems.

The application capabilities of CTBN models have been demonstrated in many areas. They have been applied to computer user availability prediction [Nodelman and Horvitz, 2003], large server network structure learning [Herbrich et al., 2007], social networks [Fan and Shelton, 2009], network intrusion detection systems [Xu and Shelton, 2010], and cardiogenic heart failure

problems [Gatti et al., 2012].

Additionally, the standard framework of CTBNs have been extended in several approaches. Gopalratnam et al. [2005] present an extension to model non-exponential time distributions using Erlang-Coxian approximations. Portinale and Codetta-Raiteri [2009] propose a generalization by adding variables with no temporal evolution (*immediate* variables). Another extension of CTBNs is a partition-based parameterization [Weiss et al., 2012]. The authors use partitions over joint state space, and have one rate per each element of the partition (instead of using one rate matrix per parents assignment of a local variable). They use regression trees and forests with a multiplicative assumption for the partition-based representation.

Lastly, we note that Shelton and Ciardo [2014] established a connection between CTBNs and edge-valued decision diagrams [Wan et al., 2011], and Qin and Shelton [2015] a connection between CTBNs and piecewise-constant conditional intensity models [Gunawardana et al., 2012]. Also, Portinale and Codetta-Raiteri [2009] established a correspondence between CTBNs and Petri nets. Shelton and Ciardo [2014] give more information about other structured representations and their relationships to each other.

3.1 Representation

In the representation, the initial distribution of CTMP is described by a Bayesian network. The transition model is specified as a directed and possibly cyclic graph, in which the nodes in the model represent variables of the Markov process, and the dynamics of each node depend on the state of its parents in the graph. Each node X_i has a set of parents U_i, for $i = 1, \ldots, m$.

The local dynamics of a variable, which depends on its parents, is modeled by a condi-

tional Markov process. The local dynamics are not homogeneous. At any time point t, they are a function of their parent assignments at t. The rate (or intensity) matrix of the local dynamics is in the form of a conditional intensity matrix (CIM).

Specifically, the rate matrix Q is factored into conditional intensity matrices, $Q_{i|u_i}$ for every assignment of u_i to U_i. Each CIM gives the rates of which variable X_i transitions at instants when $u_i = U_i$. The CIMs for variables can be combined to produce the joint matrix Q that represents the global dynamics of the whole system. For each element of Q, the correct rate is found from the CIMs of the transitioning variable. The CIM of the variable is chosen according to its parents' assignments. Then, the corresponding transition rate from that CIM is inserted into Q. The diagonals of Q are adjusted to be the negative row sums. This operation is called *amalgamation* No two variables can transition at exactly the same instant, so any element in the joint Q matrix describing a change of multiple variables is 0.

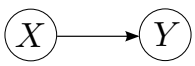


Figure 3.2: Simple CTBN network.

Example 3 *The Q matrix of Example 1 can be represented by a CTBN of two variables: X (with values x_0 and x_1) and Y (with values y_0 and y_1). X has no parents and is the parent of Y as shown in Figure 3.2. We can define CIMs associated with X and Y as*

$$Q_X = \begin{bmatrix} -1 & 1 \\ 3 & -3 \end{bmatrix}, Q_{Y|x_0} = \begin{bmatrix} -2 & 2 \\ 4 & -4 \end{bmatrix}, \text{ and } Q_{Y|x_1} = \begin{bmatrix} -4 & 4 \\ 3 & -3 \end{bmatrix}.$$

The full system can be produced by amalgamation. If the global states are ordered $x_0y_0, x_0y_1, x_1y_0, x_1y_1$,

then we obtain Q matrix from Example 1,

$$Q = \begin{bmatrix} -3 & 2 & 1 & 0 \\ 4 & -5 & 0 & 1 \\ 3 & 0 & -7 & 4 \\ 0 & 3 & 3 & -6 \end{bmatrix}.$$

If we let $\delta(\boldsymbol{x}, \boldsymbol{x}')$ be the set of variable indexes for which the assignments in $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ, the complete Q matrix can also be denoted as

$$Q(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} \sum_{i=1}^{m} Q_{i|\boldsymbol{u}_i}(x_i, x_i') & \text{if } \boldsymbol{x} = \boldsymbol{x}' \\ Q_{j|\boldsymbol{u}_j}(x_j, x_j') & \text{if } \delta(\boldsymbol{x}, \boldsymbol{x}') = \{j\} \\ 0 & \text{otherwise.} \end{cases} \tag{3.1}$$

So Q is sparse (mostly zeros) with non-zero elements on the diagonal and where only one variable changes. It has the form of a sum of factors whereas the joint transition matrix of a DBN has the form of a product of factors.

We can also use Kronecker algebra to construct Q from the conditional intensity matrices. If the nodes in the transition model were independent, we could represent Q as a Kronecker sum, $Q = \bigoplus_{i=1}^{m} Q_i$ where $Q_i = Q_{i|u_i}$ for all u_i (since dynamics do not depend on parents). However, this would not be an interesting system. In general, Q can be represented as a sum of Kronecker

products. Let $R_{i|u_i} = \bigotimes_{i'} R_{i|u_i,i'}$ be a Kronecker product of one matrix for each variable where

$$R_{i|u_i,i'} = \begin{cases} Q_{i|u_i} & \text{if } i' = i \\ \Delta_{k,k} & \text{if } i' \in \text{Pa}(X_i) \text{ \& } k \text{ is val. of } i' \text{ in } u_i \\ I & \text{otherwise .} \end{cases} \tag{3.2}$$

where $\Delta_{k,k}$ is a matrix of all zeros except a single one at location k, k. In this way, $R_{i|u_i}$ distributes the rates in $Q_{i|u_i}$ to the proper locations in Q. The full Q for the CTBN is therefore

$$Q = \sum_{i=1}^{m} \sum_{u_i} \left(\bigotimes_{i'=1}^{m} R_{i|u_i,i'} \right) . \tag{3.3}$$

3.2 Evidence

Observations for a CTBN are in the form of trajectories which contain sequences of states for each variable. If trajectories include all the assignments for the variable during the whole interval (with transitions and transition times), then this is called complete observation. However, states of variables in real-life dynamic systems are rarely observed completely.

Example 4 *Assume we are given the following partial trajectory for our running CTBN example. First, X is observed to be in state x_0 at $t = 0.1$, which is point evidence. Y is observed to be in state y_1 in the interval $[0.25, 0.4)$ which is an example of continuous evidence. Then X is observed to be in state x_1 in the interval $[0.6, 0.9)$, which is again continuous evidence.*

When the variables are observed only some of the time, then we have partial evidence. There are two types of observations in this case. First is called point evidence which is an instan-

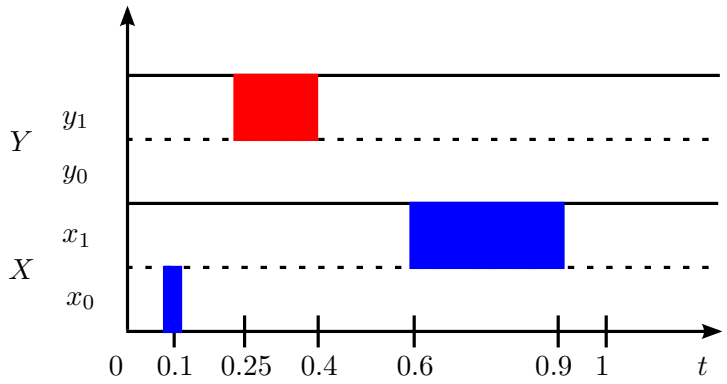


Figure 3.3: Sample CTBN evidence pattern.

taneous observation of the value or the transition of a variable. Second is continuous evidence in which variable values are observed for a time interval. Note that the beginning and ending of the interval do not imply that a transition happened at those times.

Given a set of partial observations of the system, we can learn its parameters and structure by using the expectation-maximizaton and structural expectation-maximization algorithms for CTBNs [Nodelman et al., 2005b] which are adopted from Bayesian networks.

If the parameters and structure of the CTBN are known, many types of queries can be formed. For example, we can query the marginal of a variable at any time point, or the distribution of the first transition time of a variable to a specific state.

Example 5 *Conditioned on the trajectory given in Example 4 (Figure 3.3), we can query the distribution of the state of X at $t = 1$, or the time Y first transitions to y_0.*

3.3 Inference in CTBNs

In order to answer queries we need to incorporate the partial observations to our model. If we only use observations until the query time, then this is called *filtering* If we incorporate evidence

from the whole observation interval, then this is called *smoothing*.

Example 6 *Given the evidence at Example 4, we can query the filtering result for the distribution of the state of X at $t = 1$. We can query the smoothing result for the distribution of the state of Y at $t = 0.5$. As the given evidence includes observations after $t = 0.5$, when we condition on those future observations, we obtain the smoothed distribution.*

3.3.1 Exact Inference

The rate matrix for the joint state space of a CTBN can be generated as explained in Section 3.1. We first generate the joint matrix of the whole system by amalgamation of the CIMs of all the variables (Equation 3.1). Exact inference can be performed by running the forward-backward algorithm using the joint rate matrix [Nodelman et al., 2002].

We divide the trajectory of the CTBN into subtrajectories according to the evidence and perform inference in each interval by conditioning on the evidence of each interval. We condition on point evidence by setting the states with variables having other assignments to probability zero. We condition on continuous evidence by zeroing out the rows and columns of the joint matrix that are inconsistent with the evidence. More details of the forward-backward algorithm for CTBNs can be found in Fan [2009].

Example 7 *We can divide the trajectory in Example 4 to intervals $[0, 0.1)$, $(0.1, 0.25)$, $[0.25, 0.4)$, $[0.4, 0.6)$, $[0.6, 0.9)$, and $[0.9, 1)$. We first propagate the system until $t = 0.1$. Then we set X x_0 at $t = 0.1$, and calculate the distribution at $t = 0.1$. Then we propagate this distribution to $t = 0.25$, and then we set elements that are inconsistent with Y staying at y_1. Then we propagate until $t = 0.4$ with conditioned rate matrix. Then we propagate with the unconditioned rate matrix until $t = 0.6$. We condition the rate matrix similarly for X staying at x_1, and propagate until $t = 0.9$.*

As mentioned before, exact inference is intractable for large systems since the rate matrix size is exponential [Shelton and Ciardo, 2014]. Therefore, many approximate inference methods for CTBNs have been developed.

3.4 Approximate Inference Methods for CTBNs

We can roughly group the approximate inference algorithms for CTBNs as deterministic approximation methods and sampling methods. In deterministic approximation methods, the model of approximation is fixed, and the error depends on predetermined parameters. Since they are deterministic, they are convenient to use inside parameter learning algorithms such as EM. These include variational methods which set inference as an optimization problem, maximize a free energy functional, and find a lower bound on the likelihood by using auxiliary variational parameters. Sampling methods generate samples from the posterior distribution and estimate expectations according to these samples. In contrast to deterministic approximation methods, they converge to the true result as the number of samples increase.

Nodelman et al. [2005a] use a variational algorithm based on expectation propagation [Minka, 2001]. This method divides the given partial trajectory into fixed segments according to continuous or point evidence. For each segment, it uses message passing in cluster graphs, and approximates the marginals as homogeneous Markov processes (using constant CIMs). If the segment duration is long, and the variables inside the segment change rapidly, this results in a poor approximation. Since time segments have to be predetermined, this causes a problem. Saria et al. [2007] improve upon this work and allow for different time segments for each cluster in the message passing scheme. They also dynamically change the duration of time segments depending on how

rapidly the cluster variables change. However, both of these expectation propagation methods may not converge.

Another approach to variational approximations is the mean field method [Cohn et al., 2009, 2010] which approximates the posterior distribution as a product of independent inhomogeneous Markov processes. Each inhomogeneous Markov process is for a single variable, which is more flexible than homogeneous models. An approximate energy functional is decomposed into local terms and solved as an optimization problem, with a set of ODEs for each variable (ODE integration mimics the forward-backward propagation). Therefore, complexity is linear in the number of variables in the network. This approximation converges to a consistent joint distribution (but not to the true result), and it is only good for weakly interacting variables.

Belief propagation method in El-Hay et al. [2010] combines the mean field approach in Cohn et al. [2009] with the expectation propagation method in Saria et al. [2007]. It uses continuously inhomogeneous Markov processes for clusters of variables instead of single variables. Therefore, it is a better approximation than mean field method for tightly coupled variables. As in expectation propagation, the posterier distributions on these clusters are required to be consistent in only on the expected sufficient statistics. The inhomogeneous representation results in simpler mathematics, however the method does not have convergence guarantees.

Our method of factorized filtering in Chapter 4 [Celikkaya et al., 2011] also is a deterministic approximation. It is based on projecting the joint distribution to marginals of variables, using the uniformization method. Its implementation is not extended to smoothing but it has a theoretical error bound (even though it is loose) unlike other work in this category.

As for the sampling methods, the first approach is a continuous-time particle filtering algorithm for an extension of a CTBN to a hybrid of continuous and discrete state systems [Ng

et al., 2005]. However, they assume there is only sparse point evidence.

Fan and Shelton [2008] and Fan et al. [2010] propose an importance sampling algorithm for CTBNs. In the absence of evidence, this method reduces to forward sampling. Given evidence, it uses forward sampling for the proposal distribution and weight the samples according to posterior likelihood. As the algorithm progresses on a trajectory time line, a previously unobserved variable might start being observed. In this case, it uses a predictive lookahead method to match the evidence of that variable. However, if there are unlikely observations concentrated towards the end of the interval, this might still cause a problem. This sampler is also extended to a particle filtering and a particle smoothing methods.

El-Hay et al. [2008] develop a Gibbs sampler for CTBNs. They sample a trajectory of one variable given evidence on the end points of the interval, and the full trajectory of its Markov blanket variables. This method can sample from the exact posterior, but the mixing time of the sampler might be long.

Fan and Shelton [2009] present a Metropolis-Hasting algorithm [Hastings, 1970]. This method starts from a random trajectory, and resamples a randomly selected variable's trajectory using their importance sampler, conditioned the rest of the trajectory. This is done iteratively, and the new trajectory is accepted according to a probability based on the weight from importance sampler. The convergence of this method might take long due to the disadvantage of importance sampler with unlikely evidence.

An auxiliary variable Gibbs sampler is proposed by Rao and Teh [2011, 2013], which constructs a new process by uniformization, (Section 2.5) and uses the auxiliary variables from this process to simplify posterior computations. In this method, they first use uniformization to convert the Markov jump process into a discrete one governed by a Poisson process, and sample

a trajectory from the new process. They resample this trajectory by using Markov chain forward filtering-backward sampling (FFBS) algorithm. Then, they remove self transition times from the resampled trajectory. They continue alternatingly resampling from uniformization and resampling from FFBS algorithm this way. The authors also extend their work to semi-Markov processes [Rao and Teh, 2012].

Lastly, Weiss et al. [2015] propose a rejection based importance sampling method for CTBNs. They learn models for accepting probabilities by sampling variable by variable from existing importance samplers, and accepting based on anticipated future evidence.

In contrast, our algorithm in Chapter 5 [Celikkaya and Shelton, 2014], which uses time-ordered products, has the good properties of both variational and sampling methods. It is a deterministic method that gives the same result in each run for the same problem. It is also an anytime method that can be stopped anytime for a proper result, and converges to the true result given infinite run time.

3.5 Summary

In this chapter we presented an overview of CTBN representation for structured CTMPs. This representation facilitates the computations in our approximate filtering methods. In the next chapters we explain these methods.

Chapter 4

Factored Filtering in Continuous-Time Markov Processes

In this chapter we explain our filtering method for continuous-time stochastic systems where the full distribution over states is too large to be stored or calculated [Celikkaya et al., 2011]. We use a structured representation of the rate matrix of the system, and approximate the belief state in a factored form.

In prior work, the previously described sparse uniformization method (see Section 2.5.1) works well for problems where the distribution to be tracked is highly peaked. However, in problems in which the variables may be more loosely coupled (and therefore an assignment to one does not necessarily dictate a joint assignment to all), this approximation will either be poor, or it will require a large number of states to be tracked, thereby defeating the quick runtime.

By contrast, in this work we are interested in systems with many (relatively) weakly interacting components. For such systems, a factored representation is more suitable.

The structure of the system enables us to decompose the joint belief state to states of a set of components of the system. The factored form is the product of marginals of these components. In order to keep a product of marginals, we manipulate the matrix exponential calculation. We use two of the previously mentioned matrix exponential methods: ODE integration and uniformization of the Taylor expansion. We show how they can be used to keep a factored belief state. We also show that the KL divergence of the approximation stays bounded. Our experimental results confirm our factored uniformization performs better than previously suggested uniformization methods and the mean field algorithm.

4.1 Representation

We are interested in filtering a CTMP that is described by an initial distribution p_0 over the state space and a rate matrix Q. Filtering such a system consists of keeping track of the probability distribution over the state at the current time t, given all evidence prior to t. We would like a recursive solution in which evidence before t can be discarded once the distribution (or its estimate) at t has been computed. We first demonstrate how to do filtering without any evidence, and then explain how to incorporate evidence in Section 4.5.

In the case where there is no evidence, we assume we have p_0 and wish to propagate this distribution to time t by the matrix exponential of Q, $p' = p_0 e^{Qt}$. As even p_0 and p' are not representable exactly for realistic systems with large number of components (Section 2.5.1), we will approximate them with factored representations.

As mentioned in Chapter 2, we assume a state is a set of variables (or components or properties). We follow the same notation as in the previous chapters, and denote variables as X_i

($i \in 1, \ldots, m$), and assignments as x_i. Therefore we can have a factored representation which factors the state into these variables. We approximate the state distribution p_0 as a product of marginals $\hat{p}_0(x) = \prod_i \hat{p}_{0_i}(x_i)$, and approximate p' similarly as $\hat{p}'(x) = \prod_i \hat{p}'_i(x_i)$.

We will use the ODE integrator RKF (Section 2.4), and the uniformization method (Section 2.5) to form our approximate algorithms. In RKF, the fundamental calculation is the multiplication of a current estimate of the distribution by Q to calculate the time derivative. In uniformization, a stochastic matrix M is constructed by using the maximum negative diagonal of Q, $\alpha \geq \max_i q_i$ as $M = Q/\alpha + I$. The main calculation here is the multiplication of current estimate by M.

4.2 Approximate Matrix Exponential Calculations

Both RKF and uniformization methods involve the calculation of state distributions and their multiplication by either Q or M. We will use v to denote any state distribution vector generated during the course of such a calculation. For the RKF method, v is the distribution at a particular point and vQ is its time derivative. For uniformization, v is an element of the sum and vM is the next element.

First, we force the RKF integrator to project v to the space of factored distributions. Second, we force the uniformization method to project v to the space of factored distributions. We call these factored RKF and factored uniformization.

For either factored approximate method, we need to be able to calculate the projection of vM (or vQ) onto the space of factored distributions without explicitly generating vM as an exponentially large vector.

While our methods are applicable to edge-valued decision diagrams [Wan et al., 2011],

Petri nets [Petri, 1962], and other compact representation of the rate matrix, we focus on using a continuous-time Bayesian network (CTBN) to represent Q. As we mentioned before, Shelton and Ciardo [2014] established a connection of CTBNs to edge-valued decision diagrams, and Codetta-Raiteri and Portinale [2010] to Petri nets.

We mainly use the factored representation of the matrix Q in CTBNs (see Section 3.1). The complete Q matrix is represented by using the conditional intensity matrices $Q_{X_i|\boldsymbol{u}_i}$ as in Equation 3.1, where $\boldsymbol{U}_i$ are variable X_i's parents, and $\boldsymbol{u}_i$ are parent assignments. Also $\delta(\boldsymbol{x}, \boldsymbol{x}')$ is the set of variable indexes for which the assignments in $\boldsymbol{x}$ and $\boldsymbol{x}'$ differ.

We will concentrate on approximate calculations of vM (vQ is similar). We consider one subcalculation in the form $v' = vM$. Let $\hat{v}$ be the current approximation of v in factored form: $\hat{v}(\boldsymbol{x}) = \prod_i \hat{v}_i(x_i)$. Similarly, let $\hat{v}' = \hat{v}M$, the result of multiplying $\hat{v}$ by M, which is not necessarily completely factored. Also, let $\tilde{v}$ be the projection of $\hat{v}'$ to the set of factored distributions. Finally, let $M_\perp$ be the operator that both multiplies by M and projects to the space of factored distributions. Thus $\tilde{v} = \hat{v}M_\perp$ is a factored distribution while $\hat{v}'$ is not.

We abuse notation and let any vector also stand for the distribution it embodies. Further, for any vector v, we let v_i be the marginal distribution of v over the variable x_i (even if v is not factored), v_{-i} be the marginal distribution over all variables except x_i, v_{u_i} be the marginal distribution over all of x_i's parents, and $v_{i|-i}$ be the distribution over x_i conditioned on all the other variables.

4.3 Factored Rate Matrix

4.3.1 Factored Uniformization Matrix

To construct the uniformized matrix M, we would like to select α to be the maximally negative diagonal element. In practice this could be a difficult optimization problem. So, instead we select $\alpha = \sum_{i=1}^{n} \alpha_i$ where $\alpha_i = -\min_{\boldsymbol{u}_i} \min_{x_i} Q_{X_i|\boldsymbol{u}_i}(x_i, x_i)$, the maximally negative diagonal element for X_i for any $\boldsymbol{u}_i$. We let $M_{X_i|\boldsymbol{u}_i} = Q_{X_i|\boldsymbol{u}_i}/\alpha_i + I$ be the stochastic matrix we obtain for X_i with parent assignment $\boldsymbol{u}_i$ using uniformization constant α_i. The stochastic matrix M can then be described based on Equation 3.1 as

$$M(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} \sum_k \tilde{M}_{X_k|\boldsymbol{u}_k}(x_k, x'_k) & \text{if } \boldsymbol{x} = \boldsymbol{x}' \\ \tilde{M}_{X_j|\boldsymbol{u}_j}(x_j, x'_j) & \text{if } \delta(\boldsymbol{x}, \boldsymbol{x}') = \{j\} \\ 0 & \text{otherwise.} \end{cases}$$

where $\tilde{M}_{X_i|\boldsymbol{u}_i} = \frac{\alpha_i}{\alpha} M_{X_i \boldsymbol{u}_i} + \frac{\alpha - \alpha_i}{\alpha} I$. The stochastic matrix M cannot be described as a dynamic Bayesian network (DBN): it has an additive structure, not a multiplicative one, and does not allow the transition of multiple variables. However, M can be described as a mixture of particularly simple DBN transition models: $M = \sum_i \frac{\alpha_i}{\alpha} M_i$ where

$$M_i(\boldsymbol{x}, \boldsymbol{x}') = \begin{cases} M_{X_i|\boldsymbol{u}_i}(x_i, x'_i) & \text{if } \boldsymbol{x} = \boldsymbol{x}' \\ M_{X_i|\boldsymbol{u}_i}(x_i, x'_i) & \text{if } \delta(\boldsymbol{x}, \boldsymbol{x}') = \{i\} \\ 0 & \text{otherwise.} \end{cases}$$

In particular, it describes a mixture in which with probability $\frac{\alpha_i}{\alpha}$ variable i transitions according to $M_{X_i|\boldsymbol{U}_i}$ and all other variables remain the same. The ith mixture DBN has a structure in which all variables x'_j, where $j \neq i$, have only x_j (the same node in the previous slice) as a parent. Variable x'_i has the same parents as in the CTBN, with the addition of x_i. No intra-slice arcs exist.

Example 8 *By using Examples 2 and 3, we first calculate $\alpha_X = 3$, $\alpha_Y = 4$ and so $\alpha = 7$ (as before). Then, we can decompose the same M as before into*

$$M_X = \begin{bmatrix} 0.67 & 0.33 \\ 1 & 0 \end{bmatrix}, M_{Y|x_0} = \begin{bmatrix} 0.5 & 0.5 \\ 1 & 0 \end{bmatrix}, M_{Y|x_1} = \begin{bmatrix} 0 & 1 \\ 0.75 & 0.25 \end{bmatrix}$$

which can be viewed as a Markov chain in which with probability $\frac{3}{7}$ X transitions according to M_X, otherwise Y transitions according to $M_{Y|x_0}$ or $M_{Y|x_1}$, depending on the value of X.

4.3.2 Calculating Factored vM

Given the mixture-of-DBN interpretation above of M, it is not surprising that $\tilde{v} = \hat{v}M_{\perp}$ can be calculated in one step without constructing $\hat{v}M$. As Q has the same structure, the same method works for it too.

As projection onto a particular variable, x_j, is linear, if $M_{\perp j}$ is the composition of M and the projection onto x_j, then $M_{\perp j} = \sum_i \frac{\alpha_i}{\alpha} M_{i \perp j}$ where $M_{i \perp j}$ is the composition of M_i and projection operation onto x_j. Furthermore, in M_i only variable i can change:

$$\tilde{v}_j = \hat{v}M_{\perp j} = \sum_i \frac{\alpha_i}{\alpha} \hat{v}M_{i \perp j} = (1 - \frac{\alpha_j}{\alpha})\hat{v}_j + \frac{\alpha_j}{\alpha} \hat{v}M_{j \perp j}\,.$$

$M_{j \perp j}$ is the marginal over x_j after a transition according to M_j. No other variable changes, so this

is the expectation of $M_{X_j|\boldsymbol{U}_j}$ over $\hat{v}$'s distribution over $\boldsymbol{U}_j$:

$$\begin{aligned}\tilde{v}_j &= \hat{v}_j \left[(1 - \frac{\alpha_j}{\alpha})I + \frac{\alpha_j}{\alpha}\textstyle\sum_{\boldsymbol{u}_j} \hat{v}_{u_j}(\boldsymbol{u}_j) M_{X_j|\boldsymbol{u}_j}\right] \\ &= \hat{v}_j \left[\textstyle\sum_{\boldsymbol{u}_j} \hat{v}_{u_j}(\boldsymbol{u}_j) \tilde{M}_{X_j|\boldsymbol{u}_j}\right] \end{aligned} \tag{4.1}$$

To calculate $\hat{v}Q$ and project onto x_j, Equation 4.1 holds if we change $\tilde{M}_{X_j|\boldsymbol{u}_j}$ to $Q_{X_j|\boldsymbol{u}_j}$.

Example 9 *Let starting distribution p marginals be $\hat{v}_X = \begin{bmatrix}0.6 & 0.4\end{bmatrix}$ and $\hat{v}_Y = \begin{bmatrix}0.5 & 0.5\end{bmatrix}$. Multiplying this factored approximation by M and then projecting can be done by multiplying v_X by $\tilde{M}_X$ and v_Y by $0.6\tilde{M}_{Y|x_0} + 0.4\tilde{M}_{Y|x_1}$:*

$$\tilde{M}_X = \begin{bmatrix}0.86 & 0.14 \\ 0.43 & 0.57\end{bmatrix}, \tilde{M}_{Y|x_0} = \begin{bmatrix}0.71 & 0.29 \\ 0.57 & 0.43\end{bmatrix}, \tilde{M}_{Y|x_1} = \begin{bmatrix}0.43 & 0.57 \\ 0.43 & 0.57\end{bmatrix}.$$

4.4 Bounds for Factored Uniformization

While we have not found a suitable way of bounding the approximation error for factored RKF, we can derive bounds similar to those of the BK algorithm [Boyen and Koller, 1998] for discrete-time stochastic processes to bound the error in propagation and projection through a single M matrix. However, because the process is a mixture of processes in which only a single component changes, the BK result for compound processes does not carry over.

We then use this bound to bound the error of the entire Taylor expansion and thereby the factored uniformization method.

4.4.1 Divergence Bound for Single Step

We first concentrate on bounding the error of a single multiplication by M. We wish to show that the KL-divergence between $v' = vM$ and $\hat{v}' = \hat{v}M$ is no greater than that between v and $\hat{v}$.

We begin with a simple property of the KL-divergence. The proof is omitted, but is a consequence of the fact that entropy does not increase upon conditioning.

Lemma 10 *If $q(x) = \prod_i q(x_i)$ is a factored distribution and $p(x)$ is a (non-factored) distribution over the same sample space, and x_{-i} is the set of all variables except x_i,*

$$\sum_i D_{KL}(p(x_i \mid x_{-i}) \| q(x_i)) \geq D_{KL}(p(x) \| q(x)) \ .$$

We require a mixing rate definition from Boyen and Koller [1998]:

Definition 11 *The* mixing rate *of a stochastic matrix M is defined as $\gamma \triangleq \min_{i_1,i_2} \sum_j \min(M_{i_1,j}, M_{i_2,j})$.*

We can then state that M is a contraction mapping with respect to the KL-divergence:

Theorem 12 *Let γ_i be the minimum (over $\boldsymbol{u}_i$) mixing rate of the stochastic matrix $M_{X_i|\boldsymbol{u}_i}$ and $\gamma = \min_i \frac{\alpha_i \gamma_i}{\alpha}$. Then,*

$$D_{KL}(v' \| \hat{v}') \leq (1 - \gamma) D_{KL}(v \| \hat{v}) \ .$$

Proof. If we let $v'^{(i)} = vM_i$ and $\hat{v}'^{(i)} = \hat{v}M_i$, then

$$\begin{aligned}
D_{\text{KL}}(vM\|\hat{v}M) &= D_{\text{KL}}(\textstyle\sum_i \frac{\alpha_i}{\alpha} v'^{(i)} \| \sum_i \frac{\alpha_i}{\alpha} \hat{v}'^{(i)}) \\
&\leq \textstyle\sum_i \frac{\alpha_i}{\alpha} D_{\text{KL}}(v'^{(i)} \| \hat{v}'^{(i)}) \\
&= \textstyle\sum_i \frac{\alpha_i}{\alpha} \big(D_{\text{KL}}(v'^{(i)}_{-i} \| \hat{v}'^{(i)}_{-i}) + D_{\text{KL}}(v'^{(i)}_{i|-i} \| \hat{v}'^{(i)}_{i|-i}) \big) \\
&\leq \textstyle\sum_i \frac{\alpha_i}{\alpha} \big(D_{\text{KL}}(v_{-i} \| \hat{v}_{-i}) + (1-\gamma_i) D_{\text{KL}}(v_{i|-i} \| \hat{v}_{i|-i}) \big) \\
&= \textstyle\sum_i \frac{\alpha_i}{\alpha} \big(D_{\text{KL}}(v \| \hat{v}) - \gamma_i D_{\text{KL}}(v_{i|-i} \| \hat{v}_{i|-i}) \big) \\
&\leq D_{\text{KL}}(v \| \hat{v}) - \gamma \textstyle\sum_i D_{\text{KL}}(v_{i|-i} \| \hat{v}_i) \\
&\leq D_{\text{KL}}(v \| \hat{v}) - \gamma D_{\text{KL}}(v \| \hat{v}) \; .
\end{aligned}$$

The first inequality is from the convexity of the KL-divergence [Cover and Thomas, 1991, Theorem 2.7.2]. The next inequality holds because M_i does not change any variables except x_i and the conditional KL-divergence of x_i contracts by γ_i [Boyen and Koller, 1998, Theorem 3]. The final inequality is due to the lemma above. ■

Note that unlike in BK, the global contraction rate (γ) does not depend on the in- or out-degree of model, but it is inversely proportional to n. This appears unfortunate, but the next section demonstrates that it is not a problem for the contraction rate of the entire Taylor expansion.

We upper-bound the increase from projection:

$$D_{\text{KL}}(vM\|\hat{v}M_\perp) - D_{\text{KL}}(vM\|\hat{v}M) \leq \epsilon \; .$$

As a crude upper bound, $\epsilon \leq -(n-1)\ln\eta$ where η is the smallest marginal probability. As shown by Boyen and Koller [1999], better bounds can be placed by more careful analysis or considering

the average case.

Taken together this means that after a single multiplication and projection, the KL-divergence error of our estimate can be bounded as

$$D_{\text{KL}}(vM \| \hat{v}M_{\perp}) \leq (1-\gamma) D_{\text{KL}}(v \| \hat{v}) + \epsilon \ . \tag{4.2}$$

Example 13 *We have* $\gamma_X = 0.67$*,* $\gamma_{Y|x_0} = 0.5$*,* $\gamma_{Y|x_1} = 0.25$*. Therefore* $\gamma_Y = \min(0.5, 0.25) = 0.25$*, and* $\gamma = \min(0.67 \times \frac{\alpha_X}{\alpha}, 0.25 \times \frac{\alpha_Y}{\alpha}) = \min(0.67 \times \frac{3}{7}, 0.25 \times \frac{4}{7}) = 0.14$*. Therefore the KL-divergence between the true answer and the factored approximation contracts by* $1 - 0.14 = 0.86$ *for each multiplication by* M*.*

4.4.2 Bound on Approximate Taylor Expansion

Our goal is not to bound the error on a single step, but rather the error of our entire approximation to the matrix exponential. Equation 4.2 implies that

$$\begin{aligned} D_{\text{KL}}(vM^k \| \hat{v}M_{\perp}^k) &\leq (1-\gamma)^k D_{\text{KL}}(v \| \hat{v}) + \epsilon \sum_{i=0}^{k-1} (1-\gamma)^i \\ &= (1-\gamma)^k D_{\text{KL}}(v \| \hat{v}) + \epsilon \tfrac{1-(1-\gamma)^k}{\gamma} \ . \end{aligned} \tag{4.3}$$

If we combine the bound from Equation 4.3 with the Taylor expansion of Equation 2.3, we can obtain a bound on the KL-divergence between the true matrix exponential, and an approximation of the Taylor expansion in which each vector of probabilities is approximated by a factored form and only the first l terms are evaluated (and then renormalized):

Theorem 14 *Let* α*,* ϵ*, and* γ *be as defined above. Let* p *be an arbitrary distribution over the*

state space of the process. Let $\hat{p}$ be an arbitrary factored distribution over the same. Further, let $p' = pe^{Qt}$ be the distribution at time t in the future, and let $\hat{p}' = \frac{1}{1-R_l}\sum_{k=0}^{l} e^{-\alpha t}\frac{(\alpha t)^k}{k!}\hat{p}M_\perp^k$ be the approximation of p' constructed by uniformization of a Taylor expansion truncated to l terms in which each matrix multiplication is projected back to the space of factored distributions. $R_l = \sum_{k=l+1}^{\infty} e^{-\alpha t}\frac{(\alpha t)^k}{k!}$. Then

$$D_{KL}(p'\|\hat{p}') \leq e^{-\gamma\alpha t}D_{KL}(p\|\hat{p}) + \frac{\epsilon}{\gamma}(1 - e^{-\gamma\alpha t}) + R_l(\delta + \frac{\epsilon}{\gamma})$$

where $\delta = \max_{\boldsymbol{x}} - \ln \hat{p}'_{\boldsymbol{x}}$, the maximum negative log probability over any joint assignment in the final approximate calculation.

Proof. For compactness of presentation, let $\beta_k = e^{-\alpha t}\frac{(\alpha t)^k}{k!}$, $\bar{\gamma} = (1-\gamma)$, and $\bar{R}_l = (1 - R_l)$. Then

$$\begin{aligned}
D_{\text{KL}}(p'\|\hat{p}') &= D_{\text{KL}}(\textstyle\sum_{k=0}^{\infty}\beta_k pM^k \| \frac{1}{\bar{R}_l}\sum_{k=0}^{l}\beta_k\hat{p}M_\perp^k) \\
&\leq \bar{R}_l D_{\text{KL}}(\tfrac{1}{\bar{R}_l}\textstyle\sum_{k=0}^{l}\beta_k pM^k \| \frac{1}{\bar{R}_l}\sum_{k=0}^{l}\beta_k\hat{p}M_\perp^k) \\
&\quad + R_l D_{\text{KL}}(\tfrac{1}{R_l}\textstyle\sum_{k=l+1}^{\infty} e^{-\alpha t}\beta_k pM^k \| \frac{1}{\bar{R}_l}\sum_{k=0}^{l}\beta_k\hat{p}M_\perp^k) \\
&\leq \textstyle\sum_{k=0}^{l}\beta_k D_{\text{KL}}(pM^k\|\hat{p}M_\perp^k) + R_l\delta \\
&\leq \textstyle\sum_{k=0}^{l}\beta_k[\bar{\gamma}^k D_{\text{KL}}(p\|\hat{p}) + \epsilon(1-\bar{\gamma}^k)/\gamma] + R_l\delta \\
&\leq e^{-\gamma\alpha t}D_{\text{KL}}(p\|\hat{p}) + \tfrac{\epsilon}{\gamma}(1 - e^{-\gamma\alpha t}) + R_l(\delta + \tfrac{\epsilon\bar{\gamma}^l}{\gamma})\,.
\end{aligned}$$

The first inequality is due to the convexity of the KL-divergence in only the first argument. The second is due to the convexity of the KL-divergence in both arguments, and that the KL-divergence is bounded by the negative log of smallest probability of the second argument. The third is due to Theorem 12. The final is due to bounding a finite Taylor expansion of an exponential by the

exponential. ■

Note that R_l goes to zero as l grows. In the remaining terms, γ almost always appears multiplied by α. In the previous section, we commented on how γ is inversely proportional to n. However, α is proportional to n and they exactly cancel out. Thus if l is large enough, we can let $\gamma' = \min_i \alpha_i \gamma_i$ and conclude

$$D_{\text{KL}}(p' \| \hat{p}') \leq e^{-\gamma' t} D_{\text{KL}}(p \| \hat{p}) + \frac{\alpha\epsilon}{\gamma'}(1 - e^{-\gamma' t}) \ . \tag{4.4}$$

Thus the contraction rate for the full approximation is *not* a function of n. It is unclear how $\alpha\epsilon$ scales with n.

We are then left with two error terms. The first decays with t and the second grows with t (but is bounded). The resulting distribution $\hat{p}'$ is a mixture of factored distributions. So, if filtering is to continue past time t, $\hat{p}'$ must be projected back to the space of factored distributions, introducing another similar additive error. This may happen either because evidence arrives at t and requires conditioning, or because we may wish to subdivide a propagation from 0 to t into m propagations of length t/m. Equation 4.4 implies this is not helpful for accuracy (if one considers applying the bound recursively m times over intervals of length t/m). However, our experimental results show that some interval subdivision is helpful.

Example 15 $\gamma' = \gamma\alpha = 1$. *For large l, the KL-Divergence between the true distribution and the factored approximation before (D_{KL}) and after propagation to $t = 0.5$ (D'_{KL}) is related by*

$$D'_{KL} \leq 0.61 D_{KL} + (7\epsilon/1)(1 - 0.61)$$

where we note that $e^{-1\times 0.5} \approx 0.61$. If the smallest possible marginal probability is 0.01, then a

crude upper-bound on ϵ *is* $-\ln 0.01 = 4.6$.

4.5 Adding Evidence

So far, we have discussed only how to filter without evidence. In a continuous-time process, evidence can take on two forms. First, it can be point evidence. That is, at time t we observe the values of certain variables, but for no duration. We propagate to the time of the evidence, condition the distribution on the evidence, and then continue. In this case, conditioning on the evidence in expectation reduces the error [Boyen and Koller, 1998, Fact 1].

The second form of evidence is interval evidence: a variable remains in a particular state from t_1 to t_2. In this case, at t_1 we condition on the evidence (same as for point evidence, above). From t_1 until t_2, we monitor using a modified Q matrix in which all transitions where the evidence variable changes are set to 0 (but the diagonal elements remain unchanged). The resulting $\hat{p}$ sums to the probability of the interval evidence, but it can be normalized to yield the conditional distribution of the state. The normalization makes the analysis difficult. However, we conjecture that interval evidence will also not increase the error in expectation.

Finally we note that the sparse uniformization method can have serious difficulties with evidence if none of the maintained states are consistent with the evidence.

Example 16 *Given the previous distribution* p *for* $t = 0$*, and* $Y = y_0$ *on* $t = [0.5, 1)$*, we propagate* p *to* $t = 0.5$ *using Equation 2.5, with Equation 4.1 to calculate the projected multiplications. We get the factored distribution* $\hat{p}_X = \begin{bmatrix} 0.65 & 0.35 \end{bmatrix}$, $\hat{p}_Y = \begin{bmatrix} 0.56 & 0.44 \end{bmatrix}$. *We condition on* $Y = y_0$ *by*

setting $\hat{p}_Y = \begin{bmatrix} 1 & 0 \end{bmatrix}$. *Then we similarly propagate to* $t = 1$*, but using*

$$Q_X = \begin{bmatrix} -1 & 1 \\ 3 & -3 \end{bmatrix}, Q_{Y|x_0} = \begin{bmatrix} -2 & 0 \\ 0 & 0 \end{bmatrix}, Q_{Y|x_1} = \begin{bmatrix} -4 & 0 \\ 0 & 0 \end{bmatrix}.$$

for which $\alpha_X = 3$*,* $\alpha_Y = 4$*, and* $\alpha = 7$*:*

$$M_X = \begin{bmatrix} 0.5 & 0.5 \\ 1 & 0 \end{bmatrix}, M_{Y|x_0} = \begin{bmatrix} 0.6 & 0 \\ 0 & 1 \end{bmatrix}, M_{Y|x_1} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}.$$

4.6 Experiments

We employed two synthetic networks and a network built from a real data set in our evaluations. Since exact filtering is intractable for large models, we limited the number of variables to allow for calculations of the true approximation errors.

The synthetic networks we use are the ring and toroid dynamic Ising networks of 20 binary variables from El-Hay et al. [2010]. The ring network is bidirectional, while the toroid (Figure 4.1a) is directed. Nodes try to track their parents with a coupling parameter, β, indicating the strength of the influence, and a rate parameter τ that is inversely proportional to the expected time between switching. We set $\tau = 4$ and $\beta = 1$. We set both networks to have a deterministic starting distribution. For the toroid the first 5 variables are in state 0 and the remaining variables are in state 1. The ring network's initial distribution is the reverse.

The real network we used was constructed from the British Household Panel Survey (BHPS) data set [ESRC Research Centre on Micro-social Change, 2003]. The data set records major life changes from a set of roughly 8,000 British citizens in areas including household organization,

employment, income, wealth and health. We use the same network model as in Fan et al. [2010] and Nodelman et al. [2005b] which chooses 4 variables: employment (student, employed, unemployed), children $(0, 1, \geq 2)$, married (not married, married) and smoking (non-smoker, smoker) and adds a hidden binary variable for each (Figure 4.2a). The structure and parameters of both the initial distribution and the dynamics were learned by the structural EM algorithm [Nodelman et al., 2005b] and we used the learned network model for our experiments.

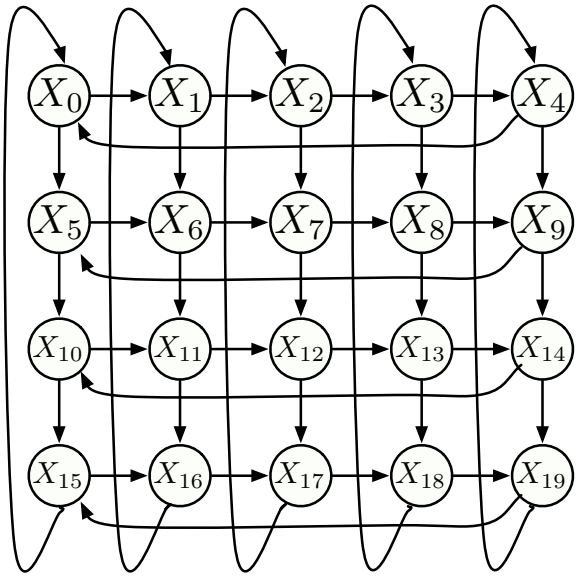
4.6.1 KL-Divergence Bound

Our first experiment tested the theoretic error bound. Figures 4.1b and 4.2b show the KL-divergence between the true marginals and the marginals computed by factored uniformization for the BHPS network and the toroid network. The bound on the KL-divergence of the full distribution is also a bound on any marginal, and experimentally the errors on the marginals grow initially and then asymptote, as per Theorem 14.

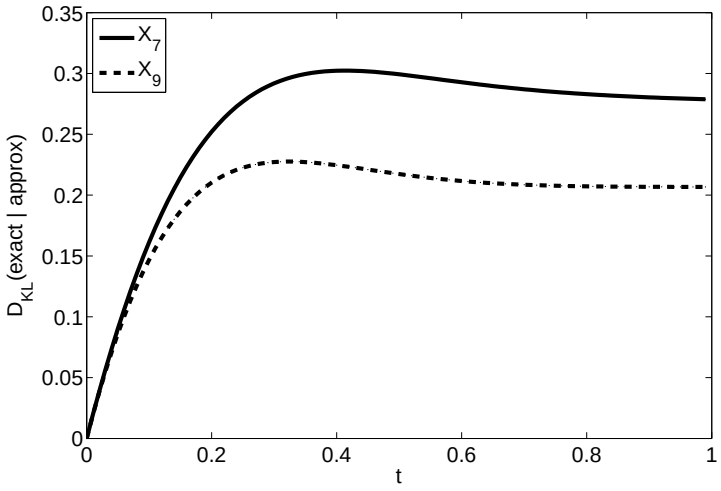
4.6.2 Approximation Comparison

We then compared our factored uniformization method (U_F) to other approximations. In particular, we compared to factored RKF (RKF_F), sparse uniformization (U_S) as explained in Section 2.5.1, and the mean field (MF) approach of Cohn et al. [2009] for CTBNs (Section 3.4). For the purpose of comparison, the MF method was extended to accept evidence on subsets of variables.

We varied an error tolerance parameter for each method to map the trade-off between error and runtime. For U_F, we varied the uniformization-specific parameter θ that determines the

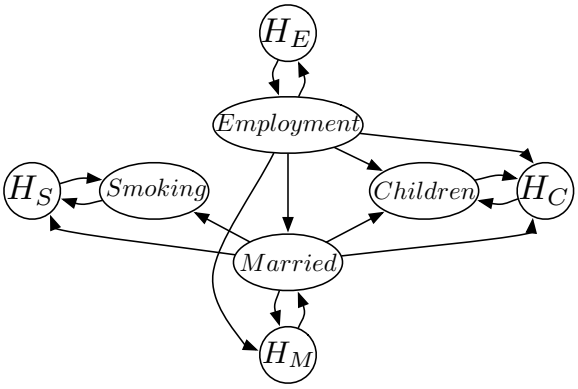


(a) Toroid network

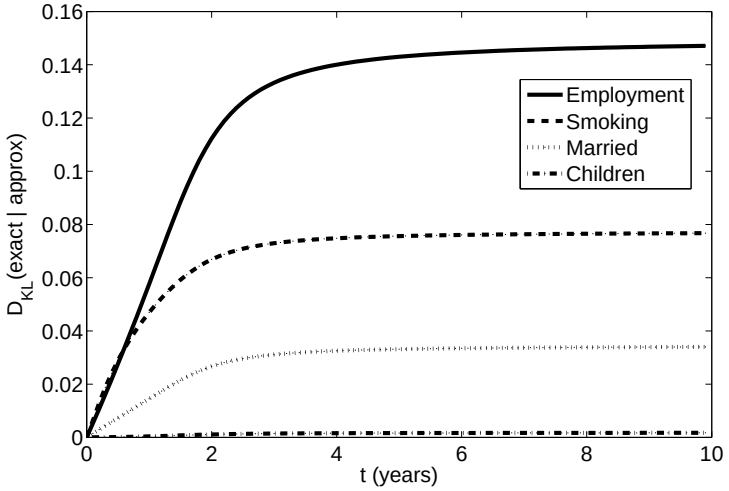


(b) Toroid results

Figure 4.1: Factored uniformization accuracy versus t (interval width) for the toroid network.



(a) BHPS network

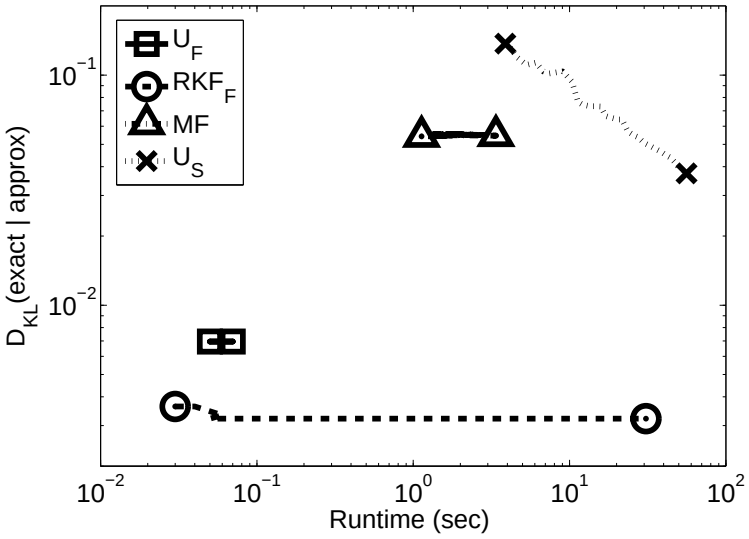


(b) BHPS results

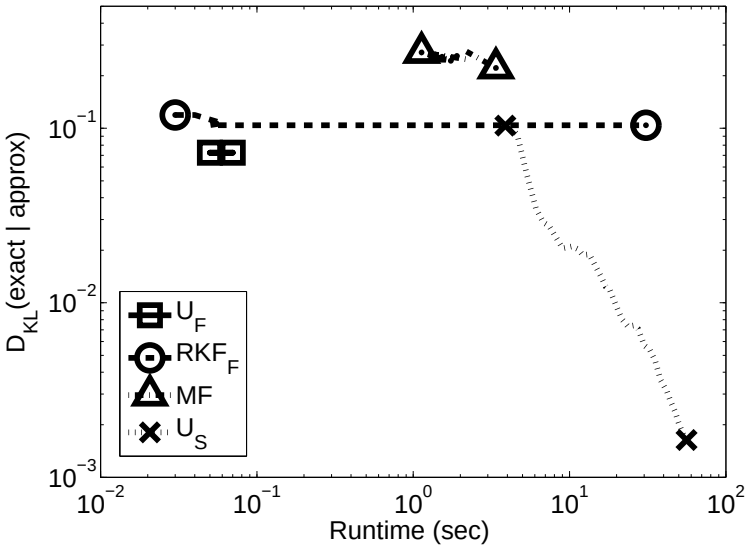
Figure 4.2: Factored uniformization accuracy versus t (interval width) for the BHPS network.

number of intervals of propagation (see Section 2.5). We fixed the number of terms of the Taylor series expansion (l) to a value that performed reasonably. Similarly, for sparse uniformization we varied θ and fixed l to a well-performing value. MF and RKF_F both use RKF for integration, so we varied the error tolerance of RKF.

The evidence for the ring and toroid networks was set to be relatively unexpected: for $t \in [0.5, 1.0)$ $x_0 = 1$ and $x_1 = 0$. For the ring, we queried the distribution of x_{10} (far from the evidence) and x_{19} (adjacent to the evidence) at time $t = 1$. For the toroid, we queried the

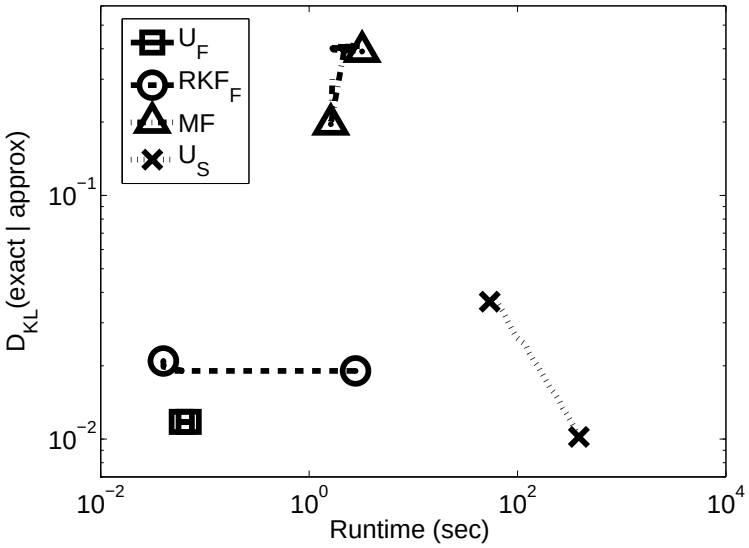


(a) Ring, x_{10}

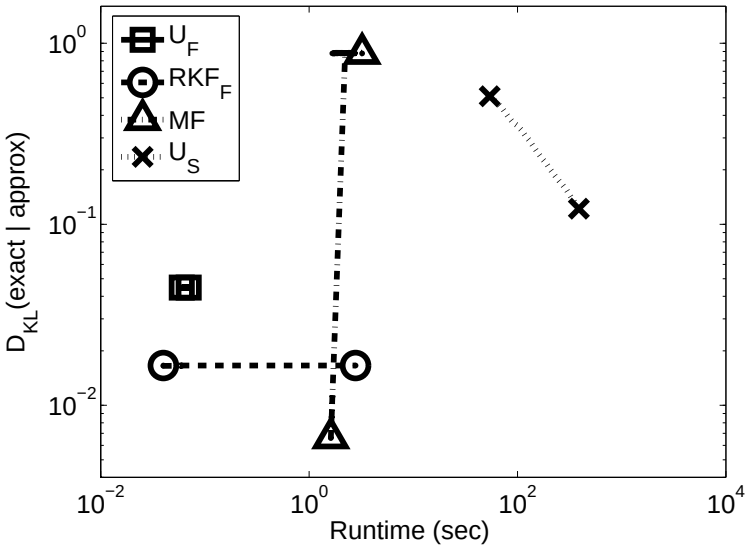


(b) Ring, x_{19}

Figure 4.3: Computation time versus accuracy comparisons for ring network.

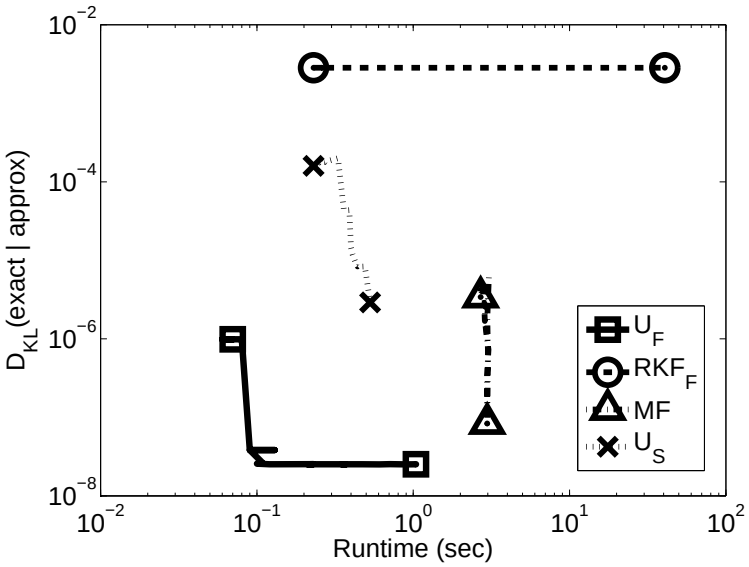


(a) Toroid, x_6

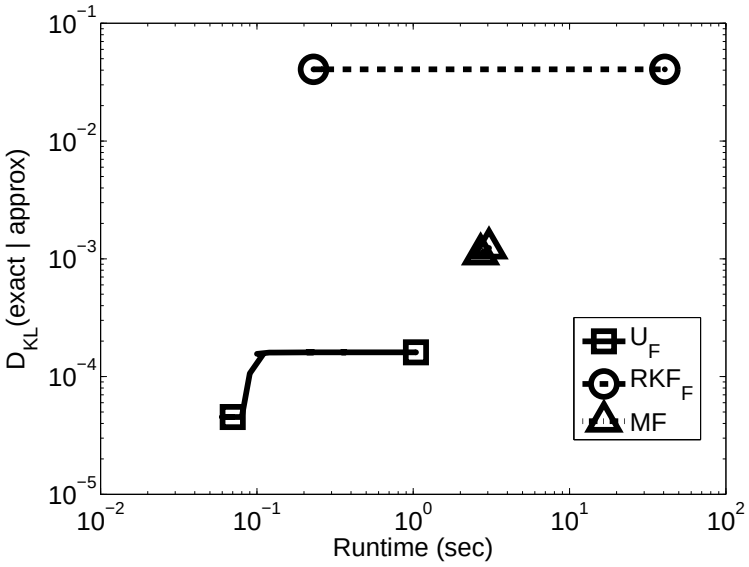


(b) Toroid, x_{13}

Figure 4.4: Computation time versus accuracy comparisons for toroid network.



(a) BHPS, smoking



(b) BHPS, children

Figure 4.5: Computation time versus accuracy comparisons for BHPS network.

distribution of x_6 (adjacent to the evidence) and x_{13} (a node more the in middle) at $t = 1$. For the BHPS network, we chose evidence where employment is observed to be in the student state continuously from year 1 to year 5. We queried the marginal distribution of the variables smoking and children at $t = 5$ years.

Figure 4.4 shows the KL-divergence between the true and approximated marginals of the query variable for each of the three networks for each query. These results are typical of other query marginals. In general, factored uniformization (U_F) performs the best, especially for smaller running times (with U_S occasionally being the method of choice for longer running times). However, when the query node is close to the evidence, RKF_F performs better (x_{10} for the ring and x_{13} for the toroid network are examples). The advantage of U_S for longer running times is predictable as it approaches the true value as more states are retained. Finally, note that for the BHPS query over the children variable, U_S has infinite KL-divergence (and hence does not appear on the plot). This is because all of the retained states have this variable equal to 0. While this is certainly the most common value at $t = 5$, it does not have probability 1 under the true distribution and thus the KL-divergence between the true distribution and the approximation of U_S is infinite.

4.7 Summary

In this chapter, we demonstrated our approximate continuous-time filtering method based on uniformization. It is simple to implement, and we have proven bounds on the KL-divergence of its error. The approximation can be made more accurate by lumping together variables into joint marginals. The bounds are similar in style to those of BK and also depend on the mixing time of the individual components (adjusted for the continuous-time nature of the system). Our experimental

results demonstrate that the theoretic bounded error holds in practice. Furthermore, our method gives superior time-accuracy trade-offs for most of the examples tested in this work.

Chapter 5

Deterministic Anytime Inference for Continuous-Time Markov Processes

In this chapter, we describe a deterministic anytime method for calculating filtered and smoothed distributions in large variable-based continuous-time Markov processes. Prior non-random algorithms do not converge to the true distribution in the limit of infinite computation time. Sampling algorithms give different results each time run, which can lead to instability when used inside expectation-maximization or other algorithms. Our method combines the anytime convergent properties of sampling with the non-random nature of variational approaches. It is built upon a sum of time-ordered products, an expansion of the matrix exponential. We demonstrate that our method performs as well as or better than the current best sampling approaches on benchmark problems.

Our method is deterministic and converges in the limit of infinite computational time. It can be viewed as a bridge between sampling and deterministic methods. The variational approaches such as expectation propagation [El-Hay et al., 2010] and mean field [Cohn et al., 2009] are deter-

ministic but they do not converge to the true value as computation time increases. Generally can only compute marginals or similar expectations. In contrast sampling approaches such as importance sampling [Fan et al., 2010] and Gibbs sampling [Rao and Teh, 2011] converge to the true value and can estimate any expectation of the distribution. However, they are random and this can cause problems when used inside other algorithms (like expectation-maximization).

In our method, we decompose the system into two pieces: a system (A) of completely independent components, and a correction (B). We reason exactly about the system A and add increasing number of correction terms derived from B. We generate a computation tree and traverse it using a priority queue, to select the larger correction terms earlier.

We first present our approach assuming that Q and probability vectors can be stored exactly. Then, we demonstrate how the calculations can be carried out efficiently when Q is structured. In section 5.4, we present a simple example to ground the derivation. Finally we demonstrate results comparing the computational efficiency of our method to other anytime convergent methods.

5.1 Calculating Matrix Exponential with Time-Ordered Products

We consider a CTMP with an initial state distribution row-vector v of size n and a rate matrix Q of size n-by-n. As we mentioned before, the calculation of e^{Qt} is intractable for large systems. Using the structure of Q for this calculation is not straightforward because it is not preserved by the matrix exponential. Additionally, the commutative property does not hold for the matrix exponential in general: For any same-sized matrices A and B, $e^{(A+B)t} \neq e^{At}e^{Bt} \neq e^{Bt}e^{At}$, unless the commutator $[A, B] = AB - BA$ vanishes. One possible decomposition comes from the Kronecker sum property: $e^{(A \oplus B)} = e^A \otimes e^B$. Yet, Kronecker sums alone can only describe rate

matrices for systems in which all variables are independent.

For the general case, $e^{(A+B)t}$ can be represented with time-ordered products (TOP) as given in Equation 2.13. We will employ the expansion to derive our deterministic method, Tree of Time-Ordered Products (TTOP).

5.2 Time-Ordered Products Computation Tree

We make two assumptions: Q can be split into $Q = A + B$, where ve^{At} is relatively simple to compute, and B is broken into J manageable terms $B = \sum_{j=1}^{J} B_j$. We will show how to realize these assumptions in the following sections. We use the TOP expansion of Equation 2.13 and apply the distributive property of matrix multiplication to move the sum outside the integral:

$$ve^{(A+\sum_{j=1}^{J} B_j)t} = ve^{At} + \sum_{j=1}^{J} \int_0^t ve^{As} B_j e^{A(t-s)}\, ds \tag{5.1}$$
$$+ \sum_{j=1}^{J} \sum_{j'=1}^{J} \int_0^t \left(\int_0^s ve^{Ar} B_j e^{A(s-r)}\, dr \right) B_{j'} e^{A(t-s)}\, ds + \ldots$$

Let $F^l(t)$ represent the l^{th} term of the expansion. Then the equation can be rewritten as

$$ve^{Qt} = \sum_{l=0}^{\infty} F^l(t) \tag{5.2}$$

where

$$F^0(t) = ve^{At}$$
$$F^l(t) = \sum_{j=1}^{J} \int_0^t F^{(l-1)}(s) B_j e^{-As}\, ds\ e^{At}. \tag{5.3}$$

By construction, the first term, ve^{At}, is simple to solve (as will be explained in Section 5.3).

5.2.1 Integral Expansion

We calculate each integral with the following expansion in which we treat a polynomial portion exactly and use adaptive quadrature to estimate the non-polynomial portion. Let $g(t)$ be a general function of time, g_0 be a constant, and $q(t)$ be a piece-wise polynomial. Further denote $\bar{q}(t) = \int_0^t q(s)\,ds$ and $q_{[r_1,r_2]}(t) = I[r_1 \leq t < r_2]q(t)$ (both also piece-wise polynomials), where $I[\cdot]$ is the indicator function. Then we can write an integral of the form $h(t; q, g, g_0) = \int_0^t q(s)(g(s) - g_0)ds$ as

$$\begin{aligned} h(t; q, g, g_0) &= \bar{q}(t)(g(s_0) - g_0) \\ &\quad + \int_0^t q_{[a,s_0]}(s)\,(g(s) - g(s_0))\;ds \\ &\quad + \int_0^t q_{[s_0,b]}(s)\,(g(s) - g(s_0))\;ds \\ &= \bar{q}(t)(g(s_0) - g_0) \\ &\quad + h(t; q_{[a,s_0]}, g, g(s_0)) + h(t; q_{[s_0,b]}, g, g(s_0)) \end{aligned} \tag{5.4}$$

for any chosen s_0, approximating g as the constant $g(s_0)$ and adding 2 correction terms (of the same form) for subparts of the interval. Here, $[a, b]$ is the support range of q, and $s_0 = \frac{a+b}{2}$. For convenience, we divide this range into two: $[a, \frac{a+b}{2}]$ and $[\frac{a+b}{2}, b]$. We use two subdivision in the following sections as well, but generalization to more than two sub-intervals is straightforward.

Recursive expansion Equation 5.4 will generate infinitely many terms in the form of

$\bar{q}(t)(g(s_0) - g_0)$:

$$h(t; q, g, g_0) = \sum_{k=1}^{\infty} q_k(t) u_k \tag{5.5}$$

where $q_k(t)$ is $\bar{q}(t)$ for a particular q, and u_k is $(g(s_0) - g_0)$ for a particular g and g_0.

5.2.2 Complete Computation Tree

Assume level l of Equation 5.2 can be expressed as

$$F^l(t) = \sum_{k=1}^{\infty} q_k^l(t) u_k^l e^{At} \tag{5.6}$$

(as is certainly true for $l = 0$: $q_0^0(t) = 1$, $u_0^0 = v$). We then show how to construct level $l + 1$ similarly:

$$\begin{aligned}
F^{l+1}(t) &= \sum_{j=1}^{J} \int_0^t F^l(s) B_j e^{-As}\, ds\, e^{At} \\
&= \sum_{j=1}^{J} \int_0^t \sum_{k'=1}^{\infty} q_{k'}^l(s) u_{k'}^l e^{As} B_j e^{-As}\, ds\, e^{At} \\
&= \sum_{k'=1}^{\infty} \sum_{j=1}^{J} \int_0^t q_{k'}^l(s) u_{k'}^l e^{As} B_j e^{-As}\, ds\, e^{At} \\
&= \sum_{k'=1}^{\infty} \sum_{j=1}^{J} h(t; q_{k'}^l, u_{k'}^l e^{At} B_j e^{-At}, 0)\, e^{At} \\
&= \sum_{k'=1}^{\infty} \sum_{j=1}^{J} \sum_{k=1}^{\infty} q_{k',k,j}^l(t) u_{k',k,j}^l e^{At}.
\end{aligned} \tag{5.7}$$

In last two lines, we have replaced the integral with the expansion of Equation 5.5. In particular, the integration of interest is $h(t; q_{k'}^l, u_{k'}^l e^{At} B_j e^{-At}, 0)$. We let the set $\{q_k, u_k\}_k$ generated for this h be denoted as $\left\{q_{k',k,j}^l, u_{k',k,j}^l\right\}_k$.

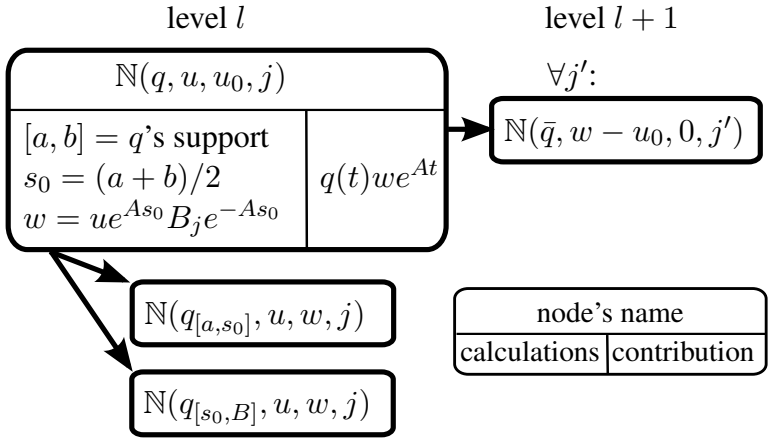


Figure 5.1: General form of the expansion.

In Equation 5.7, k' represents a node at level l in the computation tree. So, for every k', we generate J nodes in level $l+1$, each of which are the roots of trees for expansion of corresponding integrals. The result is an expansion for F^{l+1} of the same form as Equation 5.6.

We denote a term in this expansion as a compute node $\mathbb{N}(q, u, u_0, j)$. It and its descendants in the same level l represent $h(t; q, ue^{At}B_j e^{-At}, u_0)$. Its children in level $l+1$ represent new terms in F^{l+1}. Node $\mathbb{N}(q, u, u_0, j)$ contributes the term

$$q(t)we^{At}, \text{ where } w = ue^{As_0}B_j e^{-As_0}, \tag{5.8}$$

to the total sum. Its two children on the same level are $\mathbb{N}(q_{[a,s_0]}, u, w, j)$ and $\mathbb{N}(q_{[s_0,b]}, u, w, j)$ where $s_0 = \frac{a+b}{2}$. On the next level, it generates a node $\mathbb{N}(\bar{q}, w - u_0, 0, j')$, for all $1 \leq j' \leq J$. Figure 5.1 shows this expansion.

For reasons that will be clear in the next section, we cannot form $w - u_0$ (even though we can form each individually). Thus, node $\mathbb{N}(\bar{q}, w - u_0, 0, j')$ could be described by two nodes: $\mathbb{N}(\bar{q}, w, 0, j')$ and $\mathbb{N}(\bar{q}, -u_0, 0, j')$. However, the second node will essentially "undo" calculations

Algorithm 1 TTOP Filter

Initialize priority queue PQ with $\mathbb{N}(1, v, 0, 0)$
while PQ not empty **and** compute time left **do**
  Let $\mathbb{N}(q, u, u_0, j) \leftarrow \text{Pop}(PQ)$
  Set (a, b) be the support range of q
  Set $s_0 = \frac{a+b}{2}$, and $w = ue^{As_0}B_j e^{-As_0}$
  Add $q(t)we^{At}$ to the running sum (see Equation 5.8)
  {Next line can be generalized to more than 2 splits}
  Add $\mathbb{N}(q_{[a,s_0]}, u, w, j)$ and $\mathbb{N}(q_{[s_0,b]}, u, w, j)$ to PQ
  for $j' = 1$ **to** J **do**
    for $i = 1$ **to** m **do**
      Add $\mathbb{N}(\bar{q}, d_i(w, u_0), 0, j')$ to PQ

done elsewhere. We address this in the next section.

These recursively generated nodes represent an infinite tree whose sum is ve^{Qt}. Nodes have a single value, plus "same-level" children who represent finer approximations of the integral, and "next-level" children who represent one component of F^l for the next level l. If $A = 0$, then each level is one term in the Taylor expansion of Q. The traditional $\frac{1}{i!}$ coefficients in such an expansion are captured by our piece-wise polynomial integrations.

If we explore the tree such that every node will be visited in the limit of infinite computational time, then we can add each node's evaluation at time t to a running sum and compute ve^{Qt}. Algorithm 1 outlines this method using a priority queue to concentrate computation on portions of the tree with large contributions.

5.3 Structured Time-Ordered Products Calculations with CTBN

For the scenarios of interest, the vector v and matrix Q are too large to explicitly represent because the state space consists of one state for every assignment to a set of m variables, X_1 through X_m. We will assume that v is an independent distribution (although we can extend this work to

dependencies in v, but this eases the exposition). In Kronecker algebra this means $v = \bigotimes_{i=1}^{m} v_i$, where each v_i is a small row-vector of the marginal distribution over X_i.

We now show how to keep each quantity in the computation tree representable exactly as a similarly factored distribution: a Kronecker product with one term for each variable. If $A = \bigoplus_{i=1}^{m} A_i$, then $e^{At} = \bigotimes_{i=1}^{m} e^{A_i t}$. We assume that each A_i (which is only over the state space of X_i) is small enough so that calculation of $e^{A_i t}$ can be performed efficiently. If we also require that each $B_j = \bigotimes_{i=1}^{m} B_{j,i}$, then all vectors and matrices to be multiplied in the computation tree are such Kronecker products. In particular, at node $\mathbb{N}(q, u, u_0, j)$, we need to compute w (Equation 5.8) for its contribution to the sum. Because u, B_j, and e^{As_0} are all Kronecker products and $(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$, all of the matrix products can be performed efficiently by just operating on the subspaces over each variable independently. Thus w (and by extension the node's contribution to the sum) is a completely factored vector represented as a Kronecker product (that is, a distribution in which all variables are independent).

The generation of new nodes does not require any other operations, except for manipulation of one-dimensional piece-wise polynomials. Thus, our answer is a weighted sum of independent distributions (Kronecker products). It is not representable as a Kronecker product because it is a full distribution. However, any expectation of this distribution can be computed by summing up the contribution of each of these independent terms.

As we mentioned earlier, $\mathbb{N}(\bar{q}, w - u_0, 0, j')$ has the expression $w - u_0$ which is not a Kronecker product (despite that both w and u_0 are). To handle this, we note that

$$\bigotimes_{i=1}^{m} x_i - \bigotimes_{i=1}^{m} y_i = \sum_{i'=1}^{m} d_{i'}(x, y) \tag{5.9}$$

where

$$d_{i'}(x,y) = \left(\bigotimes_{i<i'} y_i\right) \otimes (x_{i'} - y_{i'}) \otimes \left(\bigotimes_{i>i'} x_i\right) \tag{5.10}$$

Because of how we select B_j (see below), $x_{i'} - y_{i'}$ is only non-zero for a few i' (the family of variable j). Furthermore, this allows us to split up the nodes by how much the deviation $(w - u_0)$ contributes by variable, which concentrates computation on those variables whose approximations are most difficult. Thus, we use this decomposition and divide the node $\mathbb{N}(\bar{q}, w - u_0, 0, j')$ (see Figure 5.1) into nodes $\mathbb{N}(\bar{q}, d_{i'}(w, u_0), 0, j')$ for all i' for which $w_{i'} \neq u_{0i'}$.

The necessary form for $Q = \bigoplus_{i=1}^{m} A_i + \sum_{j=1}^{J} \bigotimes_{i=1}^{m} B_{j,i}$ is always possible, although it might result in one B_j for each element of Q (which would in general be exponential in the number of state variables). Binary decision diagrams are often used to encode Q. In stochastic logic applications a disjunctive partitioning leads very naturally to this structure [Burch et al., 1991, Ciardo and Yu, 2005]. However, they are encoded in a form where $A = 0$. Techniques similar to those we describe next can be applied to pull intensity from the B_j matrices into A, but we will focus on the CTBN representation (Chapter 3).

5.3.1 CTBN Parameterization

As explained in Section 3.1, the global Q matrix for a CTBN can be represented with Kronecker algebra. Equation 3.3 corresponds to our TOP representation of Q where A is 0 and there is one B_j for each variable and instantiation of its parents. We can pull intensity into the A

matrix by defining

$$B_{(i|u_i),i'} = \begin{cases} Q_{i|u_i} - A_i & \text{if } i' = i \\ \Delta_{k,k} & \text{if } i' \in \text{Pa}(X_i) \text{ \& } k \text{ is val. of } i' \text{ in } u_i \\ I & \text{otherwise .} \end{cases} \tag{5.11}$$

Then,

$$Q = \bigoplus_i A_i + \sum_{i=1}^{m} \sum_{u_i} \left(\bigotimes_{i'=1}^{m} B_{(i|u_i),i'} \right) . \tag{5.12}$$

The algebra is long, but straight-forward. It holds because A_i is constant with respect to u_i and the sum over u_i represents each possible instantiation exactly once. The result is that A_i represents an independent, approximate process for X_i. A is the joint process of each of these independent approximations. The differences between the independent process and the CTBN are given by one $B_{i|u_i}$ for each variable and its parents' instantiation. Note that $B_{(i|u_i),i'} = I$ if i' is not a parent of i. Thus, most of the components of any B_j are the identity and computing $ue^{As}Be^{-As}$ for these components is trivial (they are the same as in u). Thus, the calculations for $\mathbb{N}(q, u, u_0, j)$ are local to the variable j and its parents.

5.4 CTBN Tree of Time-Ordered Products

Example 17 shows one possible decomposition of a simple 2-variable CTBN into A and B. Here the local A_i matrices are chosen to be the minimal rates. This method essentially computes the effect of A exactly and incorporates the effects of B as a Taylor expansion, adding increasing numbers of terms. Note that in Example 17, A_X and A_Y are proper intensity matrices (their rows

sum to 0). This means that $B_{Y|x_0}$ and $B_{Y|x_1}$ have negative diagonal elements. This results in a computation that corresponds to a Taylor expansion with alternating signs. This can cause computational problems. One alternative is to arrange for B to have no negative elements. For instance $A_Y = \begin{bmatrix} -4 & 2 \\ 3 & -4 \end{bmatrix}$, resulting in $B_{Y|x_0} = \begin{bmatrix} 2 & 0 \\ 1 & 0 \end{bmatrix}$ and $B_{Y|x_1} = \begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix}$. The disadvantage is that $v_2 e^{A_Y t}$ sums to less than 1. The non-root nodes add probability to the answer (instead of moving it within the answer).

Example 17 *We use the 2-variable CTBN network (Figure 3.2) from Chapter 3, where variable X is the parent of variable Y. Because X has no parents, Q_X is equal to A_X, and there are no B terms.*

$$Q_X = \begin{bmatrix} -1 & 1 \\ 3 & -3 \end{bmatrix} \rightarrow A_X = \begin{bmatrix} -1 & 1 \\ 3 & -3 \end{bmatrix} .$$

We decompose $Q_{Y|x_0}$ and $Q_{Y|x_1}$ by transferring minimal rates to A_Y,

$$Q_{Y|x_0} = \begin{bmatrix} -2 & 2 \\ 4 & -4 \end{bmatrix} , Q_{Y|x_1} = \begin{bmatrix} -4 & 4 \\ 3 & -3 \end{bmatrix} \rightarrow A_Y = \begin{bmatrix} -2 & 2 \\ 3 & -3 \end{bmatrix} .$$

The remaining rates are transferred to $B_{Y|x_i}$ matrices,

$$B_{(Y|x_0)} = \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix} , B_{(Y|x_1)} = \begin{bmatrix} -2 & 2 \\ 0 & 0 \end{bmatrix} .$$

In this case, there are two B terms: B_1 for Y and x_0, B_2 for Y and x_1. We can calculate the joint

Q matrix by using Equation 5.12,

$$A = A_X \bigotimes A_Y = \begin{bmatrix} -3 & 2 & 1 & 0 \\ 3 & -4 & 0 & 1 \\ 3 & 0 & -5 & 2 \\ 0 & 3 & 3 & -6 \end{bmatrix}.$$

$$B_1 = \Delta_{x_0,x_0} \bigotimes B_{(Y|x_0)} = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix} \bigotimes \begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$B_2 = \Delta_{x_1,x_1} \bigotimes B_{(Y|x_1)} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \bigotimes \begin{bmatrix} -2 & 2 \\ 0 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$\text{and } Q = A + B_1 + B_2 = \begin{bmatrix} -3 & 2 & 1 & 0 \\ 4 & -5 & 0 & 1 \\ 3 & 0 & -7 & 4 \\ 0 & 3 & 3 & -6 \end{bmatrix}.$$

For a CTBN, multiplication by a B_j is particularly simple. The j value indexes a variable (X_i) and an instantiation to its parents (u_i). To multiply a factored vector by B_j, multiply the X_i component by $Q_{i|u_i} - A_i$. For each component associated with a parent, zero out all elements of

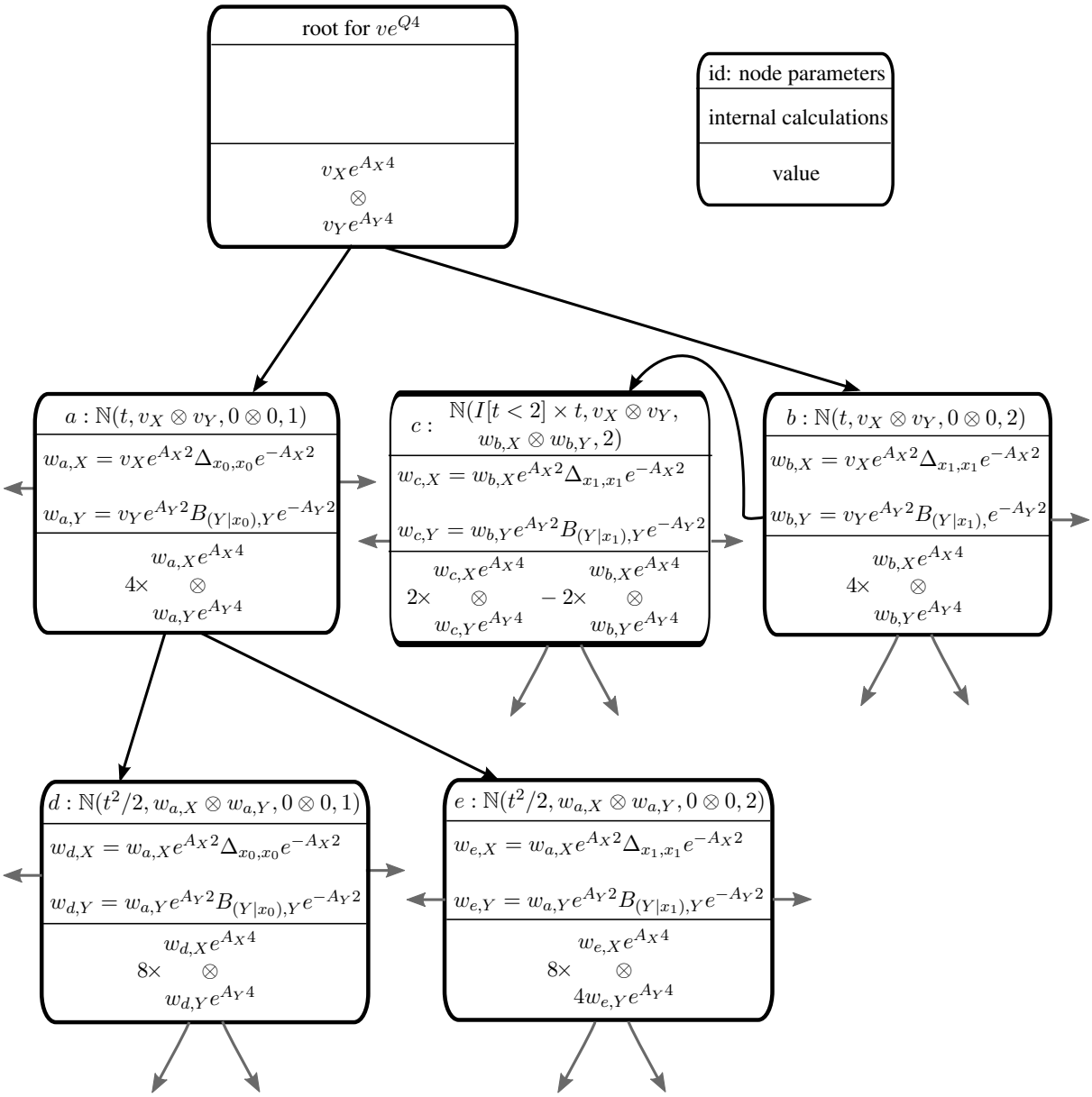


Figure 5.2: Portion of computation tree for Example 17 for $t = 4$. Each box is one node in the computation tree (see Figure 5.1). Nodes a and b are the children of the root. Nodes d and e are examples of their children at the next level ($l = 2$). Node c is an example of a refinement (of node b). Equation 5.10 dictates how the children of c for $l = 2$ are computed because $u_0 \neq 0$ for this node.

the vector except for the one consistent with u_i. Vectors for non-parent nodes are unchanged.

Example 18 *If we let $v = v_X \otimes v_Y$ (that is, v_X and v_Y are the independent marginals of X and Y), then Figure 5.2 shows a small portion of the computation tree.*

5.5 Smoothing and Computational Considerations

Our discussion so far has focused on filtering. We also perform smoothing with evidence at the beginning and at the end of the interval as explained in this section. We can also extend our method so that we can incorporate it into the framework of inference with general evidence. As we discussed in Section 3.2, this is done by dividing the trajectories to segments and conditioning on evidence in each segment.

We will limit ourselves to the case in which the initial distribution at time 0, v, is known and there is evidence at a later time T for which the vector v_T represents the probability of the evidence for each possible state of the system. We assume that both vectors factor as previously discussed.

The goal is to compute (an expectation of) the distribution at time t conditioned on the evidence at time T. This consists of computing the Hadamard (point-wise) product of ve^{Qt} and $v_T e^{Q^\top (T-t)}$ (and then normalizing the result). First, consider computing each exponential separately. Let the result of the "forward" direction be $\sum_j \alpha_j$ where the forward calculation's jth node had contribution $\alpha_j = \bigotimes_{i=1}^{m} \alpha_{j,i}$. Let the "backward" direction be similarly represented as $\sum_k \beta_k$

with $\beta_k = \bigotimes_{i=1}^{m} \beta_{k,i}$. It can easily be shown that

$$\sum_{j=1}^{N_a} \alpha_j \odot \sum_{k=1}^{N_b} \beta_k = \sum_{j=1}^{N_a} \sum_{k=1}^{N_b} \left(\bigotimes_{i=1}^{m} \alpha_{j,i} \right) \odot \left(\bigotimes_{i=1}^{m} \beta_{k,i} \right)$$
$$= \sum_{j=1}^{N_a} \sum_{k=1}^{N_b} \bigotimes_{i=1}^{m} (\alpha_{j,i} \odot \beta_{k,i}) \tag{5.13}$$

Thus, we must consider every pair of nodes, one from the forward expansion and one from the backward expansion. For each pair, the components of the factored representation are point-wise multiplied together to obtain the pair's contribution to the answer.

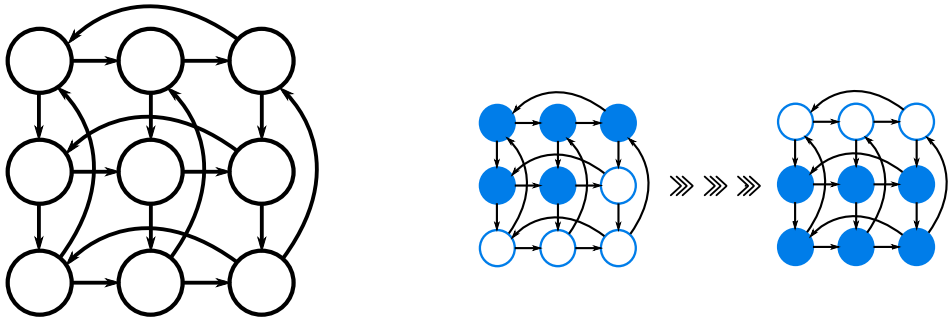
We want to include these terms judiciously to best use a finite computational budget. We do this by keeping a frontier set of pairs of computation nodes, one from the forward tree and one from the backward tree. If we have explored (added to the smoothing result) $\alpha_j \odot \beta_k$, we add to our frontier set α_j paired with all of β_k's children and α_j's children paired with β_k. We use a closed list to ensure no pair is considered twice.

The question then is how to prioritize members of the frontier. We need to have an estimate of the total contribution of this pair and all subsequent pairs in the graph. We select the sum of this node's contribution to the query value (absolute value of the change) and the product of the maximum value in each node.

5.6 Experiments

We implemented our method, TTOP (Tree of Time-Ordered Products), as part of the CTBN-RLE code base [Shelton et al., 2010]. We evaluated our method on a synthetic network of Ising model dynamics. The Ising model is a well-known interaction model with applications

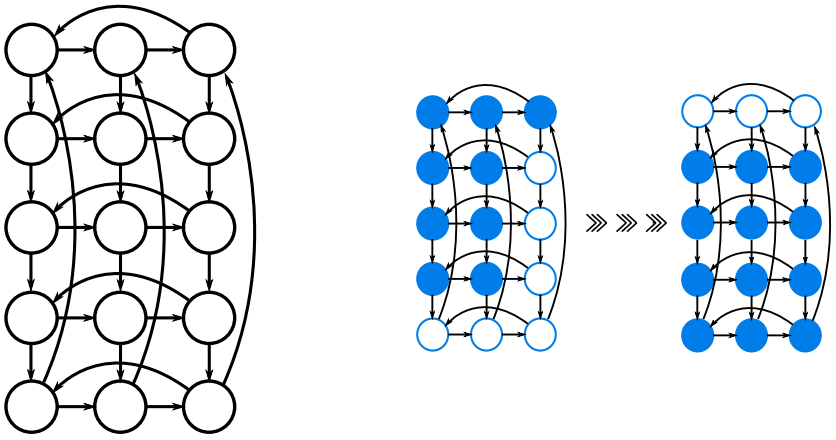
in many fields including statistical mechanics, genetics, and neuroscience [Zhou and Schmidler, 2009]. The experimental results focus on inference accuracy given a known network. The Ising model was chosen so that we could compute the true answer in a reasonable time and scale the problem size.



(a) Network structure.

(b) Evidence patterns at $t = 0$ and $t = 1$.

Figure 5.3: Toroid network with 9 nodes.

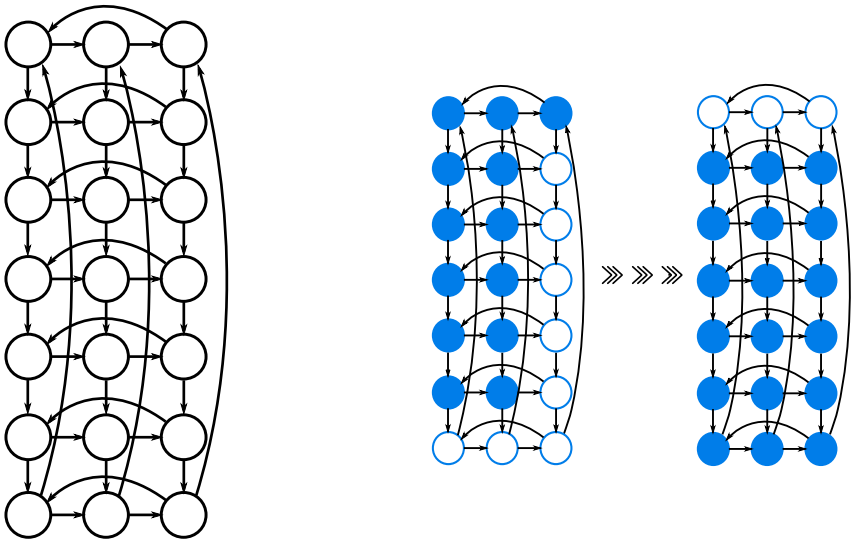


(a) Network structure.

(b) Evidence patterns at $t = 0$ and $t = 1$.

Figure 5.4: Toroid network with 15 nodes.

Using this model, we generated a directed toroid network structure with cycles following the network used by [El-Hay et al., 2010]. Nodes follow their parents' states according to a coupling strength parameter (β). A rate parameter (τ) determines how fast nodes toggle between states. We



(a) Network structure.

(b) Evidence patterns at $t = 0$ and $t = 1$.

Figure 5.5: Toroid network with 21 nodes.

scale the number of nodes in the network but limit it to 21 to be able to compare the results to exact inference. We use three networks of respectively $m = 9$, 15, and 21 binary variables (Figures 5.3a, 5.4a, and 5.5a). We could not include networks with more than 21 binary variables because we cannot do exact exponentiation on a matrix of size bigger than $2^{21} \times 2^{21}$ in a reasonable time. We scale the network by adding rows of nodes. For these networks we fix the τ parameter and vary β. Nodes can take on values -1 and $+1$.

We compare TTOP to two efficient anytime algorithms: auxiliary Gibbs sampling (AuxGibbs) [Rao and Teh, 2011] and importance sampling (IS) [Fan et al., 2010]. We also compared to a mean field variational approach (MF) [Cohn et al., 2009]; however, error for MF is above the error range of other methods for all computation times. For this reason, we omit the MF results in the plots. We analyze the error in marginals computed by each method relative to exact inference. We focus on the computation time because in our experiments memory was not an issue and whole computation

tree occupied only a few GBs.

For TTOP, we set the number of splits for the quadrature (see Equation 5.4) to 10 because it produces a good computation time versus error performance. We vary the computation time budget to observe the trade-off between computation time and the error.

For AuxGibbs, we vary the sample size between 50 and 5000, and set the burn-in period to be 10% of this value. For IS, the sample size varies between 500 to 50000. We ran the experiments 100 times for each test for both sampling methods. The computation time shown in the plots is the average of these runs for a given number of samples. The error is the sum of the KL-divergences of all the marginals from their true values.

Our experiments focus on smoothing. The networks start from a deterministic state, for $m = 9$: at $t = 0$ variables 1–5 are $+1$ and 6–9 are -1. At $t = 1$, variables 1–3 have switched to -1, 4–5 remain $+1$, and 6–9 have switched to $+1$. The evidence pattern can be seen in Figure 5.3b. For $m = 15$ and 21 we use a similar pattern of evidence for comparison reasons. For $m = 15$ (Figure 5.4b), the variables 1–5, 7–8, 10–11 start at $+1$ and the remaining variables start at -1. The variables 1–3 switch to -1, while 4–5,7–8, and 10–11 stay at $+1$, and 6, 9 and 12–15 switch to $+1$. The evidence for $m = 21$ also follows the same pattern scaled to 21 nodes (Figure 5.5b).

The nodes are not observed between $t = 0$ and $t = 1$. We query the marginal distributions of nodes at $t = 0.5$. Figures 5.6, 5.7, and 5.8 show computation time versus sum of KL-divergence of marginals. We focus on the first 20 seconds of computation time because usually a few seconds are enough for our inference tasks. The lines in the plots continue their trend and cross at some point except for Figure 5.7-a. A KL-divergence sum of 10^{-2} is generally accurate for these networks.

Figures 5.6-a, 5.7-a, and 5.8-a show the results for $\tau = 2$, $\beta = 0.5$. For most of these

experiments, TTOP performs better than sampling methods. When the coupling strength of the network is increased to $\beta = 1$ as shown in Figures 5.6-b, 5.7-b, and 5.8-b, TTOP has more variations in the error as the computation time increases but still has better performance overall. The occasional peaks in the error happen because sometimes a part of the computation tree is expanded and added to the sum, without the part that balances it since the time budget expired. This can be seen as more computation time is given to the algorithm, the errors decrease with the addition of the balancing part.

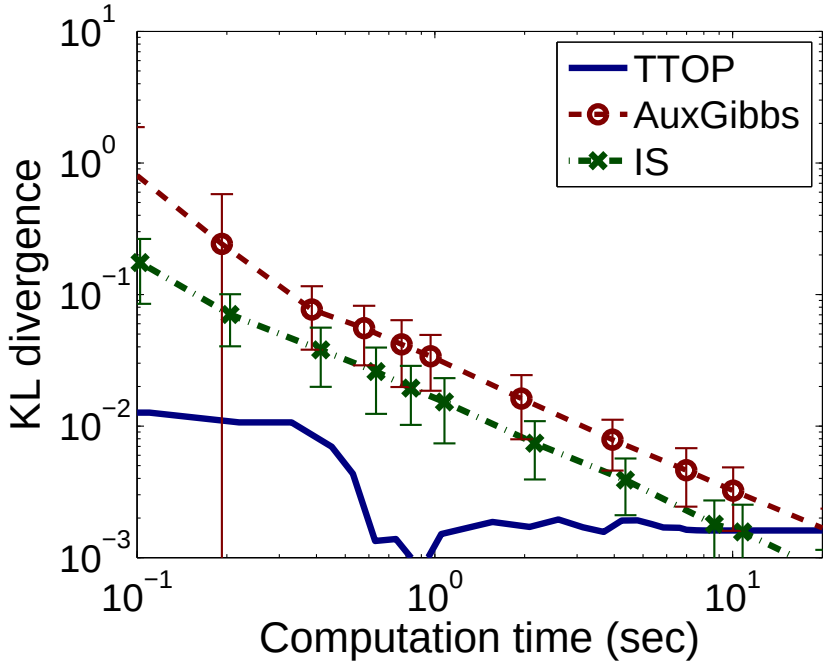
As the number of nodes in the network increase, our method keeps the computation time versus error advantage. Additionally, the gap between our method and others increases with the network size. Especially when $\beta = 1$, it performs better for the larger networks. While we cannot perform exact inference for larger networks, we expect these trends would continue as the problem size scales.

TTOP is also much better for short computation times, because it solves $e^{(At)}$ directly by integration while the sampling methods can generate only a few samples. Although the derivatives are smaller for the TTOP lines, this could potentially be fixed with better node prioritization. The best node prioritization would be one that looked at the contribution of the whole subtree rooted at a node rather than only the contribution of that node. Our heuristic is good for the first few levels of the tree, but it does not do as well as we go deeper in the tree.

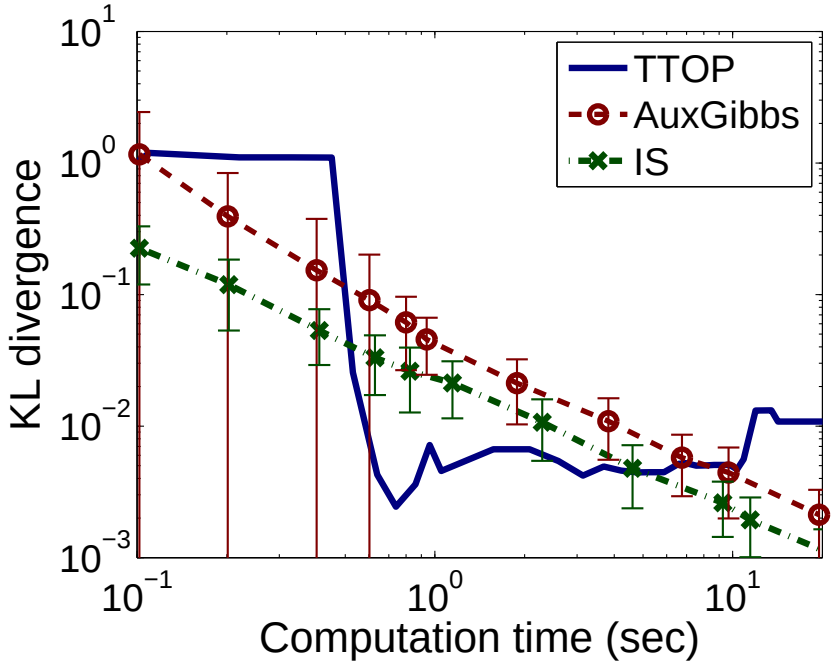
The fluctuations in the error of TTOP are expected. The error from a single run of sampling would fluctuate as well. The plotted results of our method are from a single run compared to the averaged results of sampling methods which are 100 runs.

5.7 Summary

In this chapter, we demonstrated an anytime algorithm for structured CTMP filtering and smoothing. Unlike prior work, it is deterministic, which can be of benefit when used inside learning methods. In the experiments, it has better computation time versus error performance than prior anytime convergent methods, especially for loosely coupled systems. Also as network size increases and coupling strength stays the same, our method's advantage increases as well.

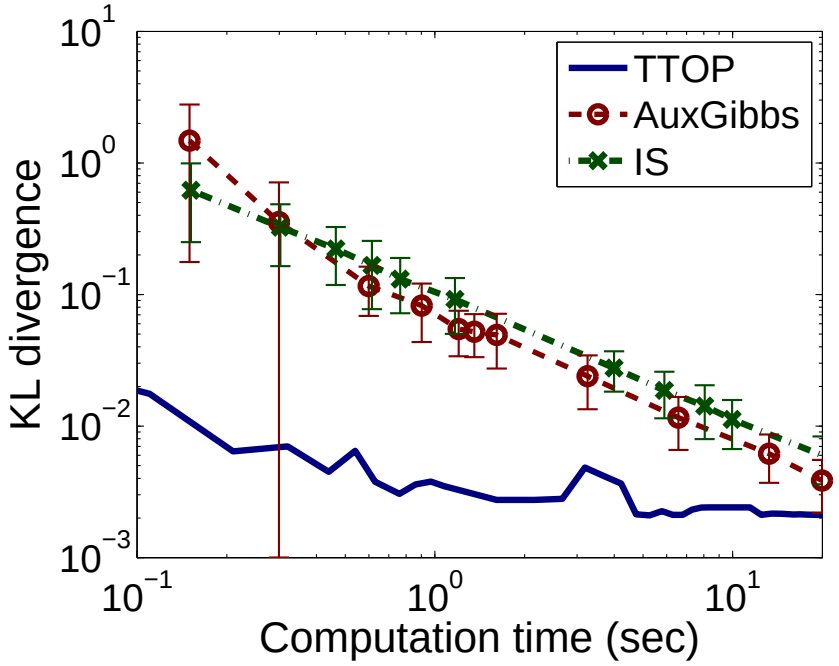


(a) $m = 9, \tau = 2, \beta = 0.5$.

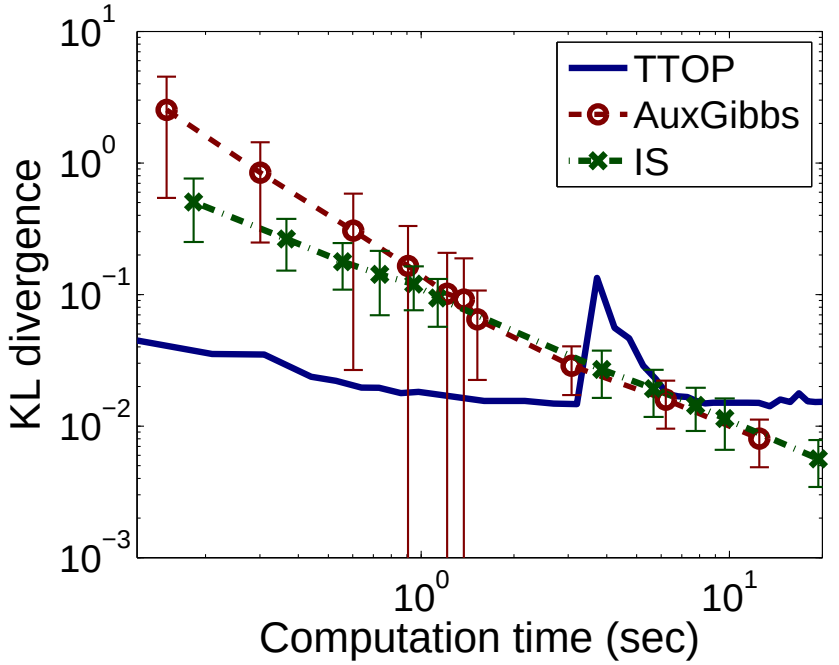


(b) $m = 9, \tau = 2, \beta = 1$.

Figure 5.6: Computation time versus KL divergence for the 9 node toroid networks with different coupling parameters.

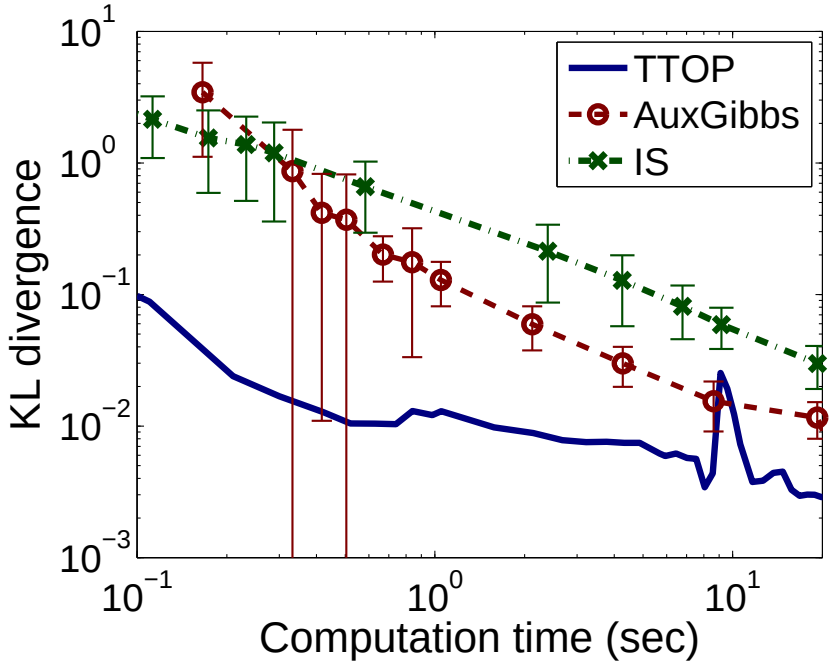


(a) $m = 15$, $\tau = 2$, $\beta = 0.5$.

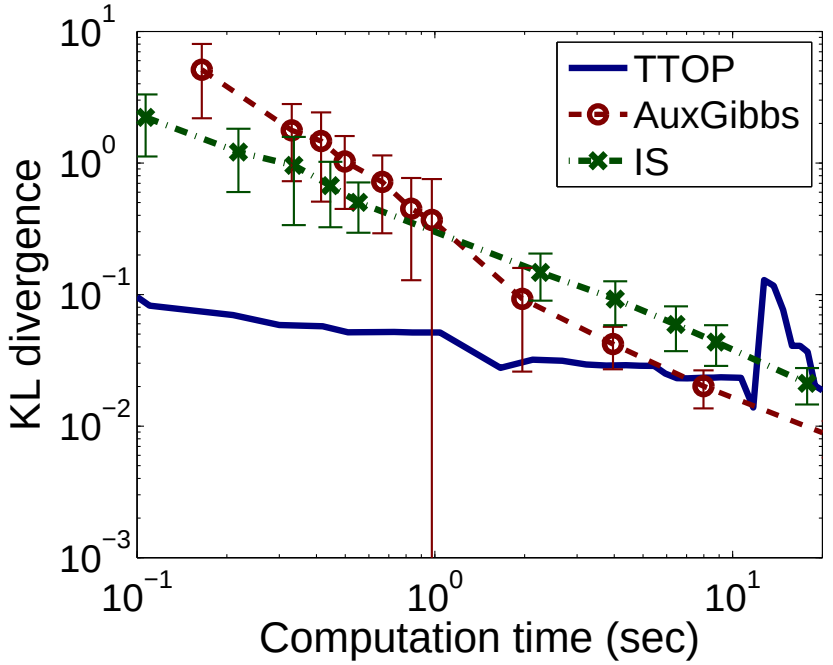


(b) $m = 15$, $\tau = 2$, $\beta = 1$.

Figure 5.7: Computation time versus KL divergence for the 15 node toroid networks with different coupling parameters.



(a) $m = 21$, $\tau = 2$, $\beta = 0.5$.



(b) $m = 21$, $\tau = 2$, $\beta = 1$.

Figure 5.8: Computation time versus KL divergence for the 21 node toroid networks with different coupling parameters.

Chapter 6

Non-Invasive Blood Gas Estimation for Pediatric Mechanical Ventilation

Applications of machine learning to electronic medical data have great potential for helping medical practitioners make more supported diagnostic decisions. They can give insight into symptoms, medications, and outcomes of specific patients by exploiting data from similar patients [Cruz and Wishart, 2006, Davis et al., 2008, Holzinger et al., 2014]. Especially in an intensive care unit, where patients are continuously monitored and decisions are time sensitive, the effects can be monumental.

Medical data has many forms: vital signs, text notes, lab results, medical imaging data, and measurements from medical monitors. Apart from many challenges associated with this data such as privacy and complexity, the dynamic nature of it also poses a problem. In this work, we use a dynamic system model to utilize the past information to estimate the current state of the patient.

We focus on two types of medical data: vitals and lab results recorded in irregular intervals

in an intensive care unit. Together they form a multi-dimensional time series. Since the measurements are not taken at regular intervals, we model them using continuous-time models. Specifically, we apply a multivariate Gaussian process model (MGP), which is a continuous-time model with continuous variables, to the problem of respiratory ventilator management for children with lung injuries. Our aim is to estimate some of the blood test results that are necessary during ventilation management. In this setting, the ventilator machine is used to support breathing for children with acute lung injury (ALI) [Flori et al., 2005] or acute respiratory distress syndrome (ARDS) [Yang et al., 2005] in a pediatric intensive care unit (PICU).

We choose an MGP model for this problem because using continuous variables is more natural for this setting, as the measurements are continuous valued. We have applied CTBN models (with discrete variables), and the results were similar, but not as competitive as the MGP model.

6.1 Motivation

Monitoring and managing ventilation are very important in a PICU. At Children's Hospital Los Angeles (CHLA), 39% of the children admitted to the PICU are mechanically ventilated at some time during their stay. This percentage is high in studies from other PICUs as well (55% in Farias et al., 2012 and 30% in Khemani et al., 2009). Although very helpful, mechnanical ventilation also comes with risks and complications, such as lung injuries. Also, it might be uncomfortable to the patient to such a degree that they have to be sedated. The ventilator settings are chosen mostly based on invasive blood tests, which are not done very frequently and they are especially challenging in children. Therefore, clinicians sometimes have to make decisions without accurate and up-to-date information on the patient's condition. This causes longer durations of ventilation,

whereas more frequent informed decisions on ventilator settings could lead to faster and safer weaning of the patient. Our model in this chapter estimates invasive blood test results from all available previous and present vitals and test results for this purpose. Accurate estimations of these blood test results could also open the way to semi-automated or even automated mechanical ventilation systems.

Machine learning techniques in general have been applied to a great variety of problems in medicine [Meyfroidt et al., 2009]. Continuous-time models are used in developing a personalized risk stratification score for prematurely born babies [Saria et al., 2010], in modeling cardiovascular system [Weiss et al., 2012], and modeling heart failure [Gatti et al., 2012].

Previous work on applying machine learning to ventilation focuses on very different prediction problems. A dynamic Bayesian network model is proposed by Charitos et al. [2009] to predict if a patient has ventilator-associated pneumonia. Guiza Grandas et al. [2008] use a Gaussian process (GP) classifier is used to predict the time frame in which a patient reaches stability, in order to wean a patient off mechanical ventilation. Also, for mechanical ventilation of adults with lung injury, a GP is used to model the pressure-volume curve which was used to titrate PEEP to optimal compliance [Guttmann, 2009]. However, to our knowledge, ours is the first application of machine learning to predict blood test results for patients with lung injury.

6.2 Mechanical Ventilation

6.2.1 Overview

Mechanical ventilation is the process of supporting or replacing breathing in patients who cannot adequately breath on their own. Ventilation methods vary depending on the patient condition;

Name	Definition
Tidal volume (V_T)	Volume of air from the lungs through the ventilator
Ventilator rate (VR)	Frequency of breaths from the ventilator
Positive end-expiratory pressure (PEEP)	Basal pressure support to the lungs at max. deflation
Peak inspiratory pressure (PIP)	Pressure support to the lungs at max. inflation
Inspiratory flow rate ($\dot{V}_i$)	Volume of air flow toward the lungs in one minute
Fraction of inspiratory O_2 (FiO_2)	Concentration of O_2 given to the lungs

Table 6.1: Common ventilator settings and their definitions.

however we are interested in the case where a ventilator machine is used. In this case, the patient is connected to the ventilator with an endotracheal tube inserted through the mouth.

Ventilators have different modes according to what is being controlled, such as pressure control or volume control. For example, in pressure control, pressure given to the lungs is kept at a target value, and volume is varied automatically by the ventilator to maintain the pressure. In volume control, volume of air given into the lungs is set, and pressure is varied automatically to maintain the target volume. There are additional settings that are common in all modes, such as the ventilator rate setting which controls the number of supported breaths in a minute, and the fraction of inspired oxygen setting for the given breath. The common ventilator settings can be found in Table 6.1. The mode of ventilation is chosen according to the patient disease and symptoms, and the clinician's familiarity with the modes.

The measurements monitored by the ventilator (Table 6.2) along with the cardiac monitor and blood pressure values are usually recorded by the nurses at intervals recommended by the physician. Monitoring the patient condition is critical during ventilation, since ventilator settings should be adjusted carefully. Optimization of ventilator settings is important to reduce the risk of lung injury, such as over inflation of lungs from excessive pressure. For example, FiO_2 is set to

Name	Definition
Pulse oximetry (SpO_2)	Surrogate for blood O_2 saturation
End-tidal CO_2 ($ETCO_2$)	The concentration of CO_2 exhaled from ventilator
Minute ventilation (MV)	Volume of gas inhaled from lungs in a minute
Oxygen saturation index (OSI)	Derived measure of oxygenation from SpO_2
Dynamic compliance (Cdyn)	Derived measure of the compliance of lungs

Table 6.2: Monitored and derived variables during ventilation.

Name	Definition
pH	Acidity level of arterial blood
Partial pressure of CO_2 ($PaCO_2$)	CO_2 pressure in arterial blood
Partial pressure of O_2 (PaO_2)	O_2 pressure in arterial blood plasma

Table 6.3: Arterial blood gas (ABG) measurements for ventilation.

the minimum value to maintain the blood O_2 saturation (SaO_2). However, since SaO_2 requires an invasive procedure, it is not done frequently. Instead, pulse oximetry (SpO_2, see Table 6.2) variable is used as an accurate surrogate for SaO_2 [Louw et al., 2001].

The other settings in the ventilator generally require an initial blood test for arterial blood gas (ABG) values. This is an invasive test where arterial blood is drawn from a catheter, in order to obtain ABG pH and partial pressure of CO_2 ($PaCO_2$, see Table 6.3) values. These measurements show the lung's CO_2 removal condition, and need to be measured frequently if patient is not stable. Unfortunately no surrogate exists for pH or $PaCO_2$ that is accurate at all times. The end-tidal CO_2 ($ETCO_2$) variable which is usually available continuously from the ventilator monitor is the closest surrogate for $PaCO_2$. $ETCO_2$ is the concentration of CO_2 at the end of the exhaled breath, measured at the ventilator. However, it is not accurate in patients with lung disease as we will explain in the next section.

6.2.2 Lung Injury and Dead Space

During ventilation, the inhaled air travels through the mouth, trachae, bronchital tubes and lastly to the alveoli in lungs. Also, the pulmonary arteries coming from the heart carries blood to the lungs. Gas exchange occurs only in the alveoli between the O_2 dense inhaled air and CO_2 dense blood in pulmonary artery capilllaries. Since gas exchange does not occur in airways including mouth, trachae and bronchital tubes, the sum of their volume is called the *dead space*.

Gas exchange is dependent upon the partial pressures of O_2 and CO_2 in the alveoli and in the blood in pulmonary capilleries. Once the partial pressures on either side are equalized, the gas exchange stops. Therefore $PaCO_2$ and alveolar partial pressure of CO_2 are in equilibrium at the end of inhalation.

When air starts to move out from the mouth at the beginning of exhalation, the pressure of CO_2 in exhaled air is initially zero because of dead space. As CO_2 from the alveoli mixes with the air in dead space and exhaled, the pressure increases. At the end of exhalation CO_2 concentration reaches the maximum value, which is the monitored value of $ETCO_2$. As the alveolar pressure is diluted with the air in dead space, $ETCO_2$ is generally a little less than the initial alveolar pressure, and therefore $PaCO_2$.

In patients with lung injury, not all alveoli participate in gas exchange. Cardiac output might get so low that blood flow to some alveoli becomes inadequate for gas exchange. On the other end of the spectrum, alveoli might not get enough air, so the blood flows without any contact with alveoli (pulmonary shunt). The total volume of alveoli that does not exchange gas is called alveolar dead space (V_{ADS}). V_{ADS} further dilutes exhaled air, increasing the difference between $ETCO_2$ and $PaCO_2$. The instantaneous ratio of V_{ADS} to V_T can be found by using the $PaCO_2$ and

$ETCO_2$ measurements of the same instant,

$$\frac{V_{ADS}}{V_T} = \frac{PaCO_2 - EtCO_2}{PaCO_2}, \tag{6.1}$$

which is called the alveolar dead-space fraction (AVDSF).

At the bedside, clinicians estimate current $PaCO_2$ from the last known value of AVDSF. In the weaning phase of ventilation, when the patient condition is not critical, this AVDSF estimation is more accurate and can be used [Khemani et al., 2014]. However, if the patient is not stable, which happens frequently in acute phase of ventilation, V_{ADS} might change rapidly. Additionally, poorly perfused alveoli might not empty at the same rate as others [Severinghaus et al., 1957]. Therefore, this estimate is not accurate in acute phase.

Estimates of pH at the bed side are even less accurate than $PaCO_2$ estimates. The standard equation for pH, the Henderson-Hasselbalch (HH) equation [Po and Senozan, 2001],

$$pH^{HH} = 6.1 + \log \frac{HCO_3}{0.03 \times PACO_2} \tag{6.2}$$

requires AVDSF. For the AVDSF value, the estimate from last known values is used, which adds to the error. pH also depends on the changes in the bicarbonate (HCO_3) measurement. Again, the last known estimate for HCO_3 is used, making the result less accurate.

This is unfortunate, since frequent accurate estimates of $PaCO_2$ and pH are necessary for lung protection and optimal ventilation management. For example, in the case of permissive hypercapnia (controlled elevation of $PaCO_2$) when the patient has hypoxemia (life threatening levels of insufficient oxygenation), frequent accurate estimates of $PaCO_2$ or pH can be life saving.

6.2.3 Pediatric Challenges

Pediatric ventilation protocol is different and less studied than adult ventilation. A PICU accepts infants and children up to age 18. This poses problems on how to measure certain parameters needed for ventilation settings, such as weight, V_T, and FiO_2. For example, V_T is calculated based on the weight of the adult patients, but it is not clear if using weight is accurate for children. Additionally, the hospitals or individual clinicians usually have tendencies to use only certain modes. So, there is no standard protocol for ventilation management for PICUs [Khemani et al., 2011b].

In the case of testing for ABG measurements, taking frequent arterial blood draws from children is very challenging. Therefore, we can use accurate estimates of pH and $PaCO_2$ to help execute a standard protocol, such as the one proposed for PICUs by Khemani et al. [2011b]. The use of estimates from a model like ours could lead to semi-automation of ventilation. Clinicians can make more informed decisions, and also give feedback to the system, which could lead the way to closed-loop automation of ventilation.

6.3 Datasets

In our model, we worked on two separate datasets. They are both collected from PICU electronic medical records. The first one, D_S dataset [Khemani et al., 2011a], is from a single institution retrospective study and has 275 patients with an average of 5 observations per patient. The second, D_M dataset [Khemani et al., 2012], is constructed from a six-center prospective study and has 84 patients with an average of 4 observations per patient. In both datasets, ventilator setting parameters and other non-invasive measurements are recorded at the time of blood tests for ABG

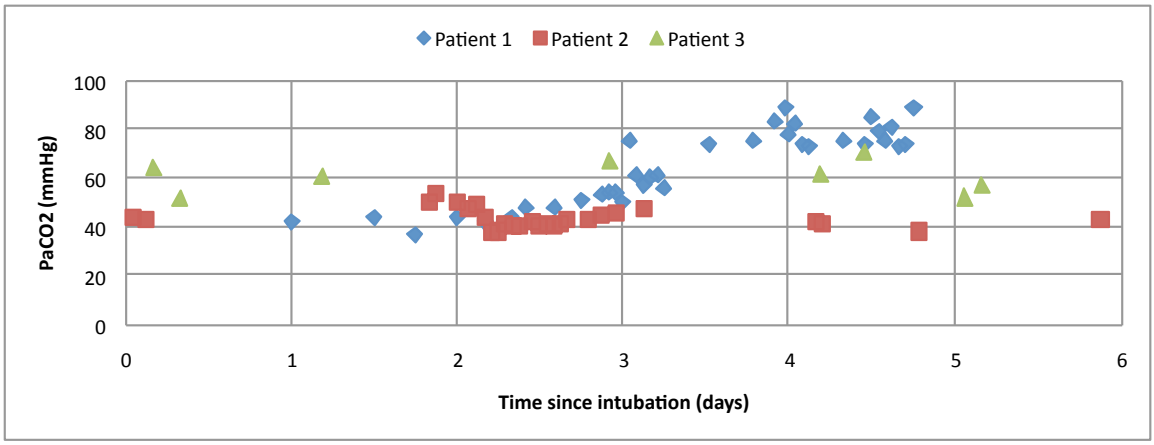


Figure 6.1: Sample $PaCO_2$ values for three patients. Note the variation in the time between measurements.

measurements. Consequently, all the variable measurements are aligned, and only available at the time of blood tests.

When constructing the datasets, out of all ventilated patients, only the critical ones with lung injury are selected. So the selected patients are the sicker ones. Also, only patients with more than one blood test for ABG are chosen.

Although the measurements are aligned due to how the datasets were constructed, since the blood tests are done more frequently when the patient is more critical, the intervals between measurements are irregular. A sample of $PaCO_2$ values for three different patients from the datasets can be seen in Figure 6.3

6.4 Multivariate Gaussian Process Model

A Gaussian process (GP) is a non-parametric model that is widely used for non-linear regression tasks [Rasmussen, 2006]. A random real-valued function $f(t)$ is distributed as a GP,

if for any finite subset of indices $t_1, \ldots, t_m$, the random vector $[f(t_1), \ldots, f(t_m)]$ has a Gaussian distribution. A GP is described by a mean function $\mu(t)$ and a covariance function $K(t, t')$.

In our model, X_i, $i \in \{1, \ldots, N\}$ denote the measured variables such as pH. For all X_i, there is a set of measurement times, $S_i = \{t_{i,1}, \ldots, t_{i,T_i}\}$. Let $\mathcal{T} = \sum_i T_i$. Then, for all $t \in S_i$ and $t' \in S_j$,

$$\mathcal{X} \sim \mathcal{GP}(\mu, K_{ij}(t, t')) \,, \tag{6.3}$$

where $\mathcal{X} \in \mathbb{R}^{\mathcal{T}}$ is a multivariate Gaussian process (MGP) with constant mean function μ. Here, the covariance function $K_{ij}(t, t')$ shows the correlation between variable X_i at time t and variable X_j at time t'.

6.4.1 Kronecker Decomposition

We assume the covariance function K is separable, and therefore can be described by the product of a covariance matrix between the variables and a temporal covariance function [Gelfand and Banerjee, 2010],

$$K_{ij}(t, t') = \mathbf{C}_{ij} R(t, t'), \tag{6.4}$$

where $\mathbf{C} \in \mathbb{R}^{N \times N}$ is the covariance matrix between variables $X_1, \ldots, X_N$, and $R(t, t')$ is the temporal covariance function between two time points.

As mentioned before, in each patient sample in our dataset, times of measurements for all the variables are aligned by construction (see Figure 6.2). Therefore, we have a common set of observation times $S = S_i$ for all i, where $S = \{t_1, \ldots, t_T\}$. For this reason, we drop the dependency on the variables in the observation times. Given S, we can use a temporal covariance

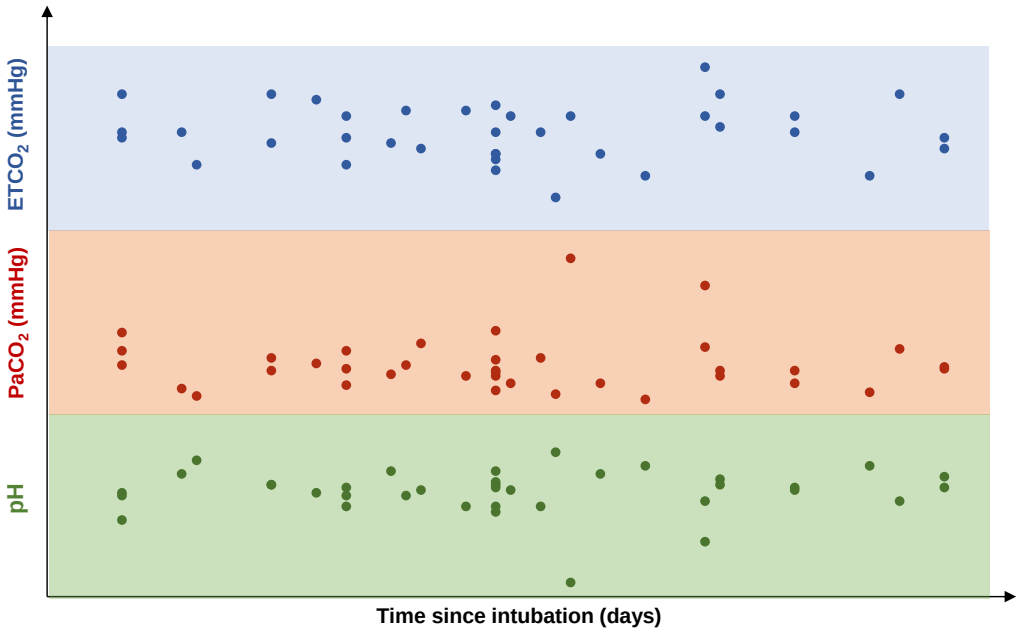


Figure 6.2: Sample measurement times for $ETCO_2$, $PaCO_2$, and pH variables for a patient from D_S dataset.

matrix $\mathbf{R} \in \mathbb{R}^{T \times T}$ instead of the function R, where $\mathbf{R}_{l,l'} = R(t_l, t_{l'})$. Then, the covariance function becomes a matrix as well, and if we arrange the indices of $\mathbf{R}$ appropriately, we can decompose the covariance matrix $\mathbf{K}$ as a Kronecker product,

$$\mathbf{K} = \mathbf{C} \otimes \mathbf{R} = \begin{bmatrix} \mathbf{C}_{11}\mathbf{R} & \dots & \mathbf{C}_{1N}\mathbf{R} \\ \mathbf{C}_{21}\mathbf{R} & \dots & \mathbf{C}_{2N}\mathbf{R} \\ \vdots & \ddots & \vdots \\ \mathbf{C}_{N1}\mathbf{R} & \dots & \mathbf{C}_{NN}\mathbf{R} \end{bmatrix},$$

using both the separability assumption and aligned measurement times.

6.4.2 Alternating Optimization

We are given a dataset with samples from each patient, where one sample $\mathbf{x}$ is a set of observations $x_{i,l}$, where i indexes the variables, and $l \in \{1, \ldots, T\}$ indexes the measurement times $t_l \in S$. For one patient sample

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_1 & \ldots & \mathbf{x}_N \end{bmatrix} \text{ where } \mathbf{x}_i = \begin{bmatrix} x_{i,1} & \ldots & x_{i,T} \end{bmatrix},$$

the log-likelihood is

$$\ln \mathcal{L} = -\frac{1}{2}\ln(|\mathbf{C}|^T|\mathbf{R}|^N) - \frac{1}{2}(\mathbf{x}(\mathbf{C}^{-1} \otimes \mathbf{R}^{-1})\mathbf{x}^\top) - \frac{d}{2}\ln(2\pi) \tag{6.5}$$

$$= -\frac{T}{2}\ln|\mathbf{C}| - \frac{N}{2}\ln|\mathbf{R}| - \frac{1}{2}\sum_{i,j}(\mathbf{C}^{-1})_{i,j}(\mathbf{x}_i\mathbf{R}^{-1}\mathbf{x}_j^\top) - \frac{d}{2}\ln(2\pi). \tag{6.6}$$

Then, the maximum likelihood estimation for $\mathbf{C}$ is

$$\mathbf{C}_{ij} = \frac{1}{T}\sum_{i,j}(\mathbf{x}_i\mathbf{R}^{-1}\mathbf{x}_j^\top)\,. \tag{6.7}$$

So, we can optimize $\mathbf{C}$ and $\mathbf{R}$ alternatingly because of the Kronecker product decomposition. We choose a squared exponential kernel for $\mathbf{R}$,

$$R(t, t') = \exp\left[-\frac{1}{\sigma^2}\|t - t'\|^2\right], \tag{6.8}$$

where σ is the width parameter. First we use an initial value of $\mathbf{C}$, and do a line search on σ on Equation 6.6 while holding $\mathbf{C}$ fixed. Then, we estimate $\mathbf{C}$ using Equation 6.7 holding $\mathbf{R}$ fixed. In this way, we alternatingly optimize $\mathbf{C}$ and $\mathbf{R}$ until they converge.

Therefore when learning the covariance matrix $\mathbf{K}$ from the dataset, we avoid generating the full $\mathbf{K}$ matrix explicitly and taking the inverse of a high-dimensional matrix. Instead we work on small matrices $\mathbf{C}$ and $\mathbf{R}$, and calculate only their inverses.

6.5 Experiments

We use the D_S dataset as training, and D_M dataset as testing dataset. D_S has more patients, and it is from a single PICU, which is why we chose it as a training set. D_M is a smaller dataset and it is collected from multiple PICUs, therefore we chose it as a test set to demonstrate the transferability of our results.

We estimate the covariance matrix K of our MGP model from the D_S dataset. For each $PaCO_2$ and pH reading in D_M, we estimated the current $PaCO_2$ and pH, using only the patients' currently available non-invasive measures, and previously available $PaCO_2$, pH and non-invasive measures. Specifically, we included the following variables: $ETCO_2$ OSI, PIP, PEEP, VR, V_T, MV, Cdyn, and previous values for pH, $PaCO_2$, $ETCO_2$. See Tables 6.1, 6.2, 6.3 for definitions.

When testing on the D_M dataset, we assume we know the measurements for all components except for the current values of $PaCO_2$ and pH. We jointly model all points of interest, and condition on the known values. Specifically, we predict the mean and covariance of the marginal distributions of $PaCO_2$ and pH at the current time given all known measurements for all the components up until and including the current time.

For comparison of prediction rates, we created baseline models which are based on standard derivations clinicians use. For $PaCO_2$, we use the AVDSF model (see Section 6.2.2) which assumes the ($PaCO_2$ - $ETCO_2$) / $PaCO_2$ ratio is constant from the last observation to the current.

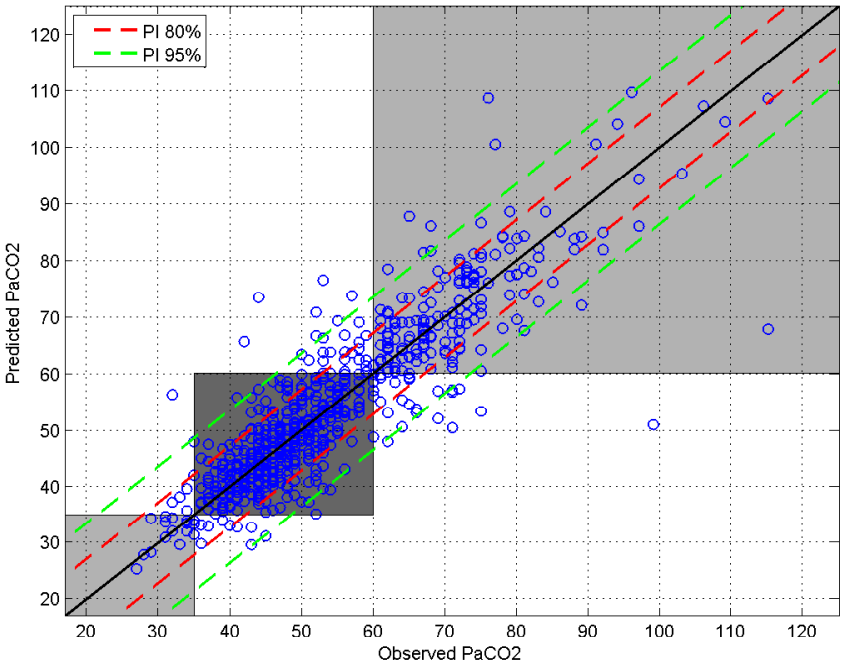
Baseline model formulations	
PaCO$_2$	Use previous (PaCO$_2$ - ETCO$_2$) / PaCO$_2$ ratio on AVDSF model
	$PaCO_2^{AVDSF} = \frac{PaCO_2^{Previous}}{EtCO_2^{Previous}} \times EtCO_2^{Current}$
pH	Use previous value of HCO_3 and $PaCO_2^{AVDSF}$ in HH equation
	$pH^{HH} = 6.1 + log\frac{HCO_3^{Previous}}{0.03 \times PaCO_2^{AVDSF}}$

Table 6.4: Baseline models for PaCO$_2$ and pH that are used by clinicians at bedside.

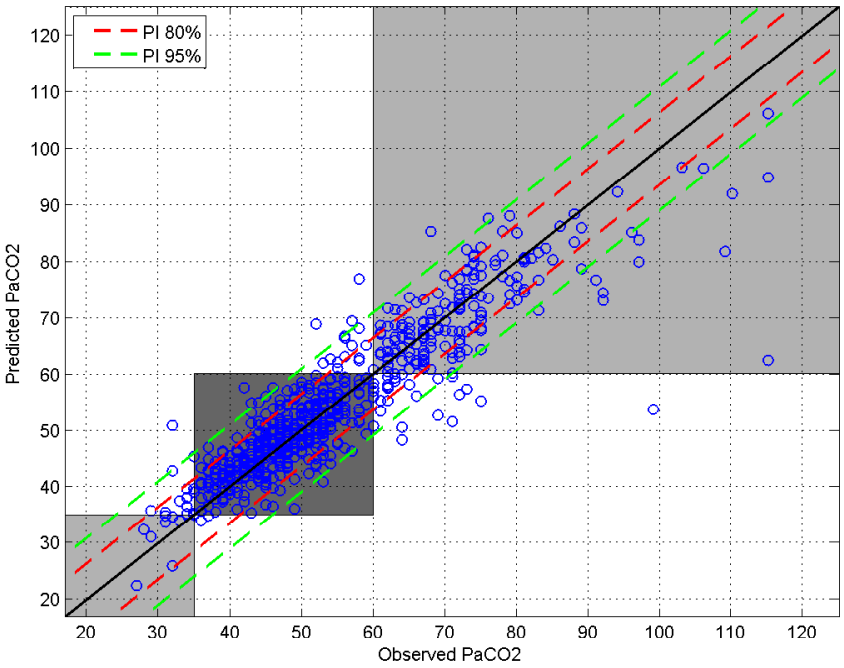
	Baseline Models		**MGP Model**	
	80% PI	**95% PI**	**80% PI**	**95% PI**
PaCO$_2$	$\pm$7 mmHg	$\pm$13 mmHg	$\pm$6 mmHg	$\pm$11 mmHg
pH	$\pm$0.07	$\pm$0.13	$\pm$0.05	$\pm$0.10

Table 6.5: The 80% and 95% prediction intervals for the baseline and MGP models to predict PaCO$_2$ and pH.

This ratio is then used to predict the expected current value for PaCO$_2$, based on the new value of ETCO$_2$. The baseline model for pH uses this prediction in the HH equation which is a standard equation for this case. In order to evaluate the prediction rates, we generated 80% and 95% prediction intervals around the estimates. We also binned the observed and predicted values of PaCO$_2$ and pH into three bins based on the proposed ventilator protocol in Khemani et al. [2011b]. According to this protocol, the ventilator setting changes are the same for the values inside the same bin. If the predicted value falls into the same bin as the observed value, then the protocol could be followed by using the predicted value. The ranges for the bins are as follows: PaCO$_2$ ranges are $[< 35, 35 - 60, > 60]$ mmHg, and the pH ranges are $[\leq 7.3, 7.3 - 7.44, \geq 7.44]$.

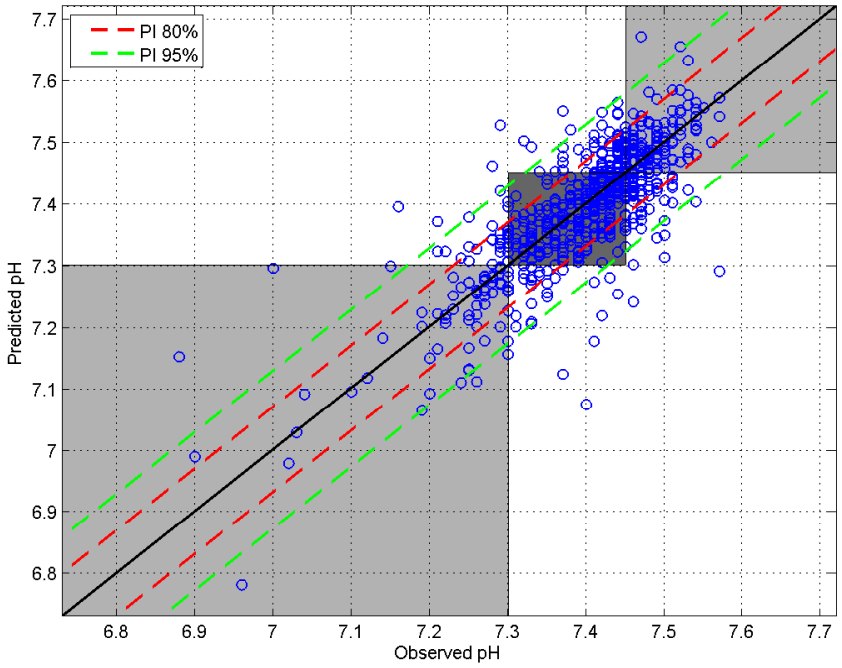


(a) $PaCO_2$ estimate based on the AVDSF equation

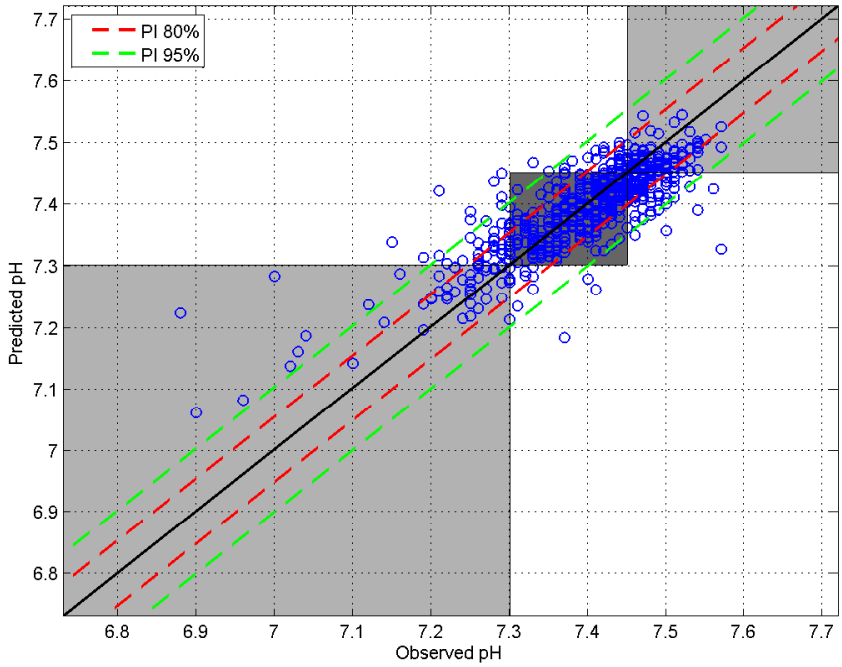


(b) $PaCO_2$ estimate based on the MGP model

Figure 6.3: Observed versus predicted values of $PaCO_2$.



(a) pH estimate based on the Henderson-Hasselbalch equation



(b) pH estimate based on the MGP model

Figure 6.4: Observed versus predicted values of pH.

	CLIA Standard
$PaCO_2$	$\max(\pm 5\text{mmHg}, \pm 8\%)$
pH	± 0.04

Table 6.6: The Clinical Laboratory Improvement Amendments (CLIA) regulations require blood test results to be in the given ranges. The MGP 80% prediction intervals are very close to these acceptable ranges.

Our MGP model performed reasonably well, and better than the baseline models. Figures 6.3 and 6.4 show the observed versus predicted values for $PaCO_2$ and pH respectively. In each figure, the red line represents the 80% prediction interval, while the green line represents the 95% prediction interval. The bins are shown by the shaded boxes, and the points that lie in the boxes are classified into the correct bin.

For $PaCO_2$, MGP predicted 95% of the values within ± 11 mmHg of observed values, while the baseline AVDSF model predicted them within ± 13 mmHg. For 80%, the interval is ± 6 mmHg for MGP and ± 7 mmHg for the baseline AVDSF. As seen in Figure 6.3, a larger number of $PaCO_2$ values predicted by MGP fall in the correct bin than the baseline AVDSF model, especially in the normal bin. As seen in Figure 6.4, prediction intervals for pH are narrower for MGP than the HH model for each bin. In the MGP model 95% of the predicted pH values within ± 0.10, and in HH equation, they are within ± 0.13. The range for 80% is ± 0.05 in the MGP model, and ± 0.07 in the HH equation.

The Clinical Laboratory Improvement Amendments (CLIA), which are regulations for all clinicial laboratory testing in the U.S., require the blood test results to be in the predetermined range of the target value for each blood gas. The equipment is given 5 samples to test, and at least 80% of the results have to be in the required range. The required ranges are: greater of ± 5 mmHg or

$\pm 8\%$ for $PaCO_2$, and ± 0.04 for pH [Medicare, 1992]. The MGP 80% prediction intervals are very close to these required ranges. However, our model depends on a history of previous and current measurements, and the current blood sample is not sufficient. Therefore, a different regulation standard is needed for models like ours. Nonetheless, CLIA standards give us a way to evaluate our MGP model and show when our results can be trusted by clinicians.

6.6 Summary

Ventilator support, critical for children with respiratory failure, requires close monitoring of the ventilator settings, vital signs, and blood gas lab tests, particularly pH and $PaCO_2$. As the blood gas measurements are invasive and therefore not so frequent, estimating them from non-invasive measurements could help clinicians make informed decisions more frequently, or even semi-automate ventilation support. In this work, we developed an MGP model using the available non-invasive measurements that predicts pH and $PaCO_2$. The results show that our MGP model improves prediction compared to baseline models that clinicians currently use.

Chapter 7

Conclusion

In this thesis, we explored inference methods and applications of continuous-time models. Continuous-time models are increasing in popularity because they provide efficient computations when modeling and reasoning about real-life temporal data. They alleviate the need to find a global time slice width (discretization) for the whole system as in discrete-time models.

We developed two inference methods for Markovian models. The first one is a filtering algorithm with deterministic approximation. It uses uniformization with the CTBN factored representation to keep the state distribution in a factored state space. We provided a theoretical analysis of its error bound, and show that it does not increase asymptotically.

The second inference method we developed is both deterministic and also can be stopped anytime to obtain a valid solution. It fills the need for an anytime algorithm that converges to the true result without being random. This is especially useful in parameter estimation algorithms such as EM. In this method, we use CTBN representation with the time-ordered products, and we can calculate any expectation of the marginal state distributions.

These inference methods can be extended to sufficient statistics algorithms, and used for

efficient learning of model parameters and structures. This is especially useful for learning large models as the model size can be a restricting factor in developing accurate representations of real-life systems. Because our methods are deterministic, they will be more reliable for learning algorithms. Additionally, our second algorithm can be used to check the accuracy of other approximate inference algorithms. When the model is too large to run exact inference, our algorithm can give a reliable result.

Lastly, we apply continuous-time models to medical informatics. The setting of our problem necessitates using continuous valued variables in our model. Therefore, we use a multivariate Gaussian process instead of a discrete state Markovian model. Our method estimates the blood gas values of pediatric patients with lung disease, during the time they are mechanically ventilated in the PICU. The blood gas values are required for optimum ventilation management, but they involve invasive blood tests. Frequent blood tests are challenging for patient in PICU, therefore our algorithm's estimates might enable better and shorter mechanical ventilation.

We believe our work in this thesis show that continuous-time models can be efficient and accurate, and they can handle the temporal data streams that are generated everyday in many fields. Especially in healthcare, these models can bring us one step closer to significantly improve patient care and life quality.

Bibliography

Adnan Aziz, Kumud Sanwal, Vigyan Singhal, and Robert Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic (TOCL)*, 1(1):162–170, 2000.

Xavier Boyen and Daphne Koller. Tractable inference for complex stochastic processes. In *Proceedings of the Fourteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 33–42, 1998.

Xavier Boyen and Daphne Koller. Exploiting the architecture of dynamic systems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence*, pages 313–320, 1999.

Jerry R. Burch, Edmund M. Clarke, and David E. Long. Symbolic model checking with partitioned transition relations. In *International Conference on Very Large Scale Integration*, pages 49–58, August 1991.

E. Busra Celikkaya and Christian R. Shelton. Deterministic anytime inference for stochastic continuous-time Markov processes. In *Proceedings of the Thirty-First International Conference on Machine Learning*, 2014.

E. Busra Celikkaya, Christian R. Shelton, and William Lam. Factored filtering of continuous-time systems. In *Proceedings of the Twenty-Seventh International Conference on Uncertainty in Artificial Intelligence*, 2011.

Theodore Charitos, Linda C Van Der Gaag, Stefan Visscher, Karin AM Schurink, and Peter JF Lucas. A dynamic Bayesian network for diagnosing ventilator-associated pneumonia in ICU patients. *Expert Systems with Applications*, 36(2):1249–1258, 2009.

Gianfranco Ciardo and Andrew S Miner. Implicit data structures for logic and stochastic systems analysis. *ACM SIGMETRICS Performance Evaluation Review*, 32(4):4–9, 2005.

Gianfranco Ciardo and Andy Jinqing Yu. Saturation-based symbolic reachability analysis using conjunctive and disjunctive partitioning. In *Proceedings of Correct Hardware Design and Verification Methods*, pages 146–161, 2005.

Daniele Codetta-Raiteri and Luigi Portinale. Generalized continuous time Bayesian networks and their GSPN semantics. *on Probabilistic Graphical Models*, page 105, 2010.

Ido Cohn, Tal El-Hay, Raz Kupferman, and Nir Friedman. Mean field variational approximation for continuous-time Bayesian networks. In *Proceedings of the Twenty-Fifth International Conference on Uncertainty in Artificial Intelligence*, 2009.

Ido Cohn, Tal El-Hay, Nir Friedman, and Raz Kupferman. Mean field variational approximation for continuous-time Bayesian networks. *The Journal of Machine Learning Research*, 11:2745–2783, 2010.

Thomas M. Cover and Joy A. Thomas. *Elements of Information Theory*. John Wiley & Sons, Inc., 1991.

Joseph A Cruz and David S Wishart. Applications of machine learning in cancer prediction and prognosis. *Cancer informatics*, 2:59, 2006.

Jesse Davis, Eric Lantz, David Page, Jan Struyf, Peggy Peissig, Humberto Vidaillet, and Michael Caldwell. Machine learning for personalized medicine: Will this drug give me a heart attack. In *Proceedings of International Conference on Machine Learning*, 2008.

E. de Souza e Silva and R. Gail. Transient solutions for Markov chains. In *Computational Probability*, pages 43–79. Kluwer Academic, 2000.

Thomas Dean and Keiji Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150, 1989.

A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society B*, 39(1):1–38, 1977.

Darrell Duffie and Peter Glynn. Estimation of continuous-time Markov processes sampled at random time intervals. *Econometrica*, pages 1773–1808, 2004.

F. J. Dyson. The radiation theories of Tomonaga, Schwinger, and Feynman. *Physical Review*, 75 (3):486–502, 1949.

Tal El-Hay, Nir Friedman, and Raz Kupferman. Gibbs sampling in factorized continuous-time Markov processes. In *Proceedings of the Twenty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 169–178, 2008.

Tal El-Hay, Ido Cohn, Nir Friedman, and Raz Kupferman. Continuous-time belief propagation. In *Proceedings of the 27th International Conference on Machine Learning*, pages 343–350, June 2010.

ESRC Research Centre on Micro-social Change. British household panel survey. Computer Data File and Associated Documentation. http://www.iser.essex.ac.uk/bhps, 2003. Colchester: The Data Archive.

Yu Fan. *Continuous Time Bayesian Network Approximate Inference and Social Network Applications*. PhD thesis, University of California at Riverside, December 2009.

Yu Fan and Christian R. Shelton. Sampling for approximate inference in continuous time Bayesian networks. In *Proceedings of the Tenth International Symposium on Artificial Intelligence and Mathematics*, 2008.

Yu Fan and Christian R. Shelton. Learning continuous-time social network dynamics. In *Proceedings of the Twenty-Fifth International Conference on Uncertainty in Artificial Intelligence*, 2009.

Yu Fan, Jing Xu, and Christian R. Shelton. Importance sampling for continuous time Bayesian networks. *Journal of Machine Learning Research*, 11(Aug):2115–2140, 2010.

JA Farias, A. Fernndez, E. Monteverde, J.C. Flores, A Baltodano, A. Menchaca, R. Poterala, F. Pnico, M. Johnson, B. von Dessauer, et al. Mechanical ventilation in pediatric intensive care units during the season for acute lower respiratory infection: a multicenter study. *Pediatric Critical Care Medicine*, 13(2):158–64, 2012.

Heidi R Flori, David V Glidden, George W Rutherford, and Michael A Matthay. Pediatric acute lung injury: prospective evaluation of risk factors associated with mortality. *American Journal of Respiratory and Critical Care Medicine*, 171(9):995–1001, 2005.

E Gatti, D Luciani, and F Stella. A continuous-time Bayesian network model for cardiogenic heart failure. *Flexible Services and Manufacturing Journal*, 24(4):496–515, 2012.

A.E. Gelfand and S. Banerjee. Multivariate spatial process models. Technical report, Research Report 2010-9, Division of Biostatistics, University of Minnesota, In Handbook of Spatial Statistics, 2010.

Karthik Gopalratnam, Henry Kautz, and Daniel S. Weld. Extending continuous time Bayesian networks. In *Proceedings of 20th. National Conference on Artificial Intelligence-AAAI 2005*, pages 981–986, 2005.

Fabian Guiza Grandas, Hendrik Blockeel, Maurice Bruynooghe, Kristien Van Loon, Jean-Marie Aerts, Daniel Berckmans, Geert Meyfroidt, and Greet Van den Berghe. Time-series analysis techniques combined with Gaussian process classifiers for prediction of clinical stability after coronary bypass surgery. In *Proceedings of the 6th IASTED*, number 6, pages 216–221, 2008.

Asela Gunawardana, Christopher Meek, and Puyang Xu. A model for temporal dependencies in event streams. In *Advances in Neural Information Processing Systems*, volume 24, 2012.

Josef Guttmann. Prediction of mechanical lung parameters using Gaussian process models. *Artificial Intelligence in Medicine*, 5651:380, 2009.

Monir Hajiaghayi, Bonnie Kirkpatrick, Liangliang Wang, and Alexandre Bouchard-Côté. Efficient continuous-time Markov chain estimation. In *Proceedings of The 31st International Conference on Machine Learning*, pages 638–646, 2014.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57:97–109, 1970.

Ralf Herbrich, Thore Graepel, and Brendan Murphy. Structure from failure. In *Proceedings of the 2nd USENIX Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*, pages 1–6. USENIX Association, 2007.

Asger Hobolth and Eric A Stone. Simulation from endpoint-conditioned, continuous-time Markov chains on a finite state space, with applications to molecular evolution. *Annals of Applied Statistics*, 3(3):1204, 2009.

Andreas Holzinger, Matthias Dehmer, and Igor Jurisica. Knowledge discovery and interactive data mining in bioinformatics-state-of-the-art, future challenges and research directions. *BMC Bioinformatics*, 15(Suppl 6):I1, 2014.

D Kartson, Gianfranco Balbo, S Donatelli, G Franceschinis, and Giuseppe Conte. *Modelling with generalized stochastic Petri nets*. John Wiley & Sons, Inc., 1994.

Robinder G. Khemani, Barry P. Markovitz, and Martha A. Q. Curley. Characteristics of children intubated and mechanically ventilated in 16 PICUs. *American College of Chest Physicians*, 136 (3):765–71, 2009.

Robinder G. Khemani, Dave Kale, Patrick A. Ross, and Randall C. Wetzel. Predicting serum pH using non-invasive measurements to enable decision support for mechanically ventilated children with acute lung injury. Technical report, Children's Hospital Los Angeles; USC Keck School of Medicine, 2011a.

Robinder G Khemani, Katherine Sward, Alan Morris, J Michael Dean, and Christopher J L Newth. Variability in usual care mechanical ventilation for pediatric acute lung injury: the potential benefit of a lung protective computer protocol. *Intensive care medicine*, 37(11):1840–8, 2011b.

Robinder G Khemani, Neal J Thomas, Vani Venkatachalam, Jason P Scimeme, Ty Berutti, James B Schneider, Patrick A Ross, Douglas F Willson, Mark W Hall, and Christopher JL Newth. Comparison of SpO2 to PaO2 based markers of lung disease severity for children with acute lung injury. *Critical care medicine*, 40(4):1309–1316, 2012.

Robinder G. Khemani, E. Busra Celikkaya, Christian R. Shelton, Dave Kale, Patrick A. Ross, Randall C. Wetzel, and Christopher J. L. Newth. Algorithms to estimate PaCO2 and pH using non-invasive parameters for children with hypoxemic respiratory failure. *Respiratory Care*, 59(8): 1248–1257, August 2014.

A. Louw, C. Cracco, C. Cerf, A. Harf, P. Duvaldestin, F. Lemaire, and L. Brochard. Accuracy of pulse oximetry in the intensive care unit. *Intensive Care Medicine*, 27(10):1606–1613, 2001.

Ligia Mateiu and Bruce Rannala. Inferring complex DNA substitution processes on phylogenies using uniformization and data augmentation. *Systematic biology*, 55(2):259–269, 2006.

Medicaid Medicare. CLIA programs: regulations implementing the clinical laboratory improvement amendments of 1988 (CLIA). *Federal Register*, 57(40):7002–186, 1992.

Geert Meyfroidt, Fabian Güiza, Jan Ramon, and Maurice Bruynooghe. Machine learning techniques to examine large patient databases. *Best Practice & Research Clinical Anaesthesiology*, 23(1): 127–143, 2009.

Thomas P. Minka. Expectation propagation for approximate Bayesian inference. In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pages 362–369, 2001.

Eric Mjolsness and Guy Yosiphon. Stochastic process semantics for dynamical grammars. *Annals of Mathematics and Artificial Intelligence*, 47(3–4):329–395, August 2006.

Cleve Moler and Charles Van Loan. Nineteen dubious ways to compute the exponential of a matrix, twenty-five years later. *SIAM Review*, 45(1):3–49, 2003.

Igor Najfeld and Timothy F. Havel. Derivatives of the matrix exponential and their computation. *Advances in Applied Mathematics*, 16:321–375, 1995.

Brenda Ng, Avi Pfeffer, and Richard Dearden. Continuous time particle filtering. In *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence*, pages 1360–1365, 2005.

Uri Nodelman and Eric Horvitz. Continuous time Bayesian networks for inferring users' presence and activities with extensions for modeling and evaluation. Technical Report MSR-TR-2003-97, Microsoft Research, December 2003.

Uri Nodelman, Christian R. Shelton, and Daphne Koller. Continuous time Bayesian networks. In *Proceedings of the Eighteenth International Conference on Uncertainty in Artificial Intelligence*, pages 378–387, 2002.

Uri Nodelman, Daphne Koller, and Christian R. Shelton. Expectation propagation for continuous time Bayesian networks. In *Proceedings of the Twenty-First International Conference on Uncertainty in Artificial Intelligence*, pages 431–440, 2005a.

Uri Nodelman, Christian R. Shelton, and Daphne Koller. Expectation maximization and complex duration distributions for continuous time Bayesian networks. In *Proceedings of the Twenty-First International Conference on Uncertainty in Artificial Intelligence*, pages 421–430, 2005b.

C. A. Petri. *Communication with Automata*. PhD thesis, University of Bonn, 1962.

Bernard Philippe and Roger B Sidje. *Transient solutions of Markov processes by Krylov subspaces*. Springer, 1995.

Henry N. Po and N.M. Senozan. The Henderson-Hasselbalch equation: its history and limitations. *Journal of Chemical Education*, 78(11):1499, 2001.

Luigi Portinale and Daniele Codetta-Raiteri. Generalizing continuous time bayesian networks with immediate nodes. In *Proceedings of the Workshop on Graph Structure for Knowledge Represetnation and Reasoning*, pages 12–17, 2009.

William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

Zhen Qin and Christian R. Shelton. Auxiliary Gibbs sampling for inference in piecewise-constant conditional intensity models. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, 2015.

Vinayak Rao and Yee W Teh. MCMC for continuous-time discrete-state systems. In *Advances in Neural Information Processing Systems*, pages 701–709, 2012.

Vinayak Rao and Yee Whye Teh. Fast MCMC sampling for Markov jump processes and continuous time Bayesian networks. In *Proceedings of the Twenty-Seventh International Conference on Uncertainty in Artificial Intelligence*, 2011.

Vinayak Rao and Yee Whye Teh. Fast MCMC sampling for Markov jump processes and extensions. *The Journal of Machine Learning Research*, 14(1):3295–3320, January 2013.

Carl Edward Rasmussen. *Gaussian Processes for Machine Learning*. Citeseer, 2006.

Andrew Reibman and Kishor Trivedi. Numerical transient analysis of Markov models. *Computer & Operations Research*, 15(1):19–36, 1988.

Nicolas Rodrigue, Hervé Philippe, and Nicolas Lartillot. Uniformization for sampling realizations of Markov processes: applications to Bayesian implementations of codon substitution models. *Bioinformatics*, 24(1):56–62, 2008.

Youcef Saad. *Numerical Methods for Large Eigenvalue Problems*. Manchester University Press Series, 1992.

Werner Sandmann and Verena Wolf. A computational stochastic modeling formalism for biological networks. In *Enformatika Transactions on Engineering, Computing and Technology 14*, pages 132–137, 2006.

Suchi Saria, Uri Nodelman, and Daphne Koller. Reasoning at the right time granularity. In *Proceedings of the Twenty-third Conference on Uncertainty in AI*, pages 421–430, 2007.

Suchi Saria, Daphne Koller, and Anna Penn. Learning individual and population level traits from clinical temporal data. In *Proc. Neural Information Processing Systems (NIPS), Predictive Models in Personalized Medicine workshop*. Citeseer, 2010.

J. W. Severinghaus, M. A. Stupfel, and A. F. Bradley. Alveolar dead space and arterial to end-tidal carbon dioxide differences during hypothermia in dog and man. *Journal of Applied Physiology*, 10(3):349–355, 1957.

Christian R. Shelton and Gianfranco Ciardo. Tutorial on continuous-time Markov processes. *Journal of Artificial Intelligence Research*, 51:725–778, December 2014.

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

Roger B. Sidje, Kevin Burrage, and Shev MacNamara. Inexact uniformization method for computing transient distributions of Markov chains. *SIAM Journal of Scientific Computation*, 29(6): 2562–2580, 2007.

William J. Stewart. *Introduction to the Numerical Solution of Markov Chains*. Princeton University Press, Princeton, NJ, USA, 1994.

Charles A. Sutton and Michael I. Jordan. Probabilistic inference in queueing networks. In Armando Fox and Sumit Basu, editors, *Proceedings of Third Workshop on Tackling Computer Systems Problems with Machine Learning Techniques*. USENIX Association, 2008.

Min Wan, Gianfranco Ciardo, and Andrew S. Miner. Approximate steady-state analysis of large Markov models based on the structure of their decision diagram encoding. *Performance Evaluation*, 68(5):463–486, 2011.

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

Jeremy C. Weiss, Sriraam Natarajan, and David Page. Learning to reject sequential importance steps for continuous-time Bayesian networks. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, 2015.

Jing Xu and Christian R. Shelton. Intrusion detection using continuous-time Bayesian networks. *Journal of Artificial Intelligence Research*, 39:745–774, 2010.

YM Yang, WD Huang, MY Shen, and ZR Xu. Comparative study of pressure-control ventilation and volume-control ventilation in treating traumatic acute respiratory distress syndrome. *Chinese Journal of Traumatology*, 8(1):36–38, 2005.

Xiang Zhou and Scott C. Schmidler. Bayesian parameter estimation in Ising and Potts models: A comparative study with applications to protein modeling. Technical report, Duke University, 2009.