Deterministic Anytime Inference for Stochastic Continuous-Time Markov Processes

E. Busra Celikkaya CELIKKAE@CS.UCR.EDU
University of California, Riverside

Christian R. Shelton CSHELTON@CS.UCR.EDU
University of California, Riverside

Abstract

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.

1. Continuous-time stochastic systems

Continuous-time discrete-state stochastic models describe systems in which event times are not synchronized with a global clock. Examples include web searches (Gunawardana et al., 2012), computer networks (Xu & Shelton, 2010), social networks (Fan & Shelton, 2009), robotics (Ng et al., 2005), system verification (Baier et al., 2003), and phylogenetic trees (Cohn et al., 2009), among others. Discretizing time can be computationally expensive. The "time-slice" width must be much smaller than the shortest time between events. This can lead to inefficient computations during times in which events or expected events are less frequent. Much as the abstraction of real-valued numbers (and their implementation in floating-point rather than fixed-point representations) is helpful in the development of numeric algorithms, continuous-time is useful for stochastic dynamics systems.

This paper focuses on Markovian models. In a discrete-time Markov process, given a row-stochastic matrix M and a distribution v (as a row-vector), the computation of $v_n = vM^n$ propagates v forward n time steps. In a continuous-time Markov process (CTMP), given a rate (intensity) matrix Q, $v_t = ve^{Qt}$ propagates v forward t time units in the same fashion. This is the critical computation step in filtering, smoothing, and parameter estimation. We focus on how to compute this *matrix exponential* when the size of v is very large and both v and Q have structure (allowing their efficient representation).

Except for the most trivial of cases, v_t has no internal structure. In particular, assume that the state space is factored, that is composed of joint assignments to state variables. Even if v is completely independent, v_t no longer has any structure (unless Q also represents a completely independent system). This is the same problem that arises in dynamic Bayesian networks (DBNs) in which forward propagation causes all variables in the system to be coupled. We assume that a full distribution over the state space is too large to be stored, and therefore seek an approximation.

1.1. Previous work

This problem has received attention in the verification literature for decision-diagram-based representations of the intensity matrix Q. However, the assumption behind this literature is that while Q may have structure to keep it representable, an exact answer is desired and therefore v_t is represented as a full vector. The shuffle algorithm is one such example (Fernandes et al., 1998).

By contrast, we assume that representing v_t explicitly is not possible. We would like to calculate expectations with respect to the distribution v_t. In our approach, we concentrate on continuous-time Bayesian networks (Nodelman et al., 2002) (CTBNs), but the method is general to any Q that is the sum of Kronecker products. Even the simplest

Proceedings of the 31^{st} International Conference on Machine Learning, Beijing, China, 2014. JMLR: W&CP volume 32. Copyright 2014 by the author(s).

expectations (like marginals) are NP-hard to compute (the proof is a straightforward extension of the proof for general Bayesian networks), so we focus on approximations. In the literature on CTBNs, there are a number of such methods that fall roughly into two groups. The first are variational approaches such as expectation propagation (El-Hay et al., 2010) and mean field (Cohn et al., 2009). These methods are deterministic. However, they do not converge to the true value as computation time increases and generally can only compute marginals or similar expectations. The second group are sampling approaches including importance sampling (Fan et al., 2010) and Gibbs sampling (Rao & Teh, 2011). These approaches converge to the true value and can estimate any expectation of v_t. However, they are random and this can cause problems when used inside other algorithms (like expectation-maximization).

1.2. Our approach

Our proposed method is deterministic and converges in the limit of infinite computational time. It can be viewed as a bridge between sampling and deterministic methods. 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 2.4, 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.

2. Matrix exponential calculation

Consider a CTMP with discrete states which is described by an initial state distribution row-vector v of size n and a rate matrix Q of size n-by-n. The rate matrix represents the rates by which the system transitions between states. The rate of transitioning from state i to j is $q_{ij} \geq 0$ and the rate of transitioning out of state i is $q_i = \sum_j q_{ij}$. The diagonal elements of the rate matrix are the negative row sums: $q_{ii} = -q_i$.

As stated above, the distribution at time t, for $t \geq 0$, is calculated by $v_t = ve^{Qt}$ where e^{Qt} is the matrix exponential with Taylor expansion $e^{Qt} = \sum_{k=0}^{\infty} \frac{1}{k!}(Qt)^k$. The matrix exponential calculation is very widely used in applied mathematics and there are many numerical methods to solve it (Moler & Loan, 2003).

If the state-space is structured as joint assignments to m variables, its size, n, grows exponentially with the number of variables, m. This makes the e^{Qt} calculation 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 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 seen as a perturbation of e^{At} in the direction of Bt (Najfeld & Havel, 1995) and can be represented as

$$e^{(A+B)t} = e^{At} + \int_0^t e^{As} B e^{A(t-s)}\, ds + \int_0^t \left(\int_0^s e^{Ar} B e^{A(s-r)}\, dr \right) B e^{A(t-s)}\, ds + \dots \tag{1}$$

which is a sum of recursive functions. 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 & Yosiphon (2006) called it a time-ordered product expansion and used it to guide a sampling algorithm. We will employ the expansion to derive our deterministic method, Tree of Time-Ordered Products (TTOP).

2.1. TOP 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 1 and apply the distributive property of matrix multiplication to move the sum outside the integral:

$$ve^{(A+\sum_{j=1}^{J} B_j)t} = ve^{At} + \sum_{j=1}^{J} \int_0^t ve^{As} B_j e^{A(t-s)}\, ds + \sum_{j=1}^{J} \sum_{j'=1}^{J} \int_0^t \left(\int_0^s ve^{Ar} B_j e^{A(s-r)}\, dr \right) B_{j'} e^{A(t-s)}\, ds + \dots \tag{2}$$

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

$$ve^{Qt} = \sum_{l=0}^{\infty} F^l(t) \tag{3}$$

where

$$F^0(t) = ve^{At}$$
$$F^l(t) = \sum_{j=1}^{J} \int_0^t F^{(l-1)}(s) B_j e^{-As}\, ds\; e^{At}. \quad (4)$$

By construction, the first term, ve^{At}, is simple to solve (as will be explained in Section 2.2).

2.1.1. 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 $q_{[r_1,r_2]}(t) = I[r_1 \le t < r_2]q(t)$ (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

$$\begin{aligned} h(t; q, g, g_0) &= \bar{q}(t)(g(s_0) - g_0) \\ &\quad + \int_0^t q_{[a,s_0]}(s)\,(g(s) - g(s_0))\; ds \\ &\quad + \int_0^t q_{[s_0,b]}(s)\,(g(s) - g(s_0))\; ds \\ &= \bar{q}(t)(g(s_0) - g_0) \\ &\quad + h(t; q_{[a,s_0]}, g, g(s_0)) + h(t; q_{[s_0,b]}, g, g(s_0)) \end{aligned} \quad (5)$$

for any chosen g_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 will generate infinitely many terms in the form of $\bar{q}(t)(g(s_0) - g_0)$:

$$h(t; q, g, g_0) = \sum_{k=1}^{\infty} q_k(t) u_k \quad (6)$$

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.

2.1.2. Complete computation tree

Assume level l of Equation 3 can be expressed as

$$F^l(t) = \sum_{k=1}^{\infty} q_k^l(t) u_k^l e^{At} \quad (7)$$

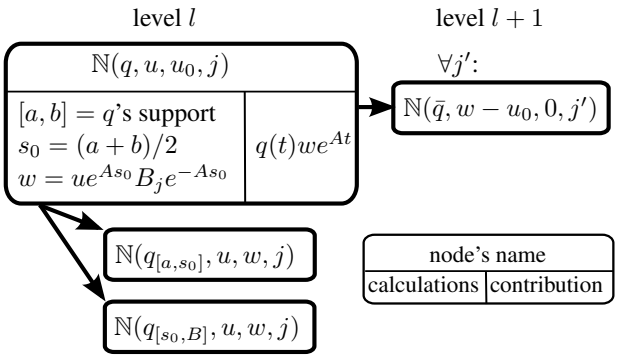


Figure 1. General form of the expansion.

(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:

$$\begin{aligned} F^{l+1}(t) &= \sum_{j=1}^{J} \int_0^t F^l(s) B_j e^{-As}\, ds\, e^{At} \\ &= \sum_{j=1}^{J} \int_0^t \sum_{k'=1}^{\infty} q_{k'}^l(s) u_{k'}^l e^{As} B_j e^{-As}\, ds\, e^{At} \\ &= \sum_{k'=1}^{\infty} \sum_{j=1}^{J} \int_0^t q_{k'}^l(s) u_{k'}^l e^{As} B_j e^{-As}\, ds\, e^{At} \\ &= \sum_{k'=1}^{\infty} \sum_{j=1}^{J} h(t; q_{k'}^l, u_{k'}^l e^{At} B_j e^{-At}, 0)\, e^{At} \\ &= \sum_{k'=1}^{\infty} \sum_{j=1}^{J} \sum_{k=1}^{\infty} q_{k',k,j}^l(t) u_{k',k,j}^l e^{At}. \end{aligned} \quad (8)$$

In last two lines, we have replaced the integral with the expansion of Equation 6. 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 $\left\{q_{k',k,j}^l, u_{k',k,j}^l\right\}_k$.

In Equation 8, 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 7.

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

$$q(t)we^{At}, \text{ where } w = ue^{As_0} B_j e^{-As_0}, \quad (9)$$

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 $s_0 = \frac{a+b}{2}$.

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 9)
  {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

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 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 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. The sum of all nodes at level l describes a system that evolves according to A (instead of Q), but for which at l time points (positioned anywhere), the correction $B = Q - A$ is applied. 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.

2.2. Structured TOP calculations

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 9) 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

$$\bigotimes_{i=1}^{m} x_i - \bigotimes_{i=1}^{m} y_i = \sum_{i'=1}^{m} d_{i'}(x, y) \tag{10}$$

where

$$d_{i'}(x, y) = \left(\bigotimes_{i<i'} y_i\right) \otimes (x_{i'} - y_{i'}) \otimes \left(\bigotimes_{i>i'} x_i\right) \tag{11}$$

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 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 & Yu, 2005). However, they are encoded in a form where $A = 0$. Techniques similar to those we describe next can be applied to

$$Q_1 = \begin{bmatrix} -1 & 1 \\ 3 & -3 \end{bmatrix} \qquad A_1 = \begin{bmatrix} -1 & 1 \\ 3 & -3 \end{bmatrix}$$

$$Q_{2|0} = \begin{bmatrix} -2 & 2 \\ 4 & -4 \end{bmatrix} \qquad B_1 = \underbrace{\begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}}_{\Delta_{1,1}} \otimes \underbrace{\begin{bmatrix} 0 & 0 \\ 1 & -1 \end{bmatrix}}_{Q_{2|1} - A_2 = B_{(2|1),2}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$A_2 = \begin{bmatrix} -2 & 2 \\ 3 & -3 \end{bmatrix}$$

$$Q_{2|1} = \begin{bmatrix} -4 & 4 \\ 3 & -3 \end{bmatrix} \qquad B_2 = \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}}_{\Delta_{2,2}} \otimes \underbrace{\begin{bmatrix} -2 & 2 \\ 0 & 0 \end{bmatrix}}_{Q_{2|2} - A_2 = B_{(2|2),2}} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & -2 & 2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$Q = \begin{bmatrix} -3 & 2 & 1 & 0 \\ 4 & -5 & 0 & 1 \\ 3 & 0 & -7 & 4 \\ 0 & 3 & 3 & -6 \end{bmatrix} \qquad A = A_1 \otimes A_2 = \begin{bmatrix} -3 & 2 & 1 & 0 \\ 3 & -4 & 0 & 1 \\ 3 & 0 & -5 & 2 \\ 0 & 3 & 3 & -6 \end{bmatrix} \qquad B = B_1 + B_2$$

Figure 2. Two-node CTBN and decomposition. The large matrices (4-by-4) are implicit. Q is broken into an independent A and two correction matrices, B_1 and B_2.

pull intensity from the B_j matrices into A, but we will focus on a different representation.

2.3. Continuous time Bayesian networks

A continuous time Bayesian network (CTBN) (Nodelman et al., 2002) is a graphical model which provides a structured representation of a CTMP. The initial distribution is described by a Bayesian network which we assume has no edges (but this work can be extended to dependencies in the initial distribution). 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, for $i = 1, \ldots, m$, has a set of parents U_i. The rate matrix Q is factored into conditional rate matrices, $Q_{i|u_i}$ for every assignment of u_i to U_i. Each conditional intensity matrix gives the rates at which variable X_i transitions at instants when $U_i = u_i$. No two variables can transition at *exactly* the same instant so any element in the global Q matrix describing a change of multiple variables is 0.

The global Q matrix for a CTBN can be represented with Kronecker algebra. Let $R_{i|u_i} = \bigotimes_{i'} R_{(i|u_i),i'}$ be a Kronecker product of one matrix for each variable where

$$R_{(i|u_i),i'} = \begin{cases} Q_{i|u_i} & \text{if } i' = i \\ \Delta_{k,k} & \text{if } i' \in \text{Pa}(X_i) \;\&\; k \text{ is val. of } i' \text{ in } u_i \\ I & \text{otherwise .} \end{cases} \tag{12}$$

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

$$Q = \sum_{i=1}^{M} \sum_{u_i} \left(\bigotimes_{i'=1}^{M} R_{(i|u_i),i'} \right) . \tag{13}$$

This 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

$$B_{(i|u_i),i'} = \begin{cases} Q_{i|u_i} - A_i & \text{if } i' = i \\ \Delta_{k,k} & \text{if } i' \in \text{Pa}(X_i) \;\&\; k \text{ is val. of } i' \text{ in } u_i \\ I & \text{otherwise .} \end{cases} \tag{14}$$

Then

$$Q = \bigoplus_{i} A_i + \sum_{i=1}^{M} \sum_{u_i} \left(\bigotimes_{i'=1}^{M} B_{(i|u_i),i'} \right) . \tag{15}$$

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 $\Delta_{(i|u_i),i'} = I$ 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 u). Thus, the calculations for $\mathbb{N}(q, u, u_0, j)$ are local to the variable j and its parents.

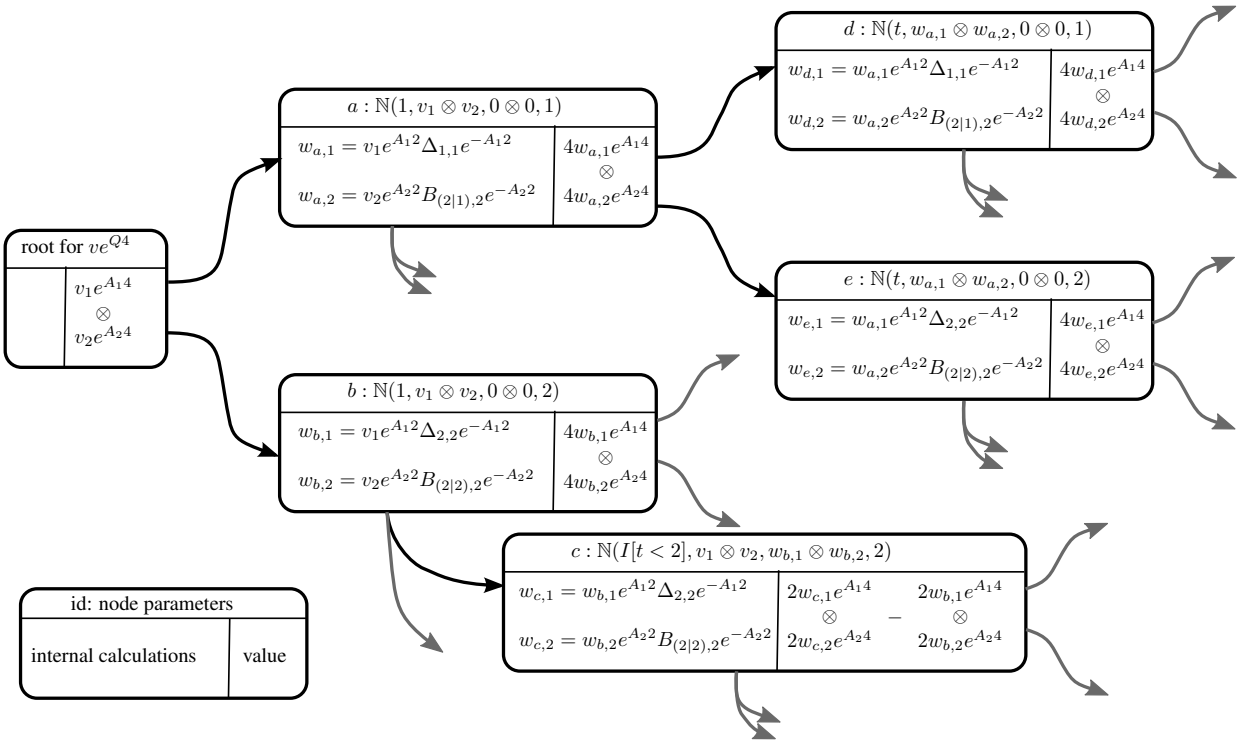


Figure 3. Portion of computation tree for example in Figure 2 for $t = 4$. Each box is one node in the computation tree (see Figure 1). Nodes a and b are the children of the root. Nodes d and e are examples of their children at the next level ($l = 2$). Node c is an example of a refinement (of node b). Equation 11 dictates how the children of c for $l = 2$ are computed because $u_0 \neq 0$ for this node.

2.4. CTBN example

Figure 2 shows a simple 2-variable CTBN and one possible decomposition into A and B. Here the local A_i matrices are chosen to be the minimal rates. Because X_1 has no parents, A_1 is exact and there are no B terms. X_2 generates 2 B terms. If we let $v = v_1 \otimes v_2$ (that is, v_1 and v_2 are the independent marginals of X_1 and X_2), then Figure 3 shows a small portion of the computation tree.

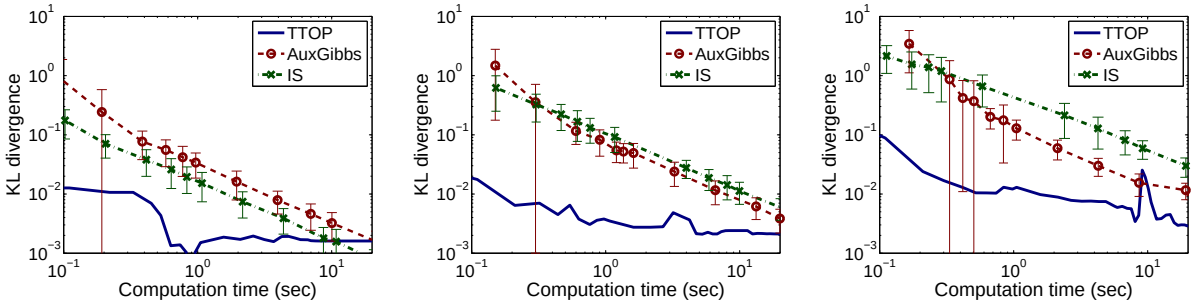
This method essentially computes the effect of A exactly and incorporates the effects of B as a Taylor expansion, adding increasing numbers of terms. Note that in Figure 2, A_1 and A_2 are proper intensity matrices (their rows sum to 0). This means that $B_{2|0}$ and $B_{2|1}$ have negative diagonal 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 $A_2 = \begin{bmatrix} -4 & 2 \\ 3 & -4 \end{bmatrix}$, resulting in $B_{2|0} = \begin{bmatrix} 2 & 0 \\ 1 & 0 \end{bmatrix}$ and $B_{2|1} = \begin{bmatrix} 0 & 2 \\ 0 & 1 \end{bmatrix}$. The disadvantage is that $v_2e^{A_2t}$ sums to less than 1. The non-root nodes add probability to the answer (instead of moving it within the answer).

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

2.5. Smoothing and computational considerations

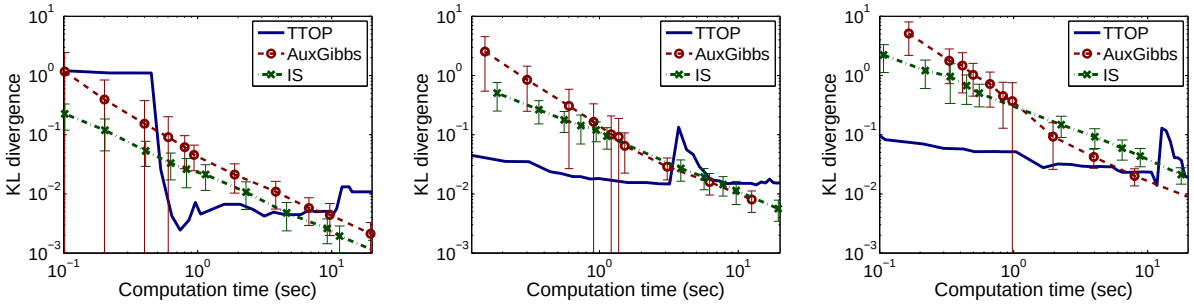
The discussion so far has focused on filtering. We would also like to perform smoothing. We will limit ourselves to the case in which the initial distribution at time 0, v, is known and there is evidence at a later time T for which the vector v_T represents the probability of the evidence for each possible 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 $v_Te^{Q^\top(T-t)}$ (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



(a) $m = 9$ (b) $m = 15$ (c) $m = 21$

Figure 4. Computation time versus KL divergence for the toroid networks when $\tau = 2$, $\beta = 0.5$.



(a) $m = 9$ (b) $m = 15$ (c) $m = 21$

Figure 5. Computation time versus KL divergence for the toroid networks when $\tau = 2$, $\beta = 1$.

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

$$\sum_{j=1}^{N_a} \alpha_j \odot \sum_{k=1}^{N_b} \beta_k = \sum_{j=1}^{N_a} \sum_{k=1}^{N_b} \left(\bigotimes_{i=1}^{m} \alpha_{j,i} \right) \odot \left(\bigotimes_{i=1}^{m} \beta_{k,i} \right) = \sum_{j=1}^{N_a} \sum_{k=1}^{N_b} \bigotimes_{i=1}^{m} (\alpha_{j,i} \odot \beta_{k,i}) \quad (16)$$

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 (absolution value of the change) and the product of the maximum value in each node.

3. Experiments

We implemented our method, TTOP (Tree of Time-Ordered Products), as part of the CTBN-RLE code base (Shelton et al., 2010), and it will be included in the next version. 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 & 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.

Using this model, we generated a directed toroid network structure with cycles following (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 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. 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 & 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) 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 $+1$. For $m = 15$ and 21 we use a similar pattern of evidence for comparison reasons. For $m = 15$, 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.

The nodes are not observed between $t = 0$ and $t = 1$. We query the marginal distributions of nodes at $t = 0.5$. Figures 4 and 5 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 4-b. A KL-divergence sum of 10^{-2} is generally accurate for these networks.

Figure 4 shows 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 Figure 5, 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.

4. Conclusion

We have 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.

Acknowledgments

This work was funded by The Laura P. and Leland K. Whittier Virtual PICU at Children's Hospital Los Angeles (awards UCR-12101024 and 8220-SGNZZ0777-00) and DARPA (award FA8750-14-2-0010).

References

Baier, Christel, Haverkort, Boudewijn, Hermanns, Holger, and Katoen, Joost-Pieter. Model checking algorithms for continuous-time Markov chains. *IEEE Transactions on Software Engineering*, 29(6):524–541, June 2003.

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

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

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

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

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

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

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

Fernandes, Paulo, Plateau, Brigitte, and Stewart, William J. Efficient descriptor-vector multiplication in stochastic automata networks. *Journal of the ACM*, 45(3):381–414, 1998.

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

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

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

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

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

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

Rao, Vinayak and Teh, Yee Whye. 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.

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

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

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