Sampling for Approximate Inference in Continuous Time Bayesian Networks

Yu Fan
University of California, Riverside
yfan@cs.ucr.edu

Christian R. Shelton
University of California, Riverside
cshelton@cs.ucr.edu

Abstract

We first present a sampling algorithm for continuous time Bayesian networks based on importance sampling. We then extend it to continuous-time particle filtering and smoothing algorithms. The three algorithms can estimate the expectation of any function of a trajectory, conditioned on any evidence set constraining the values of subsets of the variables over subsets of the timeline. We present experimental results on their accuracies and time efficiencies, and compare them to expectation propagation.

1 Introduction

Many systems evolve asynchronously in continuous time, for example computer networks, sensor networks, mobile robots, and cellular metabolisms. Continuous time Bayesian networks (CTBNs) (Nodelman, Shelton, & Koller 2002) model such stochastic systems in continuous time using graphs to represent conditional independencies among discrete-valued processes. They have been applied to human-computer interactions (Nodelman & Horvitz 2003), server farm failures (Herbrich, Graepel, & Murphy 2004), and robot monitoring (Ng, Pfeffer, & Dearden 2005). A trajectory (sample) from a CTBN consists of the starting values for the system along with the (real-valued) times at which the variables change and their corresponding new values.

Inference for CTBNs is the task of estimating the distribution over trajectories given a partial trajectory (in which some values or transitions are missing for some variables during some time intervals). Performing exact inference in CTBNs is intractable. Recently Nodelman, Koller, & Shelton (2005) presented an approximate inference method based on expectation propagation (Minka 2001). Saria, Nodelman, & Koller (2007) extended it to full believe propagation and provided a method to adapt the approximation quality. In this paper we explore a different approach. Instead of approximating the distributions involved, we use sampling to approximate the expectation of a function of the trajectory. Sampling has the advantage of being an anytime algorithm. (We can stop at any time during the computation and obtain an answer.) Furthermore, in the limit of infinite samples (computation time), it converges to the true answer. Our algorithm is simple to implement. However, the formulation of this sampling procedure is not trivial due to the infinite extent of the trajectory space, both in the transition time continuum and the number of transitions.

1.1 Previous Work

Sampling from dynamic systems is not new. However, most prior work has been in the area of discrete-time systems. Continuous-time systems pose different problems. As we note below, any evidence containing a record of the change in a variable has a zero probability under the model. Therefore rejection sampling and straightforward likelihood weighting are not generally viable methods.

Ng, Pfeffer, & Dearden (2005) developed a continuous-time particle filtering algorithm. It only handled point evidence on binary and ternary discrete variables using rejection sampling and focused primarily on the incorporation of evidence from the continuous state part of the system. By contrast our algorithm does not incorporate real-valued state information, but it allows any evidence set and performs general inference (not just filtering). Our algorithm can be adapted to a population-based filter (a particle filter).

2 Continuous Time Bayesian Networks

Continuous time Bayesian networks (Nodelman, Shelton, & Koller 2002) are based on the framework of continuous time, finite state, homogeneous Markov processes. Let X be a continuous time, finite state, homogeneous Markov process with n states $\{x_1, \ldots, x_n\}$. The behavior of X is described by the initial distribution P_X^0 and the intensity matrix

$$\mathbf{Q}_X = \begin{bmatrix} -q_{x_1} & q_{x_1x_2} & \cdots & q_{x_1x_n} \\ q_{x_2x_1} & -q_{x_2} & \cdots & q_{x_2x_n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{x_nx_1} & q_{x_nx_2} & \cdots & -q_{x_n} \end{bmatrix},$$

where $q_{x_ix_j}$ is the intensity with which X transitions from x_i to x_j and $q_{x_i} = \sum_{j \neq i} q_{x_ix_j}$. The intensity matrix $\mathbf{Q}_X$ is time invariant. Given $\mathbf{Q}_X$, the amount of time X stays at x_i follows an exponential distribution with parameter q_{x_i}. That is, the probability density function of X remaining at x_i is $f(q_{x_i}, t) = q_i \exp(-q_{x_i} t)$. The probability X transitions from state x_i to x_j is $\theta_{x_ix_j} = q_{x_ix_j}/q_{x_i}$. A conditional intensity matrix(*CIM*) $\mathbf{Q}_{X|\mathbf{U}}$ is defined as a set of intensity matrices $\mathbf{Q}_{X|\mathbf{u}}$, one for each instantiation $\mathbf{u}$ of the variable set $\mathbf{U}$. The evolution of X depends instantaneously on the values of the variables in $\mathbf{U}$.

A *continuous time Bayesian network* $\mathcal{N}$ over $\boldsymbol{X}$ consists of two components: an *initial distribution* $P^0_{\boldsymbol{X}}$, specified as a Bayesian network $\mathcal{B}$ over $\boldsymbol{X}$, and a *continuous transition model*, specified using a directed (possibly cyclic) graph $\mathcal{G}$ whose nodes are $X \in \boldsymbol{X}$. Let $\mathbf{U}_X$ denote the parents of X in $\mathcal{G}$. Each variable $X \in \boldsymbol{X}$ is associated with a conditional intensity matrix, $\mathbf{Q}_{X|\mathbf{U}_X}$.

2.1 Likelihood and Sufficient Statistics

A CTBN defines a probability density over trajectories σ of a set of variables $\boldsymbol{X}$. Therefore, one way to describe the distribution of a CTBN is to use the sufficient statistics of σ (Nodelman, Shelton, & Koller 2003). Let $T[x|\mathbf{u}]$ be the amount of time $X = x$ while $\mathbf{U}_X = \mathbf{u}$, and $M[x, x'|\mathbf{u}]$ be the number of transitions from x to x' while $\mathbf{U}_X = \mathbf{u}$. If we let $M[x|\mathbf{u}] = \sum_{x'} M[x, x'|\mathbf{u}]$, the probability density of trajectory σ (omitting the starting distribution) is

$$P_{\mathcal{N}}(\sigma) = \prod_{X \in \boldsymbol{X}} L_X(T[X|\mathbf{U}], M[X|\mathbf{U}])$$

where $L_X(T[X|\mathbf{U}], M[X|\mathbf{U}]) =$

$$\prod_{\mathbf{u}} \prod_{x} \left(q_{x|\mathbf{u}}^{M[x|\mathbf{u}]} \exp(-q_{x|\mathbf{u}} T[x|\mathbf{u}]) \prod_{x' \neq x} \theta_{xx'|\mathbf{u}}^{M[x,x'|\mathbf{u}]} \right) \quad (1)$$

is the local likelihood for variable X. The likelihood also decomposes by time. That is, the likelihood of a trajectory on $[0, T)$ is equal to the likelihood based only on sufficient statistics from time 0 to time t multiplied by the likelihood based only on sufficient statistics from time t to time T.

2.2 Evidence and Queries

Given a CTBN model, we would like to use it to answer queries conditioned on some observations. Usually there are two types of observations: point evidence and continuous evidence. Point evidence represents the observation of the value of some variables at a particular time. Continuous evidence provides the behavior of some variables throughout an interval $[t_1, t_2)$. For instance, $x = 1$ during the interval $[2, 3.5)$, or $x = 1$ from $t = 2$ to $t = 3$ and then x transitions to $x = 0$ at $t = 3$ and stays in that state until $t = 5$.

Queries can ask about the marginal distribution of some variables at a particular time, such as the distribution of x and y at $t = 2$, or questions about the timing of a transition, such as the distribution over the time that y transitions from $y = 1$ to $y = 2$ for the first time in the interval $[1, 4)$. In learning (especially expectation-maximization), we might query the expected sufficient statistics of a CTBN, which includes the total amount of time that a variable spends on a state, and the total number of times that a variable transitions from one state to another state under certain conditions. For example, we might want to know the total amount of time that $x = 0$ throughout the entire interval, or the number of times that x transitions from 1 to 2 during the time interval $[2, 3)$ when $y = 0$. In this paper, we will concentrate on answering queries given the continuous evidence, but our method can be trivially extended to point evidence.

2.3 Exact Inference in CTBN

A CTBN can be viewed as a homogeneous Markov process with a large joint intensity matrix amalgamated from the CIMs of the CTBN. Exact inference in a CTBN can be performed by generating a single joint intensity matrix over the entire state space of the CTBN and running the forward-backward algorithm on the joint intensity matrix of the homogeneous Markov process. We review this method here, but a more complete treatment can be found in Nodelman, Shelton, & Koller (2002).

Assume that we have a partially observed a trajectory σ of a CTBN $\mathcal{N}$ from 0 to T. We can divide the evidence σ into N intervals $[t_i, t_{i+1})$ ($i = 0, \ldots, N-1$) according to the observed transition times. That is, each interval contains a constant observation of the CTBN and t_i is the time that a variable begins to be observed, stops being observed, or is observed to transition. We set $t_0 = 0$ and $t_N = T$.

To perform exact inference, we first generate the intensity matrix $\mathbf{Q}$ for the joint homogeneous Markov process and incorporate the evidence into $\mathbf{Q}$. If each variable X_i in the CTBN $\mathcal{N}$ has n_i states, the number of states of the joint Markov process is $n = \prod n_i$ and $\mathbf{Q}$ is an $n \times n$ matrix. The value of the off-diagonal element q_{ij} in $\mathbf{Q}$ for which only one variable value is different between states i and j is the corresponding intensity in the CIM of that variable. All the other off-diagonal elements are zeros since two variables can not transition at the same time in a CTBN. The diagonal elements are computed to make each row sum to zero.

To incorporate the evidence, we reduce the joint intensity matrix $\mathbf{Q}$ to $\mathbf{Q}_i$ for each interval $[t_i, t_{i+1})$ by zeroing out the rows and columns of $\mathbf{Q}$ which represent states that are inconsistent with the evidence. Let $\mathbf{Q}_{i,j}$ be matrix $\mathbf{Q}$ with all element zeroed out except the off-diagonal elements that represent the intensities transitioning from non-zero rows in $\mathbf{Q}_i$ to non-zero columns in $\mathbf{Q}_j$.

$\exp(\mathbf{Q}_i(t_{i+1} - t_i))$ represents the transition matrix for interval $[t_i, t_{i+1})$ and $\mathbf{Q}_{i,i+1}$ denotes the transition probability between two consecutive intervals at time t_{i+1}. We can use the forward-backward algorithm for Markov process to answer queries.

We define the forward and backward probability vectors $\boldsymbol{\alpha}_t$ and $\boldsymbol{\beta}_t$ as

$$\begin{aligned} \boldsymbol{\alpha}_t &= p(X_t, \sigma_{[0,t]}) \\ \boldsymbol{\beta}_t &= p(\sigma_{[t,T)} | X_t) \ . \end{aligned}$$

Let $\boldsymbol{\alpha}_0$ be the initial distribution P^0_X over the state and $\boldsymbol{\beta}_T$ be a vector of ones. The forward and backward distribution vector for each interval can be calculated recursively:

$$\begin{aligned} \boldsymbol{\alpha}_{t_{i+1}} &= \boldsymbol{\alpha}_{t_i} \exp(\mathbf{Q}_i(t_{i+1} - t_i)) \mathbf{Q}_{i,i+1} \\ \boldsymbol{\beta}_{t_i} &= \mathbf{Q}_{i-1,i} \exp(\mathbf{Q}_i(t_{i+1} - t_i)) \boldsymbol{\beta}_{t_{i+1}} \ . \end{aligned}$$

The distribution over the state of the CTBN at time $t \in [t_i, t_{i+1})$ given the evidence $\sigma_{[0,T)}$ can be computed as

$$P(X_t = i | \sigma_{[0,T)}) = \frac{1}{Z} \boldsymbol{\alpha}_{t_i} \exp(\mathbf{Q}_i(t - t_i)) \boldsymbol{\Delta}_{i,i} \exp(\mathbf{Q}_i(t_{i+1} - t)) \boldsymbol{\beta}_{t_{i+1}}$$

where $\boldsymbol{\Delta}_{i,j}$ is a $n \times n$ matrix of zeros with one in position i, j and Z is the normalization constant. Other queries can be similarly computed.

```
Procedure CTBN-Sample($t_{end}$)
  1. $t \leftarrow 0$, $\sigma \leftarrow \emptyset$
  2. For each variable $X \in \boldsymbol{X}$
          Choose state $x(0)$ according to $\boldsymbol{\theta}^{\mathcal{B}}_{X|\mathbf{pa}_{\mathcal{B}}(X)}$.
  Loop:
  3. For each variable $X$ such that $Time(X)$ is undefined:
       Choose $\Delta t$ for next $X$ transition from an exponential
          with parameter $q_{x(t)|\mathbf{u}_X(t)}$.
       Define $Time(X) \leftarrow t + \Delta t$
  4. Let $X = \arg\min_{X \in \boldsymbol{X}} [Time(X)]$
  5. If $Time(X) \geq t_{end}$ return $\sigma$
  6. Update $t \leftarrow Time(X)$
  7. Choose $x(t)$, the next value of $X$, from the multinomial
  with parameters $\boldsymbol{\theta}_{x(t)|\mathbf{u}_X(t)}$.
       Add $\langle X \leftarrow x(t), t \rangle$ to $\sigma$.
       Undefine $Time(X)$ and $Time(Y)$ for all variables $Y$
       for which $X \in \mathbf{U}_Y$.
```

Figure 1: Forward sampling semantics for a CTBN

3 Sampling-based Inference

As we described in the previous section, exact inference in a CTBN can be performed by generating a single joint intensity matrix over the entire state space. As the number of states is exponential in the number of the nodes in the network, this approach is infeasible when the network size is large. In this section we describe an algorithm for approximate CTBN inference based on sampling.

3.1 Forward Sampling

Queries that contain no evidence can be answered by randomly sampling many trajectories and looking at the fraction that match the query. More formally, if we have a CTBN $\mathcal{N}$ we generate a set of particles $\mathcal{D} = \{\sigma[1], \ldots, \sigma[M]\}$ where each particle is a sampled trajectory. With $\mathcal{D}$ we can estimate the expectation of any function by computing

$$\hat{\mathbf{E}}_{\mathcal{N}}[f] = \frac{1}{M} \sum_{m=1}^{M} f(\sigma[m]) \; . \tag{2}$$

For example, if we let $f = \mathbf{1}\{x(5) = x_1\}$ then we could use the above formula to estimate $P_{\mathcal{N}}(x(5) = x_1)$. Or the function $f(\sigma)$ might count the total number of times that X transitions from x_1 to x_2 while its parent U has value u_1, allowing us to estimate the expected sufficient statistic $M[x_1, x_2|u_1]$. The algorithm for sampling a trajectory in CTBN is shown in Figure 1. For each variable $X \in \boldsymbol{X}$, it maintains $x(t)$ — the state of X at time t — and $Time(X)$ — the next potential transition time for X. The algorithm adds transitions one at a time, advancing t to the next earliest variable transition. When a variable X (or one of its parents) undergoes a transition, $Time(X)$ is resampled from the new exponential waiting time distribution. We will use $\mathbf{u}_X(t)$ to represent the instantiation to parents of X at time t.

If we want to obtain a conditional probability of a query given evidence, the situation is more complicated. We might try to use *rejection sampling*: forward sample to generate possible trajectories, and then simply reject the ones that are inconsistent with our evidence. The remaining trajectories are sampled from the posterior distribution given the evidence, and can be used to estimate probabilities as in Equation 2. However, this approach is entirely impractical in our setting, as in any setting involving an observation of a continuous quantity — in our case, time. In particular, suppose we observe that X transitions from x_1 to x_2 at time t. The probability of sampling a trajectory in which that transition occurs at precisely that time is zero. Thus, if we have evidence about transitions, with probability 1, none of our sampled trajectories will be relevant.

3.2 Importance Sampling

A more practical approach to sampling in the presence of evidence is *importance sampling*. In importance sampling, we generate samples from a proposal distribution P' which guarantees that our sampled trajectories will conform to our evidence $\mathbf{e}$. We must weight our samples to correct for the fact that we are drawing them from P' instead of the target distribution $P_{\mathcal{N}}$ defined by the CTBN. In particular, if σ is a sample from P' we set its weight to be

$$w(\sigma) = \frac{P_{\mathcal{N}}(\sigma, \mathbf{e})}{P'(\sigma)} \; . \tag{3}$$

In normalized importance sampling, we draw a set of samples $\mathcal{D} = \{\sigma[1], \ldots, \sigma[M]\}$ i.i.d. from the proposal distribution, and estimate the conditional expectation of a function f given evidence $\mathbf{e}$ as

$$\hat{\mathbf{E}}_{\mathcal{N}}[f \mid \mathbf{e}] = \frac{1}{W} \sum_{m=1}^{M} f(\sigma[m]) w(\sigma[m]) \tag{4}$$

where W is the sum of the weights.

This estimator is consistent if the support of P' is a superset of the support of $P_{\mathcal{N}}$. In general, $\hat{\mathbf{E}}_{\mathcal{N}}$ is biased and the bias deceases as $O(M^{-1})$. The variance of the estimator also decreases as $O(M^{-1})$. For more information on this and related sampling estimates, see Hesterberg (1995).

For our algorithm, we base the proposal distribution on the forward sampling algorithm. As we are sampling a trajectory, we occasionally depart from the regular forward sampling algorithm and "force" the behavior of one or more variables to ensure consistency with the evidence.

3.3 Simple Evidence

The simplest query involves evidence over some subset of variables $\boldsymbol{V} \subset \boldsymbol{X}$ for the total length the trajectory. We force only the behavior of the variables $\boldsymbol{V}$ and there are no choices about how to do that. In particular, we use the following proposal distribution: forward sample the behavior of variables $X \in (\boldsymbol{X} \setminus \boldsymbol{V})$ inserting the known transitions at known times for variables in $\boldsymbol{V}$ as determined by the evidence. As there were no choices in our forcing, the likelihood of drawing σ from the proposal distribution is just the likelihood contribution of forward sampling the behavior of the variables $X \in (\boldsymbol{X} \setminus \boldsymbol{V})$, in the context of the total behavior of the system. But what is the likelihood contribution?

To be more precise, let $x[t_1 : t_2]$ be the behavior of variable X on the interval $[t_1, t_2)$; this behavior can be summarized by the sufficient statistics over X on the interval. Let $\tilde{L}_X(x[t_1 : t_2])$ be a partial likelihood contribution function, computed by plugging the sufficient statistics of $x[t_1 : t_2]$ into Equation 1. The partial contribution function can be defined over a collection of intervals $\mathcal{I}$ by setting $\tilde{L}_{\mathcal{N}}(\mathcal{I}) = \prod_{x[t_1 : t_2] \in \mathcal{I}} \tilde{L}_X(x[t_1 : t_2])$. We can partition σ into two pieces. Let σ_e be the collection for all variables $X \in \boldsymbol{X}$ of intervals $x[t_1 : t_2]$ where the behavior of X is set by the evidence. Let σ_s be the complement of σ_e containing the collection of intervals of unobserved behavior for all variables. Returning to our simple evidence above, the likelihood contribution is just $\tilde{L}_{\mathcal{N}}(\sigma_s)$. To compute the proper weight $w(\sigma)$ we substitute in Equation 3, cancel terms and are left with $\tilde{L}_{\mathcal{N}}(\sigma_e)$. Since σ_e contains all and only the evidence, this algorithm exactly corresponds to *likelihood weighting* in Bayesian networks (Shachter & Peot 1989; Fung & Chang 1989). Intuitively, this makes sense because we can account for all the evidence by simply assigning the observed trajectories to the observed variables.

3.4 General Evidence

Now, consider a general evidence pattern $\mathbf{e}$. How can we force our trajectory to be consistent with $\mathbf{e}$? Suppose there is a set of variables which has evidence beginning at t_e. We can not simply force a transition at time t_e to make the variables consistent with the evidence $\mathbf{e}$: if the set contains more than one variable, the sample would have multiple simultaneous transitions, an event whose likelihood is zero.

Instead, we look ahead for each variable we sample. If the current state of the variable does not agree with the upcoming evidence, we force the next sampled transition time to fall before the time of the conflicting evidence by sampling from a truncated exponential distribution, instead of the full exponential distribution. In particular, if we are currently at time t and there is conflicting evidence for X at time t_e, we sample from an exponential distribution with the same q value as the normal sampling procedure, but where the sample for Δt (the time to the next transition) is required to be less that $t_e - t$. The probability of sampling Δt from this truncated exponential is $\frac{q \exp(-q\Delta t)}{1-\exp(-q(t_e - t))}$ where q is the relevant intensity for the current state of X (the diagonal element of $\mathbf{Q}_{X|\mathbf{U}_X}$ corresponding to the current state of X).

The probability of sampling σ from the proposal distribution is the partial contribution of the regularly forward sampled behavior on the unforced part of the trajectory $\tilde{L}_{\mathcal{N}}(\sigma_s)$ — which will cancel the corresponding term in the numerator of Equation 3 — times an extra factor for the probability of choosing the forcing transitions that we selected. For any variable x whose value is given in the evidence during the interval $[t, t + \Delta t)$, as we discussed above, the contribution to the trajectory weight is just $\tilde{L}_{\mathcal{N}}(x[t_1 : t_2])$. Note that in our case, this likelihood has a simple form. Only one variable can change during an iteration of the algorithm and it must change at the end of the time interval.

For variables that are forced to transition due to upcoming evidence, things are slightly more complicated. Each time we add a new transition to the trajectory, we advance time from t to $t + \Delta t$. For each variable x whose "next time" was sampled from a truncated exponential distribution, we must update the weight of trajectory to reflect the likelihood ratio for $x[t : t + \Delta t]$. Each such variable can be considered separately as their times are sampled independently.

If the variable is part of the transition, the weight must be multiplied by the probability of sampling the transition in $P_{\mathcal{N}}$ divided by the the probability in the sampling algorithm. The former is an exponential distribution and the latter is the same exponential distribution, truncated to be less than $t_e - t$. The ratio of these two probabilities is $1 - \exp(-q(t_e - t))$, where q is the relevant intensity.

Otherwise, the next time for the variable was sampled from a truncated exponential but was longer than Δt. In this case, the ratio of the probabilities of a sample being greater than Δt is $\frac{1-\exp(-q(t_e - t))}{1-\exp(-q(t_e - t - \Delta t))}$. Note that when Δt is small (relative to $t_e - t$, the time to the next evidence point for this variable), the ratio is almost 1. So, while the trajectory's weight is multiplied by this ratio for every transition for every variable that does not agree with the evidence, it does not overly reduce the weight of the entire trajectory.

The algorithm for CTBN importance sampling is shown in Figure 2. To more easily describe the evidence, we define a few helper functions:

$\mathbf{e}_X^{\mathit{val}}(t)$ is the value of X at time t according to the evidence, or undefined if X has no evidence at t.

$\mathbf{e}_X^{\mathit{time}}(t)$ is the first time after t when $\mathbf{e}_X^{\mathit{val}}(t)$ is defined.

$\mathbf{e}_X^{\mathit{end}}(t)$ is the first time after or equal to t when $\mathbf{e}_X^{\mathit{val}}(t)$ changes value or becomes undefined.

Note that $\mathbf{e}_X^{\mathit{end}}(t) = t$ when there is point evidence at t, when t is the end of an interval of evidence, and when there is a transition in the evidence at time t.

The line numbers follow those given in the forward sampling algorithm with new or changed lines marked with an asterisk. *Time*(X) might be set to the end of an interval of evidence which is not a transition time but simply a time when we need to resample a next potential transition. This means that we will not update σ with a new transition every time through the loop. The algorithm differs from the forward sampling procedure as follows. Step 2 now accounts for evidence at the beginning of the trajectory (using standard likelihood weighting for Bayesian networks). In step 3, we draw Δt from the truncated exponential if the current value disagrees with upcoming evidence. If the current evidence includes this variable, Δt is set to the duration of such evidence. Step 5 updates the weights using the procedure *Update-Weight*. Finally, step 7 now deals with variables that are just leaving the evidence set.

3.5 Predictive Lookahead

The algorithm in Figure 2 draws the next state for a variable from the same distribution as the forward sampling algorithm. This may cause a variable to transition several times in a short interval before evidence as the variable "searches" to find a way to transition into the evidence. Thus, we may generate many unlikely samples, making the algorithm inefficient. We can help mitigate this problem by trying to force the variable into a state that will lead to the evidence.

When sampling the next state for variable X at time t, instead of sampling from the multinomial according to $\boldsymbol{\theta}_{x(t)|\mathbf{u}_X(t)}$, we would like to sample from the distribution of

```
Procedure CTBN-Importance-Sample($t_{end}, \mathbf{e}$)
  1. $t \leftarrow 0, \sigma \leftarrow \emptyset, w \leftarrow 1$                                   *
  2. For each variable $X \in \boldsymbol{X}$
       If $\mathbf{e}_X^{val}(0)$ defined, set $x(0) \leftarrow \mathbf{e}_X^{val}(0)$,
       and then set $w \leftarrow w \cdot \theta^{\mathcal{B}}_{x(0)|\mathbf{pa}_{\mathcal{B}}(0)}$                  *
       Else choose state $x(0)$ according to $\boldsymbol{\theta}^{\mathcal{B}}_{X|\mathbf{pa}_{\mathcal{B}}(X)}$
  Loop:
  3. For each $X \in \boldsymbol{X}$ such that $Time(X)$ is undefined:
       If $\mathbf{e}_X^{val}(t)$ is defined, set $\Delta t \leftarrow \mathbf{e}_X^{end}(t) - t$            *
       Elseif $\mathbf{e}_X^{val}(t_e)$ is defined where
       $t_e = \mathbf{e}_X^{time}(t), x(t) \neq \mathbf{e}_X^{val}(t_e)$,
         choose $\Delta t$ from an exponential distribution with
         parameter $q_{x(t)|\mathbf{u}_X(t)}$ given $\Delta t < (t_e - t)$.             *
       Else choose $\Delta t$ from an exponential w/ param. $q_{x(t)|\mathbf{u}_X(t)}$
       Define $Time(X) \leftarrow t + \Delta t$
  4. Let $X = \arg\min_{X \in \boldsymbol{X}} [Time(X)]$
  5. If $Time(X) \geq t_{end}$
         $w \leftarrow Update\text{-}Weight(X, w, t, t_{end})$                          *
         return $(\sigma, w)$
     Else                                                                  *
         $w \leftarrow Update\text{-}Weight(X, w, t, Time(X))$                        *
  6. Update $t \leftarrow Time(X)$
  7. If $\mathbf{e}_X^{end}(t) \neq t$ or $\mathbf{e}_X^{val}(t)$ is defined                         *
       If $\mathbf{e}_X^{val}(t)$ is defined, set $x(t) \leftarrow \mathbf{e}_X^{val}(t)$                     *
       Else choose $x(t)$, the next value of $X$, from a
         multinomial with parameter $\boldsymbol{\theta}_{x(t)|\mathbf{u}_X(t)}$
       Add $\langle X \leftarrow x(t), t \rangle$ to $\sigma$.
       Undefine $Time(X)$ and $Time(Y)$ for all variables $Y$
       for which $X \in \mathbf{U}_Y$
     Else                                                                  *
       Undefine $Time(X)$.                                                 *

  Procedure Update-Weight($Y, w, t_1, t_2$)
  1. For each $X \in \boldsymbol{X}$ such that $\mathbf{e}_X^{val}(t)$ is defined for $t \in [t_1, t_2)$:
       $w \leftarrow w \cdot \tilde{L}_X(x[t_1 : t_2])$
  2. For each $X \in \boldsymbol{X}$ such that $\mathbf{e}_X^{val}(t_e)$ is defined,
       where $t_e = \mathbf{e}_X^{time}(t_1)$, and $x(t_1) \neq \mathbf{e}_X^{val}(t_e)$:
       If $X = Y, w \leftarrow w \cdot (1 - \exp(-q_{x(t_1)|\mathbf{u}_X(t_1)}(t_e - t_1)))$
       Else $w \leftarrow w \cdot \frac{1-\exp(-q_{x(t_1)|\mathbf{u}_X(t_1)}(t_e-t_1))}{1-\exp(-q_{x(t_1)|\mathbf{u}_X(t_1)}(t_e-t_2))}$
  3. return $w$
```

Figure 2: Importance sampling for CTBNs

the next state conditioned on the upcoming evidence. Suppose X is in state x_i at time t, and the next evidence for X is state x_k at t_e. Assuming the parents of X do not change before t_e and ignoring evidence over the children of X, the distribution of the state of X at t given only the evidence is $b = \exp(-\mathbf{Q}_X(t_e - t))\boldsymbol{\delta_k}$, where $\boldsymbol{\delta_k}$ is the vector of zeros with a single one at element k. The probability that X shifts from $x_i = x(t)$ to x_j is $\beta_{i,j} = b_j q_{x_i x_j} / \sum_{k \neq i} b_k q_{x_i x_k}$. We can therefore select our new state according to the distribution of β_i and, assuming state x_j is selected, multiply the weight by $\frac{\theta_{x_i x_j | \mathbf{u}_X(t)}}{\beta_{i,j}}$ to account for the difference between the target and sampling distributions.

3.6 Particle Filtering

The algorithm in Figure 2 allows us to generate a single trajectory and its weight, given the evidence. To apply this algorithm to the task of online inference in a dynamic system, we can generate multiple trajectories in parallel, advancing time forward as evidence is obtained.

```
Procedure CTBN-Particle-Filtering($\{X_0^i, w_0^i\}_{i=1\ldots N}, t_{end}, \mathbf{e}$)
  1. $k \leftarrow 0, W_t \leftarrow 1, N_r \leftarrow N$
  2. For $i \leftarrow 1$ to $N$: $Pa_0^i \leftarrow i$, $w^i \leftarrow 1/N$
  Loop:
  3. For each i such that $t_k^i < t_{end}$ :
       $(X_{k+1}^i, t_{k+1}^i, w_{k+1}^i) \leftarrow$
         Sample-Segment($X_k^{Pa_k^i}, t_k^{Pa_k^i}, w^i, t_{end}, \mathbf{e}$)
       If $t_{k+1}^i \geq t_{end}$
         $N_{remain} \leftarrow N_r - 1$,
         $W_t \leftarrow W_t - w_{k+1}^i$
  4. $k \leftarrow k + 1$
  5. If $N_r = 0$
       return $\{X_{m_i}^i, t_{m_i}^i, w_{m_i}^i, Pa_{m_i}^i\}_{i=1\ldots N, m_i=1\ldots n_i}$,
       where $n_i$ is the number of transitions of the $i^{th}$ particle
  6. Calculate $\widehat{N_{eff}}$ of all incomplete particles
  7. If $\widehat{N_{eff}} < N_{thr}$
       Sample $Pa_k^i$ according to $w_k^i$
       $w^i \leftarrow W_t \times 1/N_r$
     Else
       $w^i \leftarrow w_k^i$, $Pa_k^i \leftarrow Pa_{k-1}^i$
```

Figure 3: Particle Filtering for CTBNs

The resulting algorithm is an instance of sequential importance sampling, and therefore suffers from its characteristic flaw: As the trajectory length increases, the distribution of the importance weights gets increasingly skewed, with most importance weights converging to zero exponentially quickly. Thus, the number of "relevant" samples gets increasingly small, and the estimates provided by the set of samples quickly become meaningless. A family of methods, commonly known as sequential Monte Carlo or particle filtering (Doucet, de Freitas, & Gordon 2001), have been proposed in the setting of discrete-time processes to address this flaw. At a high level, these methods re-apportion our samples to focus more efforts on more relevant samples — those with higher weight.

The application of this idea to our setting introduces some subtleties because different samples are not generally synchronized. We could pick a time t and run the algorithm in Figure 2 with $t_{end} = t$ so that samples are synchronized at t. We would re-apportion the weights and continue each trajectory from its state at t, first setting $Time(X)$ to be undefined for all X. However, choosing the proper synchronization time t is a non-trivial problem which may depend on the evidence and the speed the system evolves.

Instead of synchronizing all the particles by the time, we can align particles by the number of transitions. If we let t_i be the i^{th} transition time and X_i be the value of X from t_{i-1} to t_i, the following recursion holds.

$$P(X_{1:n}, t_{1:n}, e_{[0:t_n)}) = P(X_{1:n-1}, t_{1:n-1}, e_{[0:t_{n-1})})$$
$$\times P(X_n|X_{n-1})P(X_{[t_{n-1},t_n)}, e_{[t_{n-1},t_n)}|X_{n-1}, e_{t_{n-1}})$$

Thus, to sample multiple trajectories in parallel, we apply the CTBN importance sampling algorithm to each trajectory until a transition occurs. To avoid the degeneracy of the weights, we resample the particles when the estimated effective sample size $\widehat{N_{eff}} = \frac{1}{\sum_i (w_k^i)^2}$ is below a threshold N_{thr}. This procedure is similar to the regular particle filtering algorithm except that all particles are not synchronized by time but the number of transitions. To answer queries in the time interval $[0, T)$, we propagate the particles until all of their last transitions are greater than T.

Figure 3 shows the algorithm for generating N trajectories from 0 to T in a CTBN. It assumes that the initial values and the weights have already been sampled. The procedure *Sample-Segment* loops from line 3 to 7 in Figure 2 until a transition occurs, returns the transition time and variables value, and updates the corresponding weight for that segment. Note that we are approximating the distribution $P(X_{1:n}, t_{1:n}, e_{[0:t_n)})$ for all possible n. Therefore, we only propagate and re-apportion weights for particles that have not yet reached time T. Particles that have been sampled past T are left untouched.

3.7 Particle Smoothing

Although the resampling step in the particle filtering algorithm reduces the skew of the weights, it leads to another problem: the diversity of the trajectories is also reduced since particles with higher weights are likely to be duplicated multiple times in the resampling step. Many trajectories share the same ancestor after the filtering procedure. A Monte Carlo smoothing algorithm using backward simulation addresses this problem (Godsill, Doucet, & West 2004).

The smoothing algorithm generates trajectories using N weighted particles $\{x_t^i, w_t^i\}$ using the particle filtering algorithm. It starts with the particles at time T, moves backward one step each time and samples a particle according to the product of its weight and the probability of it transitioning to the previously sampled particle. Specifically, in the first step, it samples $\widetilde{x}_T$ from particles x_T^i at time T with probability w_T^i. In the backward smoothing steps it samples $\widetilde{x}_t$ according to $w_{t|t+1}^i = w_t^i f(\widetilde{x}_{t+1}|x_t^i)$, where $f(\widetilde{x}_{t+1}|x_t^i)$ is the probability that the particle transitions from state x_t^i to $\widetilde{x}_{t+1}$. The resulting trajectories are an approximation of $P(x_{1:T}|y_{1:T})$ where $y_{1:T}$ is the observation.

This idea can be used in our setting with some modification. Given the filtered particles $\{X_{m_i}^i, t_{m_i}^i, w_{m_i}^i\}$,we need to sample both variable values and transition time at each step when we move backward. There are two main differences from the algorithm in Godsill, Doucet, & West (2004): There are fewer than N particles that can be used at the beginning steps of the backward smoothing since the trajectories do not have exactly the same number of transitions, and not all particles at step n can be considered as candidates to move backward. A particle $\{X_n^i, t_n^i, w_n^i\}$ is a valid candidate as the predecessor for $\{\widetilde{X}_{n+1}, \widetilde{t}_{n+1}\}$ only if (1) $t_n^i < \widetilde{t}_{n+1}$, (2) the values of X_n^i and $\widetilde{X}_{n+1}$ differ in only one variable (thus a single transition is possible), and (3) $\mathbf{e}_{(t_n^i, \widetilde{t}_{n+1})}$ contains no transitions.

Figure 4 shows the smoothing algorithm which generates a trajectory from the filtering particles. We apply the algorithm N times to sample N trajectories. These equally weighted trajectories can be used to approximate the smoothing distribution $P(X_{[0,T)}|\mathbf{e})$. Generating one trajectory with this smoothing process requires considering all the particles at each step. The running time of sampling N trajectories using particle smoothing is N times of that of particle filtering.

```
Procedure CTBN-Particle-Smoothing($\{X_{m_i}^i, t_{m_i}^i, w_{m_i}^i\}, t_{end}, \mathbf{e}$)
        $i = 1 \ldots N, m_i = 1 \ldots M_i$
    1. $\sigma \leftarrow \emptyset$
    2. Choose $k$ with probability $w_i^{M_i}$
    3. set $Y = X_{M_k^k}$, $s \leftarrow M_k$, $t \leftarrow t_s^k$
    Loop:
    4. $\sigma_{[t_{s-1}, s)} \leftarrow Y$
    5. If $\sigma$ is complete
          return $\sigma$
    6. For $j \leftarrow 1$ to $N$
          $w'_j \leftarrow$ Check-Weight($Y, t, X_{s-1}^j, t_{s-1}^j, w_{s-1}^j$)
    7. Choose $i$ with probability $w'_i$
    8. $S \leftarrow S - 1$, $Y \leftarrow X_i^s$, $t \leftarrow t_s^j$

    Procedure Check-Weight($X, t, X_s, t_s, w_s$)
    1. If $t \leq t_s$ or $\mathbf{e}_{(ts,t)}$ contains a transition, or
          the value of $X$ and $X_s$ do not differ by only one variable
          return 0
    2. $\sigma_{[t_s, t)} \leftarrow X_s$, $\sigma(t) \leftarrow X$
    3. $w \leftarrow w_s \cdot \tilde{L}_X(\sigma_{[t_s, t_2]})$
    4. return $w$
```

Figure 4: Particle Smoothing for CTBNs

4 Experimental Results

To test our algorithms, we used two CTBN networks: the drug effect network in Nodelman, Shelton, & Koller (2002) and a chain-structured network. Both are at the limit for the exact inference algorithm.

We first tested the importance sampling algorithm and the predictive lookahead modification using the drug effect network. It has 8 (mostly binary) variables modeling the effect of a pain-relief medicine. At $t = 0$ the person is not hungry, is not eating, has an empty stomach and is not drowsy. He has joint pain due to the falling barometric pressure and takes the drug to alleviate the pain. We set the observed evidence: on $t = [0, 1)$ the stomach is empty, on $t = [0.5, 1.2)$ the barometer is falling, and on $t = [1.5, 2.5)$ he is drowsy. Our query is the total amount of time that the person has no joint pain on $[0, 2.5)$. (The true value is 0.1093). We ran the two algorithms with sample sizes, M, from 5 to 90000. For each sample size, we ran the algorithms 1000 times. We calculated our query according to Equation 3 and compared the result to the true value calculated using exact inference. We evaluated the two algorithms in two ways: the relative bias $\frac{|\bar{v}_M - v^*|}{v^*}$, where $\bar{v}_M$ is the average query value of the 1000 runs with sample size M, and v^* is the true value; and the relative standard deviation $\frac{\sigma_M}{v^*}$ where σ_M is the standard deviation when sample size is M.

The results are shown in Figure 5. The bias and standard deviation decease at a rate of $O(\frac{1}{\sqrt{M}})$ (shown by the thin

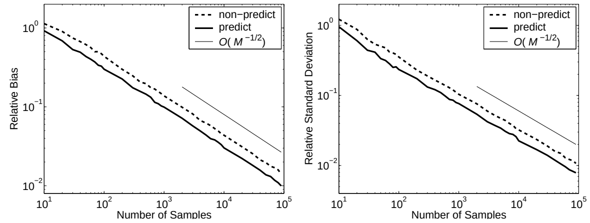
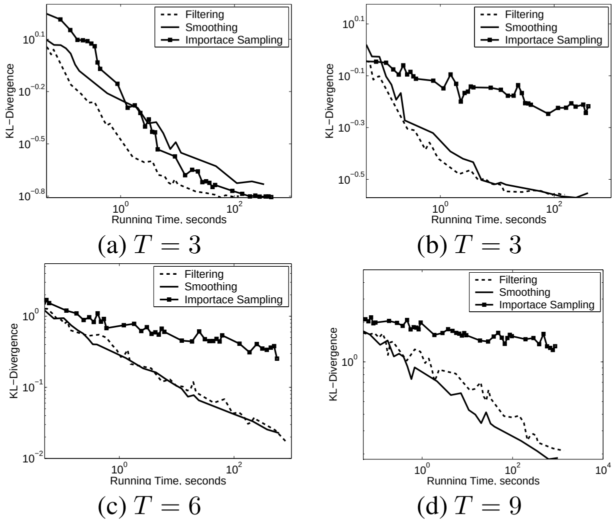


Figure 5: Relative bias and standard deviation of sampling.



(a) $T = 3$

(b) $T = 3$

(c) $T = 6$

(d) $T = 9$

Figure 6: Time-efficiency comparison.

solid lines). The sampling algorithm with prediction outperforms the non-prediction version.

We then used the chain network to evaluate the efficiency of importance sampling, particle filtering, and smoothing algorithms. The chain network contains five nodes $X_0, \ldots, X_4$, where X_i is the parent of X_{i+1} for $i < 4$. Each node has five states, $s_0, \ldots, s_4$. X_0 (usually) cycles in two loops: $s_0 \to s_1 \to s_3 \to s_0$ and $s_0 \to s_2 \to s_4 \to s_0$. All the other nodes stay at their current state if it matches their parent and otherwise transition to their parents' state with high intensity. Each variable starts in state s_0. Note that this is a difficult case with near determinism. We only observe the behavior of X_4 and set our query to be $P(X_2(\frac{T}{2})|\mathbf{e}_{[0,T)})$ where $T = 3, 6, 9$. We recorded the mean running time and KL-divergence between the estimated and true distributions, for each sample size across 300 trials.

Figure 6 shows the efficiency of the three algorithms. In Figure 6(a), we use simple evidence: only part of the behavior of X_4 is observed: on $[1, 1.7)$, $X_4 = s_3$, and on $[2, 2.5)$, $X_4 = s_2$. In Figure 6 (b)-(d), the behavior of X_4 is fully observed during the interval $[0, T)$: we sampled a trajectory from 0 to T and kept only the information about X_4. In all four cases, the particle filtering and smoothing algorithms both outperform the importance sampling algorithm when the sample size is small (small running time). For simple evidence (Figure 6(a)), the importance sampling algorithm achieves comparable performance when the sample size is large. When the evidence is complicated (Figure 6 (b)-(d)), the error of importance sampling is large even we use very large sample sizes. When the trajectory is short, the particle filtering algorithm is slightly better than the particle smoothing algorithm. This is because the filtering algorithm can generate more samples than the smoothing algorithm with the same running time. However, as the trajectory length increases, the particle smoothing algorithm then outperforms the filtering algorithm due to particle diversity problems.

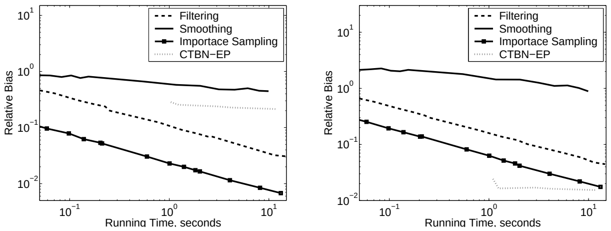
We also compared the three sampling algorithm to the approximate inference algorithm based on expectation propagation in Saria, Nodelman, & Koller (2007). We did not use their adaptive splitting method (for reasons we explain below). Even without the adaptive splitting, their method still differs from that of Nodelman, Koller, & Shelton (2005), in that it allows asynchronous propagation of messages along time.

We used the same evidence on the effective drug network and answered two queries: the total amount of time that the concentration is low and the total amount of time the person has no joint pain. For the EP algorithm, we first tried segmentations that were split at the evidence. We then gradually decreased the time interval of the segments to 0.15. The results of accuracy with respect to running time are shown in Figure 7. The importance sampling algorithm outperforms the EP algorithm in answering query about concentration (a variable in the center of the network) and is only slightly less efficient than the EP algorithm in answering the query about the joint pain (a variable at the edge). Among the sampling based algorithms, the importance sampling algorithm performs the best and the smoothing algorithm is the worst. This is not surprising given that most of the nodes are binary. At each transition time, the sampled trajectory has no choice as to the next state. Therefore, smoothing (or filtering) has less effect as there is no need to intelligently select the next state. However, the extra computation time for resampling and backward simulation makes the filtering and smoothing algorithm less efficient.

As mentioned above, we did not employ the adaptive splitting method of Saria, Nodelman, & Koller (2007). It would not have changed our results much. The left-most points in our EP plots correspond to the minimum number of splits. (They are as fast as possible.) The right-most points of the EP plots correspond to many fine splits, and are about as accurate as possible, and we can see that the accuracy has flattened out. So, while the horizontal widths of the EP curves would have been shortened (by allowing for the better accuracy in less time), the vertical spread would have been approximately the same. In neither plot of Figure 7 would this have made a large difference in the comparisons to our sampling method.

5 Conclusion

The networks used in this paper are at the upper size limit for exact computation. Thus, approximate inference methods are critical for tracking, prediction, and learning in continuous time Bayesian networks for real applications. Our importance sampling, filtering, and smoothing algorithms



(a) Concentration

(b) Joint pain

Figure 7: Comparison to expectation propagation.

are fast, simple to implement and can be used to calculate the expected value of any function of the trajectory, including the expected sufficient statistics necessary for employing expectation maximization for learning with missing data.

6 Acknowledgments

We would like to thank Uri Nodelman for his invaluable comments and suggestions throughout the development of this paper. We would also like to thank Daphne Koller for her helpful reviews of initial versions of the work and Suchi Saria for sharing her CTBN EP code. This work was funded by the DAF Air Force Office of Scientific Research (Young Investigator Award #FA9550-07-1-0076) and Intel Corporation.

References

Doucet, A.; de Freitas, N.; and Gordon, N., eds. 2001. *Sequential Monte Carlo Methods in Practice*. Springer-Verlag Telos.

Fung, R. M., and Chang, K.-C. 1989. Weighing and integrating evidence for stochastic simulation in Bayesian networks. In *UAI*, 209–220.

Godsill, S.; Doucet, A.; and West, M. 2004. Monte carlo smoothing for non-linear time series. *Journal of the American Statistical Association* 99:156–168.

Herbrich, R.; Graepel, T.; and Murphy, B. 2004. Structure from failure. unpublished.

Hesterberg, T. 1995. Weighted average importance sampling and defensive mixture distributions. *Technometrics* 37(2):185–194.

Minka, T. P. 2001. Expectation propagation for approximate Bayesian inference. In *UAI*, 362–369.

Ng, B.; Pfeffer, A.; and Dearden, R. 2005. Continuous time particle filtering. In *IJCAI*.

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

Nodelman, U.; Koller, D.; and Shelton, C. R. 2005. Expectation propagation for continuous time Bayesian networks. In *UAI*, 431–440.

Nodelman, U.; Shelton, C. R.; and Koller, D. 2002. Continuous time Bayesian networks. In *UAI*, 378–387.

Nodelman, U.; Shelton, C. R.; and Koller, D. 2003. Learning continuous time Bayesian networks. In *UAI*, 451–458.

Saria, S.; Nodelman, U.; and Koller, D. 2007. Reasoning at the right time granularity. In *UAI*.

Shachter, R. D., and Peot, M. A. 1989. Simulation approaches to general probabilistic inference on belief networks. In *UAI*, 221–234.