A Hybrid Factored Frontier Algorithm for Dynamic Bayesian Networks with a Biopathways Application

Sucheendra K. Palaniappan[a], S. Akshay[a], Bing Liu[a], Blaise Genest[b], P. S. Thiagarajan[a]

[a] *School of Computing, National University of Singapore.*
[b] *CNRS, IPAL UMI, joint with NUS-I2R-A*STAR, Singapore.*

Abstract

Dynamic Bayesian Networks (DBNs) can serve as succinct probabilistic dynamic models of biochemical networks [1]. To analyze these models, one must compute the probability distribution over system states at a given time point. Doing this exactly is infeasible for large models and hence one must use approximate algorithms. The Factored Frontier algorithm (FF) is one such algorithm [2]. However FF as well as the earlier Boyen-Koller (BK) algorithm [3] can incur large errors.

To address this, we present here a new approximate algorithm called the Hybrid Factored Frontier (HFF) algorithm. At each time slice, in addition to maintaining probability distributions over local states -as FF does- HFF explicitly maintains the probabilities of a small number of global states called spikes. When the number of spikes is 0, we get FF and with all global states as spikes, we get the exact inference algorithm. We show that by increasing the number of spikes one can reduce errors while the additional computational effort required is only quadratic in the number of spikes. We have validated the performance of our algorithm on large DBN models of biopathways. Each pathway has more than 30 species and the corresponding DBN has more than 3000 nodes. Comparisons with the performances of FF and BK show that HFF is a useful and powerful approximation algorithm for analyzing DBN models.

1. Introduction

A system of deterministic Ordinary Differential Equations (ODE) can often be used to describe the dynamics of a biochemical network. Such ODE systems are difficult to analyze and hence a technique to approximate them as Dynamic Bayesian Networks (DBNs) was developed in [1]. This approximation -explained in more detail later- is derived by discretizing both the time and value domains, sampling the assumed set of initial states and using numerical integration to generate a large number of representative trajectories. Then based on the network structure and simple counting, the generated trajectories are compactly stored as a DBN. One can then study the biochemical network by applying Bayesian inferencing techniques to the DBN approximation. This approach scales well and has been used to aid biological studies [4].

In this scheme, the random variables associated with each time point t represent the discretized concentrations of the associated proteins. To perform tasks such as parameter estimation and sensitivity analysis one must repeatedly compute the probability of a random variable assuming a specific value at a time point. Due to conditional dependencies between the variables, this will require an effort exponential in the number of species in the biochemical network. Hence for large networks, exact computation is infeasible and one must resort to approximate methods. There is an efficient approximate inferencing technique called the Factored Frontier algorithm (FF, for short) [2, 5] which was used extensively in [1, 4].

The crucial role played by the FF algorithm led us to consider its error behavior. Surprisingly, we could not find in the literature a rigorous error analysis. Further, we found that in the models we considered -as detailed later- though

Email addresses: suchee@comp.nus.edu.sg (Sucheendra K. Palaniappan), akshay@comp.nus.edu.sg (S. Akshay), lb@nus.edu.sg (Bing Liu), bgenest@irisa.fr (Blaise Genest), thiagu@comp.nus.edu.sg (P. S. Thiagarajan)

FF performed well for many variables, there were a few where it incurred significant errors. This motivated us to construct an improved approximate inferencing algorithm for DBNs called the *Hybrid Factored Frontier* algorithm (HFF, for short) and it is the focus of this paper. HFF is a parameterized version of FF and to bring out its main features, it will be helpful to first see how FF works.

A DBN has a finite set of random variables with each variable having a finite domain of values. The value of a variable at time t only depends on the values of its parents at time $t-1$. The probabilistic dynamics is captured by a Conditional Probability Table (CPT) associated with each variable at each time point (see Figure 1(c) for an example). This table will specify how the value of a variable at t is conditioned by the values of its parent variables at time $t-1$. The global state of the system at time t is a tuple of values with each component denoting the value assumed by the corresponding variable at t. One is interested in the marginal probability, i.e., the probability of a variable X taking value v at time t. To compute this exactly, we need P^t, the joint probability distribution over global states at time t. This can be computed by propagating P^{t-1} through the CPTs . FF maintains and propagates these joint probability distributions approximately. Such approximate distributions are usually called *belief states*.

FF is a simplified and more efficient version of the earlier Boyen-Koller algorithm (BK, for short) [3]. In BK, a belief state is maintained compactly as a product of the probability distributions of independent clusters of variables. This belief state is then propagated *exactly* at each step through the CPTs. Then the new belief state is compacted again into a product of the probability distributions of the clusters. In contrast, the FF algorithm maintains a belief state as a product of the marginal distributions of the individual variables. Instead of computing first the new belief state as done by BK, the FF algorithm computes the new marginal distributions directly via the propagation of the current marginal distributions through the CPTs.

FF is attractive in terms of simplicity and efficiency but unlike BK [3], it lacks a rigorous error analysis. More importantly, as we observe in Section 4, FF can exhibit significant errors. As for BK, its accuracy crucially depends on how one clusters the variables. Further, computing the next belief state exactly is computationally infeasible when the size of clusters is large. Identifying the right set of clusters is a difficult problem and there seem to be no efficient techniques for doing this well. One could avoid the problem of identifying clusters by just using singleton clusters (the so called fully factored BK algorithm). However, as we report in Section 4, this can also leads to significant errors.

The error analysis for BK [3] suggests that the key to minimizing the overall error is to reduce the error incurred in one step. HFF attempts to achieve this by maintaining the current belief state as a hybrid entity; for a small number of global states called *spikes*, their current probabilities are maintained. The probability distribution over the remaining states is represented, as in FF, as a product of the marginal probability distributions. The key insight underlying this idea is that when the error produced by one step of the inferencing algorithm is large for a global state, then actual or the estimated probability of this state must itself be high. If such states are chosen to be the spikes then since the total probability is bounded by 1, the number of spikes at each time point must be small. However, it is infeasible to explicitly identify and exactly compute the probabilities of the spikes. The main technical component of HFF accomplishes this approximately.

A pleasing feature of HFF is that it is a parametrized version of FF with σ, the number of spikes, being the parameter. When $\sigma = 0$, we get FF and when $\sigma = N$ where N is the total number of global states, we get the exact inferencing algorithm. Thus by tuning σ, one can gain control over the error behavior. We have derived the single step error bound for HFF, which then also leads to an error analysis for FF. We show that the worst case one step error of HFF is lower than that of FF. The time complexity of HFF is $O(n \cdot (\sigma^2 + K^{D+1}))$ where n is the number of nodes in the DBN, σ is the number of spikes, K is the maximum number of values that a random variable (associated with each node) can assume and D is the maximum number of parents that a node can have. This compares well with the complexity of FF which is $O(n \cdot K^{D+1})$. Since the complexity of HFF (and FF) is *linear* in n, it scales well in terms of network size. The factor D is determined by the maximum number of reactions that a species takes part in as a product or reactant. For most of the networks we have encountered, D is much smaller than n.

Our experimental results confirm that HFF is a useful and efficient algorithm. We considered four large DBNs. Three of them arise from the EGF-NGF pathway [6] with one model for NGF stimulation, the second for EGF stimulation and the third for EGF-NGF co-stimulation. The fourth DBN captures the behavior of the Epo mediated ERK signaling pathway [7]. Starting from their ODE based models (each of which had more than 30 species) we applied the technique developed in [1] to obtain the DBNs each of which had more than 3000 nodes. In all four cases, we found that the errors suffered by FF and BK (with singleton clusters) were high for marginal distributions of some

biologically significant species. The errors incurred by HFF was always lower and they reduced monotonically when the number of spikes were increased.

1.1. Related work

DBNs are used extensively in domains such as AI, computer vision, signal processing and computational biology [5, 8, 9, 10]. HFF is a generic inferencing algorithm and it can be applied to compute and maintain belief states in these settings as well. As done in HFF, capturing a large probability mass using only a few values has been considered in [11] and [12]. The main idea of [11] is to use stochastic sampling methods to look for cut-sets in the graph structure that have high probability mass. The approach of [12] consists of predictive pruning to remove all but a few high probability nodes. Loosely speaking, these methods select the spikes with methods that differ from HFF's. Further, they are not guaranteed to improve the accuracy in theory as is the case for HFF.

There is a rich vein of work that uses Continuous Time Markov Chains to study the dynamics of biochemical networks [13, 14, 15, 16, 17, 18, 19, 20]. Typically, these studies are population-based in that one maintains the *number* of molecules of each species in the network and tracks their changes as reactions occur one by one. This approach is needed when the number of molecules of the species is too low to support the "smoothness" assumptions made by the ODE based models. As might be expected, the exact inferencing problem for large CTMCs is computationally infeasible. Analysis methods based on Monte Carlo simulations [21, 22, 19, 23] as well as numerically solving the Chemical Master Equation describing a CTMC [18, 24] are being developed. Our overall approach may be viewed as Monte Carlo based but with the crucial additional feature that we generate a pool of representative trajectories just *once*, compactly store these trajectories as a DBN, and query this representation using an approximate inferencing method. The technique based on the Chemical Master Equation [18, 24] represents probabilistic dynamics of a CTMC as a *stochastic* differential equation. One then uses approximate numerical integration methods to compute and propagate belief states from one time point to the next. In all the studies cited above, the CTMCs are presented implicitly but the analysis is carried out on the full state space of the CTMCs and hence it does not seem possible to deal with large biochemical networks (say with more than 30 species) in these approaches. At present it is not clear whether approximate inferencing algorithms such as FF, BK and HFF can be deployed to analyze populations-based models of biochemical networks.

1.2. Plan of the paper

In the next section we introduce DBNs and explain in more detail how they arise in our context as models of biochemical networks. In Section 3, we sketch the FF algorithm and then present our HFF algorithm. This is followed by an error analysis for the two algorithms. The experimental results are presented in Section 4 and we conclude with a discussion in Section 5. Additional technical material and details concerning the case studies can be found in [25].

This is an expanded and improved version of the conference paper to be presented at CMSB'11 [26]. The present paper contains the main proofs and many additional technical details including a complete error analysis of FF and HFF. In the experimental part of the work, we have used here a more sophisticated sampling method -as explained in Section 4 to generate the initial states of the trajectories. Further, we have validated the quality of the DBN approximations in all our case studies.

2. DBN models of biopathways

We start with an introduction to DBNs and related notions. We fix an ordered set of n random variables $\{X_1, \ldots, X_n\}$ and let i, j range over $\{1, 2, \ldots, n\}$. We denote by $\mathbf{X}$ the tuple $(X_1, \ldots, X_n)$. The random variables are assumed to take values from the finite set V of cardinality K. We let x_i, u_i, v_i to denote a value taken by X_i. Our dynamic Bayesian networks will be time variant but with a regular structure [2]. They will be unrolled over a finite number of time points. Further, there will be no distinction between hidden and observable variables.

A *Dynamic Bayesian Network (DBN)* is a structure $\mathcal{D} = (\mathcal{X}, T, Pa, \{C_i^t\})$ where,

- T is a positive integer with t ranging over the set of time points $\{0, 1, \ldots, T\}$.
- $\mathcal{X} = \{X_i^t \mid 1 \leq i \leq n, 0 \leq t \leq T\}$ is the set of random variables. As usual, these variables will be identified with the nodes of the DBN. X_i^t is the instance of X_i at time slice t.

- Pa assigns a set of parents to each node and satisfies: (i) $Pa(X_i^0 = \emptyset)$ (ii) If $X_j^{t'} \in Pa(X_i^t)$ then $t' = t-1$. (iii) If $X_j^{t-1} \in Pa(X_i^t)$ for some t then $X_j^{t'-1} \in Pa(X_i^{t'})$ for every $t' \in \{1, 2, \ldots, T\}$. Thus the way nodes at the $(t-1)^{th}$ time slice are connected to nodes at the t^{th} time slice remains invariant as t ranges over $\{1, 2, \ldots, T\}$.
- C_i^t is the *Conditional Probability Table (CPT)* associated with node X_i^t specifying the probabilities $P(X_i^t \mid Pa(X_i^t))$. Suppose $Pa(X_i^t) = \{X_{j_1}^{t-1}, X_{j_2}^{t-1}, \ldots, X_{j_m}^{t-1}\}$ and $(x_{j_1}, x_{j_2}, \ldots, x_{j_m}) \in V^m$. Then as usual we require, $\sum_{x_i \in V} C_i^t(x_i \mid x_{j_1}, x_{j_2}, \ldots, x_{j_m}) = 1$. Since our DBNs are time-variant, in general C_i^t will be different from $C_i^{t'}$ if $t \neq t'$.

A state of the DBN at t will be a member of V^n, say $\mathbf{s} = (x_1, x_2, \ldots, x_n)$ specifying that $X_i^t = x_i$ for $1 \leq i \leq n$. This in turn stands for $X_i = x_i$ for $1 \leq i \leq n$ at t. Suppose $Pa(X_i^t) = \{X_{j_1}^{t-1}, X_{j_2}^{t-1}, \ldots, X_{j_m}^{t-1}\}$. Then a CPT entry of the form $C_i^t(x_i \mid x_{j_1}, x_{j_2}, x_{j_m}) = p$ says that if the system is in a state at $t-1$ in which $X_{j_l} = x_{j_l}$ for $1 \leq l \leq m$, then the probability of $X_i = x_i$ being the case at t is p. In this sense the CPTs specify the probabilistic dynamics locally.

The regular structure of our DBNs induces the function PA given by: $X_j \in PA(X_i)$ iff $X_j^{t-1} \in Pa(X_i^t)$. We define $\hat{i} = \{j \mid X_j \in PA(X_i)\}$ to capture Pa in terms of the corresponding indices.

In the following, $\mathbf{x}_I$ will denote a vector of values over the index set $I \subseteq \{1, 2, \ldots, n\}$. It will be viewed as a map $\mathbf{x}_I : I \to V$. We will often denote $\mathbf{x}_I(i)$ as $\mathbf{x}_{I,i}$ or just $\mathbf{x}_i$ if I is clear from the context. If $I = \{i\}$ and $\mathbf{x}_I(i) = x_i$, we will identify $\mathbf{x}_I$ with x_i. If I is the full index set $\{1, 2, \ldots, n\}$, we will simply write $\mathbf{x}$. Further, we denote by $\mathbf{X}^t$ the vector of random variables $(X_1^t, \ldots, X_n^t)$.

Using these notations, we can write $C_i^t(x_i \mid \mathbf{u}_{\hat{i}}) = p$ to mean that p is the probability that $X_i = x_i$ at time t given that at time $t-1$, $X_{j_1} = \mathbf{u}_{j_1}, X_{j_2} = \mathbf{u}_{j_2}, \ldots, X_{j_m} = \mathbf{u}_{j_m}$ with $\hat{i} = \{j_1, j_2, \ldots, j_m\}$.

The probability distribution $P(X_1^t, X_2^t, \ldots, X_n^t)$ describes the possible states of the system at time t. In other words, $P(\mathbf{X}^t = \mathbf{x})$ is the probability that the system will reach the state $\mathbf{x}$ at t. Starting from $P(\mathbf{X}^0)$ at time 0, given by $P(\mathbf{X}^0 = \mathbf{x}) = \prod_i C_i^0(\mathbf{x}_i)$, one would like to compute $P(X_1^t, \ldots, X_n^t)$ for a given t.

We can use the CPTs to inductively compute this:

$$P(\mathbf{X}^t = \mathbf{x}) = \sum_{\mathbf{u}} \Big(\prod_i C_i^t(\mathbf{x}_i \mid \mathbf{u}_{\hat{i}}) \Big) P(\mathbf{X}^{t-1} = \mathbf{u}) \tag{1}$$

with $\mathbf{u}$ ranging over V^n.

Since $V = K$, the number of possible states at t is K^n. Hence explicitly computing and maintaining the probability distributions is feasible only if n is small or if the underlying graph of the DBN falls apart into many disjoint components. Neither restriction is realistic and hence one needs approximate ways to maintain $P(\mathbf{X}^t)$ compactly and compute it efficiently. Before we get into methods for doing this, we shall describe how we use DBNs to model the dynamics of biopathways.

2.1. DBN models of biopathways

Biological pathways are often described as a network of biochemical reactions. The dynamics of such a network can often be modeled as a system of deterministic ODE; one equation of the form $\frac{dy}{dt} = f(\mathbf{y}, \mathbf{r})$ for each molecular species y, with f describing the kinetics of the reactions that produce and consume y, while $\mathbf{y}$ is the set (vector) of molecular species taking part in these reactions and $\mathbf{r}$ are the rate constants (parameters) associated with these reactions. For large pathways, this system of ODE will not admit a closed form solution. Hence one will have to resort to large scale numerical simulations to perform analysis. A crucial analysis task will be parameter estimation since many of the rate constants and initial concentrations will be unknown and have to be estimated. This in turn will involve searching through a high dimensional space accompanied by repeated numerical simulations to evaluate the quality of the candidate parameter values. Further, model calibration and validation will have to be carried out using limited data that has only crude precision. Motivated by these considerations, we have developed a method for approximating a system of ODE as a dynamic Bayesian network which we shall explain now. Some of its main features are illustrated by the example shown in Figure 1.

The first step in the approximation procedure is to discretize the time domain. For biopathways, experimental data will be available only for a few time points with the value measured at the final time point typically signifying the

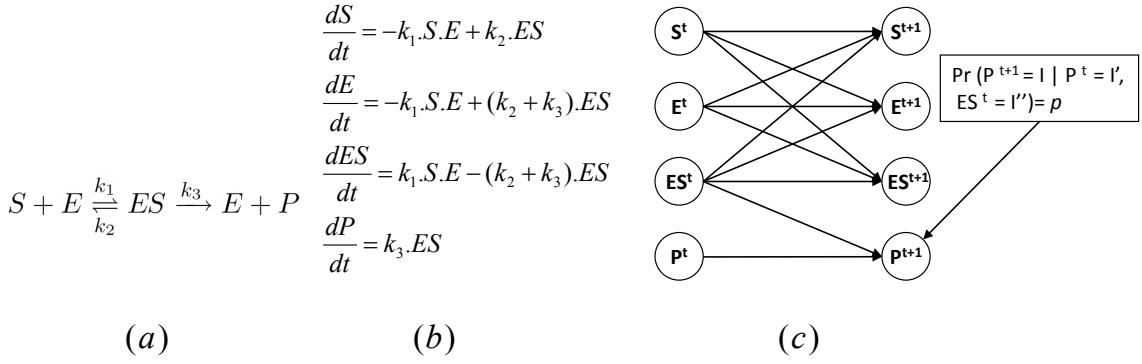

Figure 1: (a) The enzyme catalytic reaction network (b) The ODE model (c) The DBN approximation for 2 successive time slices

steady state value. Hence we assume the dynamics is of interest only for discrete time points and, furthermore, only up to a maximal time point. We denote these time points as $\{0, 1, \ldots, T\}$. It is *not* necessary to uniformly discretize the time domain though we shall often do so for convenience.

Next we assume that the values of the variables can be observed with only finite precision and accordingly partition the range of each variable y_i (rate constant r_j) into a set of intervals $\mathbf{I}_i$ ($\mathbf{I}_j$). Again, it is not necessary to partition the value range of each variable evenly but we will often do so for convenience. The initial values as well as the parameters of the ODE system are assumed to be distributions (usually uniform) over certain intervals. We then sample the initial states of the system many times [1] and generate a trajectory by numerical integration for each sampled initial state. The resulting set of trajectories is then treated as an approximation of the dynamics of ODE system.

To handle unknown rate constants we assume their minimum and maximum values are known. We then partition these ranges of values also into a finite numbers of intervals, and fix a uniform distribution over *all* the intervals. After building the DBN, we use a Bayesian inference based technique to perform parameter estimation to complete the construction of the model (the details can be found in [1]). However, unlike the variables, once the initial value of a rate constant has been sampled, this value will not change during the process of generating a trajectory. Naturally, different trajectories can have different initial values for an unknown rate constant. Similar considerations apply to unknown initial concentrations.

A key idea is to compactly store the generated set of trajectories as a dynamic Bayesian network. This is achieved by exploiting the network structure and by simple counting. First we specify one random variable $Y_i(R_j)$ for each variable y_i (parameter r_j). The node $Y_k^{t-1}(R_j^{t-1})$ will be in $Pa(Y_i^t)$ iff $k = i$ or $y_k(r_j)$ appears in the equation for y_i. On the other hand R_j^{t-1} will be the only parent of the parameter node R_j^t since a parameter value does not change once its initial value has been fixed. (In Figure 1 we have suppressed rate constant nodes to avoid clutter.)

A CPT entry of the form $C_i^t(I \mid \mathbf{I}_{\hat{i}}) = p$ says that p is the probability of the value of y_i falling in the interval I at time t, given that the value of Z_{k_l} was in $\mathbf{I}_{k_l}$ for each $Z_{k_l}^{t-1}$ in $Pa(Y_i^t)$. The probability p is calculated through simple counting. Suppose N is the number of generated trajectories. We record, for how many of these trajectories, the value of Z_{k_l} falls in the interval $\mathbf{I}_{k_l}$ simultaneously for each $k_l \in \hat{i}$. Suppose this number is J. We then determine for how many of these J trajectories, the value of Y_i falls in the interval I at time t. If this number is J' then p is set to be $\frac{J'}{J}$.

The one time cost of constructing the DBN can be easily recovered through the substantial gains obtained in doing parameter estimation and sensitivity analysis [1]. This method can cope with large biochemical networks with many unknown parameters. It has been used to uncover new biological facts about the complement system [4] where the starting point was a ODE based model with 71 unknown parameters. In these studies FF was used as the core inferencing algorithm. It was then we began to consider its shortcomings and started to seek an improved version.

3. The Hybrid Factored Frontier algorithm

In this section, we present the HFF algorithm. Since it is a parametrized version of FF, we first present FF. In doing so we will use the notations developed in Section 2.

Approximate probability distributions will be called belief states and denoted by B, B^t etc. Exact probability distributions will be denoted by P, P^t etc. Formally, a belief state B is a map from $V^n \rightarrow [0,1]$ such that $\sum_{\mathbf{u}\in V^n} B(\mathbf{u}) = 1$. Thus a belief state is just a probability distribution but it will be convenient to linguistically separate them.

The FF algorithm uses marginal functions to represent belief states. A marginal function is a map $M : \{1,\ldots,n\} \times V \rightarrow [0,1]$ such that $\sum_{v\in V} M(i,v) = 1$ for each i. In what follows, u, v will range over V while $\mathbf{u}$ and $\mathbf{v}$ will range over V^n. From a marginal function M, one can obtain the belief state B_M via $B_M(\mathbf{u}) = \prod_i M(i,\mathbf{u}_i)$. On the other hand, a belief state B induces the marginal function M_B via $M_B(i,v) = \sum_{\mathbf{u}|\mathbf{u}_i=v} B(\mathbf{u})$. It is easy to see that for a marginal function M we have $M_{B_M} = M$, but in general $B_{M_B} \neq B$ for a belief state B.

Given a DBN $\mathcal{D} = (\mathcal{X}, T, Pa, \{C_i^t\})$, FF computes inductively a sequence M^t of marginal functions as:

- $M^0(i,u) = C_i^0(u)$,
- $M^{t+1}(i,u) = \sum_{\mathbf{v}\in V_{\widehat{i}}} [C_i^t(u \mid \mathbf{v}) \prod_{j\in\widehat{i}} M^t(j,\mathbf{v}_j)]$.

It is easy to check that these are indeed marginal functions, i.e., $\sum_{u\in V} M^t(i,u) = 1$ for all t and i. Thus FF maintains B^t, the belief state at t, compactly via the marginal function M^t. More precisely, $B^t(\mathbf{u}) = \prod_j M^t(j,\mathbf{u}_j) = B_{M^t}(\mathbf{u})$.

Now suppose the DBN transforms the belief state B^{t-1} into the new belief state $\widehat{B}^t$. In other words, $\widehat{B}^t$ is the belief state obtained by performing $t-1$ steps of FF and exact computation at the t^{th} step. Then by Equation (1), we have for all $t \geq 1$:

$$\widehat{B}^t(\mathbf{x}) = \sum_{\mathbf{u}} \Big(\prod_i C_i^t(\mathbf{x}_i \mid \mathbf{u}_{\widehat{i}}) \Big) B^{t-1}(\mathbf{u}) \tag{2}$$

However, the t^{th} step of FF computes directly the marginal function M^t, which then represents the new belief state at time t as $B^t = B_{M^t}$. In general, $B^t \neq \widehat{B}^t$, that is, the belief state B^t represented via M^t is an approximation of the belief state $\widehat{B}^t$ as defined above. However, the computation of M^t is itself accurate in the following sense.

Proposition 1. *For all $t \in \{1,\ldots,T\}$, $M^t(i,v) = M_{\widehat{B}^t}(i,v)$ for each i and v.*

PROOF. For $t > 0$, we have:

$$\begin{aligned}
M_{\widehat{B}^t}(i,v) &= \sum_{\mathbf{v}|\mathbf{v}_i=v} \widehat{B}^t(\mathbf{v}) = \sum_{\mathbf{v}|\mathbf{v}_i=v} \sum_{\mathbf{u}} \prod_j (C_j^t(\mathbf{v}_j \mid \mathbf{u}_{\widehat{j}})) B^{t-1}(\mathbf{u}) && \text{(by Equation (2))} \\
&= \sum_{\mathbf{u}} B^{t-1}(\mathbf{u}) \sum_{\mathbf{v}|\mathbf{v}_i=v} \prod_j (C_j^t(\mathbf{v}_j \mid \mathbf{u}_{\widehat{j}})) \\
&= \sum_{\mathbf{u}} B^{t-1}(\mathbf{u}) \Big(\sum_{\mathbf{v}_n} C_n^t(\mathbf{v}_n \mid \mathbf{u}_{\widehat{n}}) \Big) \ldots \Big(C_i^t(v \mid \mathbf{u}_{\widehat{i}}) \Big) \ldots \Big(\sum_{\mathbf{v}_1} C_1^t(\mathbf{v}_1 \mid \mathbf{u}_{\widehat{1}}) \Big) \\
&= \sum_{\mathbf{u}} B^{t-1}(\mathbf{u}) \Big(C_i^t(v \mid \mathbf{u}_{\widehat{i}}) \Big)
\end{aligned}$$

The last of the above equalities follows since each of the summands within the expression add up to 1. Now, using $B^{t-1}(\mathbf{u}) = \prod_k M^{t-1}(k,\mathbf{u}_k)$ and splitting the above summation, we obtain:

$$\begin{aligned}
M_{\widehat{B}^t}(i,v) &= \sum_{\mathbf{u}\in V_{\widehat{i}}} \sum_{\mathbf{u}\notin V_{\widehat{i}}} \prod_k M^{t-1}(k,\mathbf{u}_k) \Big(C_i^t(v \mid \mathbf{u}_{\widehat{i}}) \Big) \\
&= \sum_{\mathbf{u}\in V_{\widehat{i}}} \prod_{k\in\widehat{i}} M^{t-1}(k,\mathbf{u}_k) \Big(C_i^t(v \mid \mathbf{u}_{\widehat{i}}) \Big) \sum_{\mathbf{u}\notin V_{\widehat{i}}} \prod_{k\notin\widehat{i}} M^{t-1}(k,\mathbf{u}_k) \\
&= \sum_{\mathbf{u}\in V_{\widehat{i}}} \prod_{k\in\widehat{i}} M^{t-1}(k,\mathbf{u}_k) \Big(C_i^t(v \mid \mathbf{u}_{\widehat{i}}) \Big) \prod_{k\notin\widehat{i}} \sum_{\mathbf{u}_k} M^{t-1}(k,\mathbf{u}_k) \\
&= \sum_{\mathbf{u}\in V_{\widehat{i}}} \prod_{k\in\widehat{i}} M^{t-1}(k,\mathbf{u}_k) \Big(C_i^t(v \mid \mathbf{u}_{\widehat{i}}) \Big) = M^t(i,v)
\end{aligned}$$

The second factor above is just a product of 1's (by the definition of marginals) and the proposition follows. □

As B^0 is accurate by definition, M^1 will also be accurate but not necessarily B^1. A simple but crucial observation is that whenever the error $max_{\mathbf{u}\in V^n}\{|\widehat{B}^t(\mathbf{u}) - B^t(\mathbf{u})|\}$ incurred by FF at step $t > 0$ (ignoring the error made in the previous steps) is large for some $\mathbf{u}$ then $M^t(i, \mathbf{u}_i)$ is large for *every* i. This is so since, $M^t(j, \mathbf{u}_j) = M_{\widehat{B}^t}(j, \mathbf{u}_j) \geq \max(\widehat{B}^t(\mathbf{u}), B^t(\mathbf{u}))$, which follows from Proposition 1 and the definition of marginals. A second important observation is that there can only be a few instances of $\mathbf{u}$ with large values of $B^t(\mathbf{u})$, since these values sum to 1. Consequently there can only be a few instances of $\mathbf{u}$ such that $M^t(i, \mathbf{u}_i)$ is large for every i. For instance, there can be only one such $\mathbf{u}$ if we want $M^t(i, \mathbf{u}_i) > \frac{1}{2}$ for each i. Hence, by maintaining $B^t(\mathbf{u})$ explicitly for a small subset of V^n for which M^t is high for all dimensions, one can hope to reduce the one step error incurred FF and hence the overall error too. This is the intuition underlying the HFF algorithm.

3.1. The Hybrid Factored Frontier algorithm

The overall structure of HFF is as follows. Starting with $t = 0$, we inductively compute and maintain the tuple $(M^t, S^t, B^t_H, \alpha^t)$, where:

- M^t is a marginal function.
- $S^t \subseteq V^n$ is a set of tuples called *spikes*.
- $B^t_H : V^n \to [0, 1]$ is a function s.t. $B^t_H(\mathbf{u}) = 0$ if $\mathbf{u} \notin S^t$ and $\sum_{\mathbf{u}\in S^t} B^t_H(\mathbf{u}) < 1$.
- $\alpha^t = \sum_{\mathbf{u}\in S^t} B^t_H(\mathbf{u})$.

This hybrid state $(M^t, S^t, B^t_H, \alpha^t)$ represents the following belief state B^t:

$$B^t(\mathbf{u}) = B^t_H(\mathbf{u}) + (1 - \alpha^t)\prod_i M^t_H(i, \mathbf{u}_i), \text{ where}$$

$$M^t_H(i, v) = [M^t(i, v) - \sum_{\{\mathbf{u}\in S^t | \mathbf{u}_i = v\}} B^t_H(\mathbf{u})]/(1 - \alpha^t)$$

Notice that we need to use M^t_H rather than M^t since the cumulative weight of the contribution made by the spikes needs to be discounted from M^t. As will be demonstrated subsequently, B^t is a belief state and M^t_H a marginal function.

The HFF algorithm

We initialize with $M^0 = C^0$, $S^0 = \emptyset$, $B^0_H = \mathbf{0}$ and $\alpha^0 = 0$ and fix a parameter σ. This σ will be the number of spikes we choose to maintain. It is a crucial parameter as our results will show. We inductively compute $(M^{t+1}, S^{t+1}, B^{t+1}_H, \alpha^{t+1})$ from $(M^t, S^t, B^t_H, \alpha^t)$ as follows.

Step 1: We first compute M^{t+1} as:

$$M^{t+1}(i, x) = \sum_{\mathbf{u}\in S^t}[C^{t+1}_i(x \mid \mathbf{u}_{\hat{i}}) \times B^t_H(\mathbf{u})] + (1 - \alpha^t)\Big(\sum_{\mathbf{u}_{\hat{i}}}[C^{t+1}_i(x \mid \mathbf{u}_{\hat{i}}) \times \prod_{j\in\hat{i}} M^t_H(j, \mathbf{u}_j)]\Big)$$

Step 2: We next compute a set S^{t+1} of at most σ spikes using M^{t+1}. We want to consider as spikes $\mathbf{u} \in V^n$ where $M^{t+1}(i, \mathbf{u}_i)$ is large for every i. To do so, we find a constant η^{t+1} such that $M^{t+1}(i, \mathbf{u}_i) \geq \eta^{t+1}$ for every i for a subset of V^n containing σ elements and for all other $\mathbf{u}'$, there exists i with $M^{t+1}(i, \mathbf{u}'_i) < \eta^{t+1}$. We compute η^{t+1} via binary search. First we fix the precision with which we want to compute η^{t+1} to be ξ. We have found $\xi = 10^{-6}$ to be a good choice. For this choice there will be at most 20 iterations of the loop described below. The search for η^{t+1} proceeds as follows:

- $\eta_1 = 0$ and $\eta_2 = 1$.
- While $\eta_2 - \eta_1 > \xi$ do
  1. $\eta = \frac{\eta_1+\eta_2}{2}$.

2. Determine the set of values U_i such that $v \in U_i$ iff $M^{t+1}(i, v) > \eta$.
3. Set a_i to be the cardinality of U_i.
4. If $\prod_i(a_i) > \sigma$ then $\eta_1 = \eta$; otherwise $\eta_2 = \eta$

- endwhile
- Return $\eta^{t+1} = \eta_2$ and $S^{t+1} = \prod_i U_i$

Step 3: Finally, we compute $B_H^{t+1}(\mathbf{u})$ for each $\mathbf{u}$ in S^{t+1} as follows.

$$B_H^{t+1}(\mathbf{u}) = \sum_{\mathbf{v} \in S^t} (B^t(\mathbf{v}) \times \prod_i C_i^{t+1}(\mathbf{u}_i \mid \mathbf{v}_{\widehat{i}}))$$

End of Algorithm

As in the case of FF, we denote by $\widehat{B}^{t+1}$ the belief state obtained from B^t through one step of the DBN:

$$\widehat{B}^{t+1}(\mathbf{u}) = \sum_{\mathbf{v} \in V^n} (B^t(\mathbf{v}) \times \prod_i C_i^{t+1}(\mathbf{u}_i \mid \mathbf{v}_{\widehat{i}}))$$

Notice that $B_H^{t+1}(\mathbf{u}) \leq \widehat{B}^{t+1}(\mathbf{u})$ for all $\mathbf{u}$. To establish the key properties of HFF, we let D be the maximum in-degree of the DBN graph. We also recall that T is the number of time points, σ the number of spikes, n the number of variables and V is the set of values with $K = |V|$.

Theorem 2. *HFF has the following properties.*

1. *if $\sigma = 0$, the HFF algorithm is the same as FF and if $\sigma = K^n$, it is the exact algorithm.*
2. *$M^t(i, v) = M_{\widehat{B}^t}(i, v)$ for every v. Further, B^t is a belief state while M_H^t and M^t are marginal functions for every t.*
3. *The time complexity of HFF is $O(T \cdot n \cdot (\sigma^2 + K^{D+1}))$.*

PROOF. $\sigma = 0$ implies that the set of spikes $S^t = \emptyset$ for all t. This implies that $\alpha^t = 0$ and the computation done by HFF is the same as FF. If $\sigma = K^n$, then $S^t = V$ for all t and $\alpha^t = 1$ (of course, M_H^t is then not computed). Thus, this boils down to perform exact inferencing. We have now established part (1).

We prove that for all $t \geq 1$, if B^{t-1} is a belief state and M^{t-1}, M_H^{t-1} are marginals, then $M^t = M_{\widehat{B}^t}$ and B^t is a belief state and M^t, M_H^t are marginals. We thus obtain Part (2) by induction on t, using the fact that B^0 is a belief state and M^0, M_H^0 are marginals by definitions. For $t \geq 0$, let $M^t(i, v), M_H^t(i, v)$ be marginals and B^t be a belief state. Then at $t + 1$, let us start by proving $M_{\widehat{B}^{t+1}}(i, v) = M^{t+1}(i, v)$. The first step is the same as in Proposition 1:

$$M_{\widehat{B}^{t+1}}(i, v) = \sum_{\mathbf{v} | \mathbf{v}_i = v} \widehat{B}^{t+1}(\mathbf{v}) = \sum_{\mathbf{u}} B^t(\mathbf{u}) \Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}}) \Big) \qquad \text{(by Equation (2))} \tag{3}$$

Now however, the definition of B^t is different for HFF and so we have from (3) above:

$$M_{\widehat{B}^{t+1}}(i, v) = \sum_{\mathbf{u}} B_H^t(\mathbf{u}) \Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}}) \Big) + (1 - \alpha^t) \sum_{\mathbf{u}} \Big(\prod_{k=1}^{n} M_H^t(k, \mathbf{u}_k) \Big) \Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}}) \Big) \tag{4}$$

But if $\mathbf{u} \notin S^t$, then $B_H^t(\mathbf{u}) = 0$. Further splitting the second term as in Proposition 1, we obtain:

$$
\begin{aligned}
M_{\widehat{B}^{t+1}}(i,v) &= \sum_{\mathbf{u}\in S^t} B_H^t(\mathbf{u})\Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}})\Big) + (1-\alpha^t)\sum_{\mathbf{u}\in V_{\widehat{i}}}\sum_{\mathbf{u}\notin V_{\widehat{i}}}\prod_k M_H^t(k,\mathbf{u}_k)\Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}})\Big) \\
&= \sum_{\mathbf{u}\in S^t} B_H^t(\mathbf{u})\Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}})\Big) + (1-\alpha^t)\sum_{\mathbf{u}\in V_{\widehat{i}}}\prod_{k\in\widehat{i}} M_H^t(k,\mathbf{u}_k)\Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}})\Big)\sum_{\mathbf{u}\notin V_{\widehat{i}}}\prod_{k\notin\widehat{i}} M_H^t(k,\mathbf{u}_k) \\
&= \sum_{\mathbf{u}\in S^t} B_H^t(\mathbf{u})\Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}})\Big) + (1-\alpha^t)\sum_{\mathbf{u}\in V_{\widehat{i}}}\prod_{k\in\widehat{i}} M_H^t(k,\mathbf{u}_k)\Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}})\Big)\prod_{k\notin\widehat{i}}\sum_{\mathbf{u}_k} M_H^t(k,\mathbf{u}_k) \\
&= \sum_{\mathbf{u}\in S^t} B_H^t(\mathbf{u})\Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}})\Big) + (1-\alpha^t)\sum_{\mathbf{u}\in V_{\widehat{i}}}\prod_{k\in\widehat{i}} M_H^t(k,\mathbf{u}_k)\Big(C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}})\Big)\times 1 \\
&= M^{t+1}(i,v)
\end{aligned}
$$

In the step above, $\sum_{\mathbf{u}_k} M_H^t(k,\mathbf{u}_k) = 1$ follows from our inductive hypothesis that M_H^t is a marginal. Now, we will prove the remainder of this part, i.e., M^{t+1}, M_H^{t+1} are marginals and B^{t+1} is a belief state. For all i, again from $\alpha^t = \sum_{\mathbf{u}\in S^t} B_H^t(\mathbf{u})$ and $\sum_{v\in V} C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}}) = 1$, we have:

$$
\begin{aligned}
\sum_{v\in V} M^{t+1}(i,v) &= \sum_{\mathbf{u}\in S^t}[\sum_{v\in V} C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}}) \times B_H^t(\mathbf{u})] + (1-\alpha^t)\Big(\sum_{\mathbf{u}_{\widehat{i}}}[\sum_{v\in V} C_i^{t+1}(v \mid \mathbf{u}_{\widehat{i}}) \times \prod_{j\in\widehat{i}} M_H^t(j,\mathbf{u}_j)]\Big) \\
&= \sum_{\mathbf{u}\in S^t} B_H^t(\mathbf{u}) + (1-\alpha^t)\sum_{\mathbf{u}_{\widehat{i}}}\prod_{j\in\widehat{i}} M_H^t(j,\mathbf{u}_j) = \alpha^t + (1-\alpha^t)\prod_{j\in\widehat{i}}\sum_{\mathbf{u}_j} M_H^t(j,\mathbf{u}_j) = 1
\end{aligned}
$$

Now, using the above and $\alpha^{t+1} = \sum_{\mathbf{u}\in S^{t+1}} B_H^{t+1}(\mathbf{u})$ (assuming $\alpha^{t+1} \neq 1$), we have:

$$
\begin{aligned}
\sum_{v\in V} M_H^{t+1}(i,v) &= \Big(\sum_{v\in V} M^{t+1}(i,v) - \sum_{v\in V}\sum_{\mathbf{u}\in S^{t+1}|\mathbf{u}_i=v} B_H^{t+1}(\mathbf{u})\Big) \times \frac{1}{1-\alpha^{t+1}} \\
&= \Big(1 - \sum_{\mathbf{u}\in S^{t+1}} B_H^{t+1}(\mathbf{u})\Big)\frac{1}{1-\alpha^{t+1}} = 1
\end{aligned}
$$

$$
\sum_{\mathbf{u}\in V^n} B^{t+1}(\mathbf{u}) = \sum_{\mathbf{u}\in V^n} B_H^{t+1}(\mathbf{u}) + (1-\alpha^{t+1})\sum_{\mathbf{u}\in V^n}\prod_i M_H^{t+1}(i,\mathbf{u}_i) = \sum_{\mathbf{u}\in S^{t+1}} B_H^{t+1}(\mathbf{u}) + (1-\alpha^{t+1})\times 1 = 1
$$

It now follows easily that for any i, v, $1 \geq M^{t+1}(i,v) \geq 0$ and $1 \geq M_H^{t+1}(i,v)$. It remains to prove that $M_H^{t+1}(i,v) \geq 0$, that is $M^{t+1}(i,v) \geq \sum_{\mathbf{u}\in S^{t+1}|\mathbf{u}_i=v} B_H^{t+1}(\mathbf{u})$. As $B_H^{t+1}(\mathbf{u}) \leq \widehat{B}^{t+1}(\mathbf{u})$ for all $\mathbf{u}$, we have,

$$
\sum_{\mathbf{u}\in S^{t+1}|\mathbf{u}_i=v} B_H^{t+1}(\mathbf{u}) \leq \sum_{\mathbf{u}\in S^{t+1}|\mathbf{u}_i=v} \widehat{B}^{t+1}(\mathbf{u}) = M_{\widehat{B}^{t+1}}(i,v) = M^{t+1}(i,v)
$$

which completes the proof of part(2).

Turning to part (3), we note that at each time point, the step 1 of HFF has the same complexity as FF together with the spikes contributing: $O(K \cdot n \cdot (K^D + \sigma))$. Step 2 makes at most $K \times n$ comparisons for each iteration of the loop and there are only a constant number of iterations of the loop. Thus the complexity of this step per time point is $O(K \times n)$. Step 3 computes for each spike, $B^t(u)$ from the values of $B_H^t(u)$ and $M^t(i,u)$ which takes $O(Kn + \sigma n)$. Then, we sum over all the spikes the value computed by multiplying n values of the CPT which takes $O(\sigma \times n)$. Thus, this step overall takes $O(\sigma K n + n\sigma^2)$. Hence the overall time complexity of HFF is the sum of all these quantities which is $O(T \cdot n \cdot (K^{D+1} + K\sigma + \sigma^2))$ which is bounded by $O(T \cdot n \cdot (K^{D+1} + \sigma^2))$. □

HFF gathers in one sweep -just as FF does- the required information about the belief states. However, it can take more time than FF depending on the number of spikes but the added complexity is only quadratic in the number of spikes.

3.2. Error analysis

It is easy to see that with each time slice t of the DBN one can associate a stochastic matrix $\mathcal{T}_t$. This stochastic matrix will capture the transformation of probability distributions effected by the n CPTs associated with the time slice t as dictated by Equation 1 in Section 2. In particular, we will have $P(X^t) = \mathcal{T}_t(P(X^{t-1}))$.

We now denote the cumulative error at t as Δ^t and define it to be: $\Delta^t = max_{\mathbf{u}\in V^n}(|P(X^t = \mathbf{u}) - B^t(u)|)$. Towards deriving an upper bound for Δ^t, we first note that Markov chain theory (for instance, using the Dobrushin's coefficient, see chapter 6.7 in [27]) guarantees the following:

Theorem 3. *Let $\mathcal{T}$ be an n-dimensional stochastic matrix. Then for two probability distributions A, B, we have $||\mathcal{T}(A) - \mathcal{T}(B)||_\infty \leq \beta_\mathcal{T}||A - B||_\infty$ where $0 \leq \beta_\mathcal{T} \leq 1$ is a constant that depends only on $\mathcal{T}$.*

$\beta_\mathcal{T}$ is called the *contraction factor*. In what follows we shall write β^t for the contraction factor associated with $\mathcal{T}_t$ and set $\beta = \max_t \beta^t$. Using a reasoning similar to [3] we shall show that Δ^t can be bounded by $\epsilon_0(\sum_{j=0}^t \beta^j)$ where ϵ_0 is the maximum one step error given by: $\epsilon_0 = \max_t ||B^t - \mathcal{T}_t(B^{t-1})||_\infty$. (We note that previously $\mathcal{T}_t(B^{t-1})$ was denoted as $\widehat{B}^t$).

Lemma 4. *$\Delta^t \leq \epsilon_0(\sum_{j=0}^t \beta^j)$. Further if $\beta < 1$, we have $\Delta^t \leq \frac{\epsilon_0}{1-\beta}$.*

PROOF. By definition of overall error and the above stated property of Markov chains,

$$\Delta^t = |B^t - P(X^t)| \leq |B^t - \mathcal{T}_t(B^{t-1})| + |\mathcal{T}_t(B^{t-1}) - \mathcal{T}_t(P(X^{t-1}))| \leq \epsilon_0 + \beta_t \Delta^{t-1}$$

Then by recursively computing the second factor, we obtain,

$$\Delta^t \leq \epsilon_0 + \beta_t\epsilon_0 + \beta_t\beta_{t-1}\epsilon_0 + \ldots + (\beta_t\beta_{t-1}\cdots\beta_1)\epsilon_0 \leq \epsilon_0(\sum_{j=0}^{t} \beta^j)$$

Further if $\beta < 1$, we have:

$$\Delta^t \leq \epsilon_0(\sum_{j=0}^{t} \beta^j) \leq \epsilon_0(\sum_{j=0}^{\infty} \beta^j) = \frac{\epsilon_0}{1-\beta}$$

□

We note that $\sum_{j=0}^t \beta^j$ depends only on the DBN. Hence, theoretically comparing the error behaviors of FF and HFF amounts to comparing their single step errors. To do so, we shall next analyze single step error of FF followed by that of HFF.

Recall that for FF, $B^t = B_{M_{\widehat{B}^t}}$ and that the one-step error incurred by FF at step t is $max_{\mathbf{u}\in V^n}\{|\widehat{B}^t(\mathbf{u}) - B_{M_{\widehat{B}^t}}(\mathbf{u})|\}$. We can bound this from above by ϵ_0 where : $\epsilon_0 = max\{|B(\mathbf{u}) - B_{M_B}(\mathbf{u})|\}$ with B ranging over the set of all possible belief states and $\mathbf{u}$ ranging over V^n. It turns out that ϵ_0 can be made arbitrarily close to 1 as n, the number of variables, tends to ∞. To see this, fix $0 < \delta < 1$ and consider the belief state B defined by $B(\mathbf{u}) = 1-\delta$, $B(\mathbf{u}') = \delta$ for some $\mathbf{u}, \mathbf{u}' \in V^n$ such that for all i, $\mathbf{u}_i \neq \mathbf{u}_j$ and $B(\mathbf{v}) = 0$ for all $\mathbf{v} \in V^n \setminus \{\mathbf{u}, \mathbf{u}'\}$. Then, $M(i, \mathbf{u}_i) = 1 - \delta$ for all $i \in \{1, \ldots, n\}$ and so $B_{M_B}(\mathbf{u}) = (1-\delta)^n$. As a result we have $\epsilon_0 = max|B - B_{M_B}| \geq (1-\delta) - (1-\delta)^n$ which tends to $1 - \delta$ as n tends to ∞. Now if we choose δ to be close to 0, $1 - \delta$ is close to 1. Thus ϵ_0 can be made as close to 1 as we want, with n tending to ∞. We found that the cumulative errors made by FF can be large in practice too as shown in the next section.

Notice that for HFF too we have $B^t = B_{M_{\widehat{B}^t}}$, and its one step error can be bound by ϵ_0. However, the spikes can be used to bound the single step error of HFF more precisely as follows:

Claim 1. *The one step error made by HFF is bounded by $\hat{\epsilon}_0$ with $\hat{\epsilon}_0 \leq \min\{(1-\alpha), \eta\}$, where $\alpha = \min_t(\alpha^t)$ and $\eta = \max_t(\eta^t)$.*

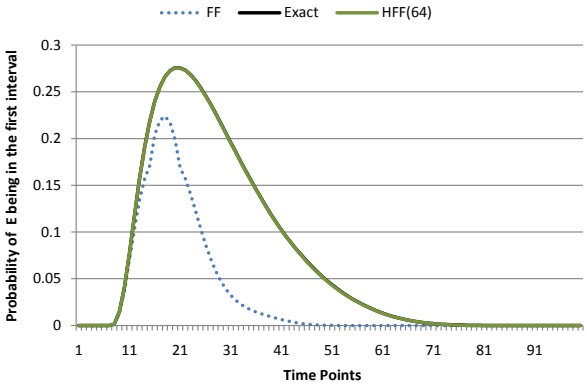

Figure 2: $M^t(E \in [0,1))$

PROOF (SKETCH). If α is large, then the value of $\widehat{B}^t(\mathbf{u}) \leq 1-\alpha$ for $\mathbf{u} \notin S^t$. Also, as $B^t(\mathbf{u}) \leq \widehat{B}^t(\mathbf{u})$, we have $\widehat{B}^t(\mathbf{u}) - B^t(\mathbf{u}) \leq 1-\alpha$. Finally, if η is small, then by construction for all $\mathbf{u} \notin S^t$, $M^{t+1}(i,v) \leq \eta$ for some i with $\mathbf{u}_i = v$, and hence $\widehat{B}^t(\mathbf{u}) \leq M^{t+1}(i,v) \leq \eta$. Also, $\widehat{B}^t(\mathbf{u}) - B^t(\mathbf{u}) \leq \eta \times \sum_{\mathbf{v} \notin S^t} \prod_i C_i^{t+1}(\mathbf{u}_i \mid \mathbf{v}_i)) \leq \eta$ for $\mathbf{u} \in S^t$.

Thus, the worst case error for HFF with at least two spikes (implying $\eta < 1/2$) is smaller than for FF. Taking more spikes will increase α and decrease η , reducing the worst case error. Experiments in the next section show that the practical accuracy is also improved as we increase the number of spikes.

4. Experimental evaluation

We have implemented our algorithm in C++. The experiments reported here were carried out on a Opteron 2.2Ghz Processor, with 3GB memory. The algorithms were evaluated on five DBN models of biochemical networks: the small enzyme catalytic reaction network shown in Figure 1 for initial experimentation, the EGF-NGF pathway [6] under (a) EGF-stimulation (b) NGF-stimulation (c) co-stimulation of EGF and NGF, and the Epo mediated ERK signaling pathway. The ODE model for the EGF-NGF pathway was obtained from the BioModels database [28] and the Epo mediated ERK signaling pathway from [7]. For all these models, there were no unknown parameters and this enabled us to focus on the main issue of evaluating the performance of HFF. The DBNs were constructed using the method presented in Section 2 [1]. To improve the quality of the approximations for the large pathway models, we constructed the DBNs using the *equation based subinterval sampling* method explained in more detail below. In what follows, we highlight the main findings of our experiments. More details can be found in the Supplementary Materials[25].

4.1. Enzyme catalytic kinetics

For initial validation, we started with the enzyme catalytic reaction network shown in Figure 1 which has only 4 species/equations and 3 rate constants. The value space of each variable was divided into 5 equally wide intervals ($\{[0,1),[1,2),\ldots,[4,5]\}$). We assumed the initial distributions of variables to be uniform over certain intervals. We then fixed the time horizon of interest to be 10 minutes and divided this interval evenly into $[0,1,\ldots,100]$ time points. The conditional probability tables associated with each node of the DBN were filled by generating 10^6 trajectories by direct random sampling over the initial states [1].

This being a small example, we could compute the marginal distributions for each species exactly. We ran FF and HFF(σ) with various choices of σ, the number of spikes. The resulting estimates were then compared against the exact marginals. We also ran the fully factored version of BK (which we call BK in this section), using the implementation provided in the Bayes Net Toolbox of MATLAB [29].

In what follows we report the errors in terms of the absolute difference between the marginal probabilities computed by the exact and approximate methods. Thus if we say the error is 0.15 then this means that the actual marginal probability was p and the marginal probability computed by the approximate algorithm was p' with $|p-p'| = 0.15$.

Even for this small network, FF and BK deviated from some of the exact marginals by as much as 0.169. Figure 2 shows the profile of the marginal distribution of E (the enzyme) assuming a value in the first interval as computed by FF, BK, HFF(64) and the exact method. The profiles of exact and HFF(64) were almost the same while FF and BK (whose curve practically coincides with that of FF and is hence not shown) make noticeable errors. The computation times for all the algorithms were negligible. The maximum error incurred for the 4 species taken over all the interval values and all time points was 0.169 for FF and 0.024 for HFF(16) and 0.003 for HFF(64). Further, the number of errors greater than 0.1 taken over all the species, intervals and time points reduced from 72 for FF to 0 for HFF(16).

4.2. The large pathway models

As explained in Section 2, during the construction of the DBN we assume that the initial values are distributed along certain predefined intervals of a variable's value space. The vector of initial states for large systems will hence be high dimensional. To ensure that the ODE dynamics is well explored, one needs to draw a large number of representative trajectories. Naive direct sampling where we randomly pick values from the initial intervals vector cannot ensure that all parts of the initial states region are sufficiently probed. Hence we used a more sophisticated sampling method called *equation based subinterval sampling* which is a variant of the method proposed in [1]. Suppose the ODE equation for the variable x_i involves variables x_j and x_k. We then subdivide the initial intervals of the variables x_i, x_j and x_k into J finer subintervals. Then for *every combination of subintervals* say, (I_i, I_j, I_k), we pick H samples each of which will have its x_i-value falling in I_i, x_j-value falling in I_j and its x_k-value falling in I_k while the values for the other variables are picked randomly from within their initial intervals. This ensures a coverage of at least H samples for every combination of the subintervals of the variables governing each equation which in turn ensures that ODE dynamics is being explored systematically along each dimension at least. In general, if an equation has R variables on its right hand side, and there are n equations and H is the required degree of coverage per equation, we pick $n \cdot H \cdot J^{R+1}$ samples.

To assess the quality of the constructed DBNs in terms of the original ODE dynamics, we used Monte Carlo integration to generate random trajectories from the prior (initial states distribution) using the ODE. We then computed the average values of each variable at the time points $0 \leq t \leq T$. We term the resulting time series for each variable as a *nominal profile*. We then used marginal probability values derived from the DBN approximation to compute expected values as follows $E(M^t(i,u)) = \sum_{u=u_j}(M^t(i,u_j) \cdot L)$, where L is the mid-point of the interval u_j. For each variable, the resulting time series of expected values was compared with its nominal profile. For all the models studied below the quality of the DBN approximation measured this way was high. Due to space limitations, the comparison plots will be shown in what follows here only for a few chosen species in the case of the NGF stimulated EGF-NGF pathway and the Epo mediated ERK pathway. The detailed comparison plots for other DBNs can be found in [25].

Finally, for the DBNs arising from EGF-NGF pathway and Epo mediated ERK pathway, exact inference is infeasible due to the large sizes of the corresponding DBNs. To get around this, we used simulation based inferencing of the DBN to obtain an estimate of the exact marginal distribution. We generated around 200 million trajectories from the underlying DBN to obtain the various marginal probabilities. This took around 2 days for each DBN. These marginals were used -in place of exact marginals- as benchmarks to compare the performance of the various algorithms. Here again we compared HFF(σ) for various choices of σ with FF and BK. We discuss towards the end of this section the performance of the clustered version of BK. In what follows, we write HFF(cK) to mean the HFF(σ) with $\sigma = c \cdot 1000$.

4.2.1. The EGF-NGF pathway

The EGF-NGF pathway describes the behavior of PC12 cells under multiple stimulations. In response to EGF stimulation they proliferate but differentiate into sympathetic neurons in response to NGF stimulation. This phenomenon has been intensively studied [30] and the network structure of this pathway is as shown in Figure 3. The ODE model of this pathway [28] consists of 32 differential equations and 48 associated rate constants (estimated from multiple sets of experimental data as reported in [28]).

To construct the three DBNs arising out of EGF, NGF and co-stimulation, we divided as before the value domains of the variables into 5 equally wide intervals and assumed the initial distributions to be uniformly distributed over some of these intervals. The time horizon of each model was set at 10 minutes which was evenly divided into 100

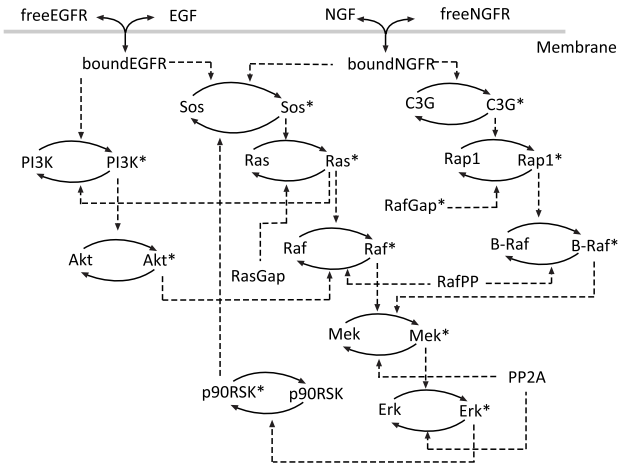

Figure 3: EGF-NGF pathway

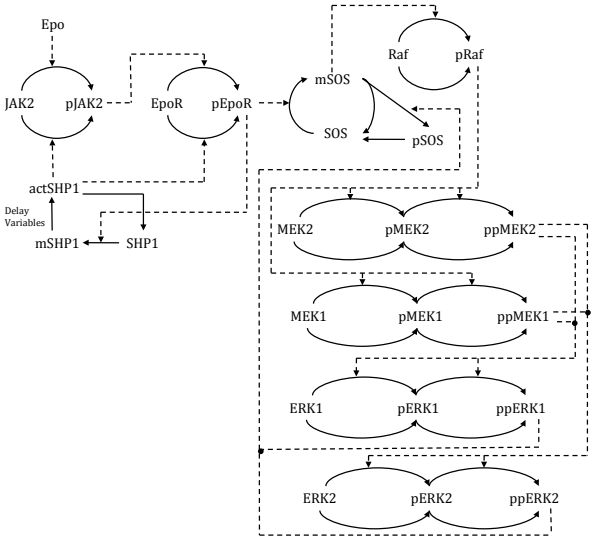

Figure 4: Epo mediated ERK Signaling pathway

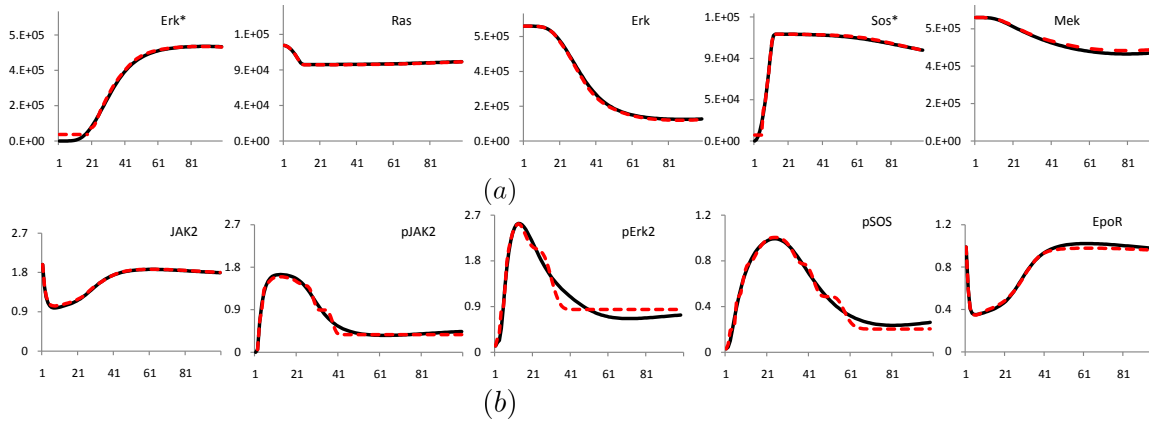

Figure 5: Comparison of ODE dynamics with DBN approximation. Solid black line represents nominal ODE profiles and dashed red lines represent the DBN simulation profiles for (a) NGF stimulated EGF-NGF Pathway (b) Epo mediated ERK pathway

time points. To fill up the conditional probability tables, we used the equation based subinterval sampling. We subdivided each of the initial states into 4 subintervals. 2.1 million trajectories were generated to get a coverage of 500 per combination of the subinterval. As shown in Figure 5(a), the quality of the approximations relative to the original ODE dynamics was high. Once we had the DBNs, we ran FF, BK and HFF(σ) for various choices of σ.

For the DBN obtained for the pathway under NGF-stimulation, for 6 of the 32 species there were significant differences between FF and BK on one hand and HFF on the other, including some biologically important proteins such as *Sos* and *Erk*. In Figure 6, we show for *Erk*, the marginal probability of the concentration falling in the interval $[1, 2)$ at various time points as computed by FF, BK, HFF($3K$) and HFF($32K$) as well as the pseudo-exact marginals obtained via massive Monte Carlo simulations. We observe that HFF tends to the exact values as the number of spikes increases.

To measure the overall error behavior, noting that HFF always did better than FF, we fixed the error incurred by FF as the base (100%) and normalized all other errors relative to this base. Under this regime, the relationship between computation time and normalized mean error for *Erk*'s value to fall in $[1, 2)$ is shown in Figure 7. We observe that the mean error reduces to 22% for HFF($32K$) at the cost of approximately 10^4 seconds increase in running time. For HFF(σ) the errors did not decrease linearly as the number of spikes were increased. This is to be expected since the

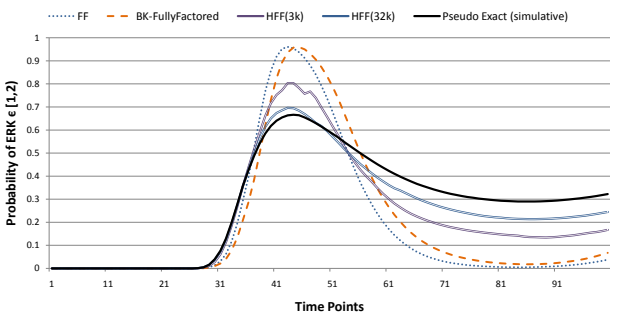

Figure 6: $M^t(Erk \in [1,2))$ under NGF-stimulation

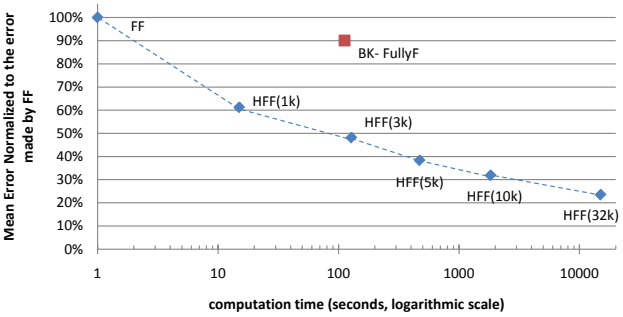

Figure 7: Normalized mean error for $M^t(Erk \in [1,2))$ under NGF-stimulation.

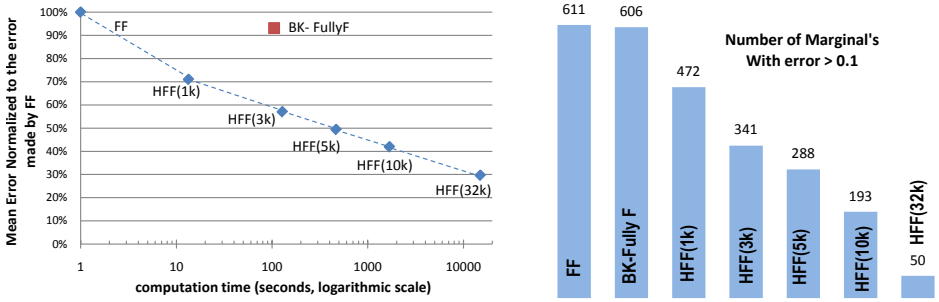

Figure 8: (a) Normalized mean errors over all marginals, (b) Number of marginals with error greater than 0.1: NGF-stimulation

probability mass captured by the additional spikes will be less than what is captured by the initial spikes. Overall, the maximum error over *all* the marginals ($32 \times 5 \times 100 = 16000$ data points) reduced from 0.42 for FF to 0.3 for HFF($3K$) and to 0.12 for HFF($32K$). The normalized mean error over all marginals went down to 60% for HFF($3K$) and 30% for HFF($32K$) as shown in Figure 8(a) which also displays the corresponding computation times. Further, when we computed the *number of marginals* with errors greater than 0.1, we found that this number reduced to about half for HFF($3K$) and by more than a factor of 10 for HFF($32K$) compared to FF as shown in Figure 8(b).

For the DBN obtained for the pathway under EGF-stimulation we found similar results. Overall, the maximum error over *all* the marginals reduced from 0.35 for FF to 0.14 for HFF($3K$) and to 0.07 for HFF($32K$). The normalized mean error over all marginals went down to 40% for HFF($3K$) spikes and 20% for HFF($32K$) spikes as shown in Figure 9(a) which also displays the corresponding computation times. Further, when we computed the number of marginals with errors greater than 0.1, we found that this number reduced by more than a factor of 4 for HFF($3K$) and to 0 for HFF($32K$) as shown in Figure 9(b). Similar results were obtained for the DBN describing the dynamics of the EGF-NGF pathway under co-stimulation of both NGF and EGF.

4.2.2. The Epo mediated ERK pathway

Next we considered the DBN model of Epo mediated ERK Signaling pathway as shown in Figure 4. *Erk* and its related kinase isoforms play a crucial role in cell proliferation, differentiation and survival. This pathway describes the effect of these isoforms on the Epo (cytokine) induced ERK cascade. The ODE model of this pathway [7] consists of 32 differential equations and 24 associated rate constants. To construct the DBN, we divided the value domain of variables into 5 intervals. Here the interval sizes for variables were not all kept equal. For 23 species that have very low basal concentration level, we set the first interval of the corresponding variables to be smaller ($\sim 20\%$) compared to the other 4 intervals (equal sized) (See Supplementary Materials [25]). The rest 9 variables all have equal sized intervals as before. Time horizon was fixed at 60 minutes which was then divided into 100 time points. We constructed the DBN using equation based subinterval sampling. As Figure 5(b) indicates, the quality of the approximation relative to the original ODE dynamics was again high (more comparison plots can be found in [25]).

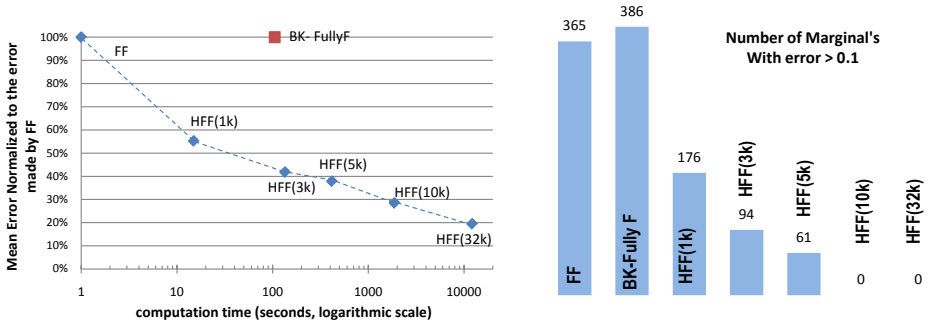

Figure 9: (a) Normalized mean error mean error over all marginals (b) Number of marginals with error greater than 0.1: EGF- stimulation

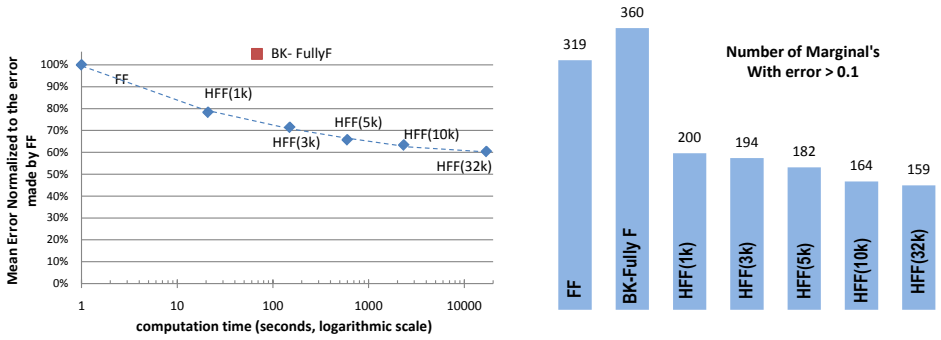

Figure 10: (a) Normalized mean errors over all marginals, (b) Number of marginals with error greater than 0.1: Epo stimulated ERK pathway

We then ran FF, BK and HFF(σ) for various choices of σ.

FF and BK were quite accurate for many of the species. However, for some species such as *JAK2*, phosphorylated *JAK2, EpoR, SHP1* and *mSHP1* etc. which are biologically relevant, FF and BK incurred a max error of 0.49. On the other hand, HFF$(3K)$ incurred a max error of 0.45 while HFF$(32K)$ incurred a max error of 0.31. The normalized mean error over all marginals went down to $\sim 70\%$ for HFF$(3K)$ and $\sim 60\%$ for HFF$(32K)$ as shown in Figure 10(a). Further, when we computed the number of marginals with errors greater than 0.1, we found that this number reduced by around half for HFF$(32K)$ compared to FF as shown in Figure 10(b).

It is worth noting that our current implementation is quite crude and sequential. We believe significant performance gains can be expected from an optimized version.

4.3. Comparison with clustered BK

An important component of the BK algorithm is the grouping of the variables into clusters. The idea is to choose the clusters in such a way that there is not much interaction between variables belonging to two different clusters. When this is done well, BK can also perform well. However, choosing the right clusters seems to be a difficult task. The easy option, namely, the fully factored BK in which each cluster is a singleton performs in our case studies as badly (or well) as FF.

We tried to gain a better understanding of BK augmented with non-trivial clusters by using the structure of the pathway to come up with good clusters. A natural way to form 2-clusters seemed to be to pair together the activated (phosphorylated) and inactivated (dephosphorylated) counterparts of a species in the pathway. For the EGF-NGF pathway, this clustering indeed reduced overall errors compared to FF and HFF($3K$). However, we found that HFF(σ) with $\sigma > 5000$ outperformed this version of BK. We did not consider bigger clusters for two reasons: first, when we tried to increase the sizes and the number of clusters in different ways, BK ran out of the 3GB memory. Second, there seemed to be no biological criterion using which one could improve the error performance of BK.

For the Epo mediated ERK pathway too we tried similar clustering. Here the natural clusters were of size 3. Unfortunately, the results were as bad as for fully factored BK. HFF, even with $1K$ spikes ($\sigma = 1000$) was able to

perform better than this clustered version of BK. This suggests that the clusters we chose were not the right ones. Hence in our setting, a clustered version of BK that performs well in terms of the computational resources required and the errors incurred appears to be difficult to realize.

5. Discussion

DBNs are a widely used class of probabilistic graphical models and can often be a succinct and more natural representation of the underlying Markov chains. Computing the probability distribution over the global states of a DBN is a basic analysis task. But this can be performed only approximately for high dimensional systems. FF and BK are two attractive approximate algorithms that have been proposed in this context. However they can incur significant errors. To cope with this, we have developed here the Hybrid Factored Frontier (HFF) algorithm. In HFF, in addition to maintaining and propagating belief states in a factored form, we also maintain a small number of full dimensional state vectors called spikes and their probabilities at each time slice. By tuning the number of spikes, one can gain accuracy at the cost of increased but polynomial (quadratic) computational cost. We have provided an error analysis for HFF as well as FF which shows that HFF is more accurate. Our experimental results confirm that HFF outperforms both FF and fully factored BK in realistic biological settings.

One may consider BK also to be a parametrized algorithm with the number of clusters and their sizes constituting the parameters. However identifying the clusters is a difficult problem and our experimental results suggest that as the sizes of the clusters increase the errors may reduce but the memory consumption could raise rapidly. In contrast, HFF's parameter can be computed in an efficient, approximate and *automated* fashion. In our case studies we have found that by using HFF, the accuracy of results can be improved with an increase in computational times. Further, for tasks such as parameter estimation that require repeated executions, one can first deploy FF to get an initial estimate and then run HFF with a suitable number of spikes just once to achieve a sufficiently accurate estimate.

In our future work, an important goal will be to deploy HFF to perform tasks such as parameter estimation and sensitivity analysis. An equally important goal will be to develop approximate probabilistic verification methods for DBN models of biochemical networks and evaluate them with respect to approaches developed in related settings [21, 19, 22, 23].

References

[1] B. Liu, D. Hsu, P. S. Thiagarajan, Probabilistic approximations of ODEs based bio-pathway dynamics, Theoretical Computer Science 412 (2011) 2188–2206.

[2] K. P. Murphy, Y. Weiss, The factored frontier algorithm for approximate inference in DBNs, in: Proceedings of the 17^{th} Annual Conference on Uncertainty in Artificial Intelligence (UAI), San Francisco, CA, USA, 2001, pp. 378–385.

[3] X. Boyen, D. Koller, Tractable inference for complex stochastic processes, in: Proceedings of the 14^{th} Annual Conference on Uncertainty in Artificial Intelligence (UAI-98), Madison, Wisconsin, USA, 1998, pp. 33–42.

[4] B. Liu, J. Zhang, P. Y. Tan, D. Hsu, A. M. Blom, B. Leong, S. Sethi, B. Ho, J. L. Ding, P. S. Thiagarajan, A computational and experimental study of the regulatory mechanisms of the complement system, PLoS Computational Biology 7 (1) (2011) e1001059.

[5] D. Koller, N. Friedman, Probabilistic Graphical Models - Principles and Techniques, MIT Press, 2009.

[6] K. S. Brown, C. C. Hill, G. A. Calero, C. R. Myers, K. H. Lee, R. A. Cerione, The statistical mechanics of complex signaling networks : nerve growth factor signaling, Physical Biology 1 (2004) 184–195.

[7] M. Schilling, T. Maiwald, S. Hengl, D. Winter, C. Kreutz, W. Kolch, W. D. Lehmann, J. Timmer, U. Klingmller, Theoretical and experimental analysis links isoform- specific ERK signalling to cell fate decisions, Molecular Systems Biology (2009) 5:334.

[8] P. F. Felzenszwalb, D. P. Huttenlocher, Efficient belief propagation for early vision, International Journal of Computer Vision 70 (2006) 41–54.

[9] R. J. Mceliece, D. J. C. Mackay, J.fu Cheng, Turbo decoding as an instance of Pearl's "Belief Propagation" algorithm, IEEE Journal on Selected Areas in Communications 16 (1998) 140–152.

[10] N. Friedman, Inferring cellular networks using probabilistic graphical models, Science 303 (2004) 799–805.

[11] B. Bidyuk, R. Dechter, An anytime scheme for bounding posterior beliefs, in: National Conference on Artificial Intelligence, 2006.

[12] J. Bilmes, H. Lin, Online adaptive learning for speech recognition decoding, in: Annual Conference of the International Speech Communication Association, 2010, pp. 1958–1961.

[13] M. Z. Kwiatkowska, G. Norman, D. Parker, PRISM: Probabilistic symbolic model checker, in: Computer Performance Evaluation, 2002, pp. 200–204.

[14] M. Calder, V. Vyshemirsky, D. Gilbert, R. J. Orton, Analysis of signalling pathways using continuous time Markov chains, Transactions on Computational Systems Biology (2006) 44–67.

[15] W. S. Hlavacek, J. R. Faeder, M. L. Blinov, R. G. Posner, M. Hucka, W. Fontana, Rules for modeling signal-transduction systems, Science STKE 2006 (2006) re6.

[16] V. Danos, J. Feret, W. Fontana, R. Harmer, J. Krivine, Rule-based modelling of cellular signalling, in: International Conference on Concurrency Theory, 2007, pp. 17–41.
[17] B. Liu, P. S. Thiagarajan, D. Hsu, Probabilistic approximations of signaling pathway dynamics, in: Computational Methods in Systems Biology, 2009, pp. 251–265.
[18] T. A. Henzinger, M. Mateescu, V. Wolf, Sliding window abstraction for infinite Markov chains, in: Computer Aided Verification, 2009, pp. 337–352.
[19] S. K. Jha, E. M. Clarke, C. J. Langmead, A. Legay, A. Platzer, P. Zuliani, A Bayesian approach to model checking biological systems, in: Computational Methods in Systems Biology, 2009, pp. 218–234.
[20] M. Z. Kwiatkowska, G. Norman, D. Parker, Symbolic Systems Biology, Jones and Bartlett, 2010, Ch. Probabilistic Model Checking for Systems Biology.
[21] R. Grosu, S. A. Smolka, Monte carlo model checking, in: Tools and Algorithms for Construction and Analysis of Systems, 2005, pp. 271–286.
[22] F. Fages, A. Rizk, On the analysis of numerical data time series in temporal logic, in: Computational Methods in Systems Biology, 2007, pp. 48–63.
[23] R. Donaldson, D. Gilbert, A model checking approach to the parameter estimation of biochemical pathways, in: Computational Methods in Systems Biology, 2008, pp. 269–287.
[24] F. Didier, T. A. Henzinger, M. Mateescu, V. Wolf, Approximation of event probabilities in noisy cellular processes, Theoretical Computer Science 412 (2011) 2128–2141.
[25] Supplementary materials, available at: `http://www.comp.nus.edu.sg/~suchee/HFF/` (2011).
[26] S. K. Palaniappan, S. Akshay, B. Genest, P. S. Thiagarajan, A Hybrid Factored Frontier Algorithm for Dynamic Bayesian Network Models of Biopathways, to appear - Computational Methods in Systems Biology 2011 (2011).
[27] P. Bremaud, Markov chains: Gibbs fields, Monte Carlo simulation and queues, Springer, 1999.
[28] N. L. Novre, B. J. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, L. Li, H. M. Sauro, M. J. Schilstra, B. E. Shapiro, J. L. Snoep, M. Hucka, BioModels Database: a free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems, Nucleic Acids Research 34 (2006) 689–691.
[29] K. P. Murphy, Bayes Net Toolbox for Matlab, http://bnt.googlecode.com.
[30] B. N. Kholodenko, Untangling the signalling wires, Nature Cell Biology 9 (2007) 247–249.