A Hybrid Factored Frontier Algorithm for Dynamic Bayesian Network Models of Biopathways

Sucheendra K. Palaniappan
School of Computing
National University of Singapore
suchee@comp.nus.edu.sg

S. Akshay
School of Computing
National University of Singapore
akshay@comp.nus.edu.sg

Blaise Genest
CNRS, IPAL UMI
joint with NUS-I2R-A*STAR, Singapore
bgenest@irisa.fr

P. S. Thiagarajan
School of Computing
National University of Singapore
thiagu@comp.nus.edu.sg

ABSTRACT

Dynamic Bayesian Networks (DBNs) can serve as succinct models of large biochemical networks [19]. 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 approximate methods are needed. The Factored Frontier algorithm (FF) is a simple and efficient approximate algorithm [25] that has been designed to meet this need. However the errors it incurs can be quite large. The earlier Boyen-Koller (BK) algorithm [3] can also incur significant errors.

To address this, we present here a novel approximation algorithm called the Hybrid Factored Frontier (HFF) algorithm. HFF may be viewed as a parametrized version of FF. At each time slice, in addition to maintaining probability distributions over local states -as FF does- we also maintain explicitly 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 -computationally infeasible- 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 can be a useful and powerful approximation algorithm for analyzing DBN models of biopathways.

Categories and Subject Descriptors

G.3 [**Mathematics of Computing**]: Probability and Statistics; I.1.2 [**Computing Methodologies**]: Symbolic and Algebraic Manipulation—*Algorithms*; J.3.1 [**Computer Applications**]: Life and Medical Sciences—*Biology and Genetics*

CMSB '11, September 21-23, 2011 Paris, FR

1. INTRODUCTION

The probabilistic graphical model known as Dynamic Bayesian Network (DBN) was used in [19] to compactly approximate the dynamics of a biochemical network originally given as a system of (deterministic) Ordinary Differential Equations (ODEs). 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 then studies the dynamics of the biochemical network by applying Bayesian inferencing techniques to the DBN approximation. This approach has been developed extensively in [19], it scales well and has been used to aid biological studies [21].

The random variables associated with each discretized time point t in the DBN are used to represent the discretized concentrations of the proteins of the biochemical network at time t. Tasks such as parameter estimation and sensitivity analysis can be carried out directly on the DBN by repeatedly computing the probability of a random variable assuming a specific value at a given time point. Due to conditional dependencies between the variables, carrying out this computation exactly will require -roughly speaking- an effort which is exponential in the number of species in the biochemical network. Hence for large networks, exact computation will be infeasible and one must resort to approximate methods. An efficient approximate inferencing technique called the Factored Frontier algorithm (FF, for short) developed for DBNs [16, 25] was used to perform these computations in [19, 21].

The crucial role played by the FF algorithm and the fact that it was an approximate 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 all the models we considered (as detailed later), though 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) that we present here. As the name suggests, HFF is based on FF and is a parametrized version of FF. To bring out its main features, it will be convenient to briefly describe how FF works in our DBN setting.

The DBNs we consider have 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 a few other variables (called its parents) at time $t-1$. Further, the probabilistic dynamics is captured by a Conditional Probability Table (CPT) associated with each variable at each time point (see fig. 1.c for an example). Thus, 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 the joint probability distribution over global states at time t. This can be computed from the joint probability distribution over the global states at time $t-1$ by propagating it through the CPTs. FF maintains and propagates these joint probability distributions approximately. Such approximate joint probability distributions are usually called belief states.

FF is a simplified and more efficient version of the earlier Boyen-Koller algorithm [3] (BK, for short). 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 probability distributions of the individual variables usually called marginal distributions. 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.

In our context, both BK and FF have drawbacks. FF is attractive in terms of simplicity and computational effort 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 with guaranteed performance. 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 also leads to significant errors.

The main idea of our new algorithm, HFF, is to maintain the 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 spikes are chosen to be those global states which have high probabilities currently. This is based on the insight that when the error produced by one step of FF is large for a global state, then actual probability of this state must itself be high. Further, since the total probability of the spikes is bounded by 1, the number of spikes at any time point must be small. For instance, there can at most be 2 spikes each with probability greater than 0.4. However, it is infeasible to explicitly identify and exactly compute the probabilities of the spikes. HFF does this approximately, as well as propagating the hybrid belief state through the CPTs. The main goal is to keep the one step error low since the error analysis for BK [3] suggests that this is the key to minimizing the overall error.

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 brute-force 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 in the DBN. This compares well with the time complexity of FF which is $O(n \cdot K^{D+1})$. Since the time complexity of HFF (and FF) is *linear* in n, our algorithm 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 (≈ 4) is much smaller than n (≈ 30). 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 [4] 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 Epor mediated ERK signaling pathway [22]. Each of these pathways contained more than 30 species and we traced their behaviors through 100 time points. Starting from their deterministic ODEs based models (with all rate constants and initial concentrations known) we applied the technique developed in [19] 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.

Turning to related work, DBNs are used extensively in domains such as as AI, computer vision, signal processing and computational biology [10, 11, 16, 23]. 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 [1] and [2]. The main idea of [1] is to use stochastic sampling methods to look for cut-sets in the graph structure that have high probability mass. The approach of [2] 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 [5, 6, 13, 14, 17, 18, 20, 28]. 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 ODEs based models. As might be expected, the exact inferencing problem for large CTMCs is computationally infeasible. Analysis

methods based on Monte Carlo simulations [8, 9, 12, 14] as well as numerically solving the Chemical Master Equation describing a CTMC [7, 13] are being developed. In these studies the CTMCs are presented implicitly while our DBNs are available explicitly. Hence it is not clear at present whether inferencing algorithms such as FF,BK and HFF can be deployed to analyze CTMCs arising as populations-based models of biochemical networks.

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. Many of the technical details can be found in [27].

2. DBN MODELS OF BIOPATHWAYS

We first introduce DBNs and related notations. We shall then describe how DBNs arise as models of biochemical networks through the approximation technique reported in [19].

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 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 [25]. 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 $\{1, 2, \ldots, n\}$.
- C_i^t is the *Conditional Probability Table (CPT)* associated with node X_i^t specifying the probabilities $P(X_i^t \mid Pa(X_i^t))$. 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 $t-1$ such that $X_{j_l} = x_{j_l}$ 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 which will then induce in a canonical way the global dynamics as a Markov chain [16].

We introduce some notations for later use. To start with, 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 $(X_1^t, \ldots, X_n^t)$.

Thus using these notations, we can write $C_i^t(x_i \mid \mathbf{u}_{\hat{i}}) = p$ to mean that p is the probability of $X_i = x_i$ 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:

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

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

Since $\mid V \mid = K$, the number of possible states at t is K^n. Hence explicitly computing and maintaining the probability distributions is feasible only when 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 ways of doing this, we first 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 a pathway can be often modeled as a system of deterministic ODEs; 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, $\mathbf{y}$ being the molecules taking part in these reactions and $\mathbf{r}$ denoting the rate constants (parameters) associated with these reactions. For large pathways, this system of ODEs 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 will have to be estimated. This in turn will involve searching through high dimensional spaces 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. Guided by these considerations, we have developed a method for deriving a dynamic Bayesian network from a system of ODEs that models a biopathway [19, 20]. Its main features are illustrated by the example shown in Figure 1.

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 steady state value. Hence we assume the dy-

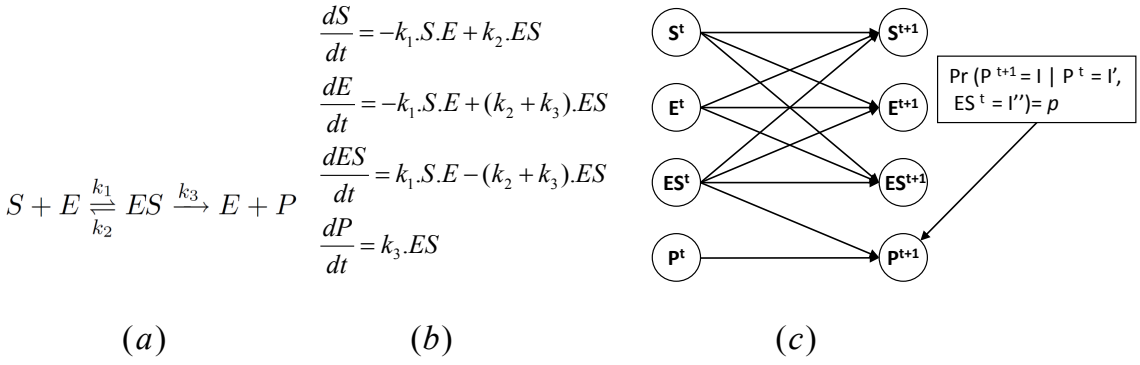

Figure 1: a) The enzyme catalytic reaction network (b) The ODEs model (c) The DBN approximation for 2 successive time slices

namics 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 [19] 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 that the minimum and maximum values of these constants 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 [19]). 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 this 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 the parameter values don't change once their initial values have been fixed. (Hence 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 gains obtained in doing parameter estimation and sensitivity analysis [19] and 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 [21] where the starting point was a ODEs based model with 71 unknown parameters.

In these studies FF was used as the core inferencing algorithm. It was then we began to notice its shortcomings and started to work toward an improved version of FF.

3. THE HYBRID FACTORED FRONTIER ALGORITHM

In this section, we present the HFF algorithm. As it is a parametrized version of FF, it will be convenient to begin with a brief formal description of FF. Throughout this section we will assume the DBN notations developed in Section 2.

As mentioned earlier, approximate probability distributions will be called belief states. They will be denoted by B, B^t etc. while 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 it is just a probability distribution over V^n but it will be convenient to linguistically distinguish between belief states and probability distributions.

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 $M_B(i, v) = \sum_{\mathbf{u} \mid \mathbf{u}_i = v} B(\mathbf{u})$. 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. However, when the variables are all mutually independent, we will have $B_{M_B} = B$.

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

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

Thus FF maintains B^t, the belief state at t, compactly via the marginal function M^t. More precisely, $B^t(\mathbf{u}) = \prod_j M^t(j, \mathbf{u}_j) = B_{M^t}$ for every $\mathbf{u}$.

Now suppose the DBN transforms the belief state B^{t-1} into the new belief state $\widehat{B}^t$. It is easy to show that $\widehat{B}^t$ is given by:

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

However, FF computes only the marginal function $M^t = M_{\widehat{B}^t}$, which then abstractly represents the new belief state $B^t = B_{M^t}$. This belief state B^t represented via M^t is an approximation of the required belief state $\widehat{B}^t$.

One can show that if B^{t-1} is accurate then M^t as computed by FF will also be accurate. More precisely, if $B^{t-1} = P^{t-1}$ then $M^t = M_{P^t}$ (see Prop.1 in [27]). We note that B^0 is accurate by definition and hence M^1 will also be accurate but not necessarily B^1. Due to $B^t = B_{M_{\widehat{B}^t}}$, the one step error ϵ_t incurred by FF at step t is bounded by:

$$max_{\mathbf{u} \in V^n} \{ |\widehat{B}^t(\mathbf{u}) - B_{M_{\widehat{B}^t}}(\mathbf{u})| \}$$

The simple but crucial observation is whenever ϵ_t is large for some $\mathbf{u}$ then $M^t(i, \mathbf{u}_i)$ is large for *every* i. This is so since $M^t(j, \mathbf{u}_j) = M_{\widehat{B}^t}(j, \mathbf{u}_j) \geq \max(\widehat{B}^t(\mathbf{u}), B^t(\mathbf{u}))$. The 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, if we can record $B^t(\mathbf{u})$ explicitly for a small subset of V^n for which M^t is high for all dimensions then one can hope to improve FF in terms of its error behavior. This is the intuition underlying the HFF algorithm.

3.1 The Hybrid Factored Frontier algorithm

As observed above, if during the execution of FF one can record $B^t(\mathbf{v})$ explicitly for members of V^n for which M^t is high for all dimensions (and there can only be a few such) then one can hope to reduce the errors incurred by FF. Unfortunately, this cannot be done exactly since it would involve an exhaustive search through V^n. Hence, we must do this approximately.

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

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

We next define the marginal function $M_H^t(i,v)$ for all i and v via:

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

It is easy to observe that this is indeed a marginal function. We next define (but not compute!) the belief state B^t represented by $(M^t, S^t, B_H^t, \alpha^t)$ as follows:

$$B^t(\mathbf{u}) = B_H^t(\mathbf{u}) + (1-\alpha^t) \prod_i M_H^t(i, \mathbf{u}_i)$$

We need to use M_H^t rather than M^t since the cumulative weight of the contribution made by the spikes needs to be discounted from M^t. It is easy to check that B^t is a belief state.

3.1.1 The HFF algorithm

We initialize with $M^0 = C^0$, $S^0 = \emptyset$, $B_H^0 = \mathbf{0}$ 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 $(M^{t+1}, S^{t+1}, B_H^{t+1}, \alpha^{t+1})$ from $(M^t, S^t, B_H^t, \alpha^t)$ as follows.

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

$$M^{t+1}(i,v) = \sum_{\mathbf{u} \in S^t} [C_i^{t+1}(x \mid \mathbf{u}_{\hat{i}}) \times B_H^t(\mathbf{u})]$$

$$+(1-\alpha^t) \Big(\sum_{\mathbf{u}_{\hat{i}}} [C_i^{t+1}(x \mid \mathbf{u}_{\hat{i}}) \times \prod_{j \in \hat{i}} B_H^t(j, \mathbf{u}_j)] \Big)$$

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 and 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.

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

It is not difficult to establish the following properties of our algorithm. We refer the reader to the full paper [27] for the proof.

PROPOSITION 1. *Let D be the maximum in-degree of the DBN graph. Then, recalling that T is the number of time points, σ is the number of spikes, n is the number of variables, V is the set of values, $K = |V|$ is the number of values, we have,*

1. *if $\sigma = 0$, the HFF algorithm is the same as FF and if $\sigma = K^n$, it is the exact algorithm.*
2. *Suppose $P^t = B^t$. Then $P^{t+1}(X_i = v) = M^{t+1}(i, v)$ for every $i \in \{1, \ldots, n\}, v \in V$.*
3. *The time complexity of HFF is $O(T \cdot n \cdot (\sigma^2 + K^{D+1}))$.*

It is easy to see that the time complexity of FF is $O(n \cdot K^{D+1})$ and hence the additional computational effort required by HFF is $O(T \cdot n \cdot \sigma^2)$. HFF gathers in one sweep -just as FF does- the required information about the belief states. Hence, where repeated executions are required, such as for parameter estimation ([19]) one can initially run FF repeatedly to narrow down the range of possibilities and then run HFF once to get a more accurate estimate.

3.2 Error analysis

The overall error at time t, denoted Δ^t is given by $\Delta^t = max_{\mathbf{u} \in V^n}(|P(X^t = \mathbf{u}) - B^t(u)|)$. Using a reasoning similar to [3], this error can be bounded as: $\epsilon_0(\sum_{j=0}^{t} \beta^j)$, where $0 \leq \beta \leq 1$ is a constant determined by the stochastic transition matrix associated with the DBN, and ϵ_0 is an upper bound over the maximum one step error (different for FF, HFF, BK). Further, $\beta < 1$ under fairly mild restrictions placed on this matrix [3]. In this case we have $\sum_{j=0}^{t} \beta^j < 1/(1-\beta)$. We note that $\sum_{j=0}^{t} \beta^j$ depends only on the DBN. Hence, comparing FF and HFF theoretically amounts to comparing their single step errors ϵ_0. In fact, this remark holds for comparison with BK as well.

We then analyze the one-step error $\epsilon_t = max_{\mathbf{u} \in V^n}\{|\widehat{B}^t(\mathbf{u}) - B^t(\mathbf{u})|\}$ made by FF and HFF at step t. For both algorithm, we have $B^t = B_{M_{\widehat{B}^t}}$. For FF, we can bound ϵ_t 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. A technical analysis of ϵ_0 for FF shows that it will tend to 1 as n tends to ∞. The FF error can be large in practice too as we demonstrate in the next section. For HFF, the one step error is bounded by $\hat{\epsilon}_0$ with $\hat{\epsilon}_0 \leq \min\{(1-\alpha), \eta\}$, where $\alpha = \min_t(\alpha^t)$ and $\eta = \max_t(\eta^t)$. Thus, the worst case error for HFF is smaller than the one for FF (with at least 2 spikes). Also, taking more spikes makes α increase and η decrease, which reduces the worse case error. Experiments in the next section shows that the effective accuracy is also improved.

4. RESULTS

We have implemented our algorithm in C++. The experiments reported here were carried out on a Opteron 2.2Ghz Processor, with 3GB memory. We evaluated our algorithms on five DBN models of biochemical networks: the small enzyme catalytic reaction network shown in Figure 1 for initial experimentation, the EGF-NGF pathway [4] 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 [26] and the Epo mediated ERK signaling pathway from [22]. 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.

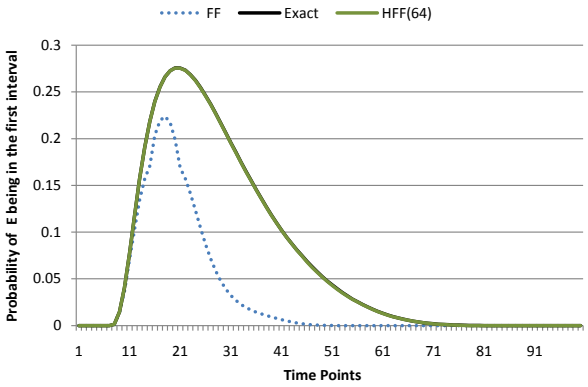

Figure 2: $M^t(E \in [0, 1))$

The DBNs were constructed using the method developed in [19] and explained in Section 3. As mentioned above, we used synthetic models with no unknown parameters and with no experimental data available for validation. Further, our focus was on evaluating the quality of HFF as an inferencing algorithm for DBNs regardless of how these DBNs arise. Hence we did not focus here on the orthogonal issue of validating the accuracy of the DBN approximation reactive to the original ODEs dynamics. The methods developed in [19] can be used to ensure that a large enough representative samples of trajectories is generated so that the resulting DBN is sufficiently accurate.

In what follows, we highlight the main findings of our experiments which compare the different DBN inferencing algorithms. All the details can be found in [27].

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.

We divided the value space of each variable and rate constant 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 up to 10 minutes by sampling the distribution over the initial states.

This being a small example, we could compute the probability distributions over the states for each time point exactly and hence derive the marginal distributions for each species also exactly. Then, we ran FF and HFF(σ) with various choices of σ where σ is 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 tool box of MATLAB [24].

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 if the actual marginal probability was p then the marginal probability computed by the approximate algorithm was p' with $\mid p - p' \mid = 0.15$.

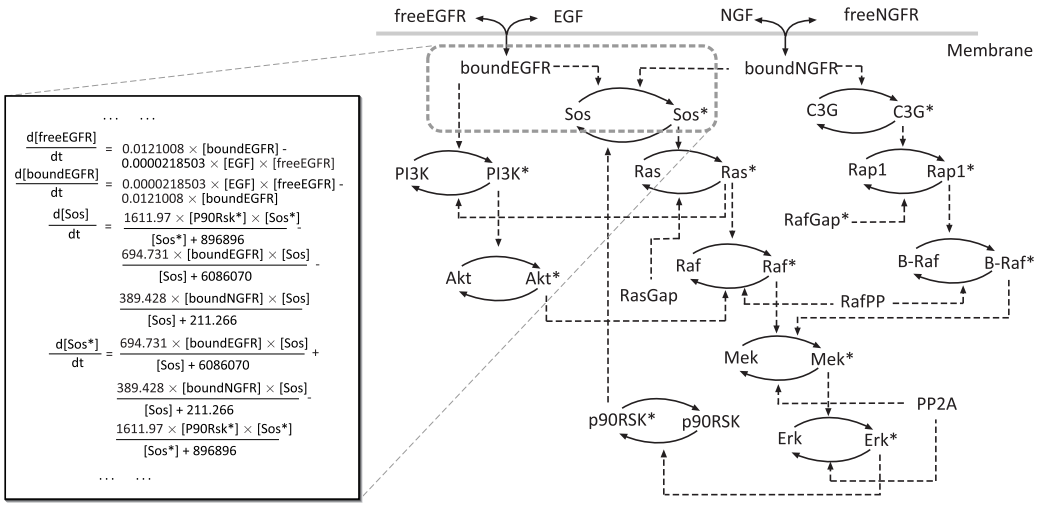

Figure 3: EGF-NGF pathway

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 to 0 for HFF(16).

4.2 The large pathway models

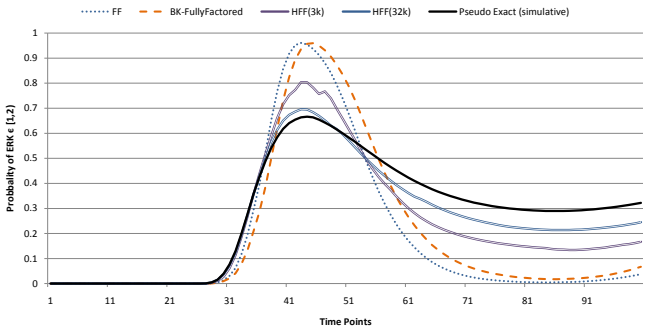

Figure 4: $M^t(Erk \in [1,2))$ **under NGF-stimulation**

For the remaining DBNs (arising from EGF-NGF pathway and Epo mediated ERK pathway), exact inference is infeasible due to the large sizes of the corresponding biochemical networks. To get around this, we used simulation based inferencing of the DBN to obtain an estimate of the exact marginal distribution against which we could compare the performance of the different algorithms. 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 then 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 [15] and the network structure of this pathway is as shown in Figure 3. The ODEs model of this pathway [26] consists of 32 differential equations and 48 associated rate constants (estimated from multiple sets of experimental data as reported in [26]).

To construct the three DBNs arising out of EGF, NGF and co-stimulation, we divided as before the value domains of the variables and rate constants 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 time points. To fill up the conditional probability tables, $5 \cdot 10^6$ trajectories were generated up to 10 mins by sampling the assumed initial distributions. Once we had the DBNs, we ran FF, BK and HFF(σ) for various choices of σ.

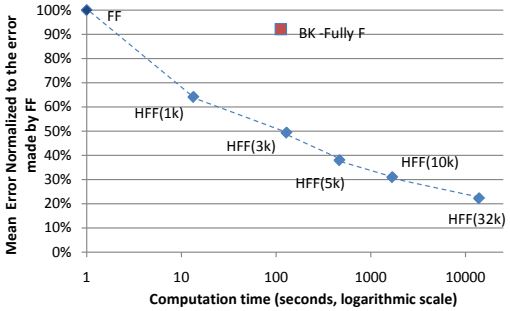

Figure 5: Normalized mean error for $M^t(Erk \in [1,2))$ **under NGF-stimulation.**

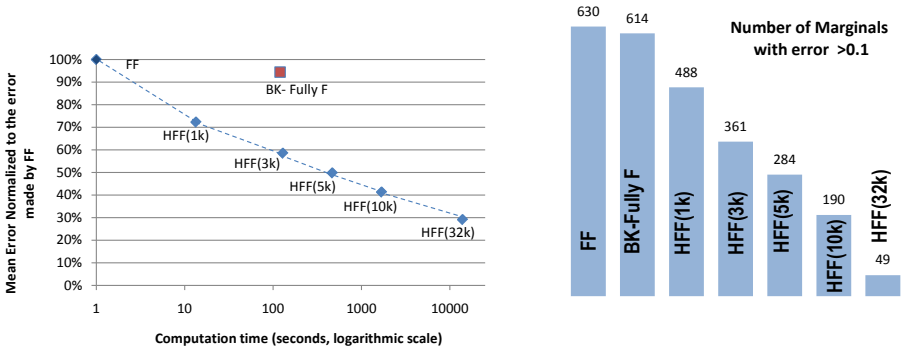

Figure 6: (a) Normalized mean errors over all marginals, (b) Number of marginals with error greater than 0.1: NGF-stimulation

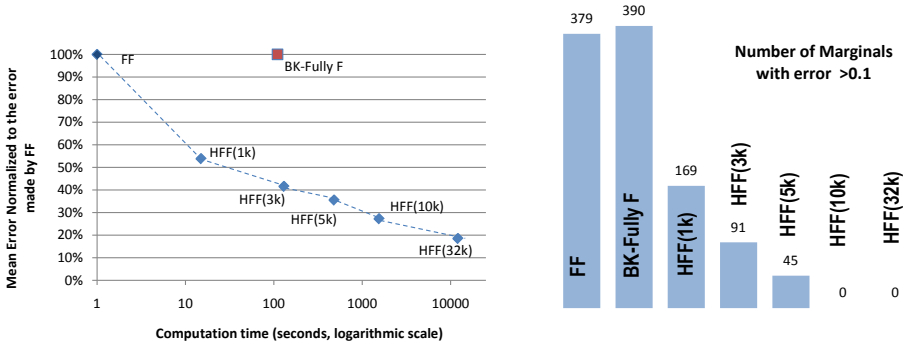

Figure 7: (a) Normalized mean error mean error over all marginals (b) Number of marginals with error greater than 0.1: EGF-stimulation

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 4, 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 we noted that HFF always did better than FF. Hence 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 5. We observe that the mean error reduces to 22% for HFF$(32K)$ at the cost of $\sim 10^4$ seconds increase in running time. Similar trends were observed for other marginals associated with the 6 species for which FF and BK incurred significant errors. For HFF(σ) the errors did not decrease linearly with as the number of spikes were increased. This is to be expected since the probability mass captured by the additional spikes will 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 6(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 half for HFF$(3K)$ and by more than a factor of 10 for HFF$(32K)$ as shown in Figure 6(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 0.36 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 7(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 7(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 below in Figure 8. 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 ODEs model of this pathway [22] consists of 32 differential equations and 24 associated rate constants. As before the value domains were divided into 5 intervals but the time horizon was fixed at 60 minutes which was then divided into 100 time points.

Again, we ran FF, BK and HFF(σ) for various choices of σ. FF and BK were quite accurate for many of the species. However, for *JAK2*, phosphorylated *JAK2, EpoR, SHP1* and *mSHP1* which

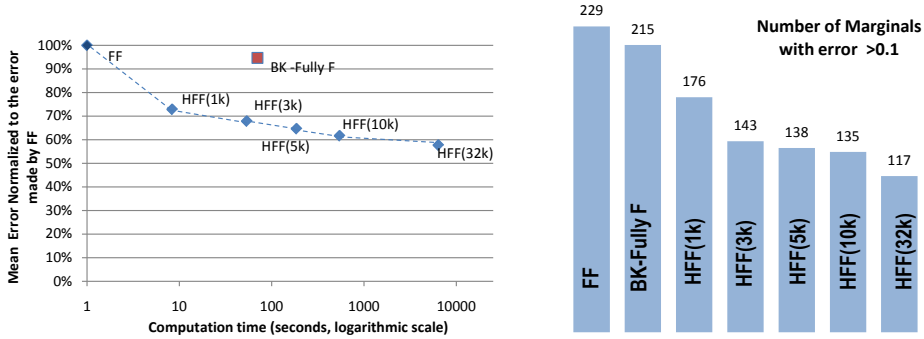

Figure 9: (a) Normalized mean errors over all marginals, (b) Number of marginals with error greater than 0.1: Epo stimulated ERK pathway

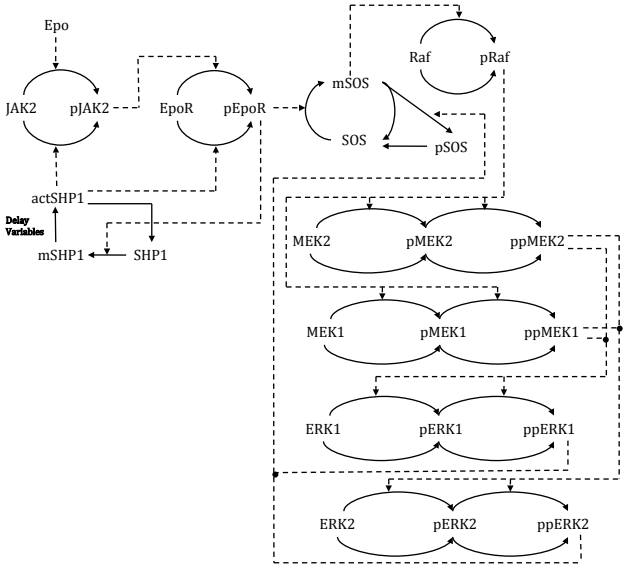

Figure 8: Epo mediated ERK Signaling pathway

are biologically relevant, FF and BK incurred a max error of 0.34. On the other hand, HFF($1K$) incurred a max error of 0.22 while HFF($32K$) incurred a max error of 0.18. The normalized mean error over all marginals went down to 70% for HFF($3K$) and 60% for HFF($32K$) as shown in Figure 9(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 9(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 the ones 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 an important class of probabilistic graphical models and can often be a succinct representation of the underlying Markov chains. Computing the probability distribution over the global states of the Markov chain underlying 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. Using this parameter, 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 biologically relevant practical 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 albeit with an increase in computational times. Further, for tasks such as parameter estimation that require repeated executions, one can first deploy FF to get good approximations and then run HFF with a suitable number of spikes just once to achieve the required error bounds.

In terms of 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 [8, 9, 12, 14].

6. ACKNOWLEDGEMENTS

The research reported here has been partially supported by the Singapore Ministry of Education grant T208A2104 and CNRS-PEPS AABS.

7. REFERENCES

[1] Bozhena Bidyuk and Rina Dechter. An anytime scheme for bounding posterior beliefs. In *AAAI*, 2006.

[2] Jeff Bilmes and Hui Lin. Online adaptive learning for speech recognition decoding. In *INTERSPEECH*, pages 1958–1961, 2010.

[3] Xavier Boyen and Daphne 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*, pages 33–42, 1998.

[4] K. S. Brown, C. C. Hill, G. A. Calero, K. H. Lee, J. P. Sethna, and R. A. Cerione. The statistical mechanics of complex signaling networks: nerve growth factor signaling. *Physical Biology 1*, pages 184–195, 2004.

[5] Muffy Calder, Vladislav Vyshemirsky, David Gilbert, and Richard J. Orton. Analysis of signalling pathways using continuous time markov chains. *T. Comp. Sys. Biology*, pages 44–67, 2006.

[6] Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Rule-based modelling of cellular signalling. In *CONCUR*, pages 17–41, 2007.

[7] F. Didier, Thomas A Henzinger, M. Mateescu, and V. Wolf. Approximation of event probabilities in noisy cellular processes. *Theor. Comput. Sci.*, 412, 2011.

[8] Robin Donaldson and David Gilbert. A model checking approach to the parameter estimation of biochemical pathways. In *CMSB*, pages 269–287, 2008.

[9] Francois Fages and Aurelién Rizk. On the analysis of numerical data time series in temporal logic. In *M. Calder and S. Gilmore (Eds.), CMSB, Vol. 4695 of Lecture Notes in Bioinformatics, Springer*, pages 48–63, 2007.

[10] Pedro F. Felzenszwalb and Daniel P. Huttenlocher. Efficient belief propagation for early vision. *nt J Computer Vision*, (70(1)):41–54, 2004.

[11] Nir Friedman. Inferring cellular networks using probabilistic graphical models. *Science*, (303(5659)):799–805, 2004.

[12] Radu Grosu and Scott A. Smolka. Monte carlo model checking. In *TACAS*, pages 271–286, 2005.

[13] Thomas A. Henzinger, Maria Mateescu, and Verena Wolf. Sliding window abstraction for infinite markov chains. In *CAV*, pages 337–352, 2009.

[14] Sumit Kumar Jha, Edmund M. Clarke, Christopher James Langmead, Axel Legay, André Platzer, and Paolo Zuliani. A bayesian approach to model checking biological systems. In *CMSB*, pages 218–234, 2009.

[15] B. N. Kholodenko. Untangling the signalling wires. *Nature Cell Biology 9 (3)*, pages 247–249, 2007.

[16] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[17] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. PRISM: Probabilistic symbolic model checker. In *T. Field, P. G. Harrison, J. T. Bradley, U. Harder (Eds.), Computer Performance Evaluation / TOOLS, Vol. 2324 of Lecture Notes in Computer Science, Springer*, pages 200–204, 2002.

[18] Marta Z. Kwiatkowska, Gethin Norman, and David Parker. *Symbolic Systems Biology*, chapter Probabilistic Model Checking for Systems Biology. Jones and Bartlett, 2010.

[19] Bing Liu, David Hsu, and P. S. Thiagarajan. Probabilistic approximations of ODEs based bio-pathway dynamics. *Theor. Comput. Sci.*, 412:2188–2206, May 2011.

[20] Bing Liu, P. S. Thiagarajan, and David Hsu. Probabilistic approximations of signaling pathway dynamics. In *P. Degano, R. Gorrieri (Eds.), CMSB, Vol. 5688 of Lecture Notes in Computer Science, Springer*, pages 251–265, 2009.

[21] Bing Liu, Jing Zhang, Pei Yi Tan, David Hsu, Anna M. Blom, Benjamin Leong, Sunil Sethi, Bow Ho, Jeak Ling Ding, and P. S. Thiagarajan. A computational and experimental study of the regulatory mechanisms of the complement system. *PLoS Comput Biol*, 7(1):e1001059, 01 2011.

[22] Schilling M, Maiwald T, Hengl S, Winter D, Kreutz C, Kolch W, Lehmann WD, Timmer J, and Klingmüller U. Theoretical and experimental analysis links isoform-specific ERK signalling to cell fate decisions. *Mol. Syst. Biol.*, 5:334, 2009.

[23] Robert J. McEliece, David J.C. MacKay, and Jung-Fu Cheng. Turbo decoding as an instance of Pearl's "Belief Propagation" algorithm. *IEEE J on Selected Areas in Communications*, (16(2)):140–152, 1998.

[24] Kevin P. Murphy. Bayes Net Toolbox for Matlab. http://bnt.googlecode.com.

[25] Kevin P. Murphy and Yair Weiss. The factored frontier algorithm for approximate inference in DBNs. In *Proceedings of the* 17^{th} *Annual Conference on Uncertainty in Artificial Intelligence, San Francisco, CA, USA*, pages 378–385, 2001.

[26] N. Le Novere, B. Bornstein, A. Broicher, M. Courtot, M. Donizelli, H. Dharuri, H. Sauro L. Li, M. Schilstra, B. Shapiro, J. Snoep, and M. Hucka. Biomodels database: A free, centralized database of curated, published, quantitative kinetic models of biochemical and cellular systems. *Nucleic Acids Research 34*, pages D689– D691, 2006.

[27] Sucheendra K. Palaniappan, S. Akshay, Blaise Genest, and P. S. Thiagarajan. A hybrid factored frontier algorithm, 2011. Available at: `www.comp.nus.edu.sg/~suchee/CMSB/`.

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