A Hybrid Factored Frontier Algorithm for Dynamic Bayesian Networks

Sucheendra K. Palaniappan[1] , S. Akshay[1], Blaise Genest[2]
and P. S. Thiagarajan[1]

[1]School of computing, National University of Singapore,
{suchee,akshay,thiagu}@comp.nus.edu.sg
[2] CNRS, IPAL UMI, joint with NUS-I2R-A*STAR, Singapore.
bgenest@irisa.fr

Abstract. Probabilistic models are often used to describe the dynamics of biochemical networks. To analyze these models one must compute the probability of a state at a given time. Doing this exactly is intractable for large networks and hence approximate methods are needed. In this light, the Factored Frontier algorithm (FF) is a simple and efficient approximate algorithm. But its error behavior has not been analyzed so far. We first observe that theoretically the errors incurred by FF can be quite large. In practice too, even for a small biochemical network, FF's performance can be rather inaccurate. To overcome this, we propose an algorithm which works like FF but at each time slice, we maintain more explicitly the probabilities of a small number of states. We show that by tuning the sizes of these sets we can reduce errors at the cost of paying a higher -but still polynomial time- computational price. Our algorithm performs quite well when applied to the EGF-NGF pathway model [3], consisting of 100 time slices and 32 variables at each time slice.

1 Introduction

Probabilistic models are increasingly used to describe the dynamics of biochemical networks [4, 5, 7, 8, 12, 13, 15, 19]. The common theme in these approaches is that the behavior of the network can be described by a Markov chain in which each state consists of a vector of finite valued random variables. Each random variable will correspond to a molecular species (protein, gene etc.) and its value will reflect its current concentration. To analyze the behavior of the network, one needs to compute at each time point of interest, the current *probability distribution* of the states. The number of states will be exponential in the number of variables. Consequently, for large networks, it is infeasible to explicitly maintain and compute this probability distribution. In fact, it is also infeasible to spell out the transition matrix of the Markov chain.

However, each molecular species will often take part only in a few reactions and hence the next value of a variable will be directly influenced by the current values of only a few other variables. In this case, one can factor the Markov chain into a probabilistic graphical model such as a Dynamic Bayesian Network (DBN) [15] in which the dynamics is captured locally through Conditional Probability Tables (CPTs). This solves the problem of representing the Markov Chain compactly. However the time complexity of exactly inferring the next probability distribution from the current one is still exponential in the size of the network [1]. Hence one must resort to maintaining and computing approximate probability distributions which are usually called *belief states*.

In the literature two attractive approximate algorithms have been proposed for DBNs and related models. In the Boyen-Koller algorithm [1] (BK for short), a belief state is maintained compactly as a product of marginal functions. Roughly speaking, this belief state is then propagated exactly at each step through the transition model consisting of the CPTs and then the new belief state is compacted again into a product of marginal functions. BK is accompanied by a sophisticated error analysis [1, 2] which shows that the discrepancy produced between the belief state and the actual probability distribution in one step does not affect future discrepancies too much. Hence the key is to bound the one step error.

Unfortunately, for large DBNs, exact propagation of the belief state at a step is itself infeasible. The Factored Frontier algorithm (FF, for short) [16] overcomes this by directly computing the new marginal functions via the propagation of the current marginal functions through the CPTs. This ensures that the time complexity is exponential in the maximal in-degree of the DBN graph and not in the size of the whole network. Further, FF is very simple and easy to implement.

In settings where DBNs model the dynamics of biochemical networks, it is important that (i) one has a good understanding on the error incurred in a single step of the algorithm (ii) the errors are not large (iii) measures can be taken by spending more computational effort if necessary to reduce the errors. To the best of our knowledge, no analysis is available regarding the single step error incurred by FF. Secondly, even on a small system (as we show in Section 5), it can produce errors as high as 0.16 on the marginals. Further, FF does not have any tunable parameters using which one can improve accuracy by increasing the computational effort.

In this paper we present an enhanced version of FF called the *Hybrid Factored Frontier* algorithm (HFF, for short) which addresses these limitations of FF. It is a parametrized algorithm in which by tuning the parameters one can reduce the errors. We also derive the single step error bound for our algorithm. The main idea behind HFF can be explained as follows. Suppose the error $\mid \widehat{B}(\mathbf{s}) - B(\mathbf{s}) \mid$ is large where $\widehat{B}$ is the actual belief state and B is the approximated belief state. Since both are probability distributions, for the error to be large, either $\widehat{B}(\mathbf{s})$ or $B(\mathbf{s})$ has to be large and hence the number of such vectors $\mathbf{s}$ -called *spikes*- has to be small. Thus, at each step, in addition to maintaining a belief state just as FF does, we also maintain a small set of spikes and their probabilities. It is however computationally infeasible to determine the set of spikes and their joint probabilities exactly. Hence HFF achieves this approximately. The time complexity of

FF is linear in n, the number of variables and exponential in D, the maximal in-degree of the DBN graph. For HFF it is quadratic in the number of spikes, linear in n and exponential in D. On the other hand, our error analysis shows that the worst case single step error made by HFF is lower than that for FF. In addition, our experimental results using a bio-pathway model based on 32 variables and 100 time points (thus the DBN has 3200 nodes) also show that whenever FF makes a "large" error, the error made by HFF is never as much and often considerably lower. Further, by increasing the number of spikes, we can reduce the errors. In fact when the number of spikes is 0, HFF is exactly FF and when the set of spikes is the set of all states then HFF corresponds to exact inference. In this sense, HFF is a parametrized extension of FF.

In the next section we introduce DBNs and explain how they arise in our approach to bio-pathways modeling. In Section 3, we discuss the FF algorithm and analyze its one step and overall error. In Section 4, we present HFF together with its error analysis. Section 5 presents our experimental results based on the EGF-NGF pathway [3] and we conclude with a discussion in Section 6. Many of the technical details and background information can be found in [18].

2 Dynamic Bayesian Networks

Through the next three sections 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 $\boldsymbol{X}$ the tuple $(X_1, \ldots, X_n)$. The random variables take values from the set V of cardinality K. As usual, we let x_i and sometimes v_i to denote a value taken by X_i. Our dynamic Bayesian networks will be time variant but with a regular structure [16]. 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 $t-1$ 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 the node X_i^t specifying the probabilities $P(X_i^t \mid Pa(X_i^t))$.

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

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

Thus $C_i^t(x_i \mid \boldsymbol{u}_{\hat{i}}) = p$ specifies p to be the probability of $X_i = x_i$ at time t given that at time $t-1$, $X_{j_1} = \boldsymbol{u}_{j_1}, X_{j_2} = \boldsymbol{u}_{j_2}, \ldots, X_{j_m} = \boldsymbol{u}_{j_m}$ 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 - probabilistically- the possible states of the system at time t. In other words, $P(\boldsymbol{X}^t = \boldsymbol{x})$ is the probability that the system will reach the state $\boldsymbol{x}$ at t. Starting from $P(\boldsymbol{X}^0)$ at time 0, given by $P(\boldsymbol{X}^0 = \boldsymbol{x}) = \prod_i C_i^0(\boldsymbol{x}_i)$, 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:

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

with $\boldsymbol{u}$ 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 distribution is feasible only when n is small. This motivates the search for maintaining $P(\boldsymbol{X}^t)$ compactly and computing it approximately but efficiently. Before we describe how our algorithm achieves this, we first describe how we use DBNs to model the dynamics of bio-pathways.

2.1 DBN Models of Bio-pathways

Biological pathways are often described as a network of bio-chemical reactions. Consequently the dynamics of a reactions network can be modelled as a system of 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. Further, model calibration and validation will have to be carried out using limited data that has only crude precision. Guided by this we have developed a method for deriving a dynamic Bayesian network from a system of ODEs that models a bio-pathway [15].

We assume the states of the system are observed only at a finite number of time points $\{0, 1, \ldots, T\}$. Next we partition the range of each variable y_i (rate constant r_j) into a set of intervals $\mathbf{I}_i$ ($\mathbf{I}_j$). 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 sufficiently many times [15] 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.

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 simple counting. First we specify one random variable $Y_i(R_j)$ for

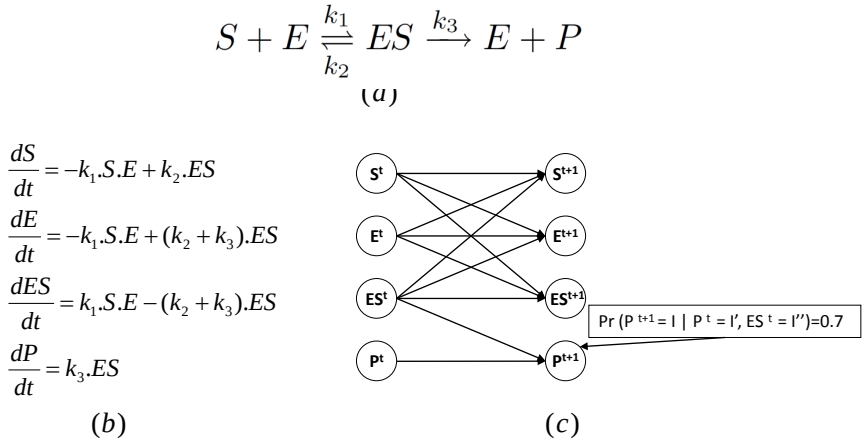

Fig. 1. a) The enzyme catalytic reaction network (b) The ODEs model (c) The DBN approximation for 2 successive time slices

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 $y_k(r_j)$ 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. In Figure 1, we show a simple enzymatic reaction network, its ODE model and the structure of its DBN approximation for 2 successive time points.

An entry in its CPT 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 value 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} simultaneously falls in the interval $\mathbf{I}_{k_l}$ for each $k_l \in \hat{i}$. Suppose this number is d. We then determine for how many of these d trajectories, the value of Y_i falls in the interval I at time t. If this number is d' then p is set to be $\frac{d}{d'}$.

The one time cost of constructing the DBN can be easily recovered through the gains obtained in doing parameter estimation and sensitivity analysis [15]. Moreover, new experimental data can be more easily integrated into the DBN approximation -typically by updating the parameter estimates- through belief propagation [10].

3 The Factored Frontier Algorithm

In what follows, we term the approximate probability distributions to be belief states and denote them as B, B^t etc. while exact probability distributions will be denoted as P, P^t etc. Thus a belief state B is just a map from $V^n \rightarrow [0,1]$ s.t. $\sum_{\boldsymbol{u} \in V^n} B(u) = 1$. Marginal functions are used by FF to represent belief states. A marginal function is a map $M : \{1, \ldots, n\} \times V \rightarrow [0,1]$ s.t. $\sum_{v \in V} M(i,v) = 1$ for each i. In what follows, u, v will range over V while $\boldsymbol{u}$ and $\boldsymbol{v}$ will range over V^n.

From a marginal function M, one can obtain the belief state B_M via $B_M(\boldsymbol{u}) = \prod_i M(i, \boldsymbol{u}_i)$. On the other hand, a belief state B induces the marginal function M_B via $M_B(i,v) = \sum_{\boldsymbol{u} \mid \boldsymbol{u}_i = v} B(\boldsymbol{u})$. 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. When the variables are all mutually independent however, we will have $B_{M_B} = B$.

Suppose we are given a DBN as described in the previous section. FF computes inductively a sequence M^t of marginal functions as follows:

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

Thus, FF maintains B^t, the belief state at t, compactly as the marginal function M^t. In other words, $B^t(\boldsymbol{u}) = \prod_j M^t(j, \boldsymbol{u}_j) = B_{M^t}$. To estimate FF's error behavior, 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:

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

FF however computes only the marginal function $M^t = M_{\widehat{B}^t}$, which then abstractly represents the new belief state $B^t = B_{M^t}$. 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 [18]). We note that B^0 is accurate by definition and hence M^1 will also be accurate but not necessarily B^1.

Consequently, due to $B^t = B_{M_{\widehat{B}^t}}$, the one step error ϵ_t incurred by FF at step t is bounded by $max_{\boldsymbol{u} \in V^n} \{|\widehat{B}^t(\boldsymbol{u}) - B_{M_{\widehat{B}^t}}(\boldsymbol{u})|\}$. We can bound ϵ_t from above by ϵ_0 where : $\epsilon_0 = max\{|B(\boldsymbol{u}) - B_{M_B}(\boldsymbol{u})|\}$ with B ranging over the set of all possible belief states and $\boldsymbol{u}$ ranging over V^n.

The overall error at time t, denoted Δ^t is given by $\Delta^t = max_{\boldsymbol{u} \in V^n}(|P(X^t = \boldsymbol{u}) - B^t(u)|)$. Using a reasoning similar to [1], 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. Further, $\beta < 1$ under fairly mild restrictions placed on the underlying Markov chain and in this case we have $\sum_{j=0}^{t} \beta^j < 1/(1-\beta)$.

A technical analysis of ϵ_0 shows that ϵ_0 converges to 1 as n, the number of variables, tends to ∞. Interestingly, the FF error on a marginal can be large in practice too. Specifically, for the simple network of Figure 1, we get an error as high as 0.16 on one of the marginals.

4 The Hybrid Factored Frontier Algorithm

During the error analysis for FF, we observed that if ϵ_t is large then $B^t(\boldsymbol{v})$ is large for some $\boldsymbol{v}$ and hence $M^t(i, \boldsymbol{v}_i)$ is large for every i. But then there can't be too many such $\boldsymbol{v}$. For instance, there can be only one such $\boldsymbol{v}$ if we want $M^t(i, \boldsymbol{v}_i) > \frac{1}{2}$ for each i. Thus if we can record $B^t(\boldsymbol{v})$ explicitly for a small subset of V^n for which M^t is high for all dimensions then one can significantly improve FF. Unfortunately this can not be done exactly since it will involve an exhaustive search through V^n. Instead we will have to do this approximately.

Accordingly, the Hybrid FF algorithm works 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 such that $B_H^t(\boldsymbol{u}) = 0$ if $\boldsymbol{u} \notin S^t$ and $\sum_{\boldsymbol{u} \in S^t} B_H^t(\boldsymbol{u}) < 1$.
- $\alpha^t = \sum_{\boldsymbol{u} \in S^t} B_H^t(\boldsymbol{u})$.

We define $M_H^t(i, v) = [M^t(i, v) - \sum_{\{\boldsymbol{u} \in S^t | \boldsymbol{u}_i = v\}} B_H^t(\boldsymbol{u})]/(1 - \alpha^t)$ for all i and v. It is easy to observe that this is a marginal function. We next define B^t as follows:

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

We need to use M_H^t rather than M^t since cumulative weight of the contribution made by the spikes needs to be discounted from M^t. This will ensure that B^t is a well defined belief state. The crucial parameter for our algorithm is σ, the number of spikes we choose to maintain. The accuracy of the algorithm improves as σ increases but so does the running time. We have found $\sigma = n^3$ to be more than ample and still computationally feasible for a large network as shown in the next section.

4.1 The algorithm

We initialize with $M^0 = C^0$, $S^0 = \emptyset$, $B_H^0 = \mathbf{0}$ and $\alpha^0 = 0$ and fix σ.

Then, 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: Compute M^{t+1} as

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

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

Step 2: We then compute a set S^{t+1} of at most σ spikes using M^{t+1} as follows.

We want to consider as spikes $\boldsymbol{u} \in V^n$ where $M^{t+1}(i, \boldsymbol{u}_i)$ is large for every i. To do so, we find a constant η^{t+1} such that $M^{t+1}(i, \boldsymbol{u}_i) \geq \eta^{t+1}$ for every i for a subset of V^n containing σ elements and for all other $\boldsymbol{u}'$, there exists i with $M^{t+1}(i, \boldsymbol{u}'_i) < \eta^{t+1}$. 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 choose $\xi = 10^{-6}$, which implies 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. Set a_i to be the number of values v with $M^{t+1}(i, v) > \eta$.
  3. Set U_i to be this set of values.
  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 $B_H^{t+1}(\boldsymbol{u})$ for each $\boldsymbol{u}$ in S^{t+1} as follows.

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

4.2 Analysis of the Hybrid FF

One can establish the following properties of our algorithm. We refer the reader to the full paper [18] for the details.

Proposition 1. *1. For $\sigma = 0$, the hybrid FF algorithm is the same as FF and for $\sigma = K^n$, it is the exact algorithm.*
2. B^t is a belief state for every t.
3. Suppose $P^t = B^t$. Then $P^{t+1}(v) = M^{t+1}(i, v)$ for every i, v.
4. The time complexity of hybrid FF is $O(T \cdot n \cdot (\sigma^2 + K^{R+1}))$, where T is the number of time points, n is the number of variables, σ is the number of spikes, $K = |V|$ and R is the maximum in-degree of the DBN.

We recall the time complexity of FF is $O(n \cdot K^{R+1})$ and hence the additional computational effort required by HFF is $O(T \cdot n \cdot \sigma^2)$. However, in our applications HFF will be run as an off-line computation where in one sweep, the required information about the belief states can be gathered. Where repeated executions are required, such as for combinations of parameter values that best match experimental data ([15]) one can initially run FF repeatedly to narrow down the range of possibilities and then run HFF once to get an accurate estimate.

The error analysis for hybrid FF proceeds along the lines for FF presented in the previous section. The one step error, can be bounded from above by ϵ'_0 with $\epsilon'_0 \leq \min\{(1 - \alpha), \eta\}$, where $\alpha = \min_t(\alpha^t)$ and $\eta = \max_t(\eta^t)$. In particular, if $\sigma \geq 1$, then $\eta \leq 1/2$ and thus $\epsilon'_0 \leq 1/2$.

The cumulative error at t is then given by: $\Delta'^t \leq \epsilon'_0(\sum_{j=0}^{t} \beta^j)$ where β is as specified in the previous section. Thus, the worst case error is better than the one for FF, and can be much better if α is large enough or η small enough, which we can monitor on the fly.

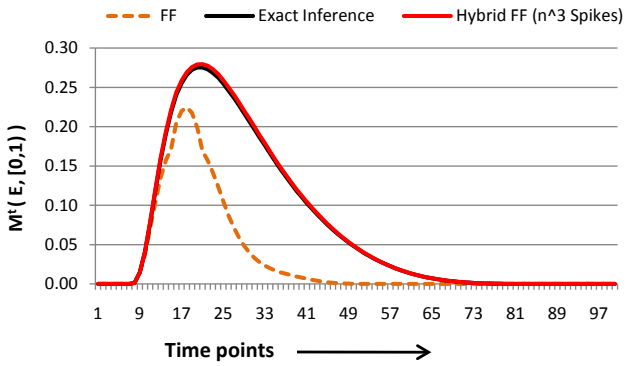

Fig. 2. $M^t(E, [0, 1])$ for all t

5 Results

We have implemented our algorithm in C++. The experiments reported here were carried out on a Opteron 2.2Ghz Processor. First we ran HFF on the DBN for the simple enzyme catalytic system shown in 1.

We fixed the parameters of the system using known values and divided the value space of each variable into 5 equal intervals $[0, 1), [1, 2), \ldots, [4, 5]$ and assumed the initial distributions to be uniformly distributed over certain intervals (see [18]). The time scale of the system was set to be 10 minutes which was evenly divided into time points ranging over $[0, 1, \ldots, 100]$. We fixed the number of spikes to be $4^3 = 64$ and ran HFF and FF. This being a small example, we could compute the probability distributions over the states for each time point exactly. From this we derived the exact marginal distributions for each species. FF performed well for the product species P. However for E, ES and S it deviated from the actual distribution for certain marginals. For instance, for E and the interval $[0, 1)$, it deviated by as much as 0.168 for the marginal $M^t(E, [0, 1))$ as shown in Figure 2. This figure also shows the time evolution of this marginal for HFF and the exact one. As can be seen, the HFF profile is almost the same as that of the exact one (in fact this was already the case for $\sigma = n^2 = 16$). Figure 3 shows the maximum error curves for FF and HFF relative to the exact one; it the maximum of the errors taken over all 5 intervals at each time point. As can be seen, the maximum error incurred by HFF is lower.

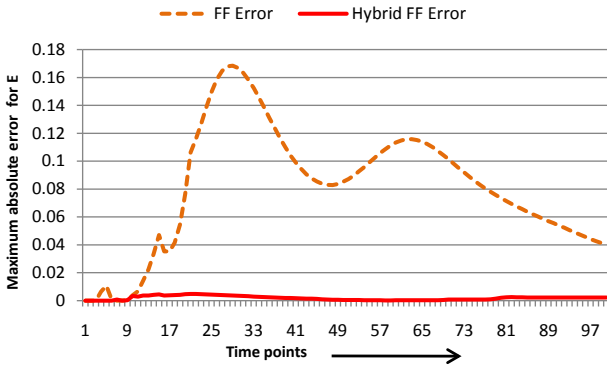

Fig. 3. Maximum Error of FF and HFF with respect to exact for E

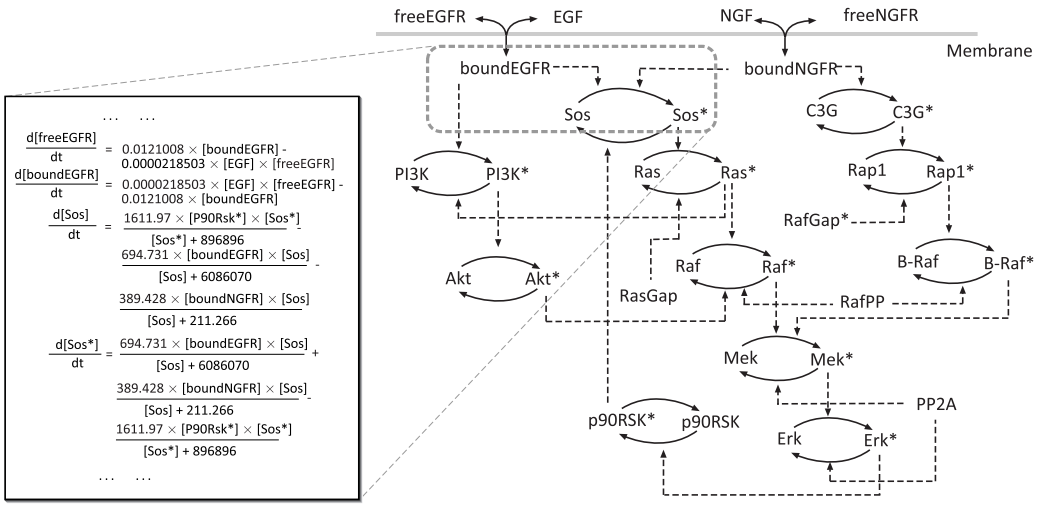

Fig. 4. EGF-NGF pathway

We next considered the EGF-NGF pathway in PC12 cells. This cell line is a valuable model system in neuroscience. Specifically, PC12 cells proliferate in response to EGF stimulation but differentiate into sympathetic neurons in response to NGF. This phenomenon has been intensively studied [9] with a transient activation of Erk1/2 associated with cell proliferation, while a sustained activity has been linked to differentiation. The network structure of this pathway shown in figure 4. The ODE model of this pathway is available in the BioModels database [17]. It consists of 32 differential equations (one for each molecular species) and 48 associated rate parameters (estimated from multiple sets of experimental data). Its DBN approximation was adapted from the one constructed in [15] and details can be found in [14]. We computed the HFF profiles for various sizes of spikes set. Next we ran FF and compared its profiles with those of HFF. For many of the species FF did quite well. However, 6 species out of 32 exhibited a significant difference, among which were important proteins such as Activated Sos and Activated Erk.

Since the DBN consisted of 3200 nodes (32 nodes per time slice; 100 time slices), with each node's variable assuming 5 possible values, it was not possible to compute the exact probability distributions over 5^{32} states at each time point. However, to compare the accuracy of FF and HFF (with $\sigma = n^3 = 32768$ for this DBN), one does not necessarily need the exact distribution. Denoting M^t_σ (respectively, M^t_{FF}) the marginal at time t computed by HFF with σ spikes (respectively, FF), the quantity $|M^t(i,v) - P(X^t_i = v)| - |M^t_{FF}(i,v) - P(X^t_i = v)|$ does not depend on the exact marginal $P(X^t_i = v)$. Rather, it depends on the marginals $(M^t_\sigma(i,v), M^t_{FF}(i,v))$ and their relative position with respect to the exact marginal.

Now, as σ approaches K^n (where $K = |V|$), α approaches 1 and thus the belief states computed by HFF approach the exact distributions. Hence, we ran HFF with different values of σ -denoted HFF(σ)- ranging from HFF(3072) upto HFF(100000) and looked for a pattern. From the relative positions of the curves, and the high value of α (always bigger than 0.6) for HFF(100000), we infer that the curve for HFF(100000) is on the same side of FF and HFF(32768) as the exact marginal. Hence, the quantity $|M^t_{32768}(i,v) - M^t_{100000}| - |M^t_{FF}(i,v) - M^t_{100000}|$ is a good approximation for $|M^t_{32768}(i,v) - P(X^t_i = v)| - |M^t_{FF}(i,v) - P(X^t_i = v)|$. In figure 5 we show for Activated Erk, the computed marginals of the concentration of this protein falling in the interval $[2,3)$ for FF as well as HFF(3072), HFF(32768) and HFF(100000).

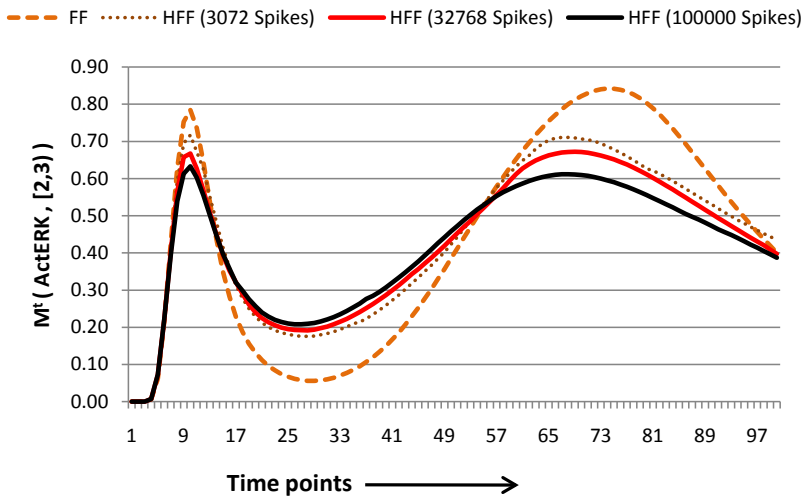

Fig. 5. Time profile of $M^t(ActErk \in [2,3])$ for FF and HFF with various choices for σ

We also gathered and compared the errors incurred by HFF($n^3 = 32768$) and FF relative to HFF(100000) for ActErk over all 5 intervals consisting of $100 \times 5 = 500$ points as well as all the proteins consisting of $32 \times 5 \times 100 = 16000$ points. The comparison of the errors is shown in figure 6. The 80 data points at which FF does better by more than 0.01 compared to HFF(n^3) can be explained by examining figure 5. As can be seen, when the FF curve crosses the HFF(100000) curve, the HFF(n^3) curve will be further away.

Further, we computed α^t which sums up the approximated probabilities of the spikes at t. The result is shown in figure 7. The relatively high values of α^t for most t indicate that our approximate method of computing the spikes (step 2) and their probabilities (step 3) in HFF is of good quality. It also shows that the single step error is quite low for HFF in this pathway model.

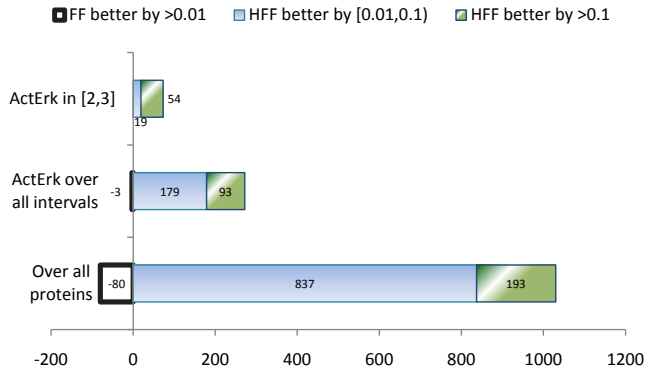

Fig. 6. Relative error of HFF(32768) and FF with respect to HFF(100000) (in terms of no. of time points)

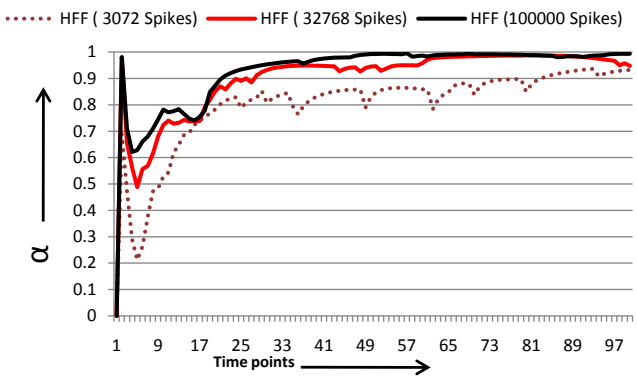

Fig. 7. The evolution of α for the EGF/NGF pathway model.

It took 40 hours to compute HFF(100000) to establish the standard for comparing FF and HFF(32768). For HFF(32768), it took 3.8 hours, for HFF(3072) it took 2.5 minutes, while FF took just 0.2 seconds. However, the current implementation of HFF is quite naive and sequential and there are significant opportunities to improve its accuracy and performance. Further as mentioned earlier, in our applications HFF is an off-line computation and hence the accuracy gained through this extra computational effort is quite encouraging.

6 Discussion

A variety of approaches to modeling bio-pathway dynamics use Markov chains as the underlying model. A key piece of information that is needed for many analysis tasks is the probability distribution of the states at each time point. The state of the Markov chain will consist of a vector of random variables and the dimension of this vector will correspond to the number of entities participating in the pathway. Hence, for large pathways, explicitly representing the probability distributions and exactly computing them, are both infeasible. One must resort to approximate methods. In this light, the Factored Frontier algorithm is a simple and efficient approximate algorithm but its error behavior was not well understood till the present. It also seems to incur significant errors even for small biochemical networks. To overcome this we have presented here an extension of FF called the Hybrid Factored Frontier algorithm. In addition to maintaining and propagating belief states in a factored form, HFF also maintains a small number of full dimensional state vectors called spikes and their probabilities at each time slice. This adds a tunable parameter to FF using which one can improve the accuracy at the price of increased but polynomial time computational cost. We have provided an error analysis for HFF as well as FF which shows that HFF can perform better. This is also validated by our experiments on a fairly large pathway model.

Here we have used probabilistic approximations of ODE models as a source of our DBNs. It will be important to derive DBN models from other sources of biological relevance such as [5, 7, 13]. We plan to optimize the implementation of our algorithm and also explore alternative methods for computing the spikes and their probabilities. Secondly, we would like to estimate the contraction factor β to bound the cumulative error. Further, we would like to exploit the pathway structure of the biological system [11] to further optimize our algorithm. We are also developing other DBN models of signaling networks to which we plan to apply HFF. A related and important goal will be to develop approximated probabilistic verification techniques based on logics such as PCTL [6] using HFF.

References

1. 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.
2. Xavier Boyen and Daphne Koller. Exploiting the architecture of dynamic systems. In *AAAI/IAAI*, pages 313–320, 1999.
3. 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.
4. 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.
5. Vincent Danos, Jérôme Feret, Walter Fontana, Russell Harmer, and Jean Krivine. Rule-based modelling of cellular signalling. In *CONCUR*, pages 17–41, 2007.
6. Hans Hansson and Bengt Jonsson. A logic for reasoning about time and reliability. *Formal Asp. Comput.*, 6(5):512–535, 1994.
7. Thomas A. Henzinger, Maria Mateescu, and Verena Wolf. Sliding window abstraction for infinite markov chains. In *CAV*, pages 337–352, 2009.
8. 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.
9. B. N. Kholodenko. Untangling the signalling wires. *Nature Cell Biology 9 (3)*, pages 247–249, 2007.
10. Geoffrey Koh, David Hsu, and P. S. Thiagarajan. Incremental signaling pathway modeling by data integration. In *RECOMB*, pages 281–296, 2010.
11. Geoffrey Koh, Huey Fern Carol Teong, Marie-Vronique Clment, David Hsu, and P. S. Thiagarajan. A decompositional approach to parameter estimation in pathway modeling: a case study of the akt and mapk pathways and their crosstalk. In *ISMB (Supplement of Bioinformatics)*, pages 271–280, 2006.
12. M. Kwiatkowska, G. Norman, and D. Parker. *Symbolic Systems Biology*, chapter Probabilistic Model Checking for Systems Biology. Jones and Bartlett, 2010.
13. M. Z. Kwiatkowska, G. Norman, and D. 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.
14. Bing Liu, P. S. Thiagarajan, and David Hsu. Supplementary materials for the egf-ngf pathway modeling. http://www.comp.nus.edu.sg/ rpsysbio/tcs10.
15. 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.
16. K. P. Murphy and Y. 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.
17. 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.
18. Sucheendra K. Palaniappan, S. Akshay, Blaise Genest, and P. S. Thiagarajan. A hybrid factored frontier algorithm, 2010. Available at: `www.comp.nus.edu.sg/~suchee/hybridlong.pdf`.
19. 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.