Bayesian Networks as Approximations of Biochemical Networks

Adrien Le Coënt, Benoît Barbot, Nihal Pekergin, Cüneyt Güzeliş

Bayesian Networks as Approximations of Biochemical Networks *

Adrien Le Coënt[1], Benoît Barbot[1], Nihal Pekergin[1], and Cüneyt Güzeliş[2]

[1] Université Paris Est Créteil, LACL, F-94010 Creteil, France
[2] Yaşar University, Department of Electrical-Electronics Engineering, Izmir, Turkey.
adrien.le-coent@u-pec.fr, benoit.barbot@u-pec.fr

Abstract. Biochemical networks are usually modeled by Ordinary Differential Equations (ODEs) that describe time evolution of the concentrations of the interacting (biochemical) species for specific initial concentrations and certain values of the interaction rates. The uncertainty in the measurements of the model parameters (*i.e.* interaction rates) and the concentrations (*i.e.* state variables) is not an uncommon occurrence due to biological variability and noise. So, there is a great need to predict the evolution of the species for some intervals or probability distributions instead of specific initial conditions and parameter values. To this end, one can employ either phase portrait method together with bifurcation analysis as a dynamical system approach, or Dynamical Bayesian Networks (DBNs) in a probabilistic domain. The first approach is restricted to the case of a few number of parameters, while DBNs have recently been used for large biochemical networks. In this paper, we show that time-homogeneous ODE parameters can be efficiently estimated with Bayesian Networks. The accuracy and computation time of our approach is compared to two-slice time-invariant DBNs that have already been used for this purpose. The efficiency of our approach is demonstrated on two toy examples and the EGF-NGF signaling pathway.

Keywords: Ordinary Differential Equations based models · Markov Chains · Bayesian Networks · Biochemical Networks · Time Homogeneous Systems

1 Introduction

In-silico analyses of biochemical networks based on their ODE and Chemical Master Equation (CME) models are valuable instruments for developing the methods for diagnosis, prognosis, drug discovery, therapeutic procedures in the clinical domain as well as for efficient and reliable experiment design. ODE models describe the deterministic future of the species' concentrations given the initial concentrations and the interaction rates. On the other hand, CMEs provide predictions for the stochastic future of the probabilities of the molecule numbers for species. Both have advantages on their sides; ODE models are appropriate

* This work was financed by the join ANR-JST project CyPhAI.

when molecular species are available in large numbers whilst CMEs are more appropriate for low molecule numbers.

As a consequence of biological variability within a population of biochemical species of the same kind as well as measurement errors, environmental noise and estimation errors in determining values of unmeasurable parameters by means of directly measured ones, there is always uncertainty in the reaction rates [12].

Although both kinds of models for biochemical networks have been well-established, their analytical solutions are rarely available and their numerical analysis is computationally expensive and may become intractable due to the combinatorial explosion of the state space for CMEs. Stochastic Simulation Algorithms (SSAs) [9] for CMEs overcome this problem, however, simulations have to be repeated several times to provide accurate estimations making this approach excessively time consuming. In the literature, approximations and learning based methods have become a popular approach to accelerate the CME based models [19, 1].

The ODE models of biochemical networks consist, in general, of many variables and parameters. Their analysis becomes computationally expensive since a large number of numerical simulations must be performed for numerous values of initial conditions and parameters. Approximations of ODEs are usually done with numerical solvers [6, 5], but they usually provide approximations for a given initial condition and parameter combination value. The first probabilistic approximation of ODEs, aimed at capturing the behaviour of the evolution of the solution, and not just approximating a single simulation, has been proposed in [20] from researchers in the theoretical physics community. It has been complemented over the years [22], but their applications have been limited to very small dimensional systems, even though the dynamics can be very complex (nonlinear).

Probabilistic approximation methods have been proposed to tackle with the above-mentioned uncertainty in the ODE model parameters and also in the initial concentration levels of species. Although these approximations provide efficient solutions for low dimensional models, computing them becomes time consuming for large numbers of species and reaction rates. A recent study [14] proposes to use DBNs to efficiently encode the probabilistic approximations built, which at the same time allows to use standard Bayesian inference methods for computing marginal distributions of species, learning unobserved parameters, and performing sensitivity analysis.

In this paper, we focus on the parameter identification problem, and show that using BNs instead of DBNs allows better accuracy of the identification given the same computational power. The description of the construction of the approximations in [14] being very succinct, we first explain how to construct them with as much details as possible, so that interested readers can implement their own approximation methods easily in the future. We furthermore discuss how the structure of the approximations (*i.e.* the structure of the networks, the Markov chain time steps, and the size of the discrete states) play an important

role in the accuracy of the identification problems. We finally exhibit that the BN approach is more efficient than the DBN one.

To show that the BN method provides an effective solution to the parameter identification problem also in high-dimensional networks, the EGF-NGF signaling pathway [3] is considered as a case study. The rationale behind this choice lies in the consideration of a case study on the same application area where the method desired to be surpassed in terms of performance was applied before. In this regard, the existing works [14, 13] that introduce DBN approximations of the ODE models, were initially aimed at analysing EGF-NGF signaling pathways.

The paper is organized as follows: Section 2 is devoted to the considered problem setting and to introduce briefly the applied models. In Section 3, we detail the proposed construction of the BN and emphasize that the structure of the network depends also the method used to simulate the ODEs. The numerical examples are given in Section 4, and finally we conclude and present our perspectives.

2 Problem setting

As in [14, 13], we consider biochemical networks modeled by ODEs. However a large number of numerical simulations for all possible initial conditions and parameter values is necessary to perform parameter estimation or sensitivity analysis in such systems. The main idea in [14, 13] is to generate several trajectories of the ODE model by sampling different initial conditions and to construct an approximate discrete-time and discrete-valued stochastic model. Dynamic Bayesian inference methods are then applied to compute marginal distributions of some species, to provide parameter estimation, sensitivity analysis of large biochemical networks.

In this paper we propose a similar approach. However, we advocate that Bayesian networks would be sufficient if the underlying ODEs are autonomous (*i.e.* they do not explicitly depend on the time variable) which is hypothesized in [14, 13]. In the following subsections we briefly define the considered deterministic and stochastic models, and then explain the probabilistic approximation of ODE based models.

2.1 ODE based continuous-valued deterministic models

The biochemical systems we consider are modeled by ODEs written under the following form:

$$\dot{x_i}(t) = f_i(x(t), k), \tag{1}$$

the variables $x(t) = x_1(t), x_2(t), ..., x_n(t)$ are real-valued concentrations of species at time t. The set $k_1, k_2, ..., k_m$ is the set of (positive) real-valued parameters, they are supposed to be constant over time, but unknown. We are interested in studying the system with various combinations of parameter values. The variables $x_i(t)$ for $i \in \{1, \ldots, n\}$ and the parameters k_j for $j \in \{1, \ldots, m\}$ are supposed to take values in products of intervals $\mathbf{X} = [x_1^{min}, x_1^{max}] \times \ldots \times [x_n^{min}, x_n^{max}]$ and $\mathbf{K} = [k_1^{min}, k_1^{max}] \times \ldots \times [k_m^{min}, k_m^{max}]$.

The functions f_i are issued from mass action kinetics, we assume them to be continuously differentiable. This ensures that the flows (vector fields) that are solutions of the ODE are measurable functions. We refer the reader to [14, 10, 7] for more details on the mathematical soundness ensured by these assumptions.

Our main motivating case study is the EGF-NGF network, described in [14], but our methods are illustrated on the following simpler examples.

Example 1. Our first example is a toy with two variables and one parameter:

$$\begin{aligned} \dot{x_1} &= kx_1 \\ \dot{x_2} &= -0.9kx_1 \end{aligned} \tag{2}$$

Example 2. Our second example comes from a typical biochemical enzyme catalyzed reaction. Its continuous dynamics is given by

$$\begin{aligned} \dot{x_1} &= -k_1x_1x_2 + k_2x_3 \\ \dot{x_2} &= -k_1x_1x_2 + (k_2 + k_3)x_3 \\ \dot{x_3} &= k_1x_1x_2 - (k_2 + k_3)x_3 \\ \dot{x_4} &= k_3x_3 \end{aligned} \tag{3}$$

In the following, we need to designate how one variable affects the others in the continuous dynamics. We thus define a function pa, which stands for parents, as follows.

Definition 1. *The function $pa_x(f_i)$ (resp. $pa_k(f_i)$) maps to the set of indices of variables (resp. parameters) on which variable i depends. Each variable depends at least on itself.*

For instance, for Example 2, $pa_x(f_1) = pa_x(f_2) = pa_x(f_3) = \{1, 2, 3\}$, $pa_x(f_4) = \{3, 4\}$ and $pa_k(f_1) = \{1, 2\}$, $pa_k(f_2) = pa_k(f_3) = \{1, 2, 3\}$ and $pa_k(f_4) = \{3\}$. We will denote $\mathbf{x}_{|pa(j)}$ the projection of $\mathbf{x}$ where only the coordinates belonging to the parents of j remain, for example in the previous example $\mathbf{x}_{|pa_x(f_4)} = (x_3, x_4)$.

2.2 Discrete-valued probabilistic models

The first step of the approximation is the discretization of the value intervals (*i.e.* $\mathbf{X}$ and $\mathbf{K}$) for each variable and parameter leading them to take values in a finite set of sub-intervals. Since each sub-interval is considered as a discrete state, the model is now discrete-valued.

Let us emphasize that it is not irrelevant to consider value intervals instead of precise values in biochemical systems, since model parameters are often subject to uncertainties. Furthermore it is assumed that the system is not observed continuously but at discrete time instants, the time is thus also discretized. The

discrete time instants are an increasing sequence $(t_i)_{i=0}^{\tau}$ with $t_0 = 0$, $t_1 = \delta$, ..., and $t_\tau = T = \delta\tau$ with time step δ. The flow of the differential equation thus induces a discrete-time Markov chain (DTMC) by assuming a prior distribution of initial values [21].

Definition 2. *A time-homogeneous discrete-time Markov chain (DTMC) with finite or countable state space $\mathcal{H}$ is a sequence X_0, X_1, ... of $\mathcal{H}$-valued random variables such that for all states $i, j, x_0, x_1, \ldots$ and all times $t = 0, 1, 2, \ldots$,*

$$Pr(X_{n+1} = j | X_n = i, X_{n-1} = x_{n-1}, \ldots) = Pr(X_{n+1} = j | X_n = i) = p(i, j)$$

where the transition probability $p(i, j)$ depends only on the states i, j, and not on the time nor the previous states x_{n-1}, x_{n-2}, ...

The estimation of transition probabilities for the approximate DTMC by the statistical analysis of ODE trajectories will be described in the following sections. BN models will be used for the compact representation of the approximate DTMC of the underlying ODE model. We refer to [18] for further information.

Definition 3. *A Bayesian network (BN) is a finite acyclic directed graph $BN = (V, E)$. For each node $v \in V$, a finite-valued random variable X_v and a conditional probability table CPT_v are associated. The entries in CPT_v are of the form $Pr(X_v = x | X_{v_1} = x_1, X_{v_2} = x_2, ..., X_{v_j} = x_j)$ where $v_1, v_2, ..., v_j$ is the set of parents of v given by $pa(v) = \{u | (u, v) \in E\}$.*

BN represents the joint probability distribution over the random variables $\{X_v\}$, $1 \leq v \leq n$ given by: $Pr(\boldsymbol{X}) = \prod_{v=1}^{n} Pr(X_v | pa(X_v))$.

Dynamic Bayesian Networks (DBNs) are Bayesian Networks that allow us to model the temporal evolution of the system.

Definition 4. *A Dynamic Bayesian Network (DBN) is a tuple $(B_0, \{B_{\rightarrow}^j\}_{j=1}^{T})$, where B_0 defines the initial probability distributions of the random variables, and $\{B_{\rightarrow}^j\}$ are two-slice temporal Bayesian networks for the transition from time step $j-1$ to j: $Pr(\boldsymbol{X}^j \mid \boldsymbol{X}^{j-1}) = \prod_{v=1}^{n} Pr(X_v^j \mid pa(X_v^j))$. In the case of time-invariant DBNs, there exists one two-slice BN, $B_{\rightarrow}$ whatever the time step j is.*

DBNs are useful for computing joint distributions of species after several time steps, or estimating initial distributions. In the case of parameter identification for time-homogeneous systems, the transition model (CPT) from time j to $j+1$ should be constant. In this paper, contrary to [14] where they use DBNs, we propose to use BNs for parameter inference.

2.3 Probabilistic approximation of ODE based models

The main idea of the approximation is to estimate approximate transition probabilities from the numerical simulation of ODEs. The distribution of initial values is sampled several times and for each sampled initial value, the trajectory of the

ODE is constructed. A sufficiently large set of such trajectories provides a good approximation of the dynamics of the ODE model. The transition probabilities of the underlying approximate Markov chain are estimated by analyzing the statistical properties of the ODE trajectories.

The probabilistic approximation of ODEs proposed in [14] consists of modeling the approximate DTMC by a two-slice time-variant DBN. Our observation is that the ODE considered in [14] is autonomous, and the parameters are considered constant over time. Transition probabilities should not vary with time. Contrary to [14], we propose to use Bayesian networks to store the transition probabilities, whatever the time step is.

As that will be explained in the next section, the underlying graph is constructed by exploiting the structure of the ODEs, as well as the continuous simulation method used to generate the sample set. The entries of the CPT are specified by the probability transition probabilities of the time-homogeneous approximate Markov chain. Using BN inference techniques for parameter marginal distribution computations leads to the following main computational improvements: the inference algorithms are less time consuming for BNs than DBNs; for large networks, one can use exact inference algorithms for BNs, while they become intractable for DBNs. The discrete probabilistic model still requires numerous simulations to compute the probability tables whatever is the kind of network. Let us emphasize that more simulations used in the BN or DBN construction lead to more accurate parameter estimation.

3 Construction of the probabilistic approximation

3.1 Computation of the BN model

The BN model is built using two main steps which are the construction of both spatially and temporally discretized trajectories and the parameter estimation of the approximate Markov chain from these trajectories.

Construction of discretized trajectories First step is the discretization of the continuous state spaces in which the state variable and parameter values lie: $\mathbf{X} = [x_1^{min}, x_1^{max}] \times \ldots \times [x_n^{min}, x_n^{max}]$ for variables and $\mathbf{K} = [k_1^{min}, k_1^{max}] \times \ldots \times [k_m^{min}, k_m^{max}]$ for parameters. The value interval for each variable i, $[x_i^{min}, x_i^{max}]$ is divided in L_i sub-intervals: $[x_i^{min}, x_i^1)$, $[x_i^1, x_i^2)$, $\ldots$, $[x_i^{L_i-1}, x_i^{max})$. Similarly in M_j sub-intervals for each parameter j: $[k_j^{min}, k_j^1)$, $[k_j^1, k_j^2)$, $\ldots$, $[k_j^{M_j-1}, k_j^{max})$.

- We define the discretization function d_{L_i} (resp. d_{M_i}) such that $d_{L_i}(x_i) = a$ iff $x_i \in [x_i^{a-1}, x_i^a)$ the function which maps the i^{th} state variable (resp. parameter) to the index a of the corresponding sub-interval with $1 \leq a \leq L_i$ (resp. $1 \leq a \leq M_i$). Therefore for the vector of a continuous state $(\mathbf{x}, \mathbf{k})$ we define the vector of the corresponding discrete state $(\mathbf{d_L}(\mathbf{x}), \mathbf{d_M}(\mathbf{k}))$.
- A number of (continuous) simulations of the ODE (Eq. (1)) is then performed. Any simulation method yielding approximations of the continuous

trajectories can be used, Runge-Kutta schemes are used here. These simulations are performed from random initial conditions in $\mathbf{X}$, and using random parameter values in $\mathbf{K}$. In order to get a probabilistic model that approximates the continuous equation as closely as possible for all the possible initial conditions, the random initial conditions need to cover the initial sets as uniformly as possible. To this end, we make use of Halton sequences [8] to choose the initial conditions.

- The continuous simulations are converted spatially in discrete state simulations by applying $\mathbf{d_L}$. More precisely, consider a given continuous trajectory (or an approximation of it) $\mathbf{x}(t)$, $0 \leq t \leq T$. It is converted in a sequence of discrete states $\mathbf{d_L}(\mathbf{x}(t))$, $0 \leq t \leq T$.
- This sequence of discrete states is also converted temporally by specifying an increasing sequence of time $(t_j)_{j=0}^{\tau}$ with $t_0 = 0$ and $t_\tau = T$.
- The continuous parameters values are only converted spatially since parameters do not depend on time. By applying $\mathbf{d_M}(\mathbf{k})$, we obtain the corresponding discrete parameter values.

Thus, for a trajectory (simulation) with the initial values $\mathbf{x}(t_0)$, and parameter values $(\mathbf{k})$, we produce the pair $\langle(\mathbf{d_L}(\mathbf{x}(t_j)))_{j=0}^{\tau}; \mathbf{d_M}(\mathbf{k})\rangle$ containing the sequence of variables $\mathbf{x}$ at time t_j and parameter values $\mathbf{k}$. In the following, we note this pair for a discretized (both spatially and temporarily) trajectory of Eq. (1) by $\langle(\mathbf{l^j})_{j=0}^{\tau}; \mathbf{m}\rangle$.

Transition probabilities estimation for the Markov chain In the second step, we have a set S of discretized trajectories with elements of the form $\langle(\mathbf{l^j})_{j=0}^{\tau}; \mathbf{m}\rangle_S$ of the Eq. (1). We estimate the transition probabilities of the approximate Markov chain (see Definition 2) by counting which transitions are taken, compared to the total possible transitions from the set S.

- Given S the set of discretized trajectories, $NT(S, \mathbf{l}^{pred}, \mathbf{l}^{next}, \mathbf{m}, i)$ denotes the number of transitions taken during all of the trajectories in S which lead to a state where $l_i = l_i^{next}$ for a given variable index i from a state of $\mathbf{l}^{pred}$. The transition may happen during time step j to time step $j+1$ whatever j is. However at time step j of the trajectory, the values for the parents of variable i denoted by i: $pa_x(f_i)$ (see Definition 1) must match those in $\mathbf{l}^{pred}$. Similarly the parameters in the set $pa_k(f_i)$ must match with $\mathbf{m}$.

$$NT(S, \mathbf{l}^{pred}, \mathbf{l}^{next}, \mathbf{m}, i) = \sum_{\langle(\mathbf{l^j})_{j=0}^{\tau}; \mathbf{m'}\rangle \in S} |\{0 \leq j < \tau \textbf{ s.t. } \mathbf{l}^{pred}_{|pa_x(f_i)} = \mathbf{l}^{j}_{|pa_x(f_i)} \wedge l_i^{next} = l_i^{j+1} \wedge \mathbf{m}_{|pa_k(f_i)} = \mathbf{m'}_{|pa_k(f_i)}\}| \tag{4}$$

The transition probabilities are estimated from the number of transitions for each variable as follows:

$$p(\langle \mathbf{l}^{pred}; \mathbf{m}\rangle, \langle \mathbf{l}^{next}; \mathbf{m}\rangle) = \prod_{i=1}^{n} \frac{NT(S, \mathbf{l}^{pred}, \mathbf{l}^{next}, \mathbf{m}, i)}{\sum_{\langle \mathbf{l}; \mathbf{m}\rangle \in S} NT(S, \mathbf{l}^{pred}, \mathbf{l}, \mathbf{m}, i)} \tag{5}$$

In Figure 1, we illustrate a continuous simulation of the simple Example 1 and a realization of the approximate Markov chain.
– The approximate Markov chain built is finally efficiently stored as a BN. Its structure depends on the ODE and the numerical scheme used for Eq. (1). The structure of the BN to be used is discussed in the following section.

Inference Once a BN is computed, various algorithms can be used for inference, parameter learning, structure learning, sensitivity analysis etc. Whether BNs or DBNs are considered, parameter inference is performed with some evidence that ideally consists of real life measurements. Here, we generate some continuous simulations with given parameter values and observe how accurately we recover these parameter values. For DBNs, the evidence takes the form of a sequence of variable values (species concentrations). For BNs, the same simulations are used, except that the sequence of variable values is simply decomposed in multiple couples (values at t and $t+\delta$). For BNs, we use a standard junction tree algorithm implemented in the BNT toolbox [16]. For DBNs, we use the DBN version of junction trees for exact inference, and the the Boyen and Koller fully factored (BKFF) algorithm [2] for approximate inference.

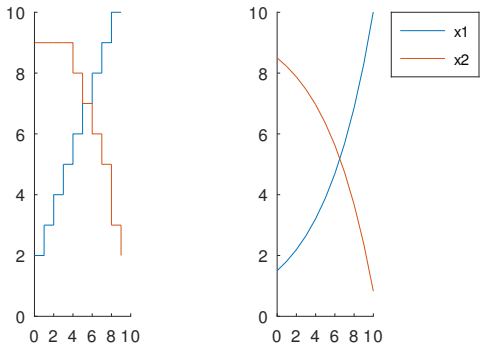

Fig. 1. Left: one realization of the approximate DTMC with 10 sub-intervals for x and 2 sub-intervals for k; Right: the continuous simulation of the ODE from the initial condition $(x_1^0, x_2^0) = (1.5, 8.5)$, using parameter $k = 0.19$. The initial sets of the continuous systems are $[0, 10] \times [0, 10]$ for xs and $[0, 0.2]$ for k.

3.2 Structure of the Bayesian Network

Let us now discuss the structure of the networks that should be used. If we consider ODE simulations performed with a simple Euler scheme, for which the time-step corresponds to Markov chain time step (*i.e.* with the abuse of notation $\delta = 1$, the solution of Eq. (1) is approximated by the series $y_{n+1} = y_n + f(y_n, p)$),

then the structure proposed in [14] can be used. However, if a higher order scheme is used, then more edges should be added to the network.

Let us recall the principle of explicit S-stage Runge-Kutta (RK) schemes [4]:

$$y_{n+1} = y_n + h \sum_{i=1}^{S} b_i \kappa_i \tag{6}$$

where

$$\begin{aligned}
\kappa_1 &= f(t_n, y_n),\\
\kappa_2 &= f(t_n + c_2 h, y_n + (a_{21}\kappa_1)h),\\
\kappa_3 &= f(t_n + c_3 h, y_n + (a_{31}\kappa_1 + a_{32}\kappa_2)h),\\
&\vdots\\
\kappa_S &= f(t_n + c_s h, y_n + (a_{S1}\kappa_1 + a_{S2}\kappa_2 + \cdots + a_{S,S-1}\kappa_{S-1})h).
\end{aligned}$$

Let us now consider Example 2, and add an exponent $i \in \{1, 2, 3, 4\}$ to the variables y_n and κ_j to denote its i^{th} component (dimension). Application of an explicit 4-stage RK scheme to this example exhibits that:

- κ_1^3 depends on y_n^1, y_n^2, y_n^3; κ_1^4 depends on y_n^3, y_n^4 (parents in ODE as defined in Definition 1);
- the definition of κ_2^4 implies that κ_2^4 depends on y_n^3, y_n^4, κ_1^3, κ_1^4;
- then κ_2^4 depends on y_n^1, y_n^2, y_n^3, y_n^4.

And in the end, y_{n+1}^4 depends on y_n^1, y_n^2, y_n^3 and y_n^4. Therefore, as illustrated in Figure 2, node X_3' should have X_1, X_2, X_3 and X_4 as parents in the BN. The following definition reformulates the parent definition for BN nodes, it was used earlier for variables in Definition 1.

Definition 5. *The one-stage parents of node X_i, denoted by $pa_{BN}(X_i)$, are the nodes corresponding to the variables given by $pa_x(f_i)$ and the parameters given by $pa_k(f_i)$.*

Definition 6. *The s-stage parents of node X_i, denoted by $pa_{BN}^s(X_i)$ for any $s \geq 1$, are defined recursively as follows :*

$$pa_{BN}^1(X_i) = pa_{BN}(X_i),$$

$$pa_{BN}^{s+1}(X_i) = \bigcup_{X \in pa_{BN}^s(X_i)} pa_{BN}(X).$$

In the above definitions, if $pa_x(f_i) = \{1, 2\}$ and $pa_k(f_i) = \{1\}$, then $pa_{BN}(X_i) = \{X_1, X_2, K_1\}$. The 2-stage parents $pa_{BN}^2(X_i)$ are the one stage parents of the one stage parents of X_i. Such node connections are then used to build the BN structure depending on the numerical scheme used. If an explicit Euler scheme

with time step δ was used as the continuous simulation scheme, then the structure of the BN would be given by $pa_{BN}(X_i)$ for all i (this is the BN structure suggested in [14], it is represented in Figure 2(a) for Example 2). The following theorem formally states the graph structure yielding the highest accuracy depending on the numerical scheme used to generate the continuous trajectories.

Theorem 1. *Consider the system of Eq.* (1) *simulated with time step* δ *using an* S*-stage RK scheme. The most accurate BN approximation is obtained with the following graph structure. Node* X_i *has parents:*

$$pa_{BN_S}(X_i) = \bigcup_{s=1}^{S} pa^s_{BN}(X_i)$$

where $pa^s_{BN}(X_i)$ *are the* s*-stage parents of node* X_i *as defined in Definition 6.*

The above theorem formalizes that computing an S-stage RK step requires at least S repeated applications of function f of Eq. (1), which the BN node connections should reflect. Application of this theorem to Example 2 yields the structure represented in Figure 2(b). One can observe that, contrary to Figure 2(a), the nodes X'_1, X'_2, X'_3, X'_4 all depend on all the parameter nodes K_1, K_2, K_3. Note however that the graph is not fully connected, repeated applications of f do not connect X_4 to the other nodes. Once a graph structure is chosen, computation of the transition probabilities should be performed with modified functions pa_x and pa_k that take into account node dependencies and not continuous variable dependencies of Definition 1.

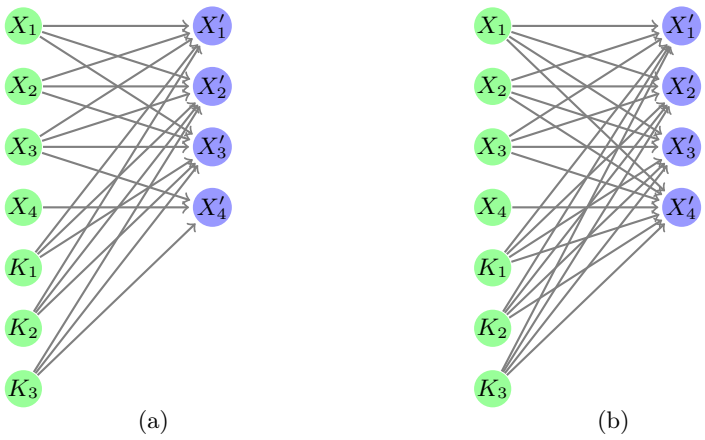

Fig. 2. The BN structure used in [14] (left) and the BN structure we suggest for an RK4 scheme (right).

3.3 Time and space discretization discussion

The time and space discretization parameters used in the present method play an important role in the accuracy of the parameter learning and the computation times, and their choice heavily depends on each other. Let us first observe that over a short discrete time step δ, very small changes in the continuous state are made. If these changes are small enough, the corresponding discrete state does not change. On the other hand, if the space discretization is fine enough, discrete state changes will be observed in the approximating Markov chain.

As a general guideline, we propose to choose the discrete time step, δ and the number of sub-intervals for variable i, L_i such that there exists at least one $i \in \{1, \ldots, n\}$ such that:

$$\frac{\int_{y_0 \in \mathbf{X}} \int_{k \in \mathbf{K}} \|y^i_\delta(y_0, k) - y^i_0\|}{|\mathbf{X}| \times |\mathbf{K}|} \geq \frac{x_i^{max} - x_i^{min}}{L_i}$$

where $|\mathbf{X}| = (x_1^{max} - x_1^{min}) \times \ldots \times (x_n^{max} - x_n^{min})$, $|\mathbf{K}| = (k_1^{max} - k_1^{min}) \times \ldots \times (k_m^{max} - k_m^{min})$, and $y_\delta(y_0, k)$ is (the Runge-Kutta approximation of) the solution at time δ of Eq. (1) from the initial condition y_0 at $t = 0$ and using parameter k. This formula ensures that, on average, the continuous state variation over a time step δ is greater than the size of at least one sub-interval considered for the Markov chain, thus inducing at least one discrete state change over time step δ. This formula cannot be computed exactly, but a few Monte-Carlo simulations should yield a satisfying first estimate of the time and space discretization parameters to use. Note that we do not have a proof that such a parameter combination exists, but we hope to have one in a future work using uniform discretizations and additional hypotheses on function f.

4 Application to biochemical networks

Experimental settings The experiments presented in this section have been performed with GNU Octave, and ran on a MacBook Pro 2017 using a dual core 2.3GHz Intel Core i5 with 8GB of RAM. The probability tables have been generated with an ad hoc prototype that we intend to share in the near future. The BN and DBN analyses have been performed using the BNT toolbox [16].

4.1 The enzyme catalyzed reaction

Consider Example 2 with initial sets $\mathbf{X} = [0, 10] \times [0, 10] \times [0, 15] \times [0, 15]$ for the variables and $\mathbf{K} = [0, 1] \times [0, 1] \times [0, 1]$ for the parameters. For this example, we first illustrate the approach by generating a BN and a DBN using 5 sub-intervals per variable and parameter. *I.e.*, the interval $[0, 10]$ for variable x_1 is divided in $[0, 2)$, $[2, 4)$, $[4, 6)$, $[6, 8)$, $[8, 10]$.

To test the accuracy of the BN and DBN learning methods, we generate a sample set, *i.e.* some continuous trajectories for a given parameter combination and some random initial conditions. They are taken as evidence to learn some

parameter probability distributions. We generate 10 continuous samples using $(k_1, k_2, k_3) = (0.9, 0.5, 0.1)$, and random initial conditions in $\mathbf{X}$. We suppose that parameter k_2 is observed (taken as evidence), the distributions for k_1 and k_3 have to be estimated.

An example of parameter combination learned is given in Table 1. For samples generated using $k_1 = 0.9$ and $k_3 = 0.1$, k_2 being known, the highest probability for parameter k_1 estimation is 0.45 (given in bold). Similarly k_3 is estimated to be in $[0.0, 0.2)$ with a 67% probability in the BN approach. With the DBN approach, the highest probability for k_1 estimation is 0.58 for interval $[0.6, 0.8)$ while the estimation to be in $[0.8, 1.0]$ is with a 31% probability. And k_3 is estimated to be in $[0.0, 0.2)$ with a 59% probability. We have tested the approach for different sets of samples and parameter combinations. We obtain similar results using 7 sub-intervals per parameter and variable.Note that the results obtained highly depend on the sample set. On average, the BN approach yields right parameter intervals more often than the DBN approach. Interestingly, the DBN approach gives results with higher confidence (probability), even though these results might be wrong. Depending on the sample set, the BN learning time is between 25% and 95% faster than the DBN learning time.

original parameters	sub-interval	BN marginal distribution	DBN marginal distribution
$k_1 = 0.9$	$[0, 0.2)$	0.0106	0.0000
	$[0.2, 0.4)$	0.1034	0.0039
	$[0.4, 0.6)$	0.1480	0.0993
	$[0.6, 0.8)$	0.2889	**0.5855**
	$[\mathbf{0.8}, \mathbf{1.0}]$	**0.4501**	0.3114
$k_3 = 0.1$	$[\mathbf{0}, \mathbf{0.2})$	**0.6770**	**0.5940**
	$[0.2, 0.4)$	0.2582	0.2845
	$[0.4, 0.6)$	0.0456	0.1215
	$[0.6, 0.8)$	0.0072	0.0000
	$[0.8, 1.0]$	0.0120	0.0000

Table 1. An example of parameter marginal distribution learned with the BN and DBN approaches, using 5 sub-intervals per parameter. The highest probability is in bold for each method.

We also compared the accuracy of the parameter identification using the BN structures suggested in [14] and our proposition (Theorem 1). We built two BNs with the structures given in Figure 2(a) and Figure 2(b), with 5 sub-intervals per variable and parameter. The sample set was generated using $k_1 = 0.9$ and $k_3 = 0.1$, k_2 being known. As expected, the graph that is more connected yields more accurate results, as seen in Table 2. The computation time was slightly higher with the more connected graph of Figure 2(a) with $3.86s$, compared to $3.32s$ with the graph of Figure 2(b), which is a 16% increase in computation time. This is also expected since the junction tree algorithm is polynomial in the number of cliques [11], which is increased by the number of edges in the graph.

original parameters	sub-interval	BN marginal distribution with structure of Theorem 1	BN marginal distribution with structure of [14]
$k_1 = 0.9$	$[0, 0.2)$	0.0000	0.0000
	$[0.2, 0.4)$	0.0806	0.0947
	$[0.4, 0.6)$	0.1875	0.2075
	$[0.6, 0.8)$	0.3159	**0.5249**
	$[\mathbf{0.8}, \mathbf{1.0}]$	**0.4160**	0.1730
$k_3 = 0.1$	$[\mathbf{0}, \mathbf{0.2})$	**0.9001**	**0.8462**
	$[0.2, 0.4)$	0.0362	0.1502
	$[0.4, 0.6)$	0.0638	0.0036
	$[0.6, 0.8)$	0.0000	0.0000
	$[0.8, 1.0]$	0.0000	0.0000

Table 2. An example of parameter marginal distribution learned with the BN with our suggested graph connection and the graph connection suggested in [14], computed with 5 sub-intervals. The highest probability is in bold for each method.

Finally, the parameter learning accuracy has been tested with different state and time discretizations. Very fine time discretization ($\delta < 0.1$) led to very random results since the BN barely captures any dynamical behaviour, this is due to the number discrete state transitions being very small compared to the number of trajectories used to generate the BNs, and confirms our suggestions in Section 3.3. On Example 2, we observe that state discretization has a noticeable influence on computation time, as shown in Table 3.

number of sub-intervals	BN	DBN
3	28s	35s
5	37s	45s
7	83s	108s

Table 3. Average parameter marginal distribution learning times (in seconds) for the BN and DBN approaches, using different numbers of sub-intervals per parameter and variable.

4.2 The EGF-NGF model

The EGF-NGF signaling pathway is a large biochemical network that has been extensively studied in [3, 14]. The model under consideration, which is described by the ODEs in Appendix in (7), is depicted as a hybrid functional Petri net (HFPN) [15] in Fig. 3. The original model has 32 variables and 48 parameters, 28 of which are known, the remaining ones needing to be learned. A first formal analysis of the network allowed us to identify that only 14 of the 20 unknown parameters are essential to fully grasp the dynamics of the system.

We have successfully computed a BN and a DBN approximation of the model in 53 hours with a very straightforward implementation. They have been con-

structed with 10^6 trajectories, using 5 sub-intervals per variables and unknown parameter. The number of trajectories is very small compared to what is used in [14]. To test the learning accuracy, we generated 5 trajectories of length 10 taken as evidence, for given randomly selected parameter values, and observed how often the learned parameters were in the right sub-interval. Note that exact inference methods quickly become intractable for large DBNs, and approximate inference methods were required to test the DBN learning approach. We used here the Boyen and Koller fully factored (BKFF) algorithm [2] which is already implemented in the BNT toolbox. More time-efficient algorithms such as the factored frontier algorithm could be considered [17], but the accuracy of the BN approach would still be higher since we can use an exact inference algorithm (the junction-tree algorithm). Our approach, despite the small number of trajectories used for generating the BN, recovers the right parameter sub-intervals for 7 of the 14 parameters on average, the others being close, compared to 5 for the DBN. Using exact inference, the learning time is around 15 seconds for the BN, and 3 hours for the DBN. Using the BKFF algorithm, the computation time is lowered to 3 minutes for the DBN, with a minor accuracy decrease.

5 Conclusions and future work

Throughout this paper, we have shown that using BNs as probabilistic approximations for biochemical networks yields better accuracy and computation times than using DBNs for parameter learning, provided that the ODEs are autonomous. We provided a detailed explanation for building the BN approximation. We compared our approach to the DBN approximations suggested in [14]. We furthermore proposed a BN structure that, depending on the numerical schemes used, provides the highest possible accuracy. We compared the accuracy of the parameter identification with BN and DBN approximations on a simple biochemical system, and computed parameters on the real life sized case study of the EGF-NGF signaling pathway model.

Our future work will be devoted to using model reduction techniques as a preliminary step before building the BN approximations, in order to use more computational power on the learning part and less on the BN construction part. Experimental measurements are potentially performed at times that are not in a perfect time sequence (*e.g.* some species concentrations are obtained at times 0, 5, 20, 50, ...). To use such experimental measurements, our method should be extended to BNs that allow learning parameters with samples that have missing time instants.

References

1. Luca Bortolussi and Luca Palmieri. Deep abstractions of chemical reaction networks. In *CMSB 2018, Brno, Czech Republic*, pages 21–38. Springer, 2018.
2. Xavier Boyen and Daphne Koller. Approximate learning of dynamic models. In *Advances in Neural Information Processing Systems 11 (NIPS 1998)*, pages 396–402. Cambridge: MIT Press, 1999.

3. Kevin S Brown, Colin C Hill, Guillermo A Calero, Christopher R Myers, Kelvin H Lee, James P Sethna, and Richard A Cerione. The statistical mechanics of complex signaling networks: nerve growth factor signaling. *Physical Biology*, 1(3):184, 2004.
4. John Charles Butcher. A history of Runge-Kutta methods. *Applied Numerical Mathematics*, 20(3):247–260, 1996.
5. John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.
6. Peter Deuflhard, Ernst Hairer, and J Zugck. One-step and extrapolation methods for differential-algebraic systems. *Numerische Mathematik*, 51:501–516, 1987.
7. Rick Durrett. *Probability: theory and examples*, volume 49. Cambridge University Press, 2019.
8. Henri Faure and Christiane Lemieux. Generalized Halton sequences in 2008: A comparative study. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 19(4):1–31, 2009.
9. Daniel T Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.
10. Morris W Hirsch, Stephen Smale, and Robert L Devaney. *Differential equations, dynamical systems, and an introduction to chaos*. Academic Press, 2012.
11. Cecil Huang and Adnan Darwiche. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning*, 15(3):225–263, 1996.
12. Klipp Liebermeister. Biochemical networks with uncertain parameters. *Systems Biology*, 152:97–107, 10 2005.
13. Bing Liu, Andrei Hagiescu, Sucheendra Palaniappan, Bipasa Chattopadhyay, Zheng Cui, Weng-Fai Wong, and P. Thiagarajan. Approximate probabilistic analysis of biopathway dynamics. *Bioinformatics*, 28:1508–16, 04 2012.
14. Bing Liu, David Hsu, and PS Thiagarajan. Probabilistic approximations of ODEs based bio-pathway dynamics. *Theoretical Computer Science*, 412(21):2188–2206, 2011.
15. Hiroshi Matsuno, Sachie Fujita, Atsushi Doi, Masao Nagasaki, and Satoru Miyano. Towards biopathway modeling and simulation. In *24th International Conference on Applications and Theory of Petri Nets, ICATPN*. Springer, 2003.
16. Kevin Murphy et al. The Bayes net toolbox for Matlab. *Computing Science and Statistics*, 33(2):1024–1034, 2001.
17. Kevin Murphy and Yair Weiss. The factored frontier algorithm for approximate inference in DBNs. In *Proceedings of the 17th conference on Uncertainty in Artificial Intelligence*, pages 378–385, 2001.
18. Kevin P. Murphy. *Probabilistic Machine Learning: An introduction*. MIT Press, 2022.
19. David Schnoerr, Guido Sanguinetti, and Ramon Grima. Approximation and inference methods for stochastic biochemical kinetics—a tutorial review. *Journal of Physics A: Mathematical and Theoretical*, 50(9):093001, 2017.
20. John Skilling. Bayesian solution of ordinary differential equations. *Maximum Entropy and Bayesian Methods: Seattle, 1991*, pages 23–37, 1992.
21. Kishor Trivedi. *Probability and Statistics with Reliability, Queuing and Computer Science Applications*. John Wiley & Sons, 2001.
22. Filip Tronarp, Hans Kersting, Simo Särkkä, and Philipp Hennig. Probabilistic solutions to ordinary differential equations as nonlinear Bayesian filtering: a new perspective. *Statistics and Computing*, 29:1297–1315, 2019.

6 Appendix

$$
\begin{aligned}
\dot{x}_1 &= -k_1 * x_1 * x_3 + k_2 * x_4 \\
\dot{x}_2 &= -k_3 * x_2 * x_5 + k_4 * x_6 \\
\dot{x}_3 &= -k_1 * x_1 * x_3 + k_2 * x_4 \\
\dot{x}_4 &= k_1 * x_1 * x_3 - k_2 * x_4 \\
\dot{x}_5 &= -k_3 * x_2 * x_5 + k_4 * x_6 \\
\dot{x}_6 &= k_3 * x_2 * x_5 - k_4 * x_6 \\
\dot{x}_7 &= k_9 * x_{10} * x_8/(x_8 + k_{10}) - k_5 * x_4 * x_7/(x_7 + k_6) - k_7 * x_6 * x_7/(x_7 + k_8) \\
\dot{x}_8 &= -k_9 * x_{10} * x_8/(x_8 + k_{10}) + k_5 * x_4 * x_7/(x_7 + k_6) + k_7 * x_6 * x_7/(x_7 + k_8) \\
\dot{x}_9 &= -k_{27} * x_{21} * x_9/(x_9 + k_{28}) \\
\dot{x}_{10} &= k_{27} * x_{21} * x_9/(x_9 + k_{28}) \\
\dot{x}_{11} &= -k_{11} * x_8 * x_{11}/(x_{11} + k_{12}) + k_{13} * x_{13} * x_{12}/(x_{12} + k_{14}) \\
\dot{x}_{12} &= k_{11} * x_8 * x_{11}/(x_{11} + k_{12}) - k_{13} * x_{13} * x_{12}/(x_{12} + k_{14}) \\
\dot{x}_{13} &= 0 \\
\dot{x}_{14} &= -k_{15} * x_{12} * x_{14}/(x_{14} + k_{16}) + k_{45} * x_{32} * x_{15}/(x_{15} + k_{46}) \\
&\quad + k_{35} * x_{25} * x_{15}/(x_{15} + k_{36}) \\
\dot{x}_{15} &= k_{15} * x_{12} * x_{14}/(x_{14} + k_{16}) - k_{45} * x_{32} * x_{15}/(x_{15} + k_{46}) \\
&\quad - k_{35} * x_{25} * x_{15}/(x_{15} + k_{36}) \\
\dot{x}_{16} &= -k_{43} * x_{29} * x_{16}/(x_{16} + k_{44}) + k_{47} * x_{32} * x_{17}/(x_{17} + k_{20}) \\
\dot{x}_{17} &= k_{43} * x_{29} * x_{16}/(x_{16} + k_{44}) - k_{47} * x_{32} * x_{17}/(x_{17} + k_{20}) \\
\dot{x}_{18} &= -k_{17} * x_{15} * x_{18}/(x_{18} + k_{18}) - k_{19} * x_{17} * x_{18}/(x_{18} + k_{48}) \\
&\quad + k_{21} * x_{31} * x_{19}/(x_{19} + k_{22}) \\
\dot{x}_{19} &= k_{17} * x_{15} * x_{18}/(x_{18} + k_{18}) + k_{19} * x_{17} * x_{18}/(x_{18} + k_{48}) \\
&\quad - k_{21} * x_{31} * x_{19}/(x_{19} + k_{22}) \\
\dot{x}_{20} &= -k_{23} * x_{19} * x_{20}/(x_{20} + k_{24}) + k_{25} * x_{31} * x_{21}/(x_{21} + k_{26}) \\
\dot{x}_{21} &= k_{23} * x_{19} * x_{20}/(x_{20} + k_{24}) - k_{25} * x_{31} * x_{21}/(x_{21} + k_{26}) \\
\dot{x}_{22} &= -k_{29} * x_4 * x_{22}/(x_{22} + k_{30}) - k_{31} * x_{12} * x_{22}/(x_{22} + k_{32}) \\
\dot{x}_{23} &= k_{29} * x_4 * x_{22}/(x_{22} + k_{30}) + k_{31} * x_{12} * x_{22}/(x_{22} + k_{32}) \\
\dot{x}_{24} &= -k_{33} * x_{23} * x_{24}/(x_{24} + k_{34}) \\
\dot{x}_{25} &= k_{33} * x_{23} * x_{24}/(x_{24} + k_{34}) \\
\dot{x}_{26} &= -k_{37} * x_6 * x_{26}/(x_{26} + k_{38}) \\
\dot{x}_{27} &= k_{37} * x_6 * x_{26}/(x_{26} + k_{38}) \\
\dot{x}_{28} &= -k_{39} * x_{27} * x_{28}/(x_{28} + k_{40}) + k_{41} * x_{30} * x_{29}/(x_{29} + k_{42}) \\
\dot{x}_{29} &= k_{39} * x_{27} * x_{28}/(x_{28} + k_{40}) - k_{41} * x_{30} * x_{29}/(x_{29} + k_{42}) \\
\dot{x}_{30} &= 0 \\
\dot{x}_{31} &= 0 \\
\dot{x}_{32} &= 0
\end{aligned}
\tag{7}
$$

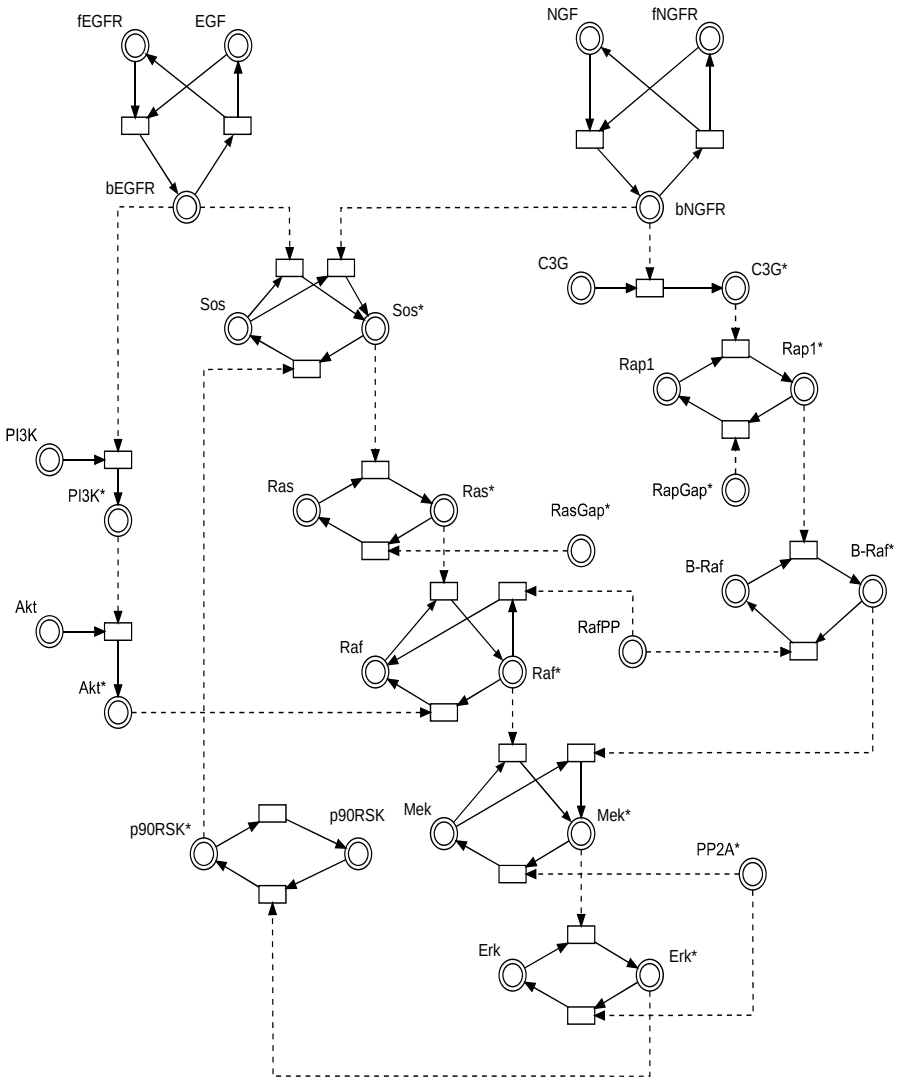

Fig. 3. The HFPN model of the EGF-NGF signaling pathway [14]. The HFPN representation of chemical networks is detailed in [15]. In a few words, circles (places) represent species, squares (transitions) represent reactions, and arrows represent how the species are involved in the reactions, dashed arrows mean that the species are not consumed (like in enzyme reactions).