A Recursive Method for Approximate Inference in Discrete Dynamic Bayesian Networks using Interface Junction Trees (submitted August 2020)

Huange Wang, Martin Neil, Norman Fenton

Abstract—Dynamic Bayesian Networks (DBNs) generalize Hidden Markov Models and Kalman Filter Models and provide a general framework for modelling and inference of discrete-time stochastic processes with Markov properties. Since exact inference in large or dense DBNs is often infeasible due to heavy computational overheads, a robust approximate inference approach is required. This paper presents the Forward-Backward Interface (FBI) algorithm, a recursive method for approximate inference in discrete DBNs, whose novelty lies in two respects: (1) it introduces the Maximum Retrospective Length, which prompts the minimum size of the temporal window required for recursive backward inference; (2) it derives a main junction tree and two interface junction trees from every temporal window, and uses these trees to propose an approximate message passing scheme. Both theoretical and empirical evidence demonstrate the validity of the FBI algorithm. Compared with Murphy's interface algorithm, the FBI algorithm can significantly reduce the space complexity of inference in discrete DBNs with large forward interface, for only a small loss in inference accuracy and compared with the Boyen-Koller algorithm, it not only supports automated decomposition of interface but also delivers more accurate inference results.

Index Terms—Graphs algorithms, Markov processes, Network problems, Probabilistic algorithms, Stochastic processes, Trees, Time series analysis

1 Introduction

Dynamic Bayesian Networks (DBNs) extend Bayesian Networks (BNs) by modeling temporal and non-temporal dependencies simultaneously to allow inference over discrete time steps. DBNs also generalize Hidden Markov Models and Kalman Filter Models allowing factored state space and arbitrary probability distributions. Given their potential to handle time series and sequential data, DBNs have a wide range of practical applications, including image and vision computing [1], [2], speech and gesture recognition [3], [4], [5], [6], traffic monitoring and prediction [7], [8], reverse engineering of biological networks [9], [10], medical decision support [11], [12], and forensic data analysis [13].

Like a BN, a DBN is a directed acyclic graph (DAG) whose nodes represent random variables and whose arcs represent conditional dependencies between the nodes. But unlike BNs, arcs in DBNs are divided into two categories: intra-slice and inter-slice, to represent temporal and non-temporal dependencies. The most common DBNs in the literature are first-order Markov and are time-invariant. The first-order Markov property states that the future is independent of the past given the present; or graphically, that the inter-slice arcs can only exist between adjacent slices. Time-invariance constrains conditional dependencies, between nodes in the same slice and between nodes in adjacent slices, such that they do not change over time. Obviously, a time-invariant first-order Markov DBN can be fully represented by its first two slices, i.e. a 2-slice Temporal BN (2-TBN) expressing $P(\boldsymbol{X}_0, \boldsymbol{y}_0)$ and $P(\boldsymbol{X}_{t+1}, \boldsymbol{y}_{t+1} | \boldsymbol{X}_t, \boldsymbol{y}_t)$, where $\boldsymbol{X}_t = \{X_t^1, \dots, X_t^n\}$ and $\boldsymbol{y}_t = \{y_t^1, \dots, y_t^m\}$ are, respectively, the hidden (also called unobservable) nodes and an instantiation of the observable nodes in slice $t \in \mathbb{Z}^{\geq 0}$, n and m are, respectively, the cardinalities of $\boldsymbol{X}_t$ and $\boldsymbol{y}_t$. According to the conditional dependencies in the 2-TBN, both $P(\boldsymbol{X}_0, \boldsymbol{y}_0)$ and $P(\boldsymbol{X}_{t+1}, \boldsymbol{y}_{t+1} | \boldsymbol{X}_t, \boldsymbol{y}_t)$ can be factorized as $\prod_{i=1}^{n} P\left(X_t^i | \boldsymbol{pa}(X_t^i)\right) \prod_{j=1}^{m} P\left(y_t^j | \boldsymbol{pa}(y_t^j)\right)$, where $\boldsymbol{pa}(X_t^i)$ are the parent nodes of X_t^i.

To our knowledge, there exist very few approaches to exact inference in discrete DBNs, and these approaches can become practically infeasible in large or dense DBNs due to the curse of dimensionality and associated computational overhead. One naive method is to unroll a DBN to an equivalent BN and apply the well-known Junction Tree (JT) algorithm [14]. The JT algorithm performs belief propagation on a JT, which is a tree decomposition that maps the nodes of a BN into a set of cliques satisfying the Running Intersection Property (RIP). When applied to large or dense DBNs, this can produce a JT with a large treewidth where inference, due to its high space complexity, is intractable. An alternative method is to use Murphy's interface algorithm [15] that implements exact inference recursively in a time-invariant first-order Markov

• *H. Wang, M. Neil and N. Fenton are with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London E1 4NS.*
E-mail: huange.wang@qmul.ac.uk, m.neil@qmul.ac.uk, n.fenton@qmul.ac.uk.

DBN. Given that $\boldsymbol{I}_t$, the forward interface in slice t (i.e. the set of nodes in slice t with outgoing arcs to slice $(t+1)$), is sufficient to d-separate the past from the future, the interface algorithm constructs a JT for every 1.5-TBN (a TBN is composed of $\boldsymbol{I}_t$, $\boldsymbol{X}_{t+1}$, $\boldsymbol{y}_{t+1}$ and related arcs), and then uses the JT algorithm to perform exact inference in each JT separately and passes the Joint Probability Distribution (JPD) of their sepset (i.e. intersection) between every two adjacent JTs. Note that the interface algorithm enforces the constraint that $\boldsymbol{I}_t$ and $\boldsymbol{I}_{t+1}$ in each 1.5-TBN form a clique in the corresponding JT separately, as $P(\boldsymbol{I}_t)$ and $P(\boldsymbol{I}_{t+1})$ are required for exact inference. This constraint, however, makes the interface algorithm unsuitable for DBNs with large forward interfaces, because the lower bound of its space complexity increases exponentially with the cardinality of the forward interface.

From a practical perspective, approximate inference may promise a reasonable compromise between computational overhead and accuracy. A mainstream approach to approximate inference in DBNs is stochastic sampling that draws, using Monte Carlo methods, a number of samples, each of which is a full instantiation of the variables studied. The empirical distribution of the samples is then taken as an approximation to the true distribution. The most representative algorithms in this class include the Metropolis-Hastings (M-H) algorithm [21], [22], Gibbs sampling [23], [24] and Particle Filtering (PF) [25]. The M-H algorithm and Gibbs sampling are instances of Markov Chain Monte Carlo (MCMC) methods that aim to generate samples in a sequential manner: where every newly obtained sample is used as a stepping stone to generate the next one, hence producing a chain with the first-order Markov property. PF differs from the MCMC methods in that it performs sequential importance sampling and resampling to approximate the true distribution by a population of weighted particles. It is widely recognized, however, that Monte Carlo methods are most likely to fail in large-scale models [16]. For example, to truly approximate a high-dimensional distribution, PF requires sample size to grow exponentially with the number of variables, and the MCMC methods suffer from an exponential growth in convergence and mixing rates. None of these challenges are technically trivial, as actual computing resources usually cannot be devoted to calculation involving very large sample sizes, or too slow convergence and mixing rates.

In contrast with stochastic sampling, another mainstream approach to approximate inference in DBNs, namely deterministic methods, resorts to tree-based message passing instead of Monte Carlo methods, and therefore performs better in terms of speed. For nonlinear and/or non-Gaussian state space where exact inference is typically intractable, deterministic methods can achieve approximate inference by first converting continuous variables into discrete ones [17], [18]. However, although it is always theoretically feasible, exact inference in discrete state space can be computationally prohibitive. Recall that Murphy's interface algorithm is inefficient for discrete DBNs with a large forward interface. In such cases a possible solution may be the Boyen-Koller (BK) algorithm [19], which is a derivative of the interface algorithm and is proposed for approximate inference in discrete time-invariant first-order Markov DBNs. In each recursive step, the BK algorithm takes $\tilde{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ as input, where $\tilde{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ is an approximate decomposition of $P(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ and $\boldsymbol{y}_{0:t}=\{\boldsymbol{y}_0,\boldsymbol{y}_1,\dots,\boldsymbol{y}_t\}$. Next, it applies the JT algorithm to the 1.5-TBN containing $\boldsymbol{I}_t$, $\boldsymbol{X}_{t+1}$ and $\boldsymbol{y}_{t+1}$ to yield $\hat{P}\left(\boldsymbol{I}_{t+1}|\boldsymbol{y}_{0:(t+1)}\right)$, which is an initial approximation to $P\left(\boldsymbol{I}_{t+1}|\boldsymbol{y}_{0:(t+1)}\right)$. Last, it decomposes $\hat{P}\left(\boldsymbol{I}_{t+1}|\boldsymbol{y}_{0:(t+1)}\right)$ into a more aggressive approximation $\tilde{P}\left(\boldsymbol{I}_{t+1}|\boldsymbol{y}_{0:(t+1)}\right)$ for use in the next recursive step. Note that the BK algorithm partitions the forward interface into smaller disjoint cliques by neglecting weak dependencies between the nodes. That is, the JPD of the forward interface is not marginalized out of a single clique's potential; instead, it is approximated by the product of a few marginals, each of which is marginalized out of the potential of a clique containing only a subset of the forward interface. Unfortunately, identifying the optimal partition of the forward interface is not automated in the BK algorithm. It remains an experimental process reliant on human guesswork, which is especially difficult where the JPD of the forward interface cannot be computed exactly in the first place.

In this paper, we present a novel recursive method called the Forward-Backward Interface (FBI) algorithm for approximate inference in discrete time-invariant first-order Markov DBNs. This method automatically decomposes a DBN to minimize the interface used for recursive backward inference while maximizing the accuracy of inference. It applies where temporal arcs can either be persistent (i.e. arcs of the form X_t^i to X_{t+1}^i) or non-persistent (i.e. arcs of the form X_t^i to X_{t+1}^j where $i \neq j$), and nodes in the forward interface can either be hidden or observable. Technically, it has two major innovations. First, it introduces the concept of Maximum Retrospective Length (MRL), which prompts the minimum size of temporal window required for recursive backward inference in a given DBN. Second, from the minimum temporal window, it constructs a main JT, two interface JTs and a message passing scheme for approximate forward and backward inference, respectively. Compared to Murphy's interface algorithm our method has little loss in inference accuracy but can greatly reduce the space complexity of inference for DBNs with a large forward interface. Compared to the BK algorithm our method produces smaller inference errors by accommodating more comprehensive associations between a larger scope of interface nodes and does so automatically without intervention.

Fig. 1 shows a graphical summary of the FBI algorithm, where each step is described in detail in the rest of the paper. Specifically, Section 2 describes how to compute the MRL for a given DBN. Section 3 covers how to construct the main JT, the forward interface JT and the backward interface JT determined by the MRL. Section 4 presents the message passing protocol designed for recursive forward inference including filtering (i.e. estimating the posterior distribution of a present state) and prediction (i.e. estimating the posterior distribution of a future state). Analogously, Section 5 presents the message passing protocol designed for recursive backward inference, also known as smoothing (i.e. estimating the posterior distribution of a past state). Section 6 proves the space complexity and inference accuracy of the proposed method. In Section 7, we apply the FBI algorithm to representative example DBNs, and empirically compare the results against those of other competing algorithms. Finally, section 8 concludes the paper and discusses future work.

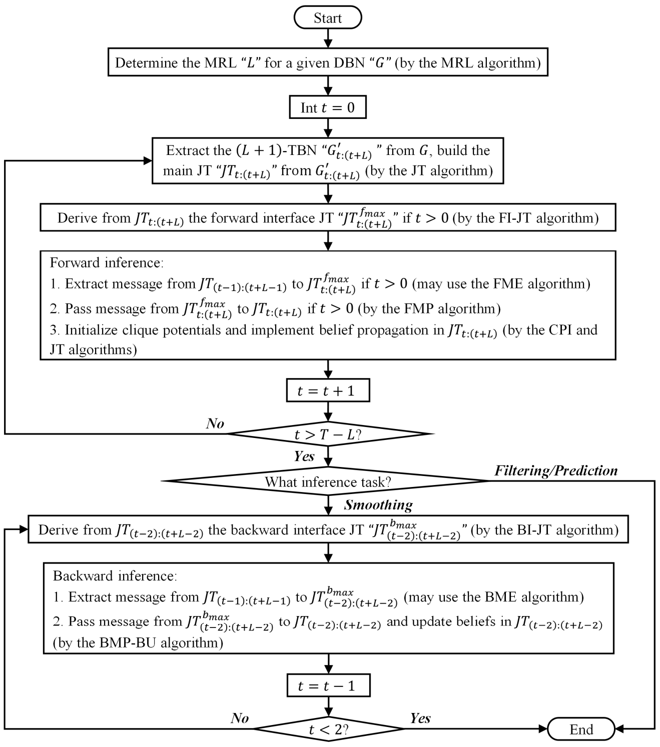

Start
Determine the MRL "L" for a given DBN "G" (by the MRL algorithm)
Int $t = 0$
Extract the $(L+1)$-TBN "$G'_{t:(t+L)}$" from G, build the main JT "$JT_{t:(t+L)}$" from $G'_{t:(t+L)}$ (by the JT algorithm)
Derive from $JT_{t:(t+L)}$ the forward interface JT "$JT^{fmax}_{t:(t+L)}$" if $t > 0$ (by the FI-JT algorithm)
Forward inference:
1. Extract message from $JT_{(t-1):(t+L-1)}$ to $JT^{fmax}_{t:(t+L)}$ if $t > 0$ (may use the FME algorithm)
2. Pass message from $JT^{fmax}_{t:(t+L)}$ to $JT_{t:(t+L)}$ if $t > 0$ (by the FMP algorithm)
3. Initialize clique potentials and implement belief propagation in $JT_{t:(t+L)}$ (by the CPI and JT algorithms)
$t = t + 1$
No
$t > T - L$?
Yes
What inference task?
Filtering/Prediction
Smoothing
Derive from $JT_{(t-2):(t+L-2)}$ the backward interface JT "$JT^{bmax}_{(t-2):(t+L-2)}$" (by the BI-JT algorithm)
Backward inference:
1. Extract message from $JT_{(t-1):(t+L-1)}$ to $JT^{bmax}_{(t-2):(t+L-2)}$ (may use the BME algorithm)
2. Pass message from $JT^{bmax}_{(t-2):(t+L-2)}$ to $JT_{(t-2):(t+L-2)}$ and update beliefs in $JT_{(t-2):(t+L-2)}$ (by the BMP-BU algorithm)
$t = t - 1$
No
$t < 2$?
Yes
End

Fig. 1. A flow chart of the FBI algorithm for approximate inference in discrete time-invariant first-order Markov DBNs.

2 Determining the Maximum Retrospective Length

Proposition *For any time-invariant first-order Markov DBN with persistent arcs only, the minimum temporal window required for recursive forward and backward inference is the 1.5-TBN, which consists of $\boldsymbol{I}_{t-1}$, $\boldsymbol{X}_t$, $\boldsymbol{y}_t$ and the arcs between them. For any time-invariant first-order Markov DBN with non-persistent arcs, the minimum temporal window required for recursive forward inference is the 1.5-TBN while that required for recursive backward inference is the $(L+1)$-TBN, which consists of $\boldsymbol{X}_{(t-L):t}$, $\boldsymbol{y}_{(t-L):t}$ and the arcs between them, where L is the Maximum Retrospective Length (MRL) of the given DBN and can be determined by the MRL algorithm described in the pseudocode shown below.*

MRL algorithm:

```
Input: a given DBN $G$
Let $L = 1$, $\boldsymbol{id}^R = \emptyset$, $\boldsymbol{X}^R_{T-L} = \emptyset$, $\boldsymbol{X}^{R\_add}_{T-L} = \emptyset$ and $\boldsymbol{id}^R_{T-L} = \emptyset$
for all $i \in \{1, \dots, n\}$, where $n$ is the cardinality of $\boldsymbol{X}_t$
    if $X^i_{T-L}$ simultaneously meets the three conditions:
        (1) It has no descendant node in slice $(T-L+1)$
        (2) All its descendant nodes in slice $(T-L)$, if any, are hidden
        (3) It has at least one parent node in slice $(T-L-1)$
        $\boldsymbol{X}^R_{T-L} = \boldsymbol{X}^R_{T-L} \cup X^i_{T-L}$
        $\boldsymbol{X}^{R\_add}_{T-L} = \boldsymbol{X}^{R\_add}_{T-L} \cup X^i_{T-L}$
        $\boldsymbol{id}^R_{T-L} = \boldsymbol{id}^R_{T-L} \cup i$
    end if
end for
while $\boldsymbol{X}^{R\_add}_{T-L} \neq \emptyset$
    Let $\boldsymbol{id}^{\bar{R}}_{T-L} = \{1, \dots, n\} \backslash \boldsymbol{id}^R_{T-L}$ and $\boldsymbol{id}^{R^*}_{T-L} = \emptyset$
    for all $i \in \boldsymbol{id}^{\bar{R}}_{T-L}$
        if $X^i_{T-L} \in \boldsymbol{des}(\boldsymbol{X}^{R\_add}_{T-L})$ where $\boldsymbol{des}(\boldsymbol{X}^{R\_add}_{T-L})$
           Denotes the descendant nodes of $\boldsymbol{X}^{R_{add}}_{T-L}$
            $\boldsymbol{X}^R_{T-L} = \boldsymbol{X}^R_{T-L} \cup X^i_{T-L}$
            $\boldsymbol{X}^{R\_add}_{T-L} = \boldsymbol{X}^{R\_add}_{T-L} \cup X^i_{T-L}$
            $\boldsymbol{id}^{R^*}_{T-L} = \boldsymbol{id}^R_{T-L} \cup i$
        end if
    end for
    if $\boldsymbol{id}^{R^*}_{T-L} \neq \emptyset$
        $\boldsymbol{id}^R_{T-L} = \boldsymbol{id}^R_{T-L} \cup \boldsymbol{id}^{R^*}_{T-L}$
        $\boldsymbol{id}^{\bar{R}}_{T-L} = \boldsymbol{id}^{\bar{R}}_{T-L} \backslash \boldsymbol{id}^{R^*}_{T-L}$
    end if
    $\boldsymbol{id}^R(L) = \boldsymbol{id}^R_{T-L}$
    $L = L + 1$
    Let $\boldsymbol{X}^R_{T-L} = \emptyset$, $\boldsymbol{X}^{R\_add}_{T-L} = \emptyset$, $\boldsymbol{id}^R_{T-L} = \boldsymbol{id}^R_{T-L+1}$ and
    $\boldsymbol{id}^{\bar{R}}_{T-L} = \boldsymbol{id}^{\bar{R}}_{T-L+1}$
    for all $i \in \boldsymbol{id}^R_{T-L}$
        $\boldsymbol{X}^R_{T-L} = \boldsymbol{X}^R_{T-L} \cup X^i_{T-L}$
    end for
    for all $i \in \boldsymbol{id}^{\bar{R}}_{T-L}$
        if $X^i_{T-L}$ simultaneously meets the three conditions:
            (1) all its descendant nodes in slice $(T-L+1)$ belong only to $\boldsymbol{X}^R_{T-L+1}$
            (2) all its descendant nodes in slice $(T-L)$, if any, are hidden
            (3) it has at least one parent node in slice $(T-L-1)$
            $\boldsymbol{X}^R_{T-L} = \boldsymbol{X}^R_{T-L} \cup X^i_{T-L}$
            $\boldsymbol{X}^{R_{add}}_{T-L} = \boldsymbol{X}^{R_{add}}_{T-L} \cup X^i_{T-L}$
            $\boldsymbol{id}^R_{T-L} = \boldsymbol{id}^R_{T-L} \cup i$
        end if
    end for
end while
$\boldsymbol{id}^R(L) = \boldsymbol{id}^R_{T-L}$
Output: $L$ and $\boldsymbol{id}^R$
```

The MRL algorithm takes a given DBN, G, as input and has two outputs:

1. L, the MRL.
2. $\boldsymbol{id}^R$, a one-dimensional array with L elements.

L and $\boldsymbol{id}^R$ are used to construct three classes of JTs from G, and the three classes of JTs can be connected for recursive forward and backward inference. Details on how to construct and connect the three classes of JTs will be given in Section 3. In order to calculate L and $\boldsymbol{id}^R$, the MRL algorithm, as shown in its pseudocode above, needs to first calculate several intermediate variables, including:

1. $\boldsymbol{X}_t^R$, the set of nodes in slice t whose posterior distribution needs to be updated with the message passed from nodes in slice $(t-1)$.
2. $\boldsymbol{id}_t^R$, the superscripts of nodes in $\boldsymbol{X}_t^R$.
3. $\boldsymbol{id}_t^{\bar{R}}$, the relative complement of $\boldsymbol{id}_t^R$ with respect to $\{1,\dots,n\}$, i.e. $\boldsymbol{id}_t^{\bar{R}} = \{1,\dots,n\}\backslash\boldsymbol{id}_t^R$.
4. $\boldsymbol{X}_t^{R\_add}$, the set of nodes in slice t whose superscripts belong to $\boldsymbol{id}_t^R\backslash\boldsymbol{id}_{t-1}^R$.
5. $\boldsymbol{id}_t^{R^*}$, the superscripts of nodes in $\boldsymbol{X}_t^{R\_add}$ that have no parent nodes in slice $(t-1)$.

For proof of the proposition, please refer to Appendix A, where two core single-step inference rules, namely the "absorption from child to parents" rule and the "absorption from parents to child" rule, are introduced as the cornerstone of our work. To put it shortly, the two rules indicates the only two possible flow directions of message passing. And in particular, the update on the child node can be transmitted to all its parent nodes, whereas the update on each parent node is essential for the update of the child node.

For illustrative purposes, Fig. 2a shows an example DBN. We apply the MRL algorithm to this DBN and summarize the values of all intermediate variables and the two outputs in Table 1. According to the proposition, we know that given $L=3$, 4-TBN is the minimum temporal window required for recursive backward inference in this example. To explain the logic of recursive backward inference succinctly here, Fig. 2b uses different node representations to divide the hidden nodes into four groups. Assume $P(\boldsymbol{X}_t|\boldsymbol{y}_{0:T})$, where $\boldsymbol{X}_t = \{X_t^1,\dots,X_t^6\}$, $\boldsymbol{y}_{0:T} = \{y_0,y_1,\dots,y_T\}$ and $t<T$, has been obtained from the previous recursive step, the Marginal Posterior Distribution (MPD) of each node with double-solid line can be sequentially obtained by following the "absorption from child to parents" rule. Then, the MPD of each node with double-dashed line can be sequentially obtained by following the "absorption from parents to child" rule. But, the MPD of each node with single-dotted line cannot be obtained either way in the present recursive step. Fortunately, we can obtain $P(\boldsymbol{X}_{t-1}|\boldsymbol{y}_{0:T})$ and also $P(\boldsymbol{X}_{t-3}^{1:2},\boldsymbol{X}_{t-2}^{1:4},\boldsymbol{X}_{t-1}|\boldsymbol{y}_{0:T})$, where $\boldsymbol{X}_{t-3}^{1:2} = \{X_{t-3}^1,X_{t-3}^2\}$ and $\boldsymbol{X}_{t-2}^{1:4} = \{X_{t-2}^1,X_{t-2}^2,X_{t-2}^3,X_{t-2}^4\}$, using the chain rule. Given that the slices are self-similar, we will then obtain $P(X_{t-2}^5|\boldsymbol{y}_{0:T})$, $P(X_{t-2}^6|\boldsymbol{y}_{0:T})$, $P(X_{t-3}^3|\boldsymbol{y}_{0:T})$ and $P(X_{t-3}^4|\boldsymbol{y}_{0:T})$ in the next recursive step by sliding the temporal window backwards to cover slices $(t-4)$ to $(t-1)$. Similarly, $P(X_{t-3}^5|\boldsymbol{y}_{0:T})$ and $P(X_{t-3}^6|\boldsymbol{y}_{0:T})$ will be obtained by further sliding the temporal window backwards to cover slices $(t-5)$ to $(t-2)$.

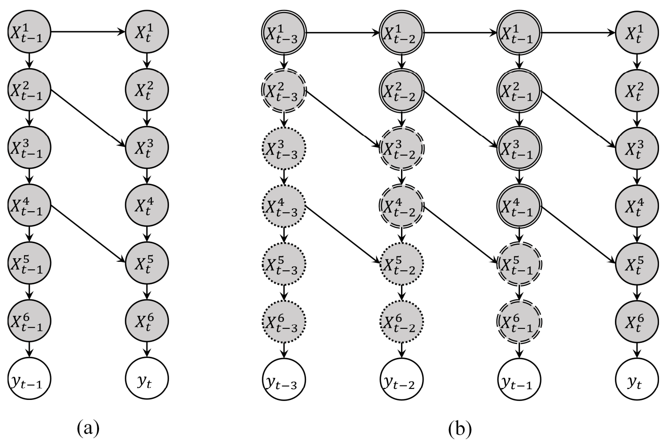


Fig. 2. (a) An example DBN, where hidden and observable nodes are shaded and unshaded, respectively. (b) A 4-TBN is extracted from the example DBN. Assume $P(\boldsymbol{X}_t|\boldsymbol{y}_{0:T})$, where $\boldsymbol{X}_t = \{X_t^1,\dots,X_t^6\}$, $\boldsymbol{y}_{0:T} = \{y_0,y_1,\dots,y_T\}$ and $t<T$, has been derived from the previous recursive step, the marginal posterior distribution of each node with double-solid line can be sequentially obtained by following the "absorption from child to parents" rule; then, the marginal posterior distribution of each node with double-dashed line can be sequentially obtained by following the "absorption from parents to child" rule; but, the marginal posterior distribution of each node with single-dotted line cannot be obtained either way in this 4-TBN.

TABLE 1
THE INTERMEDIATE VARIABLES AND TWO OUTPUTS OBTAINED BY APPLYING THE MRL ALGORITHM TO THE EXAMPLE DBN IN FIG.2A

Variable	Value
$\boldsymbol{X}_{T-1}^R$	$\{X_{T-1}^5,X_{T-1}^6\}$
$\boldsymbol{X}_{T-1}^{R\_add}$	$\{X_{T-1}^5,X_{T-1}^6\}$
$\boldsymbol{id}_{T-1}^R$	$\{5,6\}$
$\boldsymbol{id}_{T-1}^{\bar{R}}$	$\{1,2,3,4\}$
$\boldsymbol{id}_{T-1}^{R^*}$	$\{6\}$
$\boldsymbol{X}_{T-2}^R$	$\{X_{T-2}^3,X_{T-2}^4,X_{T-2}^5,X_{T-2}^6\}$
$\boldsymbol{X}_{T-2}^{R\_add}$	$\{X_{T-2}^3,\ X_{T-2}^4\}$
$\boldsymbol{id}_{T-2}^R$	$\{3,4,5,6\}$
$\boldsymbol{id}_{T-2}^{\bar{R}}$	$\{1,2\}$
$\boldsymbol{id}_{T-2}^{R^*}$	$\{4\}$
$\boldsymbol{X}_{T-3}^R$	$\{X_{T-3}^3,X_{T-3}^4,X_{T-3}^5,X_{T-3}^6\}$
$\boldsymbol{X}_{T-3}^{R\_add}$	$\emptyset$
$\boldsymbol{id}_{T-3}^R$	$\{3,4,5,6\}$
$\boldsymbol{id}_{T-3}^{\bar{R}}$	$\{1,2\}$
L	3
$\boldsymbol{id}^R$	$\{\{5,6\},\{3,4,5,6\},\{3,4,5,6\}\}$

Although extracting $P(\boldsymbol{X}_{t-1}|\boldsymbol{y}_{0:T})$ in the present step and passing it to the next step is sufficient for recursive backward inference, we recommend extracting and passing $P(\boldsymbol{X}_{t-3}^{1:2},\boldsymbol{X}_{t-2}^{1:4},\boldsymbol{X}_{t-1}|\boldsymbol{y}_{0:T})$ instead of $P(\boldsymbol{X}_{t-1}|\boldsymbol{y}_{0:T})$. This is based on the following three facts:

1. $\{\boldsymbol{X}_{t-3}^{1:2},\boldsymbol{X}_{t-2}^{1:4},\boldsymbol{X}_{t-1}\}$ are involved in both the present and the next recursive steps.
2. $P(\boldsymbol{X}_{t-3}^{1:2},\boldsymbol{X}_{t-2}^{1:4},\boldsymbol{X}_{t-1}|\boldsymbol{y}_{0:T})$ can be obtained in the present recursive step.

3. The sub-DBN composed from $\boldsymbol{X}_{t-3}^{1:2}, \boldsymbol{X}_{t-2}^{1:4}, \boldsymbol{X}_{t-1}$, and related arcs, can be converted into a JT; while the sub-DBN composed merely of $\boldsymbol{X}_{t-1}$, and related arcs, cannot be converted into a JT due to the lack of X_{t-2}^{2} and X_{t-2}^{4} in v-structures. That is to say, $P(\boldsymbol{X}_{t-3}^{1:2}, \boldsymbol{X}_{t-2}^{1:4}, \boldsymbol{X}_{t-1} | \boldsymbol{y}_{0:T})$ rather than $P(\boldsymbol{X}_{t-1} | \boldsymbol{y}_{0:T})$ can be decomposed into the product of smaller terms.

$\boldsymbol{id}^R$ can be used to identify the nodes with double-solid and double-dashed lines in any $(L+1)$-TBN of G. Generally, in the $(L+1)$-TBN consisting of slices $(t-L)$ to t, the superscripts of these nodes in slice $(t-1)$ are $\{1, \dots, n\}$, and the superscripts of these nodes in slice $(t-i)_{\forall i \in [2,L]}$ are $\left(\{1, \dots, n\} \backslash \boldsymbol{id}^R(i)\right) \cup \left(\boldsymbol{id}^R(i) \backslash \boldsymbol{id}^R(i-1)\right) = \{1, \dots, n\} \backslash \boldsymbol{id}^R(i-1)$.

3. Constructing Three Classes of Junction Trees

Given the outputs of the MRL algorithm, we can extract slices t to $(t+L)$ from G as $G'_{t:(t+L)}$, and further construct three classes of JTs from $G'_{t:(t+L)}$ for all $t \in [0, T-L]$. The first class refers to JTs constructed by the JT algorithm from each $G'_{t:(t+L)}$, and is called the main JTs. These serve as the basis for deriving the other two classes, namely the forward interface JTs and the backward interface JTs. Fig. 3a and Fig. 3b show, respectively, how different classes of JTs are connected to support recursive forward and backward inference, where $JT_{t:(t+L)}$ denotes the main JT constructed from $G'_{t:(t+L)}$, $JT^f_{t:(t+L)}$ and $JT^b_{t:(t+L)}$ denote, respectively, the forward inference JT and the backward interface JT derived from $JT_{t:(t+L)}$. See Section 3.1 and 3.2 for a detailed description of the steps to derive $JT^f_{t:(t+L)}$ and $JT^b_{t:(t+L)}$ from $JT_{t:(t+L)}$.

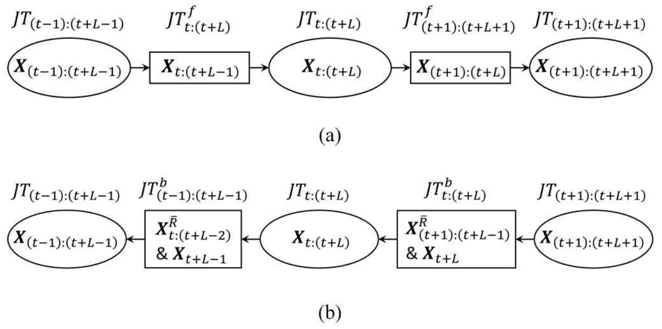

$JT_{(t-1):(t+L-1)}$
$JT^f_{t:(t+L)}$
$JT_{t:(t+L)}$
$JT^f_{(t+1):(t+L+1)}$
$JT_{(t+1):(t+L+1)}$
$\boldsymbol{X}_{(t-1):(t+L-1)}$
$\boldsymbol{X}_{t:(t+L-1)}$
$\boldsymbol{X}_{t:(t+L)}$
$\boldsymbol{X}_{(t+1):(t+L)}$
$\boldsymbol{X}_{(t+1):(t+L+1)}$
(a)
$JT_{(t-1):(t+L-1)}$
$JT^b_{(t-1):(t+L-1)}$
$JT_{t:(t+L)}$
$JT^b_{t:(t+L)}$
$JT_{(t+1):(t+L+1)}$
$\boldsymbol{X}_{(t-1):(t+L-1)}$
$\boldsymbol{X}^{\bar{R}}_{t:(t+L-2)}$ & $\boldsymbol{X}_{t+L-1}$
$\boldsymbol{X}_{t:(t+L)}$
$\boldsymbol{X}^{\bar{R}}_{(t+1):(t+L-1)}$ & $\boldsymbol{X}_{t+L}$
$\boldsymbol{X}_{(t+1):(t+L+1)}$
(b)

Fig. 3. (a) A schematic diagram shows how to connect the main JTs and the forward interface JTs to support recursive forward inference. (b) A schematic diagram shows how to connect the main JTs and the backward interface JTs to support recursive backward inference. In both diagrams, the domain of every main JT is given in an ellipse while the domain of every interface JT is given in a rectangle, where $\boldsymbol{X}_t^{\bar{R}} = \boldsymbol{X}_t \backslash \boldsymbol{X}_t^R$.

3.1 Constructing the Forward Interface Junction Tree

To derive $JT^f_{t:(t+L)}$ from $JT_{t:(t+L)}$, we follow the Forward Interface Junction Tree (FI-JT) algorithm given below.

FI-JT algorithm:

1. Remove from $JT_{t:(t+L)}$ all those cliques (and the connected sepsets and edges) that contain only nodes in slice $(t+L)$. Next, remove from each remaining clique and sepset the nodes belonging to slice $(t+L)$. Denote the resulting junction tree as $JT^f_{t:(t+L)}$.
2. Remove from $JT^f_{t:(t+L)}$ a leaf clique (and the connected sepset and edges) if it is a subset of the neighboring clique. Rewrite the resulting junction tree as $JT^f_{t:(t+L)}$. Repeat this step until every leaf clique in the newly obtained $JT^f_{t:(t+L)}$ is not a subset of its neighboring clique.

Remove from $JT^f_{t:(t+L)}$ a non-leaf clique C_i (and the connected sepsets and edges) if $C_i \subseteq C_j$, where $C_j \in \boldsymbol{nb}(C_i)$ and $\boldsymbol{nb}(C_i)$ denotes C_i's neighboring cliques. Connect C_j with each $C_k \in \boldsymbol{nb}(C_i) \backslash C_j$ by inserting the corresponding sepsets and edges. Rewrite the resulting junction tree as $JT^f_{t:(t+L)}$. Repeat this step until every non-leaf clique in the newly obtained $JT^f_{t:(t+L)}$ is not a subset of any of its neighboring cliques. At this point, $JT^f_{t:(t+L)}$ is a JT of maximal cliques of nodes belonging to slices t to $(t+L-1)$ (for proof, see Appendix B).

3.2 Constructing the Forward Interface Junction Tree

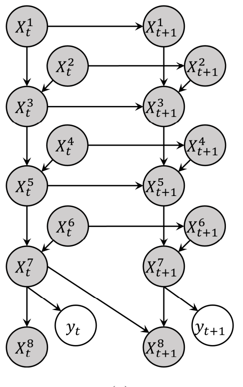
To derive $JT^b_{t:(t+L)}$ from $JT_{t:(t+L)}$, we follow the BI-JT (abbreviation of Backward Interface Junction Tree) algorithm given below.

BI-JT algorithm:

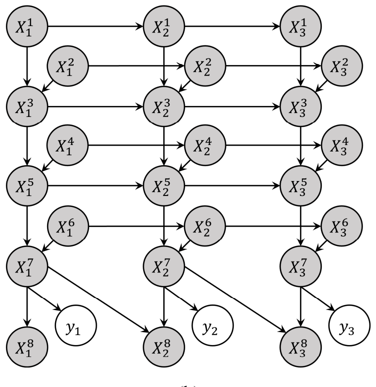
1. Remove from $JT_{t:(t+L)}$ all those cliques (and the connected sepsets and edges) that contain only nodes in $\{\boldsymbol{X}_t, \boldsymbol{y}_t, \boldsymbol{X}^R_{(t+1):(t+L-1)}\}$, where $\boldsymbol{X}^R_{t+i, \forall i \in [1, L-1]}$ can be identified by $\boldsymbol{id}^R(L-i)$, i.e. the superscripts of nodes in $\boldsymbol{X}^R_{t+i}$ correspond to $\boldsymbol{id}^R(L-i)$. Next, remove from each remaining clique and sepset the nodes belonging to $\{\boldsymbol{X}_t, \boldsymbol{y}_t, \boldsymbol{X}^R_{(t+1):(t+L-1)}\}$. Denote the resulting junction tree by $JT^b_{t:(t+L)}$.
2. Remove from $JT^b_{t:(t+L)}$ a leaf clique (and the connected sepset and edges) if it is a subset of the neighboring clique. Rewrite the resulting junction tree as $JT^b_{t:(t+L)}$. Repeat this step until every leaf clique in the newly obtained $JT^b_{t:(t+L)}$ is not a subset of its neighboring clique.
3. Remove from $JT^b_{t:(t+L)}$ a non-leaf clique C_i (and the connected sepsets and edges) if $C_i \subseteq C_j$, where $C_j \in \boldsymbol{nb}(C_i)$. Connect C_j with each $C_k \in \boldsymbol{nb}(C_i) \backslash C_j$ by inserting the corresponding sepsets and edges. Rewrite the resulting junction tree as $JT^b_{t:(t+L)}$. Repeat this step until every non-leaf clique in the newly obtained $JT^b_{t:(t+L)}$ is not a subset of any of its neighboring cliques. At this point, $JT^b_{t:(t+L)}$ is a JT of max-

imal cliques of nodes belonging to $\{\boldsymbol{X}^{\bar{R}}_{(t+1):(t+L-1)}, \boldsymbol{X}_{t+L}, \boldsymbol{y}_{(t+1):(t+L)}\}$, where $\boldsymbol{X}^{\bar{R}}_t = \boldsymbol{X}_t \backslash \boldsymbol{X}^R_t$.

In order to visually represent the mechanism of the FI-JT algorithm and the BI-JT algorithm, Fig. 4a shows a synthetic DBN, for which $L = 2$ and $\boldsymbol{id}^R = \{8,8\}$ are obtained using the MRL algorithm. $L = 2$ indicates that a main JT needs to be built for every 3-TBN using the JT algorithm as shown by the illustrative example in Fig. 4b and Fig. 5a. Given the main JT, $JT_{1:3}$ (Fig. 5a), a step-by-step walkthrough to construct the forward interface JT, $JT^f_{1:3}$ (Fig. 5b), and the backward interface JT, $JT^b_{1:3}$ (Fig. 5c), is provided in Fig. S1 and Fig. S2 (Fig. S hereafter refers to figure in the supplemental material), respectively.



(a)



(b)

Fig. 4. (a) An example DBN is expressed in the form of a 2-TBN. Hidden and observable nodes are represented by shaded and unshaded circles, respectively. Nodes in the same slice are named alphabetically. (b) For illustrative purposes, a 3-TBN is extracted from the DBN in (a).

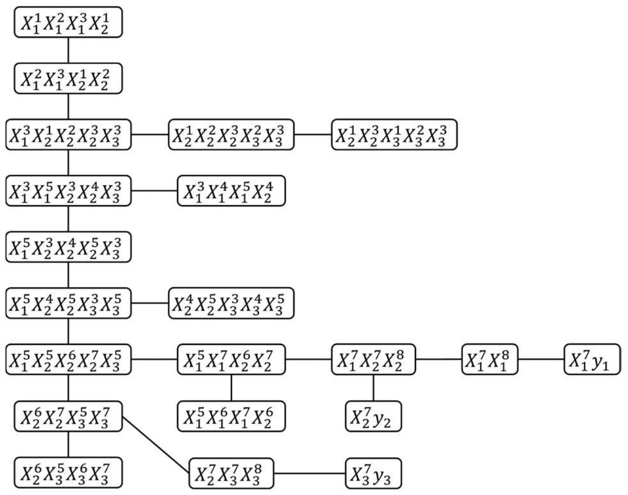
4 Forward Inference

$JT^f_{t:(t+L)}$ is designed to pass a decomposed message from $JT_{(t-1):(t+L-1)}$ to $JT_{t:(t+L)}$. Specifically, following the first round of belief propagation in $JT_{(t-1):(t+L-1)}$, all clique potentials in $JT^f_{t:(t+L)}$ can be extracted from $JT_{(t-1):(t+L-1)}$. These clique potentials, along with related sepset potentials, are then passed to $JT_{t:(t+L)}$, since they can jointly define the a priori factors of the first round of belief propagation in $JT_{t:(t+L)}$.

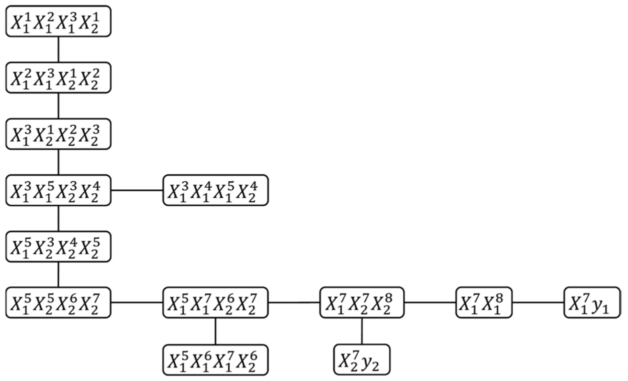
$JT^f_{t:(t+L)}$ is a JT containing nodes in slices t to $(t + L - 1)$. That is, $JT^f_{t:(t+L)}$ restores the connectivity between nodes in slices t to $(t + L - 1)$ but neglects the associations induced by slices prior to t. This indicates two facts:

First, the product of clique potentials divided by the product of sepset potentials in $JT^f_{t:(t+L)}$ represents an approximation to $P(\boldsymbol{X}_{t:(t+L-1)}|\boldsymbol{y}_{0:(t+L-1)})$.

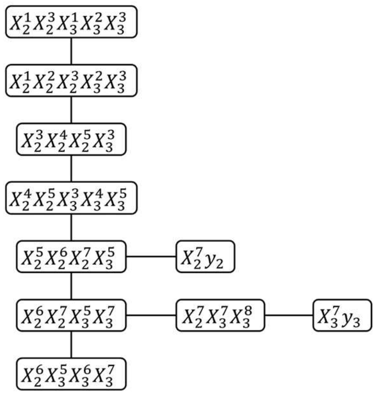
Second, the forward inference process after the first round of belief propagation in the very first main JT, $JT_{0:L}$, deals with approximate rather than exact beliefs. To be more precise, the forward inference process involves two approximations to $P(\boldsymbol{X}_{t:(t+L-1)}|\boldsymbol{y}_{0:(t+L-1)})$ for $\forall t \in [1, T - L]$. The first, denoted by $\hat{P}(\boldsymbol{X}_{t:(t+L-1)}|\boldsymbol{y}_{0:(t+L-1)})$, is the marginal obtained in the first round of belief propagation in $JT_{(t-1):(t+L-1)}$. The second, denoted by $\tilde{P}(\boldsymbol{X}_{t:(t+L-1)}|\boldsymbol{y}_{0:(t+L-1)})$, is an approximate decomposition of $\hat{P}(\boldsymbol{X}_{t:(t+L-1)}|\boldsymbol{y}_{0:(t+L-1)})$ and is passed from $JT_{(t-1):(t+L-1)}$ to $JT_{t:(t+L)}$ via $JT^f_{t:(t+L)}$.



(a)



(b)



(c)

Fig. 5. (a) For the 3-TBN in Fig. 4b, a main JT, $JT_{1:3}$, is constructed by the JT algorithm. (b) From $JT_{1:3}$, a forward interface JT, $JT^f_{1:3}$, is derived by the FI-JT algorithm. (c) From $JT_{1:3}$, a backward interface JT, $JT^b_{1:3}$, is derived by the BI-JT algorithm. For simplicity, all sepsets are omitted in (a), (b) and (c).

In this way, the approximate forward inference process can be abstractly summarized as below:

1. Pass $\tilde{P}(\boldsymbol{X}_{t:(t+L-1)}|\boldsymbol{y}_{0:(t+L-1)})$ from $JT_{(t-1):(t+L-1)}$ to $JT_{t:(t+L)}$ via $JT^{f}_{t:(t+L)}$;
2. Obtain $\hat{P}(\boldsymbol{X}_{t:(t+L)}|\boldsymbol{y}_{0:(t+L)})$ in the first round of belief propagation in $JT_{t:(t+L)}$, and then marginalize out $\hat{P}(\boldsymbol{X}_{(t+1):(t+L)}|\boldsymbol{y}_{0:(t+L)})$;
3. Decompose $\hat{P}(\boldsymbol{X}_{(t+1):(t+L)}|\boldsymbol{y}_{0:(t+L)})$ into $\tilde{P}(\boldsymbol{X}_{(t+1):(t+L)}|\boldsymbol{y}_{0:(t+L)})$ approximately so that step 1-3 can be performed recursively.

To elaborate on this recursive process, we reorganize it into the three main steps as detailed in sections 4.1 – 4.3. These three main steps are consistent with the description of forward inference in Fig. 1 and are implemented by several sub-steps, respectively.

4.1 Extracting Message from the Main JT to the Forward Interface JT

When extracting the clique potentials in $JT^{f}_{t:(t+L)}$ from $JT_{(t-1):(t+L-1)}$, we may encounter two mutually exclusive situations that require different solutions. In the first situation when clique $C_i \in JT^{f}_{t:(t+L)}$ is a subset of clique $C_j \in JT_{(t-1):(t+L-1)}$, the potential of C_i, ϕ_{C_i}, can be directly marginalized out of ϕ_{C_j}, the potential of clique C_j. In the second situation when C_i is not a subset of any clique in $JT_{(t-1):(t+L-1)}$, ϕ_{C_i} can be obtained using the Forward Message Extraction (FME) algorithm that performs belief propagation and marginalization in a temporary JT, $JT^{f_{C_i}}_{(t-1):(t+L-1)}$, where f_{C_i} denotes forward message extraction for C_i:

FME algorithm:

1. Extract from $JT_{(t-1):(t+L-1)}$ the subgraph formed by all those cliques (and the sepsets and edges between them) that contain at least one node in C_i. Denote the subgraph as the junction tree $JT^{f_{C_i}}_{(t-1):(t+L-1)}$.
2. Remove from $JT^{f_{C_i}}_{(t-1):(t+L-1)}$ a leaf clique C_j (and the connected sepsets and edges) if $(C_i \cap C_j) \subseteq (C_i \cap C_k)$ where $C_k \in \boldsymbol{nb}(C_j)$. Rewrite the resulting graph, which is still a JT satisfying the RIP, as $JT^{f_{C_i}}_{(t-1):(t+L-1)}$. Repeat this step until every leaf clique in the newly obtained $JT^{f_{C_i}}_{(t-1):(t+L-1)}$ is not a subset of its neighboring clique.
3. As nodes in C_i are all included in $JT^{f_{C_i}}_{(t-1):(t+L-1)}$, ϕ_{C_i} can be marginalized out of the JPD of all nodes in $JT^{f_{C_i}}_{(t-1):(t+L-1)}$, where that JPD can be expressed as the product of clique potentials divided by the product of sepset potentials in $JT^{f_{C_i}}_{(t-1):(t+L-1)}$.

Fig. 5b shows there are two cliques $\{X_1^5X_2^3X_2^4X_2^5\}$ and $\{X_1^5X_2^5X_2^6X_2^7\}$ in $JT^{f}_{1:3}$. Analogously, there are two cliques $\{X_2^5X_3^3X_3^4X_3^5\}$ and $\{X_2^5X_3^5X_3^6X_3^7\}$ in $JT^{f}_{2:4}$. In order to achieve approximate forward inference in $JT_{2:4}$, $\hat{P}(X_2^5X_3^3X_3^4X_3^5|\boldsymbol{y}_{0:3})$ and $\hat{P}(X_2^5X_3^5X_3^6X_3^7|\boldsymbol{y}_{0:3})$ need to be passed from $JT_{1:3}$ to $JT_{2:4}$ via $JT^{f}_{2:4}$. Since $\{X_2^5X_3^3X_3^4X_3^5\}$ is a subset of clique $\{X_2^4X_2^5X_3^3X_3^4X_3^5\}$ in $JT_{1:3}$, $\hat{P}(X_2^5X_3^3X_3^4X_3^5|\boldsymbol{y}_{0:3})$ can be simply extracted from $\hat{P}(X_2^4X_2^5X_3^3X_3^4X_3^5|\boldsymbol{y}_{0:3})$. In contrast, $\{X_2^5X_3^5X_3^6X_3^7\}$ is not a subset of any clique in $JT_{1:3}$. Thus, we need to follow the FME algorithm to generate $JT_{1:3}^{f_{\{X_2^5X_3^5X_3^6X_3^7\}}}$ as $\boxed{X_1^5X_2^5X_2^6X_2^7X_3^5} - \boxed{X_2^6X_2^7X_3^5X_3^7} - \boxed{X_2^6X_3^5X_3^6X_3^7}$, and compute $\hat{P}(X_2^5X_3^5X_3^6X_3^7|\boldsymbol{y}_{0:3})$ as the product of $\hat{P}(X_1^5X_2^5X_2^6X_2^7X_3^5|\boldsymbol{y}_{0:3})$, $\hat{P}(X_2^6X_2^7X_3^5X_3^7|\boldsymbol{y}_{0:3})$ and $\hat{P}(X_2^6X_3^5X_3^6X_3^7|\boldsymbol{y}_{0:3})$ divided by the product of $\hat{P}(X_2^6X_2^7X_3^5|\boldsymbol{y}_{0:3})$ and $\hat{P}(X_2^6X_3^5X_3^7|\boldsymbol{y}_{0:3})$, where all the five factors are available in $JT_{1:3}$.

4.2 Message Passing from the Forward Interface JT to the Main JT

Recall that $JT^{f}_{t:(t+L)}$ is a JT containing the maximal cliques of nodes in slices t to $(t+L-1)$. In other words, $JT^{f}_{t:(t+L)}$ compactly decompose the JPD of all nodes involved into the product of all clique potentials divided by the product of all sepset potentials. Thus, the message passing from $JT^{f}_{t:(t+L)}$ to $JT_{t:(t+L)}$ can be achieved through the Forward Message Passing (FMP) algorithm:

FMP algorithm:

1. Start with an arbitrary leaf clique in $JT^{f}_{t:(t+L)}$, say C_i, specify ϕ_{C_i} as the message passed from C_i to $JT_{t:(t+L)}$.
2. Move to C_i's neighboring clique in $JT^{f}_{t:(t+L)}$, say C_j, specify $\frac{\phi_{C_j}}{\phi_{S_{ij}}}$ as the message passed from C_j to $JT_{t:(t+L)}$, where S_{ij} denotes the sepset between C_i and C_j.
3. For all cliques $C_k \in \boldsymbol{nb}(C_j)\backslash C_i$, specify $\frac{\phi_{C_k}}{\phi_{S_{jk}}}$ as the message passed from C_k to $JT_{t:(t+L)}$. Continue this process along the remaining pathways in $JT^{f}_{t:(t+L)}$ until the message passed from every clique in $JT^{f}_{t:(t+L)}$ to $JT_{t:(t+L)}$ is specified.

Please note the use of different starting points in the FMP algorithm merely indicates that the clique and sepset potentials in $JT^{f}_{t:(t+L)}$ are grouped differently when passed to $JT_{t:(t+L)}$. For instance, when we start from the clique $\{X_1^1X_1^2X_1^3X_2^1\}$ in Fig. 5b, the messages passed from the five cliques $\{X_1^1X_1^2X_1^3X_2^1\}$, $\{X_1^2X_1^3X_2^1X_2^2\}$, $\{X_1^3X_2^1X_2^2X_2^3\}$, $\{X_1^3X_1^5X_2^3X_2^4\}$ and $\{X_1^3X_1^4X_1^5X_2^4\}$ to $JT_{1:3}$ are, respectively, $\hat{P}(X_1^1X_1^2X_1^3X_2^1|\boldsymbol{y}_{0:2})$, $\frac{\hat{P}(X_1^2X_1^3X_2^1X_2^2|\boldsymbol{y}_{0:2})}{\hat{P}(X_1^2X_1^3X_2^1|\boldsymbol{y}_{0:2})}$, $\frac{\hat{P}(X_1^3X_2^1X_2^2X_2^3|\boldsymbol{y}_{0:2})}{\hat{P}(X_1^3X_2^1X_2^2|\boldsymbol{y}_{0:2})}$, $\frac{\hat{P}(X_1^3X_1^5X_2^3X_2^4|\boldsymbol{y}_{0:2})}{\hat{P}(X_1^3X_2^3|\boldsymbol{y}_{0:2})}$ and $\frac{\hat{P}(X_1^3X_1^4X_1^5X_2^4|\boldsymbol{y}_{0:2})}{\hat{P}(X_1^3X_1^5X_2^4|\boldsymbol{y}_{0:2})}$. Instead, when we start from the clique $\{X_1^3X_1^4X_1^5X_2^4\}$ in Fig. 5b, the messages passed from those five cliques to $JT_{1:3}$ are, respectively, $\frac{\hat{P}(X_1^1X_1^2X_1^3X_2^1|\boldsymbol{y}_{0:2})}{\hat{P}(X_1^2X_1^3X_2^1|\boldsymbol{y}_{0:2})}$, $\frac{\hat{P}(X_1^2X_1^3X_2^1X_2^2|\boldsymbol{y}_{0:2})}{\hat{P}(X_1^3X_2^1X_2^2|\boldsymbol{y}_{0:2})}$, $\frac{\hat{P}(X_1^3X_2^1X_2^2X_2^3|\boldsymbol{y}_{0:2})}{\hat{P}(X_1^3X_2^3|\boldsymbol{y}_{0:2})}$,

$\frac{\hat{P}(X_1^3X_1^5X_2^3X_2^4|\boldsymbol{y}_{0:2})}{\hat{P}(X_1^3X_1^5X_2^4|\boldsymbol{y}_{0:2})}$ and $\hat{P}(X_1^3X_1^4X_1^5X_2^4|\boldsymbol{y}_{0:2})$. Thus, the choice of starting point makes no difference, because the set of numerators and denominators passed from $JT^f_{t:(t+L)}$ to $JT_{t:(t+L)}$ is always the same.

4.3 Clique Potential Initialization and Belief Propagation in the Main JT

The initial clique potentials in $JT_{t:(t+L)}$ are required for forward inference in $JT_{t:(t+L)}$. They can be defined by the Clique Potential Initialization (CPI) algorithm given the messages passed from the forward interface JT.

CPI algorithm:

1. Initialize each clique potential in $JT_{t:(t+L)}$ to the unit measure.
2. For all cliques $C_i \in JT^f_{t:(t+L)}$, find in $JT_{t:(t+L)}$ a clique C_j s.t. $C_i \subseteq C_j$, multiply ϕ_{C_j} by the message passed from C_i to $JT_{t:(t+L)}$.
3. For all nodes $X^i_{t+L} \in \boldsymbol{X}_{t+L}$, find in $JT_{t:(t+L)}$ a clique C_k that contains X^i_{t+L} and $\boldsymbol{pa}(X^i_{t+L})$, multiply ϕ_{C_k} by the conditional probability distribution $P\left(X^i_{t+L}|\boldsymbol{pa}(X^i_{t+L})\right)$.
4. Entre any observations $\boldsymbol{y}_{t+L}$ into $JT_{t:(t+L)}$.

In our running example, the clique $\{X_1^3X_2^1X_2^2X_2^3\}$ in $JT^f_{1:3}$ is a subset of the clique $\{X_1^3X_2^1X_2^2X_2^3X_3^3\}$ in $JT_{1:3}$, and the domain of $P(X_3^3|\boldsymbol{pa}(X_3^3))$ includes $\{X_2^3X_3^1X_3^2\}$, which is a subset of the clique $\{X_2^1X_2^3X_3^1X_3^2X_3^3\}$ in $JT_{1:3}$. Thus, by following the CPI algorithm, we initialize $\phi_{\{X_1^3X_2^1X_2^2X_2^3X_3^3\}}$ as $\hat{P}(X_1^3X_2^1X_2^2X_2^3|\boldsymbol{y}_{0:2})$ and $\phi_{\{X_2^1X_2^3X_3^1X_3^2X_3^3\}}$ as $P(X_3^3|X_2^3X_3^1X_3^2)$ in $JT_{1:3}$.

Upon initialization, $\tilde{P}(\boldsymbol{X}_{t:(t+L-1)}|\boldsymbol{y}_{0:(t+L-1)})$ (which is formally the product of all messages passed from $JT^f_{t:(t+L)}$ to $JT_{t:(t+L)}$), $P\left(X^i_{t+L}|\boldsymbol{pa}(X^i_{t+L})\right)$ for all $X^i_{t+L} \in \boldsymbol{X}_{t+L}$, and the observations $\boldsymbol{y}_{t+L}$ are all taken into account, that is, the prerequisites of belief propagation in $JT_{t:(t+L)}$ using the JT algorithm are all met. Next, we adopt the serial protocol of the JT algorithm as below:

1. Select an arbitrary clique in $JT_{t:(t+L)}$ as the root clique.
2. Collect messages from the leaf cliques to the root clique.
3. Distribute messages from the root clique to the leaf cliques.

5 Backward Inference

Subsequent to approximate forward inference described in Section 4, approximate backward inference can be achieved recursively by first extracting a set of messages from $JT_{(t+1):(t+L+1)}$, then passing these messages to $JT_{t:(t+L)}$ via $JT^b_{t:(t+L)}$and propagating them in $JT_{t:(t+L)}$.

5.1 Extracting Message from the Main JT to the Backward Interface JT

$P(\boldsymbol{X}^{\bar{R}}_{(t+1):(t+L-1)}, \boldsymbol{X}_{t+L}|\boldsymbol{y}_{0:T})$ can be obtained by belief updating in $JT_{(t+1):(t+L+1)}$ given $P(\boldsymbol{X}^{\bar{R}}_{(t+2):(t+L)}, \boldsymbol{X}_{t+L+1}|\boldsymbol{y}_{0:T})$ (see (18), (19) and (20) in Appendix A). Thus, by passing $P(\boldsymbol{X}^{\bar{R}}_{(t+1):(t+L-1)}, \boldsymbol{X}_{t+L}|\boldsymbol{y}_{0:T})$ from $JT_{(t+1):(t+L+1)}$ to $JT_{t:(t+L)}$, where $t \in [0, T-L-1]$, we can achieve a recursive implementation of backward inference. Recall that $JT^b_{t:(t+L)}$ is a JT of maximal cliques of nodes belonging to $\{\boldsymbol{X}^{\bar{R}}_{(t+1):(t+L-1)}, \boldsymbol{X}_{t+L}, \boldsymbol{y}_{(t+1):(t+L)}\}$, which means that $JT^b_{t:(t+L)}$ factorizes $P(\boldsymbol{X}^{\bar{R}}_{(t+1):(t+L-1)}, \boldsymbol{X}_{t+L}|\boldsymbol{y}_{0:T})$ as the product of clique potentials divided by the product of sepset potentials. Moreover, the construction of $JT^b_{t:(t+L)}$ shows that for all cliques $C_i \in JT^b_{t:(t+L)}$, there exists a clique C_j in $JT_{t:(t+L)}$ s.t. $C_i \subseteq C_j$. Therefore, we only need to extract from $JT_{(t+1):(t+L+1)}$ the potentials of all cliques in $JT^b_{t:(t+L)}$, and pass them to $JT_{t:(t+L)}$ to meet the prerequisite for belief updating in $JT_{t:(t+L)}$. However, given that the recursive forward inference beforehand generates approximate instead of exact beliefs for nodes in $JT_{t:(t+L)\forall t\in[1,t+L]}$, all beliefs and messages involved throughout the recursive backward inference are accordingly approximate also.

Similar to Section 4.1, when extracting from $JT_{(t+1):(t+L+1)}$ the clique potentials in $JT^b_{t:(t+L)}$, we may encounter two mutually exclusive situations that require different solutions. In the first situation when clique $C_i \in JT^b_{t:(t+L)}$ is a subset of clique $C_j \in JT_{(t+1):(t+L+1)}$, ϕ_{C_i} can be directly marginalized out of ϕ_{C_j}. In the second situation when C_i is not a subset of any clique in $JT_{(t+1):(t+L+1)}$, ϕ_{C_i} can be obtained using the Backward Message Extraction (BME) algorithm that performs belief propagation and marginalization in a temporary JT, $JT^{b_{C_i}}_{(t+1):(t+L+1)}$, where b_{C_i} denotes backward message extraction for C_i:

BME Algorithm:

1. Extract from $JT_{(t+1):(t+L+1)}$ the subgraph formed by all those cliques (and the sepsets and edges between them) that contain at least one node in C_i. Denote the subgraph as the junction tree $JT^{b_{C_i}}_{(t+1):(t+L+1)}$.
2. Remove from $JT^{b_{C_i}}_{(t+1):(t+L+1)}$ a leaf clique C_j (and the connected sepsets and edges) if $(C_i \cap C_j) \subseteq (C_i \cap C_k)$ where $C_k \in \boldsymbol{nb}(C_j)$. Rewrite the resulting graph, which is still a JT satisfying the RIP, as $JT^{b_{C_i}}_{(t+1):(t+L+1)}$. Repeat this step until every leaf clique in the newly obtained $JT^{b_{C_i}}_{(t+1):(t+L+1)}$ is not a subset of its neighboring clique.
3. As nodes in C_i are all included in $JT^{b_{C_i}}_{(t+1):(t+L+1)}$, ϕ_{C_i} can be marginalized out of the JPD of all nodes in $JT^{b_{C_i}}_{(t+1):(t+L+1)}$, where that JPD can be expressed as the product of clique potentials divided by the

product of sepset potentials in $JT_{(t+1):(t+L+1)}^{b_{C_i}}$.

Fig. 5c shows two cliques $\{X_2^7X_3^7X_3^8\}$ and $\{X_2^6X_2^7X_3^5X_3^7\}$ in $JT_{1:3}^b$. Analogously, two cliques $\{X_1^7X_2^7X_2^8\}$ and $\{X_1^6X_1^7X_2^5X_2^7\}$ exist in $JT_{0:2}^b$. In order to achieve approximate backward inference in $JT_{0:2}^b$, $\hat{P}(X_1^7X_2^7X_2^8|\boldsymbol{y}_{0:T})$ and $\hat{P}(X_1^6X_1^7X_2^5X_2^7|\boldsymbol{y}_{0:T})$ need to be passed from $JT_{1:3}$ to $JT_{0:2}$ via $JT_{0:2}^b$. Since the clique $\{X_1^7X_2^7X_2^8\}$ also exists in $JT_{1:3}$, $\hat{P}(X_1^7X_2^7X_2^8|\boldsymbol{y}_{0:T})$ is directly available in $JT_{1:3}$. In contrast, $\{X_1^6X_1^7X_2^5X_2^7\}$ is not a subset of any clique in $JT_{1:3}$. Thus, we need to follow the BME algorithm to generate $JT_{1:3}^{b_{\{X_1^6X_1^7X_2^5X_2^7\}}}$ as $\boxed{X_1^5X_2^5X_2^6X_2^7X_3^5}$ — $\boxed{X_1^5X_1^7X_2^6X_2^7}$ — $\boxed{X_1^5X_1^6X_1^7X_2^6}$, and compute $\hat{P}(X_1^6X_1^7X_2^5X_2^7|\boldsymbol{y}_{0:T})$ as the product of $\hat{P}(X_1^5X_2^5X_2^6X_2^7X_3^5|\boldsymbol{y}_{0:T})$, $\hat{P}(X_1^5X_1^7X_2^6X_2^7|\boldsymbol{y}_{0:T})$ and $\hat{P}(X_1^5X_1^6X_1^7X_2^6|\boldsymbol{y}_{0:T})$ divided by the product of $\hat{P}(X_1^5X_2^6X_2^7|\boldsymbol{y}_{0:T})$ and $\hat{P}(X_1^5X_1^7X_2^6|\boldsymbol{y}_{0:T})$, where all the five factors are available in $JT_{1:3}$.

5.2 Message Passing via the Backward Interface JT and Belief Updating in the Main JT

Message passing from $JT_{t:(t+L)}^b$ to $JT_{t:(t+L)}$ and the second round of belief propagation in $JT_{t:(t+L)}$ can be achieved synchronously by the Backward Message Passing and Belief Updating (BMP-BU) algorithm.

BMP-BU algorithm:

```
for all cliques $C_i \in JT_{t:(t+L)}^b$
    Find in $JT_{t:(t+L)}$ a clique $C_j$ s.t. $C_i \subseteq C_j$;
    Update $\phi_{C_j}$ by absorbing from $\phi_{C_i}$, i.e. compute
        $\phi_{C_j}^* = \phi_{C_j} \times \frac{\phi_{C_i}}{\sum_{X^{C_j} \setminus X^{C_i}} \phi_{C_j}}$ where $X^{C_i}$ denotes the
        hidden nodes in $C_i$;
    Let $\phi_{C_j} = \phi_{C_j}^*$;
    Distribute messages from root $C_j$ in $JT_{t:(t+L)}$, i.e.
        update all other clique potentials in $JT_{t:(t+L)}$
        given $\phi_{C_j}$;
end for
```

6 Space Complexity and Inference Accuracy

In the proposed FBI algorithm, the two interface JTs are both extracted from the main JT. This means the treewidth of either interface JT cannot exceed that of the main JT. Thus, the space complexity of the FBI algorithm depends entirely on the treewidth of the main JT. It has been proven that a N-vertex graph with separability S has treewidth $O(S \times \log N)$, where the separability of a graph refers to the maximum number of internally vertex-disjoint paths with the same non-adjacent end vertices [20]. For a given DBN, let L be its MRL, S be its separability after moralization, n be the number of nodes per slice, and K be the maximum number of states a hidden node can take, then the treewidth of the main JT is $O\left(S \times \log\left(n \times (L+1)\right)\right)$ and the space complexity of the FBI algorithm is $O\left(K^{S \times \log(n \times (L+1))}\right)$. We focus on space complexity rather than time complexity because the former is a crucial constraint on practical application. Murphy's interface algorithm has space complexity $O(K^{|\boldsymbol{I}|+n})$, where $|\boldsymbol{I}|$ is the cardinality of the forward interface [15]. By comparison, we can conclude that for a DBN with large forward interface, where $S \times \log\left(n \times (L+1)\right) < |\boldsymbol{I}| + n$, our method will achieve inference with lower space complexity.

The BK algorithm implements forward inference recursively in two main steps. First, $\tilde{P}\left(\boldsymbol{I}_{t-1}|\boldsymbol{y}_{0:(t-1)}\right)$, an approximate decomposition of $\hat{P}\left(\boldsymbol{I}_{t-1}|\boldsymbol{y}_{0:(t-1)}\right)$, is propagated in a JT built from the 1.5-TBN involving $\boldsymbol{I}_{t-1}$, $\boldsymbol{X}_t$ and $\boldsymbol{y}_t$, so as to generate $\hat{P}(\boldsymbol{I}_{t-1}, \boldsymbol{X}_t|\boldsymbol{y}_{0:t})$. Next, $\hat{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ is marginalized out of $\hat{P}(\boldsymbol{I}_{t-1}, \boldsymbol{X}_t|\boldsymbol{y}_{0:t})$ and then projected into $\tilde{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ based on a pre-determined clique decomposition of $\boldsymbol{I}_t$. The risk of this recursive process is that errors may accumulate over time, thus rendering inference results inaccurate. Intuitively, there are two sources of error: the "old" error inherited from $\tilde{P}\left(\boldsymbol{I}_{t-1}|\boldsymbol{y}_{0:(t-1)}\right)$ to $\hat{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$, and the "new" error induced by projecting $\hat{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ into $\tilde{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$. Fortunately, the stochasticity of state transitions and the diversity of observations serve to reduce the errors on expectation [19]. It turns out that the BK algorithm produces bounded errors over time because $E\left[D_{KL}\left(P(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})||\tilde{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})\right)\right] \leq \frac{\varepsilon_t}{\gamma}$, where E is the expectation taken over all possible observation sequences, D_{KL} is the KL divergence, γ is the mixing rate of the DBN, and $\varepsilon_t = D_{KL}\left(P(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})||\tilde{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})\right) - D_{KL}\left(P(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})||\hat{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})\right)$ [15].

The FBI algorithm is essentially similar to the BK algorithm, with the difference being that $\tilde{P}\left(\boldsymbol{I}_{t-1}|\boldsymbol{y}_{0:(t-1)}\right)$, $\hat{P}(\boldsymbol{I}_{t-1}, \boldsymbol{X}_t|\boldsymbol{y}_{0:t})$, $\hat{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ and $\tilde{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ in the BK algorithm are, respectively, replaced by $\tilde{P}\left(\boldsymbol{X}_{(t-L):(t-1)}|\boldsymbol{y}_{0:(t-1)}\right)$, $\hat{P}\left(\boldsymbol{X}_{(t-L):t}|\boldsymbol{y}_{0:t}\right)$, $\hat{P}\left(\boldsymbol{X}_{(t-L+1):t}|\boldsymbol{y}_{0:t}\right)$ and $\tilde{P}\left(\boldsymbol{X}_{(t-L+1):t}|\boldsymbol{y}_{0:t}\right)$ in the FBI algorithm. By comparison, the BK algorithm constructs a JT for every 1.5-TBN and uses the forward interface to pass messages both forwards and backwards; whereas the FBI algorithm constructs a main JT for every $(L+1)$-TBN, and extract two interface JTs from each main JT to pass messages forwards and backwards respectively. Moreover, in order to reduce the space complexity, the BK algorithm partitions the forward interface into smaller disjoint cliques; whereas the FBI algorithm organizes the two types of interfaces in the form of JT that satisfies the RIP. As a result, in discrete DBNs with large forward interfaces, the BK algorithm usually fails to accommodate the associations in $\{\boldsymbol{I}_{t-1}, \boldsymbol{X}_t\}$ induced by slices prior to $(t-1)$ when generating $\hat{P}(\boldsymbol{I}_{t-1}, \boldsymbol{X}_t|\boldsymbol{y}_{0:t})$ given $\tilde{P}\left(\boldsymbol{I}_{t-1}|\boldsymbol{y}_{0:(t-1)}\right)$; and it neglects the associations between the disjoint cliques of $\boldsymbol{I}_t$ when projecting $\hat{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$ into $\tilde{P}(\boldsymbol{I}_t|\boldsymbol{y}_{0:t})$. In contrast, the FBI algorithm only neglects the associations in $\boldsymbol{X}_{(t-L):t}$ induced by slices prior to $(t-L)$ when generating $\hat{P}\left(\boldsymbol{X}_{(t-L):t}|\boldsymbol{y}_{0:t}\right)$ given $\tilde{P}\left(\boldsymbol{X}_{(t-L):(t-1)}|\boldsymbol{y}_{0:(t-1)}\right)$; and it only neglects the associations in $\boldsymbol{X}_{(t-L+1):t}$ induced by slice $(t-L)$ when projecting $\hat{P}(\boldsymbol{X}_{(t-L+1):t}|\boldsymbol{y}_{0:t})$ into $\tilde{P}(\boldsymbol{X}_{(t-L+1):t}|\boldsymbol{y}_{0:t})$. Thus, it is safe to conclude that in discrete DBNs with large forward interfaces, as long as the main JT has practically tractable treewidth, the FBI algorithm also has bounded errors over time and its errors are always smaller than those of the BK algorithm.

7 EXAMPLES

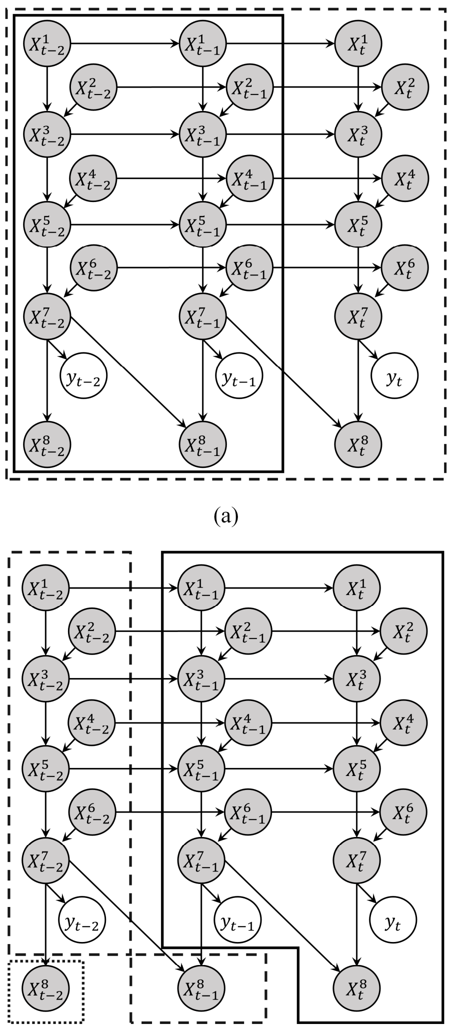
The validity of the FBI algorithm has been empirically verified by two example DBNs. The first is the example run above, which demonstrates the sub-algorithms involved. The second (Fig. 7a) was first introduced in [19] to monitor a wastewater treatment plant and was also used as a test case of the BK algorithm in [19] and [15].

7.1 The First Example DBN

Once the main JT and the forward interface JT are established for every 3-TBN, forward inference including filtering and prediction can be achieved by recursively implementing the three main steps described in Section 4. This recursive process corresponds to the top loop in Fig. 1. It can also be graphically elaborated in another way as shown in Fig. 6a, where two sets of nodes are visually marked in a 3-TBN. The first set refers to nodes in the corresponding forward interface JT, which is used to pass messages forwards. The second set refers to nodes in the corresponding main JT, and the approximate posterior distribution of these nodes, given observations up to the present, becomes available once belief propagation in the main JT is done.

After filtering is done for the last slice of interest and the backward interface JT is constructed for every 3-TBN, backward inference, also called smoothing, can be achieved by recursively implementing the two main steps described in Section 5. This recursive process corresponds to the bottom loop in Fig. 1. It can also be graphically elaborated in another way as shown in Fig. 6b, where three sets of nodes are visually marked in a 3-TBN. The first set refers to nodes in the corresponding backward interface JT, which is used to pass messages backwards. The second set refers to nodes whose approximate posterior distribution, given observations up to the last slice, becomes available once belief updating in the corresponding main JT is done. In contrast, the third set refers to nodes whose approximate posterior distribution, given observations up to the last slice, is not yet available in this 3-TBN.

For the sake of simplicity, all nodes in this example DBN are assumed to be binary. Please refer to the supplemental material for the parameter settings and the observations in slices up to $t = 15$. We used the FBI algorithm to achieve filtering and smoothing. In order to check the accuracy of inference, we unrolled the DBN to



X^1_{t-2} X^1_{t-1} X^1_t
X^2_{t-2} X^2_{t-1} X^2_t
X^3_{t-2} X^3_{t-1} X^3_t
X^4_{t-2} X^4_{t-1} X^4_t
X^5_{t-2} X^5_{t-1} X^5_t
X^6_{t-2} X^6_{t-1} X^6_t
X^7_{t-2} X^7_{t-1} X^7_t
y_{t-2} y_{t-1} y_t
X^8_{t-2} X^8_{t-1} X^8_t
(a)
(b)

Fig. 6. (a) In forward inference, the nodes in a 3-TBN can be visually marked as two sets. The first set, marked in the solid frame, refers to nodes in the forward interface JT. The second set, marked in the dashed frame, refers to nodes in the main JT. Their approximate posterior distribution, given observations up to the present, becomes available once belief propagation in the main JT is done. (b) In backward inference, the nodes in a 3-TBN can be visually marked as three sets. The first set, marked in the solid frame, refers to nodes in the backward interface JT. The second set, marked in the dashed frame, refers to nodes whose approximate posterior distribution, given observations up to the last slice, becomes available once belief updating in the corresponding main JT is done. The third set, marked in the dotted frame and excluded from the solid and dotted frames, refers to nodes whose approximate posterior distribution, given observations up to the last slice, is not yet available in this 3-TBN.

include slices up to every $t \in [0, \dots, 15]$ and applied the JT algorithm to each resulting BN individually. This benchmark exact inference is implemented in AgenaRisk [26], a commercial software for probabilistic reasoning using BN technology. We also implemented the BK algorithm in BNT, a Bayes net toolbox for Matlab [27]. But for a fair comparison, only four partitioning schemes of the forward interface, including $([X_t^1X_t^2X_t^3X_t^4],[X_t^5X_t^6X_t^7])$, $([X_t^1X_t^2X_t^3],[X_t^4X_t^5X_t^6X_t^7])$, $([X_t^1X_t^2X_t^3X_t^4X_t^5],[X_t^6X_t^7])$, $([X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6X_t^7])$, were considered in the BK algorithm based on two facts: (1) partitioning the forward interface into fewer disjoint clusters will produce smaller inference error, (2) the maximum sizes of cliques in the forward interface JT and the backward interface JT constructed by the FBI algorithm are 4 and 5, respectively (see Fig. 5b and Fig. 5c). That is, the four candidate partitioning schemes were selected for comparing the inference accuracy of the BK algorithm and the FBI algorithm given roughly the same space complexity. It turns out that the best partitioning scheme among the four candidates is $([X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6X_t^7])$, as it produces the lowest mean KL-divergence in terms of singleton marginals. Table 2 compares results for some singleton and multivariate marginals. For a comprehensive summary of all inference results achieved by the three methods, please refer to the supplemental material. Not surprisingly, the outcomes of the FBI algorithm are more accurate than those of the BK algorithm; and more impressively, they are highly consistent with the exact inference results.

In addition to unrolling the DBN, the interface algorithm can also implement exact inference in a DBN, but only if the treewidth of the JT built for 1.5-TBN is feasible in practice. To apply the interface algorithm to this example DBN, we first extracted a 1.5-TBN composed of $\boldsymbol{I}_{t-1}=\{X_{t-1}^1X_{t-1}^2X_{t-1}^3X_{t-1}^4X_{t-1}^5X_{t-1}^6X_{t-1}^7\}$, $\boldsymbol{X}_t=\{X_t^1X_t^2X_t^3X_t^4X_t^5X_t^6X_t^7X_t^8\}$ and y_t, and then enforced the constraint that $\boldsymbol{I}_{t-1}$ and $\boldsymbol{I}_t$ each belong to a clique in the JT built for the 1.5-TBN. This constraint can be ensured by simply adding two dummy nodes, which are assumed to be a common child node of $\boldsymbol{I}_{t-1}$ and $\boldsymbol{I}_t$, respectively (see Fig. S3a). In result, the maximum size of cliques in the JT constructed by the interface algorithm is 8 (see Fig. S3b). In the same way we can show that the maximum size of cliques in the JT constructed by the BK algorithm with the partitioning scheme $([X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6X_t^7])$ is 6 (see Fig. S3c and Fig. S3d). Compared with Fig. 5a, where the maximum clique size is 5, it shows that the FBI algorithm has lower space complexity than the interface algorithm and the BK algorithm in this example.

7.2 The Second Example DBN

Since each hidden node in Fig. 7a has a persistent arc from the previous slice, the MRL algorithms returns $L=1$. This means that to use the FBI algorithm, a main JT, $JT_{t:(t+1)}$ (Fig. 7b), needs to be built for every 2-TBN. Next, by following the FI-JT and BI-JT algorithms in Section 3, we derive from $JT_{t:(t+1)}$ a forward interface JT, $JT_{t:(t+1)}^f$ (Fig. 7c), and a backward interface JT, $JT_{t:(t+1)}^b$ (Fig. 7d), respectively. What makes this DBN special is that it has no intra-slice arcs between the hidden nodes. Therefore, $JT_{t:(t+1)}^f$ consists of overlapping cliques; while $JT_{t:(t+1)}^b$ consists of disjoint cliques, in particular, each of the hidden nodes in a slice belongs to a distinct clique. Clearly, the backward interface JT of this DBN represents the worst case.

Again, for simplicity, we assume all nodes in this DBN have binary states. For the parameter settings and the observations in slices up to $t=20$, please refer to the supplemental material. Table 3 shows the KL-divergence for some inferred singleton and multivariate posterior distributions compared to the corresponding exact distributions. For all inference results obtained in this example, please also refer to the supplemental material. In Table 3, one multivariate posterior distribution involves X_t^3, X_t^4, X_t^5 and X_t^6, which belong to the same clique in the BK partitions; the other multivariate posterior distribution involves X_t^2 and X_t^7, which are located in two disjoint cliques in the BK partitions. Intuitively, the BK algorithm produces lower inference accuracy on multivariate posterior distributions for nodes belonging to multiple cliques rather than a single clique. This is confirmed by the relevant results in Table 3. In contrast, the FBI algorithm has similar inference accuracy on multivariate posterior distributions for nodes belonging to one or more partitioned cliques. Furthermore, for all forward inference tasks, the FBI algorithm outperforms the BK algorithm in terms of inference accuracy. As for backward inference tasks, one would think the performance of the FBI algorithm would be poor, since the backward interface JT in this example represents the worst case, i.e. each hidden node locates in a distinct isolated clique. However, as indicated by Table 3, the FBI algorithm exhibits comparable performance to the BK algorithm, and in particular has a more accurate multivariate posterior distribution on X_t^2 and X_t^7.

Lastly, in order to compare the space complexity of different methods, we added dummy nodes to group the hidden nodes according to the specific requirements of the interface algorithm and the BK algorithm (Fig. S4a and Fig. S4c). As a result, for each 1.5-TBN, the maximum clique size in the JT constructed by the interface algorithm is 11 (Fig. S4b), and the maximum clique size in the JT constructed by the BK algorithm is 7 (Fig. S4d). In comparison, the main JT built by the FBI algorithm, as shown in Fig. 7b, has the maximum clique of 6 nodes. That is, in this example the FBI algorithm still has lower space complexity than the interface algorithm and the BK algorithm.

8 Conclusion

Exact inference can be hardly achieved in large or dense DBNs due to the curse of dimensionality and associated computational overhead. Approximate inference has thus become a hot spot in theoretical and applied research. Tree-based deterministic approximate inference outperforms the Monte Carlo method in speed but can still be computationally prohibitive in discrete DBNs with large

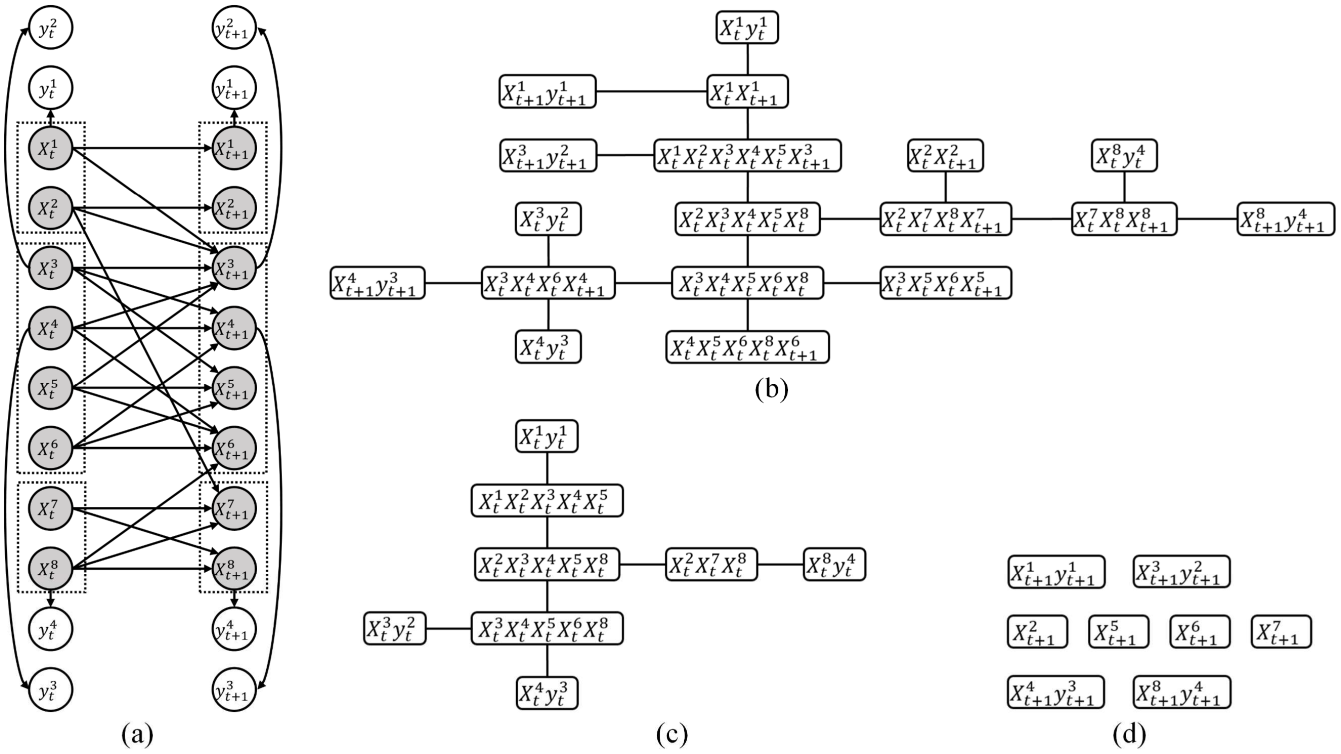

(a)
(b)
(c)
(d)
$X_t^1 y_t^1$
$X_{t+1}^1 y_{t+1}^1$
$X_t^1 X_{t+1}^1$
$X_{t+1}^3 y_{t+1}^2$
$X_t^1 X_t^2 X_t^3 X_t^4 X_t^5 X_{t+1}^3$
$X_t^2 X_{t+1}^2$
$X_t^8 y_t^4$
$X_t^3 y_t^2$
$X_t^2 X_t^3 X_t^4 X_t^5 X_t^8$
$X_t^2 X_t^7 X_t^8 X_{t+1}^7$
$X_t^7 X_t^8 X_{t+1}^8$
$X_{t+1}^8 y_{t+1}^4$
$X_{t+1}^4 y_{t+1}^3$
$X_t^3 X_t^4 X_t^6 X_{t+1}^4$
$X_t^3 X_t^4 X_t^5 X_t^6 X_t^8$
$X_t^3 X_t^5 X_t^6 X_{t+1}^5$
$X_t^4 y_t^3$
$X_t^4 X_t^5 X_t^6 X_t^8 X_{t+1}^6$
$X_t^1 X_t^2 X_t^3 X_t^4 X_t^5$
$X_t^2 X_t^3 X_t^4 X_t^5 X_t^8$
$X_t^2 X_t^7 X_t^8$
$X_t^3 X_t^4 X_t^5 X_t^6 X_t^8$
X_{t+1}^2
X_{t+1}^5
X_{t+1}^6
X_{t+1}^7

Fig. 7. (a) The water DBN, designed to monitor a waste water treatment plant. The model was originally introduced in [19] and later used in [15]. Again, the hidden and observable nodes are represented by shaded and unshaded circles, respectively. The dotted boxes group together nodes that will be used in the BK approximation. (b) From the 2-TBN in (a), a main JT, $JT_{t:(t+1)}$, is constructed by the JT algorithm. (c) From $JT_{t:(t+1)}$, a forward interface JT, $JT^{f}_{t:(t+1)}$, is derived by the FI-JT algorithm. (d) From $JT_{t:(t+1)}$, a backward interface JT, $JT^{b}_{t:(t+1)}$, is derived by the BI-JT algorithm.

TABLE 2

PARTIAL INFERENCE RESULTS OF THE FIRST EXAMPLE DBN

		$t=6$	$t=7$	$t=8$	$t=9$	$t=10$
$\frac{1}{8}\times\sum_{i\in[1,8]} KL\left(P(X_t^i\|\boldsymbol{y}_{0:t})\|\|\tilde{P}(X_t^i\|\boldsymbol{y}_{0:t})\right)$	FBI	2.585E-10	2.665E-10	3.104E-10	2.830E-11	1.125E-10
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6X_t^7]$)	1.664E-06	1.035E-05	2.720E-05	7.656E-07	1.497E-05
$KL\left(P(X_t^5X_t^6X_t^7\|\boldsymbol{y}_{0:t})\|\|\tilde{P}(X_t^5X_t^6X_t^7\|\boldsymbol{y}_{0:t})\right)$	FBI	1.567E-10	8.464E-11	4.755E-11	4.460E-11	2.057E-11
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6X_t^7]$)	8.362E-07	2.170E-07	3.041E-07	8.875E-07	4.462E-07
$\frac{1}{8}\times\sum_{i\in[1,8]} KL\left(P(X_t^i\|\boldsymbol{y}_{0:15})\|\|\tilde{P}(X_t^i\|\boldsymbol{y}_{0:15})\right)$	FBI	3.213E-09	6.403E-10	5.380E-10	1.316E-11	2.771E-10
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6X_t^7]$)	4.361E-04	1.661E-04	2.951E-05	7.747E-05	7.358E-05
$KL\left(P(X_t^5X_t^6X_t^7\|\boldsymbol{y}_{0:15})\|\|\tilde{P}(X_t^5X_t^6X_t^7\|\boldsymbol{y}_{0:15})\right)$	FBI	2.883E-10	1.195E-10	6.065E-11	4.855E-11	5.690E-12
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6X_t^7]$)	6.445E-06	1.280E-06	7.000E-07	2.512E-06	1.158E-06

TABLE 3

PARTIAL INFERENCE RESULTS OF THE SECOND EXAMPLE DBN

		$t=11$	$t=12$	$t=13$	$t=14$	$t=15$
$\frac{1}{8}\times\sum_{i\in[1,8]} KL\left(P(X_t^i\|\boldsymbol{y}_{0:t}^{1:4})\|\|\tilde{P}(X_t^i\|\boldsymbol{y}_{0:t}^{1:4})\right)$	FBI	5.674E-07	2.332E-07	3.179E-07	5.229E-07	8.714E-07
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6],[X_t^7X_t^8]$)	3.903E-04	1.824E-05	1.632E-04	6.469E-04	1.030E-03
$KL\left(P(X_t^3X_t^4X_t^5X_t^6\|\boldsymbol{y}_{0:t}^{1:4})\|\|\tilde{P}(X_t^3X_t^4X_t^5X_t^6\|\boldsymbol{y}_{0:t}^{1:4})\right)$	FBI	1.847E-06	7.537E-07	1.993E-06	2.211E-06	5.859E-06
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6],[X_t^7X_t^8]$)	1.859E-05	1.508E-06	1.242E-05	2.763E-05	6.035E-05
$KL\left(P(X_t^2X_t^7\|\boldsymbol{y}_{0:t}^{1:4})\|\|\tilde{P}(X_t^2X_t^7\|\boldsymbol{y}_{0:t}^{1:4})\right)$	FBI	1.055E-06	2.561E-07	3.509E-08	3.108E-07	4.735E-08
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6],[X_t^7X_t^8]$)	5.634E-02	5.685E-02	5.903E-02	6.290E-02	6.503E-02
$\frac{1}{8}\times\sum_{i\in[1,8]} KL\left(P(X_t^i\|\boldsymbol{y}_{0:20}^{1:4})\|\|\tilde{P}(X_t^i\|\boldsymbol{y}_{0:20}^{1:4})\right)$	FBI	4.704E-04	4.682E-04	1.291E-03	1.108E-03	7.140E-05
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6],[X_t^7X_t^8]$)	3.243E-04	1.098E-03	1.348E-03	5.911E-04	4.488E-04
$KL\left(P(X_t^3X_t^4X_t^5X_t^6\|\boldsymbol{y}_{0:20}^{1:4})\|\|\tilde{P}(X_t^3X_t^4X_t^5X_t^6\|\boldsymbol{y}_{0:20}^{1:4})\right)$	FBI	2.121E-03	2.982E-03	8.323E-03	7.457E-03	4.460E-04
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6],[X_t^7X_t^8]$)	4.855E-05	2.249E-04	2.123E-04	1.490E-04	1.778E-05
$KL\left(P(X_t^2X_t^7\|\boldsymbol{y}_{0:20}^{1:4})\|\|\tilde{P}(X_t^2X_t^7\|\boldsymbol{y}_{0:20}^{1:4})\right)$	FBI	1.307E-03	1.960E-04	1.562E-04	4.718E-04	3.824E-04
	BK($[X_t^1X_t^2],[X_t^3X_t^4X_t^5X_t^6],[X_t^7X_t^8]$)	6.162E-02	6.651E-02	6.230E-02	5.961E-02	5.905E-02

forward interfaces. Here, we present the FBI algorithm to enhance the practical feasibility of approximate inference at the expense of limited accuracy. Compared with the benchmark deterministic approximate inference method, namely the BK algorithm, the FBI algorithm achieves better inference accuracy in the following ways: (1) rearrange nodes beyond the scope of the forward interface into two interface JTs for forward and backward inference, respectively; (2) provide an automated and optimized solution to partitioning sophisticated interfaces throughout a recursive inference process. Compared with the benchmark exact inference method, i.e. Murphy's interface algorithm, the FBI algorithm causes a small loss of inference accuracy, but it can significantly reduce the space complexity of inference in DBNs with large forward interfaces, and thus has great potential for broad applications in practice.

Acknowledgment

This work was supported partly by the Leverhulme Trust under Grant RPG-2016-118 CAUSAL-DYNAMICS, and partly by the EPSRC under project EP/P009964/1: PAtient Managed decision-support using Bayesian networks. The authors also acknowledge Agena Ltd for software support.

References

[1] L. An, M. Kafai, and B. Bhanu, "Dynamic Bayesian Network for Unconstrained Face Recognition in Surveillance Camera Networks," *IEEE Journal on Emerging and Selected Topics in Circuits and Systems,* vol. 3, no. 2, pp. 155-164, June 2013.

[2] H.Y. Cheng, C.C. Weng, and Y.Y. Chen, "Vehicle Detection in Aerial Surveillance Using Dynamic Bayesian Networks," *IEEE transactions on image processing,* vol. 21, no. 4, pp. 2152-2159, Apr. 2011.

[3] J. Frankel, M. Wester, and S. King, "Articulatory Feature Recognition Using Dynamic Bayesian Networks," *Computer Speech and Language,* vol. 21, no. 4, pp. 620-640, Oct. 2007.

[4] A.V. Nefian, L. Liang, X. Pi, X. Liu, and K. Murphy, "Dynamic Bayesian Networks for Audio-Visual Speech Recognition," *EURASIP Journal on Advances in Signal Processing,* vol. 2002, no. 11, pp. 1274–1288, 2002.

[5] C. Chen, J. Liang, and X. Zhu, "Gait Recognition Based on Improved Dynamic Bayesian Networks," *Pattern Recognition,* vol. 44, no. 4, pp. 988-995, 2011.

[6] H.I. Suk, B.K. Sin, and S.W. Lee, "Hand Gesture Recognition Based on Dynamic Bayesian Network Framework," *Pattern recognition,* vol. 43, no. 9, pp. 3059-3072, 2010.

[7] J. Sun, and J. Sun, "A Dynamic Bayesian Network Model for Real-Time Crash Prediction Using Traffic Speed Conditions Data," *Transportation Research Part C: Emerging Technologies,* vol. 54, pp. 176-186, 2015.

[8] C.M. Queen and C.J. Albers, "Intervention and Causality: Forecasting Traffic Flows Using a Dynamic Bayesian Network," *Journal of the American Statistical Association,* vol. 104, no. 486, pp. 669-681, Jun. 2009.

[9] J. Schütte, H. Wang, S. Antoniou, A. Jarratt, N.K. Wilson, J. Riepsaame, F.J. Calero-Nieto, V. Moignard, S. Basilico, S.J. Kinston, R.L. Hannah, M.C. Chan, S.T. Nürnberg, W.H. Ouwehand, N. Bonzanni, M.F. de Bruijn, and B. Göttgens, "An Experimentally Validated Network of Nine Haematopoietic Transcription Factors Reveals Mechanisms of Cell State Stability," *Elife* 5: e11469, Feb. 2016.

[10] S. Eldawlatly, Y. Zhou, R. Jin, and K.G. Oweiss, "On the Use of Dynamic Bayesian Networks in Reconstructing Functional Neuronal Networks from Spike Train Ensembles," *Neural computation,* vol. 22, no. 1, pp. 158-189, Jan. 2010.

[11] M.A. Van Gerven, B.G. Taal, and P.J. Lucas, "Dynamic Bayesian Networks as Prognostic Models for Clinical Patient Management," *Journal of biomedical informatics,* vol. 41, no. 4, pp. 515-529, 2008.

[12] A. Onisko, M.J. Druzdzel, and R.M. Austin, "Application of Dynamic Bayesian Networks to Cervical Cancer Screening," *Proc. Eleventh Int. Conf. Artificial Intelligence,* pp. 5-14, 2009.

[13] P. Kozlow, N. Abid, and S. Yanushkevich, "Gait Type Analysis Using Dynamic Bayesian Networks," *Sensors,* 18(10), 3329, Oct. 2018, doi.org/10.3390/s18103329.

[14] R.G. Cowell, P. Dawid, S.L. Lauritzen, and D.J. Spiegelhalter, *Probabilistic Networks and Expert Systems: Exact Computational Methods for Bayesian Networks,* Springer Science & Business Media, 2006.

[15] K.P. Murphy, "Dynamic Bayesian Networks: Representation, Inference and Learning," PhD dissertation, Dept. of Electrical Engineering and Computer Sciences, University of California, Berkeley, 2002.

[16] T. Bengtsson, P. Bickel, and B. Li, "Curse-of-dimensionality Revisited: Collapse of the Particle Filter in Very Large Scale Systems," *In Probability and statistics: Essays in honor of David A. Freedman, Institute of Mathematical Statistics,* pp. 316-334, 2008.

[17] M. Neil, X. Chen, and N. Fenton, "Optimizing the Calculation of Conditional Probability Tables in Hybrid Bayesian Networks Using Binary Factorization," *IEEE Trans. on Knowledge and Data Engineering,* vol. 24, no. 7, pp. 1306-1312, 2011.

[18] M. Neil, M. Tailor, and D. Marquez, "Inference in Hybrid Bayesian Networks Using Dynamic Discretization," *Statistics and Computing,* vol. 17, no. 3, pp. 219-233, 2007.

[19] X. Boyen and D. Koller, "Tractable Inference for Complex Stochastic Processes," *Proc. Fourteenth Ann. Conf. on Uncertainty in Artificial Intelligence,* pp. 33-42, July 1998.

[20] O. Schaudt, R. Schrader, and V. Weil, "On the Separability of Graphs," *Discrete Mathematics,* vol. 313, no. 6, pp. 809-820, 2013.

[21] D. Madigan, J. York, and D. Allard, "Bayesian Graphical Models for Discrete Data," *International Statistical Review,* vol. 63, no. 2, pp. 215-232, 1995.

[22] P. Giudici and R. Castelo, "Improving Markov Chain Monte Carlo Model Search for Data Mining," *Machine learning,* vol. 50, no. 1-2, pp. 127-158, 2003.

[23] S. Geman and D. Geman, "Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images," *IEEE Transactions on pattern analysis and machine intelligence,* vol. 6, pp. 721-741, Nov. 1984.

[24] T. Hrycej, "Gibbs Sampling in Bayesian Networks," *Artificial Intelligence,* vol. 46, no. 3, pp. 351-363, 1990.

[25] A. Doucet, N. De Freitas, K. Murphy, and S. Russell, "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks," *Proc. Sixteenth Ann. Conf. on Uncertainty in Artificial Intelligence,* pp. 176-183, Jun. 2000.

[26] N.E. Fenton and M. Neil, "Decision Support Software for Probabilistic Risk Assessment Using Bayesian Networks," *IEEE Software,* vol. 31, no. 2, pp. 21–26, 2014.

[27] K. Murphy, "The Bayes Net Toolbox for Matlab," *Computing science and statistics,* vol. 33, no. 2, pp. 1024-1034, 2001.

Huange Wang received the PhD degree in applied mathematics and statistics from Wageningen University & Research, the Netherlands, in 2017. She is a post-doctoral research associate in Risk and Information Management research group, at the School of Electronic Engineering and Computer Science, Queen Mary University of London. Her research interest lies in the development of probabilistic graphical models in machine learning, automated inference, and decision support.

Martin Neil is a professor of Computer Science and Statistics at the School of Electronic Engineering and Computer Science, Queen Mary University of London. He is also a Director of Agena, a company that develops Bayesian probabilistic reasoning software and applies it to risky and uncertain problems.

Norman Fenton is a professor of Risk Information Management at the School of Electronic Engineering and Computer Science, Queen Mary University of London. He is also a Director of Agena, a company that develops Bayesian probabilistic reasoning software and applies it to risky and uncertain problems.