Optimizing the triangulation of Dynamic Bayesian Networks

Morgan Chopin and Pierre-Henri Wuillemin
LIP6
University of Paris 6
75016, Paris, France

Abstract

In this paper, we address the problem of finding good quality elimination orders for triangulating dynamic Bayesian networks. In Bilmes and Bartels (2003), the authors proposed a model and an algorithm to compute such orders, but in exponential time. We show that this can be done in polynomial time by casting the problem to the problem of finding a minimum s–t cut in a graph. In this approach, we propose a formal definition of an *interface* (a set of nodes which makes the past independent from the future), we link the notion of an interface with the notion of a graph cut-set. We also propose an algorithm which computes the minimum interface of a dBN in polynomial time. Given this interface, we show how to get an elimination order which guarantees, theoretically and experimentally, the triangulation quality.

1 Introduction

Many domains deal with monitoring the evolution of complex system over time: localization of a robot in a dynamic environment (Fox et al., 1999), fault diagnostics analysis (Weber, 2002), monitoring physicochemical reactions (Baudrit et al., 2009). Many formal tools have been developed for describing such systems, including *hidden Markov models*, *Kalman filters*, and *dynamic Bayesian networks* (dBNs) (Dean and Kanazawa, 1990). All of these models share the same idea of representing the state of a system and its changes over time according to a probabilistic *transition model* and a *prior* distribution. Moreover, they assume the system to be *first-order Markovian* (the state at time t only depends on the state at time $t-1$) and *homogeneous* (the parameters do not vary with time). DBNs are graphical models that generalize hidden Markov models and Kalman filters by factoring the state as a *Bayesian network* (Pearl, 1988), hence providing an easy way to model and specify the parameters of a complex system. A dBN is then used to answer some *query* we may ask by making an *inference* (*i.e.* compute the probability of a state given some observations). Numerous types of queries lead to numerous types of inference. In this paper, we study the *offline* inference problem: computing the probability of a state at time $t \in [0, T]$ given the evidences we have during the simulation.

Exact inference in Bayesian networks is computationally hard (Cooper, 1990). This is even more difficult in the case of a dBN because it monitors variables over time and thus can be arbitrarily large, depending of the duration of the process. Among all the methods that perform inference in dBNs (Murphy, 2002), some are based on triangulation such as the junction tree inference (Jensen et al., 1990). Finding a good triangulation, *i.e.* finding an elimination order such that the resulting triangulated graph has the smallest maximum clique, is crucial for tractable inference but is a NP-complete problem (Arnborg et al., 1987). Hence, we must resort on heuristic approach to find good elimination orders.

In the case of dBNs the most promising work of finding good quality elimination orders uses a constrained elimination scheme (Kjærulff, 1994; Murphy, 2002; Darwiche, 2001; Bilmes and Bartels, 2003). In this approach, the use of constraints reduce the amount of elimination orders

to consider. Furthermore, these constraints do not reduce the triangulation quality (Darwiche, 2001). In this paper, we propose a method that follows the same idea by computing such elimination orders in an efficient[1] way. We also give some theoretical and experimental results.

Section 2 gives the basic background and defines the notion of interface. In section 3, we present the state of the art inference methods in a dBN and their complexity. In section 4, we present our method. Experimental results are given in section 5 and we conclude in section 6. Throughout this paper, we assume that the reader has some knowledge in graphical models (Pearl, 1988).

2 Dynamic Bayesian networks

We begin by defining some notations. Discrete random variables will be denoted by X^1, X^2,... and X_t^i will denote the i-th variable at time t. We will use the notation $X_{a:b}$ as a shortcut for $X_a, ..., X_b$. In a Bayesian network, we will denote by $pa(X)$ the parents of a node X in the graph, *i.e.* the set of nodes with an arc to X. A set of random variables will be denoted by $\mathbf{X}$; its size, denoted $size(\mathbf{X})$, corresponds to the cardinal product of each variable in $\mathbf{X}$ and its dimension, denoted $|\mathbf{X}|$, is the number of variable in $\mathbf{X}$. Finally, we will use $\mathbf{X}_t$ to denote the set of variables at time t.

Dynamic Bayesian networks (dBNs) can be viewed as an extension of Bayesian networks for modeling dynamic systems. Since we assume that the process is first-order Markovian and homogeneous, we can write the transition model as follows:

$$P(\mathbf{X}_t|\mathbf{X}_{0:t-1}) = P(\mathbf{X}_t|\mathbf{X}_{t-1}) = P(\mathbf{X}_1|\mathbf{X}_0)$$

In addition, a dBN exploits independences into $\mathbf{X}_1$ and $\mathbf{X}_0$ to factorize $P(\mathbf{X}_1|\mathbf{X}_0)$ and $P(\mathbf{X}_0)$. This leads us to the following definition:

Definition 1. A 2-*Time-slice Bayesian Network* (2-TBN) is a Bayesian network defined as follows: the nodes are partitioned into two sets $\mathbf{X}_0$ and $\mathbf{X}_1$ called *slices*. The 2-TBN represents the *transition model* such that:

$$P(\mathbf{X}_1|\mathbf{X}_0) = \prod_{i=0}^{n-1} P(X_1^i|pa(X_1^i))$$

where $pa(X_1^i) \subset \mathbf{X}_0 \cup \mathbf{X}_1$ and the *prior* distribution:

$$P(\mathbf{X}_0) = \prod_{i=0}^{n-1} P(X_0^i|pa(X_0^i))$$

where $pa(X_0^i) \subset \mathbf{X}_0$.

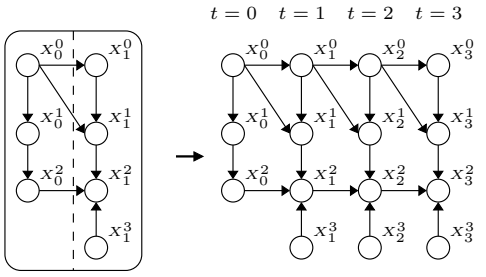

Figure 1: A 2-TBN unrolled three times to get a dBN of length $T = 3$. Here $\mathbf{I}_t^{\rightarrow} = \{X_t^0, X_t^2\}$ and $\mathbf{I}_t^{\leftarrow} = \{X_t^0, X_t^1, X_t^2\}$.

We can then define a dBN (see Figure 1):

Definition 2. A dynamic Bayesian network of length T (denoted as $\mathcal{G}_T = (\mathbf{X} = \mathbf{X}_0 \cup ... \cup \mathbf{X}_T, \mathbf{E})$) is a Bayesian network resulting by "unrolling" a 2-TBN in T time steps. Each state $\mathbf{X}_t$ of the dBN is called the slice t. We denote by $\mathcal{G}_T^m$ the *moralization* of the dBN $\mathcal{G}_T$.

In the next section, we define the notion of interface that plays a crucial role in our triangulation method.

2.1 The notion of interface

An interface is a subset of nodes such that if they were removed, the past would be independent from the future[2]. This notion admits several definitions (Kjærulff, 1994; Murphy, 2002; Darwiche, 2001; Bilmes and Bartels, 2003), we propose one that allows us to encompass several definitions found in the literature and makes explicit the link between an interface and a cut-set in a graph.

[1] By *efficient* we mean *polynomial time*.

[2] Note that we use the term interface to be consistent with the literature. Heggernes (2006) uses the term *separator*.

Definition 3. Let $\mathcal{G}_T^m = (\mathbf{X}, \mathbf{E})$ a moralized dBN of length T, a subset $\mathbf{I} \subset \mathbf{X}$ is called *interface* if every path from a node of $\mathbf{X}_0$ to a node of $\mathbf{X}_T$ in $\mathcal{G}_T^m$ contains at least one node in $\mathbf{I}$. A *minimal interface* $\mathbf{I}$ is such that for all $\mathbf{I}' \subset \mathbf{I}$, $\mathbf{I}'$ is not an interface.

Now we prove that forward and backward interfaces defined in Darwiche (2001) are also interfaces. First, we recall what are backward and forward interfaces:

Definition 4. Let $\mathcal{G}_T = (\mathbf{X}, \mathbf{E})$ a dBN of length T, the *forward interface* $\mathbf{I}_t^{\rightarrow}$ is the set of nodes in slice $t < T$ that have at least one child in slice $t+1$. The *backward interface* $\mathbf{I}_t^{\leftarrow}$ is the set of all nodes X in slice $t > 0$ such that X, or one of its children, has a parent in slice $t-1$.

Proposition 1. *Let $\mathcal{G}_T = (\mathbf{X}, \mathbf{E})$ a dBN of length T, then $\mathbf{I}_t^{\rightarrow}$ for all $t < T$ and $\mathbf{I}_{t'}^{\leftarrow}$ for all $t' > 0$ are interfaces.*

Proof. Let $t < T$, we will prove the result for $\mathbf{I}_t^{\rightarrow}$. The proof for the *backward interfaces* is similar. Let $\mathbf{N}_t^{\rightarrow} = \mathbf{X}_t - \mathbf{I}_t^{\rightarrow}$. Suppose there is a path from a node of $\mathbf{X}_0$ to a node of $\mathbf{X}_T$ that does not contain any node in $\mathbf{I}_t^{\rightarrow}$ in the moralized dBN. Thus, there must be an edge (u, v) such that $u \in \mathbf{N}_t^{\rightarrow}$ and $v \in \mathbf{X}_{t+1}$. If (u, v) was an arc in the non-moralized version, then u should be in $\mathbf{I}_t^{\rightarrow}$ by definition, but this leads to a contradiction. If (u, v) is an edge added during the moralization, then u and v have a common child in the non-moralized version. This child is either in $\mathbf{X}_t$ or in $\mathbf{X}_{t+1}$. If it belongs to $\mathbf{X}_{t+1}$ then u has a child in slice $t+1$ and should be in $\mathbf{I}_t^{\rightarrow}$ which leads to a contradiction. If it belongs to $\mathbf{X}_t$ then there exists an arc from the future to the past which is forbidden into a dBN. Thus the edge (u, v) can not exist and we deduce the result. □

Note that an interface is not necessarily included entirely in the same slice, but can span across several slices. Since the topology of a slice does not vary over time, the size of the backward interface, forward interface and a slice remain constant, so we can write $size(\mathbf{I}_t^{\rightarrow}) = i^{\rightarrow}$, $size(\mathbf{I}_t^{\leftarrow}) = i^{\leftarrow}$ and $size(\mathbf{X}_t) = s_{\text{slice}}$.

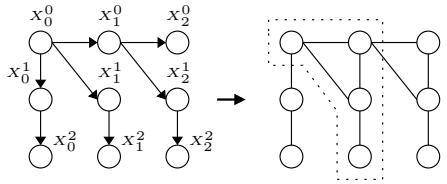

Figure 2: Triangulation of a dBN using a forward slice-by-slice elimination order where $\mathbf{I}_t^{\leftarrow} = \{X_t^0, X_t^1\}$. The subgraph surrounded in dashed line can be repeated to avoid re-triangulated the dBN if its length changes.

3 Exact inference and complexity

In this work, we have studied the *fixed interval smoothing* inference (or the *offline* inference), since all other inferences are particular cases. To perform this task, one can use two different approaches. By considering the dBN either as a stochastic process or as a Bayesian network and applying inference algorithms that are devoted for the former or the latter. In this paper, we consider the second approach and methods based on triangulation. As stated previously, the NP-hardness of finding optimal triangulation implies the use of heuristic methods. In the case of dBNs, a particular type of elimination order, called *constrained elimination order*, is distinctively interesting. A constrained elimination order is an elimination order such that we impose the elimination of a set of nodes before an other. For example, a forward (resp. backward) slice-by-slice elimination, denoted F-SS (resp. B-SS), is a constrained elimination order such that we eliminate a slice after an other from the past to the future (resp. from the future to the past) (see Figure 2). This kind of elimination orders gives a theoretical upper bound of the maximum clique size which is independent of the length of the dBN. Besides, they give good experimental results and provide a way to avoid re-triangulating the dBN for different length (Kjærulff, 1994; Bilmes and Bartels, 2003; Murphy, 2002).

In Figure 2 the backward interfaces belongs to a clique, this is a direct consequence of the following theorem:

Lemma 1 (Rose et al. (1976)). *Let $X_1, ..., X_n$ be an elimination order triangulating an undirected graph G, and let X_i and X_j two non-neighbors nodes. Then the elimination order add an edge (X_i, X_j) if and only if there exists a path with endpoints X_i and X_j such that every nodes on the path are eliminated before X_i and X_j.*

If we use a slice-by-slice elimination order then we eliminate each node in slice $t-1$ before the nodes in slice t. By definition, a backward interface is the set of nodes that have neighbors in slice $t-1$ and if there is one connected component per slice then for each pair of nodes (u, v) in the backward interface there exists a path with endpoints (u, v) where all nodes in the path are in the previous slice. Hence, by Lemma 1, (u, v) will be connected.

One can note that lemma 1 also implies that the size of the maximum clique is at least as large as the size of the backward interface. This is the main drawback of using constrained elimination order. For example, in Figure 2, the treewidth is 2 whereas the constrained triangulation gives a maximum clique of size 3. However, in practice, constrained elimination gives good results and may be superior to unconstrained elimination (Darwiche, 2001). Moreover, we have the following upper bound guarantee on the maximum clique size:

Theorem 1 (Darwiche (2001)). *Let a dBN of length T, then $m(\text{F-SS}) \leq i^{\leftarrow}.s_{slice}$, $m(\text{B-SS}) \leq i^{\rightarrow}.s_{slice}$ where $m(\sigma)$ is the size of the maximum clique obtained using the elimination order σ.*

Thus an idea to improve this result is to find a smaller interface and deduce a constrained elimination order. To illustrate this, consider Figure 3: since $\mathbf{I}_0$ is an interface we can split the dBN into two parts $\mathbf{L}_0$ and $\mathbf{R}_0$ that are respectively the nodes in the left of $\mathbf{I}_0$ and the nodes in its right. At each step, we eliminate nodes in $\mathbf{L}_i$ and slide $\mathbf{I}_i$ one time step further to get an other interface $\mathbf{I}_{i+1}$ and we repeat the process. The final step consists in eliminating nodes in $\mathbf{I}_2 \cup \mathbf{R}_2$. This elimination order denoted MIN-ELIM is then $\mathbf{L}_0, \mathbf{L}_1 - \mathbf{L}_0, \mathbf{L}_2 - \mathbf{L}_1, \mathbf{I}_2 \cup \mathbf{R}_2$. The maximum clique dimension is then 2 which is better in comparison to the one in Figure 2.

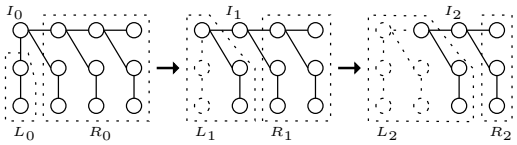

Figure 3: Finding a constrained elimination order given an interface $\mathbf{I}_0$.

This approach was followed by (Bilmes and Bartels, 2003), where the authors proposed an algorithm to find a smaller interface and an other one to build an elimination order relying on a more general model than the dBN called "GMTK-template". They show that the triangulation quality can be dramatically better than other constrained elimination.

Unfortunately, the algorithm that finds this elimination order runs in exponential time and needs to be parameterized. In the next sections, we show that finding and building such elimination order can be done in polynomial time and does not need any parameterization.

4 Minimum interface triangulation

In this section, we first prove that finding the smallest possible interface can be done in polynomial time. We then show how to build a constrained elimination order with a theoretical guarantee on the upper bound of the maximum clique size.

4.1 Finding the minimum interface

The problem of finding a minimum interface in a dBN of length T can be formally stated as:

• MINIMUM T-INTERFACE

Instance : Moralized dBN $\mathcal{G}_T^m = (\mathbf{X}_0 \cup ... \cup \mathbf{X}_T, E)$ of length T.

Solution : An interface $\mathbf{I} \subseteq \mathbf{X}_0 \cup ... \cup \mathbf{X}_T$.

Measure : $size(\mathbf{I})$

We recall the both problems MINIMUM s-t CUT and MINIMUM s-t VERTEX CUT that are used in our reduction:

• MINIMUM s-t CUT

Instance : Directed graph $G = (\{s, t\} \cup V, E)$, a weight function $w : E \to R$.

Solution : A subset $E' \subseteq E$ such that every path from s to t contains an arc in E'.

Measure : $\sum_{e \in E'} w(e)$.

• MINIMUM s-t VERTEX CUT

Instance : Graph $G = (\{s,t\} \cup V, E)$, a weight function $w : V \to N$.

Solution : A subset $V' \subseteq V - \{s,t\}$ such that every path from s to t contains a node in V'.

Measure : $\sum_{v \in V'} w(e)$.

We state and prove the following theorem:

Theorem 2. MINIMUM T-INTERFACE *polynomially reduces to* MINIMUM s-t CUT *and can be then solved in polynomial time.*

Proof. To prove this theorem, we use a composition of two polynomial transformations that are stated and proved in the two following lemma. We give for both lemma a sketch of proof since the proof is trivial. We refer to Figure 4 for a detailed transformation.

Lemma 2. MINIMUM T-INTERFACE *polynomially reduces to* MINIMUM s-t VERTEX CUT.

Proof. It is quite easy to see that an instance I of MINIMUM T-INTERFACE can be cast into an instance I' of MINIMUM s-t VERTEX CUT. Indeed, it consists essentially of adding a source and a sink to the dBN. Note that solutions of I and I' are of equal size. □

Lemma 3. MINIMUM s-t VERTEX CUT *polynomially reduce to* MINIMUM s-t CUT.

Proof. The transformation essentially consists of splitting each node $v \in V$ into two nodes v_i (*in-node*), v_o (*out-node*) with an arc (v_i, v_o) (*arc-node*) and a weight on this arc equal to the weight of v. An edge (u,v) is replaced by two arcs (u_o, v_i) and (v_o, u_i) with weights equal to $+\infty$. By construction, a solution of the former problem is a solution of equal size to the second problem by considering the corresponding *arc-node* and *vice versa*. □

By Lemma 2 and 3 the result follows. □

In Figure 2, when the value of T increases up to $T = 2$ the size of the minimum interface decreases. Hence, it remains open the question

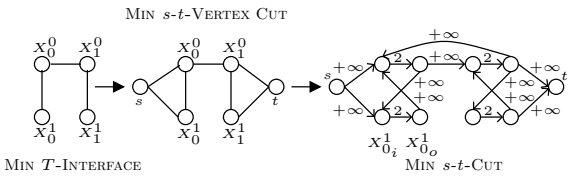

Figure 4: An example of a polynomial transformation from MINIMUM T-INTERFACE to MINIMUM s-t CUT. Variables are all binaries.

for which value of T we find the smallest interface by resolving MINIMUM T-INTERFACE. We claim that this value is less or equal to the number of nodes in a slice. Let $\mathbf{I}^{(t)}$ an optimal solution of MINIMUM T-INTERFACE with $T = t$, we first prove the following lemma:

Lemma 4. *For all $a, b \geq 0$ such that $a \geq b$,*

$$size(\mathbf{I}^{(a)}) \leq size(\mathbf{I}^{(b)})$$

Proof. For a dBN of length a, every path from a node of $\mathbf{X}_0$ to a node of $\mathbf{X}_b$ contains, by definition, a node in $\mathbf{I}^{(b)}$. Moreover, every path from a node of $\mathbf{X}_0$ to a node of $\mathbf{X}_a$ contains a node in $\mathbf{X}_b$ and then a node in $\mathbf{I}^{(b)}$. Hence, $\mathbf{I}^{(b)}$ is also an interface for the dBN of length a which implies $size(\mathbf{I}^{(a)}) \leq size(\mathbf{I}^{(b)})$. □

The following proposition give a first characterization of the optimal value for T:

Proposition 2. *Let t^* be the time such that:*

$$t^* = \min_t \{t \in N : size(\mathbf{I}^{(t)}) = size(\mathbf{I}^{(t+1)})\}$$

Then for all $t \geq 0$, we have that:

$$size(\mathbf{I}^{(t^*)}) \leq size(\mathbf{I}^{(t)})$$

Proof. By theorem 2, finding the minimum interface in a moralized dBN of length T $\mathcal{G}_T^m$ can be viewed as maximizing a flow in the directed graph $\mathcal{G}_T^{m'}$ obtained from $\mathcal{G}_T^m$ by applying the polynomial transformation given in the proof. Let α_t the value of the maximum flow in the graph $\mathcal{G}_t^{m'}$, by the max-flow min-cut theorem and since optimal solution of the two problems are of the same size, we can write $size(\mathbf{I}^{(t)}) = size(\mathbf{I}^{(t+1)}) = \alpha_t = \alpha_{t+1}$. Hence, in

the slice t of $\mathcal{G}^{m'}_{t+1}$ the value of the flow is maintained to the slice $t+1$. Since the topology of a dBN does not vary over time, if we could preserve a flow of value α_t from slice t to slice $t+1$, then we could preserve the flow from slice $t+1$ to slice $t+2$. Moreover, the value of the flow can not increase to a value $\alpha_{t+2} > \alpha_t$ by lemma 4. By inductively applying these arguments, we have $\alpha_t \leq \alpha_{t'}$ for all $t' > 0$ and thus $size(\mathbf{I}^{(t)}) \leq size(\mathbf{I}^{(t')})$ for all $t' > 0$. □

We can now give an upper bound of the optimal value of T:

Proposition 3. *Let a dBN with h nodes per slices, then for all $t \geq 0$, $size(\mathbf{I}^{(2h-1)}) \leq \mathbf{I}^{(t)}$.*

Proof. By lemma 4, if $t \leq 2h-1$ then $size(\mathbf{I}^{(2h-1)}) \leq size(\mathbf{I}^{(t)})$. Assume $t > 2h-1$ and $size(\mathbf{I}^{(2h-1)}) > \mathbf{I}^{(t)}$. By proposition 2, we must have $size(\mathbf{X}_0) = size(\mathbf{I}^{(0)}) > size(\mathbf{I}^{(1)}) > ... > size(\mathbf{I}^{(2h-1)}) > size(\mathbf{I}^{(t)})$. Hence, let $i \in \{0, ..., 2h-1\}$, either we reduce the size of $\mathbf{I}^{(i)}$ to get $\mathbf{I}^{(i+1)}$ by removing a node or by replacing a node with another one of lower size. We can remove at most h nodes, and replacing at most $h-1$ nodes, thus $size(\mathbf{I}^{(2h-1)}) = 0$ and $size(\mathbf{I}^{(t)}) = 0$ which leads to a contradiction. □

Using propositions 3 and 2, and the theorem 2, we deduce the `Min-Interface` algorithm (see Figure 5) that finds the smallest interface in polynomial time.

4.2 Building the elimination order

Once we have the interface $\mathbf{I}_0$ found by `Min-Interface`, the next step is to build the constrained elimination order. To do this, consider the sets $\mathbf{I}_i = \{X^k_{t+i} : X^k_t \in \mathbf{I}_0\}$ for $i = 1, ..., n$ of a dBN of length T. Each $\mathbf{I}_i$ is obtained by sliding $\mathbf{I}_0$ by i time step. As previously discussed in section 3, since $\mathbf{I}_i$ is straightforwardly an interface, we can split the dBN into two parts $\mathbf{L}_i$ and $\mathbf{R}_i$ where $\mathbf{L}_i$ is the set of nodes in the left of $\mathbf{I}_i$ and $\mathbf{R}_i$ the set of nodes in its right. We then define the following elimination order:

$$\text{MIN-ELIM} = \mathbf{L}_0, \mathbf{L}_1 - \mathbf{L}_0, ..., \mathbf{L}_n - \mathbf{L}_{n-1}, \mathbf{I}_n \cup \mathbf{R}_n$$

Require: A 2-TBN.
Ensure: A minimum interface
  prevSize $\leftarrow +\infty$
  $\mathbf{I} \leftarrow \mathbf{X}_0$.
  $T \leftarrow 1$
  while $T < 2h-1$ **do**
    prevSize $\leftarrow size(\mathbf{I})$
    $\mathcal{G}_T \leftarrow$ Unroll the 2-TBN T time steps
    $\mathcal{G}^m_T \leftarrow$ moralize($\mathcal{G}_T$)
    Solve MIN T-INTERFACE with $\mathcal{G}^m_T$ to get $\mathbf{I}$
    if $size(\mathbf{I})$ = prevSize **then**
      return $\mathbf{I}$
    end if
    $T \leftarrow T+1$
  end while
  return $\mathbf{I}$.

Figure 5: Pseudo-code of the `Min-Interface` algorithm. Its polynomial time complexity is ensure by the $O(h)$ iterations and by the theorem 2.

Note that we do not impose any constraints in the elimination order of nodes in each $\mathbf{L}_i - \mathbf{L}_{i-1}$, $\mathbf{L}_0$ and $\mathbf{I}_n \cup \mathbf{R}_n$. In our experiments, we used the min-fill heuristic but other approaches can be considered. We now give an upper bound of the maximum clique size if we apply the elimination sequence MIN-ELIM.

Proposition 4. *For all $i > 0$, when eliminating nodes in $\mathbf{L}_i - \mathbf{L}_{i-1}$, we create a clique of size at most $size(\mathbf{I}_i).s_{slice}$ where $size(\mathbf{I}_i) \leq min(i^{\rightarrow}, i^{\leftarrow})$.*

Proof. To get $\mathbf{L}_i$ from $\mathbf{L}_{i-1}$ it suffices to add nodes X^k_t such that $X^k_{t-1} \in \mathbf{L}_{i-1}$ or $t = 0$. Then we have $\mathbf{L}_i - \mathbf{L}_{i-1} = \{X^0_{t_1}, ..., X^n_{t_n}\}$ and thus $size(\mathbf{L}_i - \mathbf{L}_{i-1}) = size(\{X^0_{t_1}, ..., X^n_{t_n}\}) = s_{\text{slice}}$. When eliminating nodes in $\mathbf{L}_i - \mathbf{L}_{i-1}$, the set of involved nodes are $(\mathbf{L}_i - \mathbf{L}_{i-1}) \cup \mathbf{I}_i$ (since $\mathbf{I}_i$ is an interface) then the size of the created clique is less or equal to $size(\mathbf{L}_i - \mathbf{L}_{i-1}).size(\mathbf{I}_i) = size(\mathbf{I}_i).s_{\text{slice}}$. Finally, by proposition 1 and the optimality of $\mathbf{I}_i$, we have $size(\mathbf{I}_i) \leq \min(i^{\rightarrow}, i^{\leftarrow})$. □

Remark. The previous proposition is a "local" improvement of theorem 1. It is "local" be-

cause when we eliminate nodes in $\mathbf{L}_0$ or in $\mathbf{I}_n \cup \mathbf{R}_n$ the created clique could be larger than $size(\mathbf{I}_i).s_{\text{slice}}$, but (1) this concerns a fix fraction of the dBN and thus the average size of the cliques in the triangulated dBN converge to $size(\mathbf{I}_i).s_{\text{slice}}$ when $T \to +\infty$ (2) we could use the B-SS (or the F-SS) elimination order to have a guarantee on the size of the maximum clique when eliminating nodes in these sets.

5 Experiments

In this section we present triangulation results on classical dBNs and random dBNs using methods based on (Ide and Cozman, 2002). To perform these tests, we used the C++ aGrUM library (`http://agrum.lip6.fr`).

		B-SS	F-SS	MIN-ELIM
Fig. 3.3 of Murphy (2002)	Mean	3.16	**3.03**	**3.03**
	Bounds	0.69–3.46	0.69–3.46	0.69–3.46
Fig. 3.2 of Murphy (2002)	Mean	**5.54**	**5.54**	**5.54**
	Bounds	5.54–5.54	5.54–5.54	5.54–5.54
Fig. 3-D of Bilmes (2003)	Mean	**1.38**	3.00	**1.38**
	Bounds	1.39–1.39	1.39–3.46	1.39–1.39
Fig. 2 of Darwiche (2001)	Mean	**2.07**	3.45	**2.07**
	Bounds	1.39–2.08	1.39–3.46	1.39–2.08
Fig. 6 of Bilmes (2003)	Mean	3.46	3.23	**3.00**
	Bounds	2.08–4.16	2.08–3.46	2.08–3.46

Figure 6: Results on real dBNs.

Figure 6 shows results for classical dBNs. Each row represents a dBN and each column corresponds to an elimination order used for triangulation. Each dBN is unrolled up to $T = 500$ time steps. We report the mean size of the cliques in the triangulated dBN and the minimum and the maximum clique. We see that our method always gives triangulated dBN that have the minimum mean size of cliques and for the last dBN it is the only one that gives the best mean size.

Figure 7 shows results for randomly generated dBN. Each dBN contains 5 ((a) and (b)), 10 ((c) and (d)) or 15 ((e) and (f)) variables per slices with random variable cardinalities chosen uniformly between 2 and 8. All dBNs are unrolled up to $T = 500$ time steps. In figure 7 (a), (c) and (e), for each generated dBN we compute the mean size of the cliques x, y_1 and y_2 in the triangulated dBN using respectively MIN-ELIM, F-SS and B-SS, then we plot the point (x, y_1) as "$\times$" and the point (x, y_2) as "$\circ$". We report in

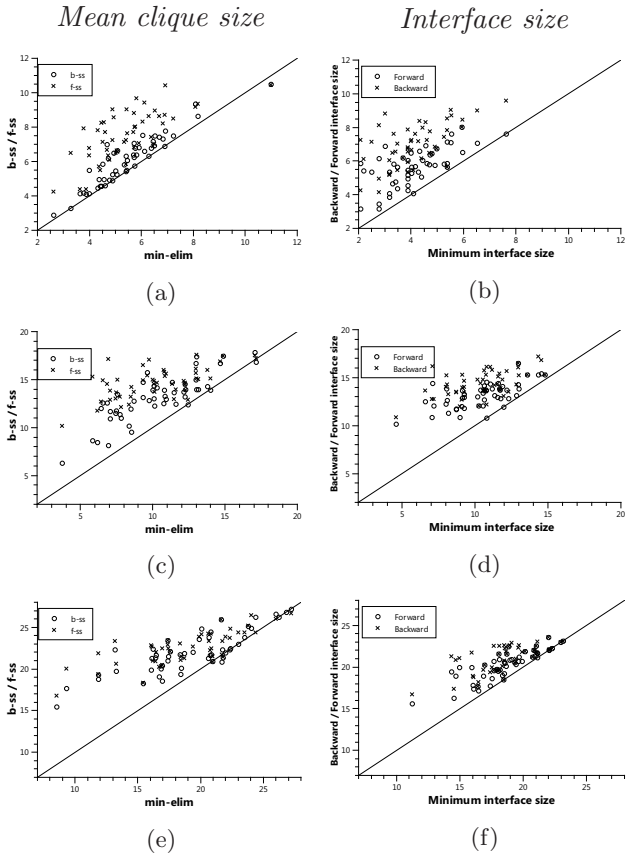

Figure 7: Results on random generated dBNs.

figure 7 (b), (d) and (f) the size of the backward, forward and minimum interface in the same way. Every points above the first diagonal shows an improvement thanks to our algorithm. As we can see, our constrained elimination order improve the triangulation quality for almost all generated dBNs.

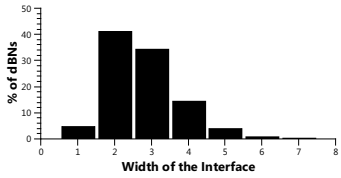

Figure 8: Minimum interfaces often span across many slices.

Figure 8 shows the percent of dBNs given the *width* of the minimum interface, *i.e.* the number of slices the interface spans across. This shows that allowing the interface to span across many slices is a useful approach.

6 Conclusion

In this paper, we studied the problem of finding good quality constrained elimination order for triangulating a dBN. We propose a polynomial algorithm to compute such order using the concept of interface in a dBN. We first show that an interface is equivalent to a cut-set in a graph, which allows us to find the minimum interface in polynomial time and then to construct a constrained elimination that have a theoretical guarantee on the maximum clique size in the triangulated graph. Experimental results show that our approach of using a polynomial time pre-treatment allows to increase the quality of the triangulation.

In future work, we plan to use this approach with approximate inference algorithms, *e.g.* loopy belief propagation, factored frontier algorithm (Murphy and Weiss, 2001), in which we could take advantage of using a small interface.

Acknowledgments

This research was supported by the ANR SKOOB project (`http://skoob.lip6.fr`).

References

Stefan Arnborg, Derek G. Corneil, and Andrzej Proskurowski. 1987. Complexity of finding embeddings in a k-tree. *SIAM Journal on Algebraic and Discrete Methods*, 8(2):277–284.

Cédric Baudrit, Mariette Sicard, Pierre-Henri Wuillemin, and Nathalie Perrot. 2009. Dynamic bayesian networks for modelling food processing: Application to the cheese ripening process. In *8th World Congress of Chemichal Engineering 09*, Montréal (Canada).

Jeff Bilmes and Chris Bartels. 2003. On Triangulating Dynamic Graphical Models. In *Uncertainty in Artificial Intelligence*, pages 47–56, Acapulco, Mexico.

Gregory F. Cooper. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405.

Adnan Darwiche. 2001. Constant-space reasoning in dynamic Bayesian networks. *International Journal of Approximate Reasoning*, 26:161–178.

Thomas Dean and Keiji Kanazawa. 1990. A model for reasoning about persistence and causation. *Computational Intelligence*, 5(3):142–150.

Dieter Fox, Wolfram Burgard, and Sebastian Thrun. 1999. Markov Localization for Mobile Robots in Dynamic Environments. *Journal of Artificial Intelligence Research*, 11:391–427.

Pinar Heggernes. 2006. Minimal triangulations of graphs: A survey. *Discrete Mathematics*, 306(3):297–317.

Jaime S. Ide and Fabio G. Cozman. 2002. Random Generation of Bayesian Networks. In *Brazilian Symposium on Artificial Intelligence*, pages 366–375. Springer-Verlag.

Finn V. Jensen, Steffen L. Lauritzen, and Kristian G. Olesen. 1990. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quaterly*, 4:269–282.

Uffe Kjærulff. 1994. dHugin: A computational system for dynamic time-sliced Bayesian networks. *International Journal of Forecasting*, 11:89–111.

Kevin Murphy and Yair Weiss. 2001. The Factored Frontier Algorithm for Approximate Inference in DBNs. In *Uncertainty in Artificial Intelligence*, pages 378–385, Seattle, WA.

Kevin Murphy. 2002. *Dynamic Bayesian Networks: Representation, Inference and Learning*. Ph.D. thesis, University of California.

Judea Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.

Donald J. Rose, Endre Tarjan, and Robert George S. Lueker. 1976. Algorithmic Aspects of Vertex Elimination on Graphs. *SIAM Journal on Computing*, 5(2):266–283.

Philippe Weber. 2002. Dynamic bayesian networks model to estimate process availability. In *8th International Conference Quality, Reliability, Maintenance*, Sinaia (Romania).

Geoffrey Zweig. 1996. A Forward-Backward Algorithm for Inference in Bayesian Networks and An Empirical Comparison with HMMs. Master's thesis, University of California.