Temporally Invariant Junction Tree for Inference in Dynamic Bayesian Network

Y. Xiang

Department of Computer Science
University of Regina
Regina, Saskatchewan, Canada S4S 0A2
Phone: (306) 585-4088, E-mail: yxiang@cs.uregina.ca

Abstract. Dynamic Bayesian networks (DBNs) extend Bayesian networks from static domains to dynamic domains. The only known generic method for *exact* inference in DBNs is based on dynamic expansion and reduction of active slices. It is effective when the domain evolves relatively slowly, but is reported to be "too expensive" for fast evolving domain where inference is under time pressure.
This study explores the *stationary* feature of problem domains to improve the efficiency of exact inference in DBNs. We propose the construction of a temporally invariant template of a DBN directly supporting exact inference and discuss issues in the construction. This method eliminates the need for the computation associated with dynamic expansion and reduction of the existing method. The method is demonstrated by experimental result.

`Keywords:` probabilistic reasoning, temporal reasoning, knowledge representation, dynamic Bayesian networks.

1 Introduction

Dynamic Bayesian networks (DBNs) [5, 9] extend Bayesian networks (BNs) [10] from static domains to dynamic domains, i.e., domains that change their states with time. A DBN consists of a finite number of "slices" each of which is a domain dependence model at a particular time interval. Slices corresponding to successive intervals are connected through arcs that represent how the state of the domain evolves with time. Collectively, the slices represent the dynamic domain over a period of time.

When inference must be performed over an extended period of time, it is not feasible to maintain all slices accumulated in the past. Kjaerulff [9] proposed a method, which we shall refer to as the *dynamic expansion and reduction* (DER) method, to perform *exact* inference by dynamically adding new slices and cutting off old slices. To the best of our knowledge, it is the only method explicitly designed for exact inference in DBNs. However, as networks become more complex, the method does not provide satisfactory performance in time-critical domains [7].

In this paper, we investigate ways to improve the efficiency of exact run time inference computation when the domain is either *stationary* or close to be stationary. In Section 2, we define the terminology. Graph-theoretic terms that may not be familiar to some readers are included in Appendix. In Section 3, we propose the construction of a temporally invariant representation to support exact run time inference computation. We discuss technical issued involved in the subsequent two sections and demonstrate our method with an experiment in Section 6.

2 Dynamic Bayesian Networks

A DBN [5, 9] is a quadruplet

$$\mathcal{G}^K = (\bigcup_{i=0}^{K} N_i, \bigcup_{i=0}^{K} E_i, \bigcup_{i=1}^{K} F_i, \bigcup_{i=0}^{K} P_i).$$

Each N_i is a set of nodes labeled by variables. N_i represents the state of a dynamic domain at time interval $t = i$ $(i = 0, \ldots, K)$. Collectively, $N = \bigcup_{i=0}^{K} N_i$ represents the states of the dynamic domain over $K + 1$ intervals. Each E_i is a set of arcs between nodes in N_i, which represent conditional independencies between domain variables at a given interval. Each F_i is a set of *temporal* arcs each of which is directed from a node in N_{i-1} to a node in N_i $(i = 1, \ldots, K)$. These arcs represent the Markov assumption: the future states of the domain is conditionally independent of the past states given the present state. The subset of N_i $(0 \leq i < K)$

$$FI_i = \{x \in N_i | (x, y) \in F_{i+1}\}$$

is called the *forward interface* of N_i, where (x, y) is a temporal arc from x to y. The subset of N_i $(0 < i \leq K)$

$$BI_i = \{y \in N_i | (x, y) \in F_i\} \cup \{z \in N_i | z \in \pi(y) \ \& \ (x, y) \in F_i\}$$

is called the *backward interface* of N_i, where $\pi(y)$ is the set of parent nodes of y. Arcs of E_i and F_i are so directed that $D_i = (N_i \cup FI_{i-1}, E_i \cup F_i)$ is a directed acyclic graph (DAG). Each P_i is a conditional probability distribution

$$P_i = \begin{cases} P(N_0) & i = 0 \\ P(N_i | FI_{i-1}) & i > 0 \end{cases}$$

specified by a set of probability tables one for each variable x in N_i conditioned on $\pi(x)$. The pair $S_i = (D_i, P_i)$ is called a *slice* of the DBN and D_i is called the structure of S_i. Collectively, the slices of a DBN define a Bayesian network, whose structure is the union of slice structures and whose joint probability distribution (jpd) is the product of probability tables in all slices.

Figure 1 shows the structure of a DBN where $N_1 = \{a_1, b_1, c_1, d_1, e_1, f_1\}$, $E_1 = \{(a_1, b_1), (b_1, c_1), (b_1, d_1), (c_1, e_1), (d_1, e_1), (e_1, f_1)\}$, $F_1 = \{(a_0, b_1), (f_0, f_1)\}$, $FI_1 = \{a_1, f_1\}$ and $BI_1 = \{a_1, b_1, e_1, f_1\}$.

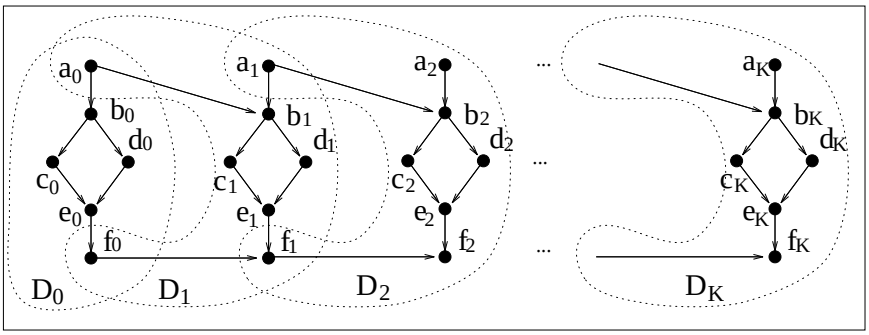

Fig. 1. A dynamic Bayesian network.

At any time $t = j \leq K$, the slices $S_0, \ldots, S_{j-1}$ represent the domain history and $S_{j+1}, \ldots, S_K$ predict the future. Evidence (observations obtained in the past and present) may be entered into $S_0, \ldots, S_j$. Limited by computational resource, normally only $S_i, \ldots, S_K$ $(i \leq j \leq K)$ are explicitly maintained, called *active slices* of the DBN.

We assume that the DBN is *connected*. Otherwise the domain can be partitioned into independent subdomains each of which can be represented by a separate DBN.

3 Temporally Invariant Template

Kjaerulff [9] proposed the DER method to perform *exact* inference in DBNs. The method dynamically adds new slices to the front of active slices, converts the expanded slices into a junction tree (JT) [8] representation, reduces the JT by removing the parts corresponding to slices in the most remote history, and uses the reduced JT to process new evidence.

The DER method is effective for domains that evolve relatively slowly, e.g., monitoring the effect of medical therapy [1] or commercial forecasting [3]. However, for fast evolving domains where inference computation is under time pressure, e.g., mobile robot navigation [6] or automated vehicles [7], the computation is "too expensive" as reported in [7].

We attribute the unsatisfactory performance of the DER method partially to the expensive computation during dynamic expansion and reduction. We argue that in many practical applications, the domain is *stationary* or at least is stationary for an extensive period of time before changing to a different (stationary) state. When the domain is stationary, the slices of the DBN are invariant with time. If the number of active slices is also a constant, then dynamic expansion/reduction by the DER method is unnecessarily repeated over and over again.

Forbes et al. [7] recognized this opportunity for improvement. They proposed to precompile the slice of a stationary DBN into a "temporally invariant net-

work". However, since the approach that they took was to replace exact inference by approximate inference using stochastic simulation, they did not deal with the issue of establishing a stable representation directly capable of *exact* inference.

Exact inference in BNs in general has been shown to be NP-hard [2]. Moreover, approximate inference is also NP-hard [4]. On the other hand, efficient algorithms for exact inference [10, 8, 13] are available when the graphical structure of a BN is sparse. The approach taken in this study is to investigate ways to improve the efficiency of exact inference. To this end, we explore the stationary property of the DBN by precompiling a run-time slice representation capable of supporting more efficient exact inference.

We assume that the number of active slices of the DBN is a constant (relaxed in Section 7) $m \geq 1$. Since the domain is stationary, the m active slices at any two time intervals are identical. For the study of inference efficiency, it makes no difference to treat the m slices as one big slice. Therefore, without loss of generality, we can consider only the case $m = 1$.

As stated, we want to precompile some slice representation of a stationary DBN that supports more efficient exact inference. The representation consists of a graphical structure and the associated conditional probabilities. Once such a representation is constructed, a copy of it can be stored as a *template* which we shall denote by T. Inference using the template works as follows:

At any time interval $t = i$, place an active copy T_i of T in the memory. T_i is identical to T except that it has absorbed all evidence acquired when $t < i$. To proceed in time, we evolve T_i into T_{i+1}. First, we cache the belief on some interface (defined below) between T_i and T_{i+1}. Then belief of T is copied into T_i. This effectively makes T_i identical to T without changing the overall data structure of T_i (e.g., the internal representation of the graphical structure). The belief of T_i is then updated using cached belief, which turns (physical) T_i into (logical) T_{i+1}. Now T_{i+1} has emerged and is ready to process new evidence while T_i has vanished.

We emphasize that the above inference uses only two *physical* copies of the template, T and T', and only the belief of T' is modified from interval to interval. Much computation required by DER method is no longer needed.

4 Defining Subnet

In order to construct the template, a portion of the DBN must be selected, which we refer to as a *subnet*. It may or may not be identical to a slice. The subnet may be multiply connected in general. To allow efficient exact inference, we convert it into a JT representation [8] as the run-time template.

Instead of defining the subnet first and then converting to template, we may *conceptually* first convert the DBN into a JT, then select a subtree T of it as the template, and finally determine the corresponding subnet. The subnet/template pair must be such that the template contains exactly the set of variables of the subnet, namely, no clique in T contains variables outside the subnet. When this

is the case, we say that the subnet and the subtree template *covers* each other. We will define the subnet in this way.

As in the standard method [8], the process of converting the DBN into a JT consists of moralization, triangulation, organizing cliques into a JT, and assigning belief to each clique. As we shall see, to ensure that the subnet is covered by a subtree, triangulation is the key step in this process.

We define some minimum separator of the moral graph of DBN as the interface between T_i and T_{i+1}, denoted by I_i. This is semantically correct since variables in a separator renders the two groups of variables it separates conditionally independent.

We use node elimination to triangulate (Appendix) the moral graph. The elimination order will be consistent with the order that each T_i emerges and vanishes. That is, for each T_i $(0 < i \leq K)$, nodes contained in T_j $(0 \leq j < i)$, except I_i, are eliminated before any node of T_i. We shall call any such order a *temporal elimination order*. We show that in the resultant triangulation, the interface I_i is complete.

Proposition 1 *Let $G = \{N, E\}$ be the moral graph of a stationary DBN. Let $I_i \subset N_{i-1} \cup N_i$ $(1 < i \leq K)$ be a minimum graph separator of G. Let $\{N_a, I_i, N_b\}$ be a partition of N such that N_a and N_b are separated by I_i. Let G be triangulated into G' by eliminating all nodes in N_a before any node in $I_i \cup N_b$ is eliminated.*

Then I_i is complete in G'.

Proof:

We show that an arbitrary pair of nodes in I_i is connected in G'. Since the DBN is stationary, there exists $I_{i-1} \subset N_{i-2} \cup N_{i-1}$ for $i \geq 2$. Consider a pair of nodes x_i and y_i in I_i and the corresponding node x_{i-1} in I_{i-1}.

Since the DBN is connected and I_i is minimum, there exists a path from x_i to x_{i-1} such that all nodes on the path are contained in N_a, except x_i. Otherwise, for every path from x_i to x_{i-1}, there exists a node $z_i \in I_i$. In that case, x_i may be removed from I_i such that I_i is still a separator, which contradicts the assumption that I_i is minimum.

For the similar argument, there exists a path from y_i to x_{i-1} such that all nodes on the path are contained in N_a. Hence, there exists a path from x_i to y_i such that all nodes on the path are contained in N_a. Due to Lemma 4 in Rose et al. [11], the link $\{x_i, y_i\}$ is in G'. □

Proposition 1 implies that I_i is contained in a clique of G' and so is I_{i-1}. If we organize cliques of G' into a JT, then all nodes of the DBN between I_i and I_{i-1} can be covered by a subtree that connects to the rest of the JT through these two cliques. This subtree (a JT) can then be used as the run-time template. This is justified in Theorem 3. Proposition 2 prepares for its proof.

Proposition 2 *Let I be a complete separator between nodes x and y in a triangulated graph G. Let C_x and C_y be two cliques of G such that $x \in C_x$ and $y \in C_y$. Then there exists a JT T of G such that I is either a sepset on the simple path from C_x to C_y in T or is contained in a clique on that path.*

Proof:

The set N of nodes of G can be partitioned into $\{N_x, I, N_y\}$ such that $x \in N_x$, $y \in N_y$, and N_x and N_y are separated by I. Since I is a complete separator, the subgraph G_x spanned by $N_x \cup I$ is triangulated and so is the subgraph G_y spanned by $N_y \cup I$. Hence, a JT T_x of G_x exists and so does a JT T_y of G_y.

The two JTs can be combined into a single JT as follows: Identify a clique Q_x in T_x containing I and a clique Q_y in T_y containing I. If one of the cliques equals I, then join the two JTs by unioning Q_x and Q_y. If none of the cliques equals I, then join the two JTs by a sepset I. The resultant is a JT that satisfies the requirement. □

The following theorem shows that a JT of a DBN can be found that consists of a sequence of (sub)JTs chained together. The subJT will be our template.

Theorem 3 *Let $G = \{N, E\}$ be the moral graph of a stationary DBN. Let $I_i \subset N_{i-1} \cup N_i$ $(1 < i \leq K)$ be a minimum graph separator of G. Let $\{N_a, I_{i-1}, N_b, I_i, N_c\}$ be a partition of N, where N_a and N_b are separated by I_{i-1}, and N_b and N_c are separated by I_i.*

Then there exists a temporal elimination order triangulating G into G', and there exists a JT $\mathcal{T}$ of G' that satisfies the following conditions:

1. *There exists a subtree T_i connected to the rest of $\mathcal{T}$ through two cliques $C_{i-1} \supseteq I_{i-1}$ and $C_i \supseteq I_i$ such that every clique in T_i is a subset of $I_{i-1} \cup N_b \cup I_i$ except that C_{i-1} may contain nodes in N_a and C_i may contain nodes in N_c.*
2. *For each $y \in N_b$, y is contained in nowhere in $\mathcal{T}$ except in T_i.*

Proof:

It suffices to show that for any $x \in N_a$ and $y \in N_b$ where x is contained in a clique C_x and y is contained in a clique C_y, it must be the case that $C_x \neq C_y$ and C_{i-1} is on the path between C_x and C_y in some $\mathcal{T}$ obtained by some temporal elimination order.

According to a temporal elimination order, x is eliminated before y. They are in a same clique of G' iff they are connected when x is eliminated. Since I_{i-1} is the separator between x and y, y is not in the adjacency of x, and hence they are not connected at the time x is eliminated. Hence $C_x \neq C_y$.

By Proposition 1, since I_{i-1} is a minimum separator, I_{i-1} is complete in G' using any temporal elimination order. Hence C_{i-1} exists in G'. By Proposition 2, it follows that C_{i-1} is on the path between C_x and C_y. □

We can now define the subnet based on such a template.

Definition 4 *Let I_i be a minimum separator in the moral graph of a DBN. Let $\{N_a, I_{i-1}, N_b, I_i, N_c\}$ be a partition of N such that N_a and N_b are separated by I_{i-1}, and N_b and N_c are separated by I_i. The subgraph spanned by $I_{i-1} \cup N_b \cup I_i$ defines the structure of a subnet relative to separator I_i.*

Through previous *conceptual* analysis, we have understood what the structure of a subnet should be. In *practice*, the subnet obtained by Definition 4 will be the starting point in the construction of a template.

5 Choosing Separator

Given the moral graph of a DBN, there are many minimum separators. We first consider two immediate choices: the forward and backward interface. The following propositions show that both can be used as the basis in choosing the separator.

Proposition 5 *Backward interface BI_i is a separator in the moral graph of DBN.*

Proof:

BI_i contains the head of each temporal arc and the parents of the head. In the moral graph, every simple path from a node in N_{i-1} to a node in N_i must contain either a temporal link or a moral link. Hence, deletion of BI_i renders them separated. □

Proposition 6 *Forward interface FI_i is a separator in the moral graph of DBN.*

Proof:

FI_i contains the tail of each temporal arc. In the moral graph, every simple path from a node in N_{i-1} to a node in N_i must pass the tail of a temporal arc, and then either the corresponding temporal link or a moral link. Hence, deletion of BI_i renders them separated. □

It should be noted that both forward and backward interface may not be minimum separators. For example, let $x \in N_i$ be the head of a temporal arc, and $y \in N_i$ be a parent of x. If y has no parent nor other child, then the minimum separator based on BI_i includes x but not y.

Similarly, if the tail of a temporal arc has no parent nor other child, then the minimum separator based on the forward interface does not include this node.

Construction of the template requires assignment of belief to cliques of the template JT. This is performed by assigning each node in the subnet to a unique clique in the JT that contains the family of the node. The belief of a clique C is initialized to the product of $P(x|\pi(x))$ for each x assigned to C. The family of a node in the subnet may not be identical to its family in the DBN. This may or may not cause problem in the belief assignment as discussed below:

First, consider a subnet defined based on forward interface FI_{i-1} and FI_i. The family of each node in this subnet is identical to that in the DBN except for nodes in FI_{i-1}. Since during inference, the belief on FI_{i-1} will be *replaced* by the belief on FI_{i-1} from the previously active template, the belief on these nodes can be left unassigned (equivalent to a constant belief). Hence, difference of family size for nodes in separator FI_{i-1} causes no problem to belief assignment.

On the other hand, if the subnet is defined based on backward interface BI_{i-1} and BI_i, the situation is different. For example, let $x \in N_i$ be the head of a temporal arc, and $y \in N_i$ be a parent of x. The parents of y in the DBN may not be contained in the subnet. Therefore, $P(y|\pi(y))$ as specified in the DBN cannot be included in the belief assignment. Without this piece of knowledge, a correct belief assignment of the template cannot be accomplished. Therefore,

the belief assignment of a template cannot be performed *locally* using only the subnet defined by backward interface.

Since forward interface separator allows local belief assignment and thus simplifies the implementation of the template constructor, it is generally preferred over backward interface separator. We shall call forward interface a *self sufficient* separator. In fact, it is not the only self sufficient separator. We characterize such separators as follows:

Definition 7 *Let I_i be a minimum separator in a DBN and S be a subnet defined by separators I_{i-1} and I_i. I_i is* `self sufficient` *if for each node in S, its family is identical to that in the DBN except nodes in I_{i-1}.*

Since self sufficient separators simplify template constructor, they are generally preferred over separators that are not self sufficient.

Among self sufficient separators, different separators may produce templates of different run-time computational complexity. It is known that the amount of inference computation in a JT of a BN increases as the size of the total state space (STSS) of the JT [12]. Hence a template of smaller STSS is preferred. As finding a JT with the minimum STSS is NP-hard [12], we have to settle for heuristic methods.

According to Proposition 1, the separator will be completed during triangulation. Therefore, a larger separator creates a larger clique and tends to increase the STSS of the resultant template. Furthermore, a larger separator needs more fill-ins to complete. These fill-ins may cause additional cycles which in turn require more fill-ins to triangulate the graph. The result is the additional increase of the STSS. Therefore, one useful heuristics is to choose the separator of the smallest state space, which we shall term as a *minimal* separator.

6 Experimental Demonstration

The method proposed has been implemented and tested in WEBWEAVR-III environment, a research testbed that supports many aspects of representation and inference with uncertain knowledge. The modules involved in this work include a Bayesian network editor for specifying a slice or subnet, a template constructor, and a dynamic inference engine. In the following, we demonstrate the method proposed using our implementation.

We shall demonstrate using the monitoring of a digital counter since understanding the problem requires very little domain knowledge. The counter consists of three D flip-flops (DFFs). The first DFF is driven by an external clock signal. Its output is used to drive the second DFF, whose output is in turn used to drive the third DFF. The circuit, a clock input and its normal output are shown in Figure 2.

The counter cannot be modeled using standard Bayesian networks. This is because the input and output of each DFF are changing with time. The state of each DFF can also change with time. A DFF may be initially normal but becomes abnormal. However, the topology of the circuit is fixed. The state of each DFF

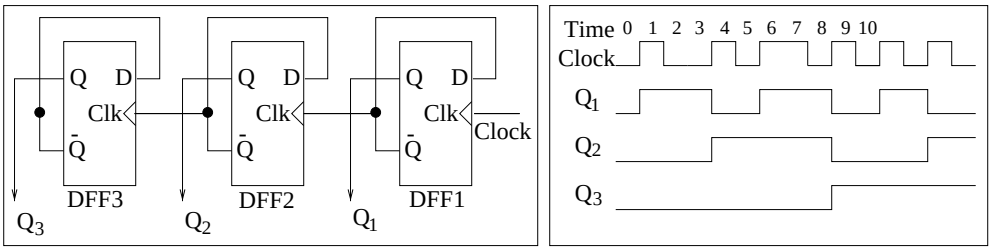

Fig. 2. Left: a digital counter made of three D flip-flops. Right: the input and output of the counter.

can be modeled as temporally changing between two types of behavior: normal and abnormal. Each type can be described without reference to time. Hence the domain is *stationary* and our proposed method is applicable.

The first DFF toggles at the positive edge (not positive level) of the clock. The edge monitoring can be modeled by a variable GotLow. At each time interval, GotLow = true if the input clock level is negative. The value implies that the next positive level will be a positive edge. When the input clock level is positive, two possible previous clock levels should be considered. If the previous clock level has been positive, then GotLow should be false since the negative level has not been seen yet. If the previous clock level has been negative, then the current positive level represents a positive edge. The value of GotLow should be reset to start the next cycle of monitoring. Hence, GotLow = false whenever the input clock level is positive. We have $P(GotLow = true|Clock = 0) = 1$ and $P(GotLow = true|Clock = 1) = 0$.

The toggling decision is made based on both GotLow value and the current clock level. This decision can be modeled by a variable Flip. Flip = yes if and only if GotLow = true and Clock = 1.

The output Q_1 is determined by the previous value of Q_1 and the Flip decision. Q_1 toggles if and only if Flip = yes.

To model the abnormal behavior of a DFF, we assume that if the DFF is abnormal, it will not toggle when it should 80% of the time. It may toggle when it shouldn't 10% of the time.

The other two DFFs can be similarly modeled. Since each of them is driven by $\overline{Q}$ of another DFF, it toggles at the negative edge of Q of the other DFF. Hence the variable GotHigh is used to model the edge monitoring.

We model the persistence of the state of a DFF as follows: If a DFF is normal at $t = i$, it may become abnormal at $t = i + 1$ with 1% probability. If it is abnormal at $t = i$, it will stay abnormal. A subnet of the DBN specified using the Bayesian network editor is shown in Figure 3, where each node is labeled by the variable name followed by the index of the node. The subnet is defined based on the forward interface. FI_{i-1} contains nodes 0, 2, 3, 4, 6, 7, 8, 10 and 11. FI_i contains nodes 13 through 21.

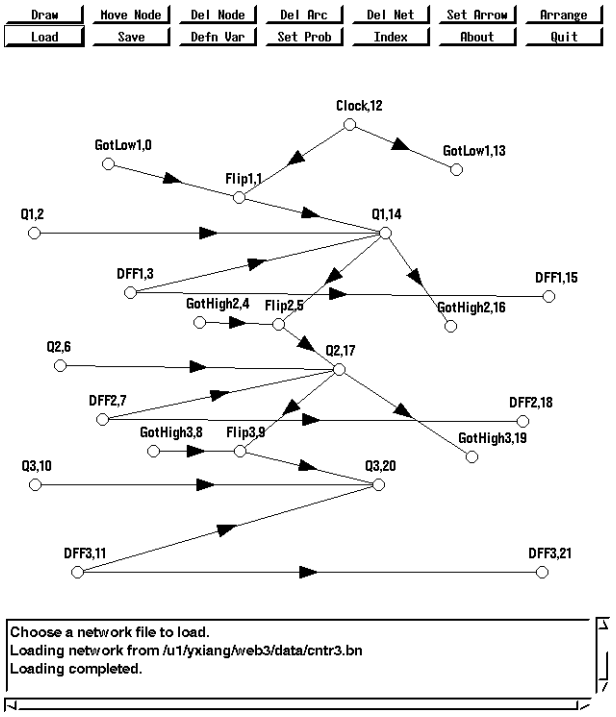

Fig. 3. A subnet of DBN for digital counter.

Once the subnet is specified, we use the template constructor to generate the template. The constructor module converts subnet into a template JT with belief assigned and initialized. The template JT generated based on the subnet is shown in Figure 4, where each clique is labeled by the indexes of member variables. The clique C_8 contains FI_{i-1} and is used to propagate evidence from the previous active template into the current template during inference. The clique C_0 contains FI_i and is used to propagate evidence from the current template into the next active template.

After the template is generated, inference can be performed using the dynamic inference engine. In our experiment, we assume that all DFFs are normal at $t = 0$. At $t = 4$, DFF2 breaks down and did not toggle. Since DFF3 is driven by the output of DFF2, the output of DFF3 is also affected. The corresponding output of DFF2 and DFF3 are shown in Figure 5. Note the difference from Figure 2.

We assume that the initial values of all variables at $t < 0$ are known, e.g., $Q_i = 0$, $GotLow1 = false$, $DFF1 = good$, etc. We assume that clock can be cheaply observed and is observed at every time interval. The observation of output of each DFF incurs a cost, and hence only one DFF is observed at a time interval. The first observation is made on Q_1 at $t = 4$. At $t = 5$, 6 and 7, Q_2, Q_3

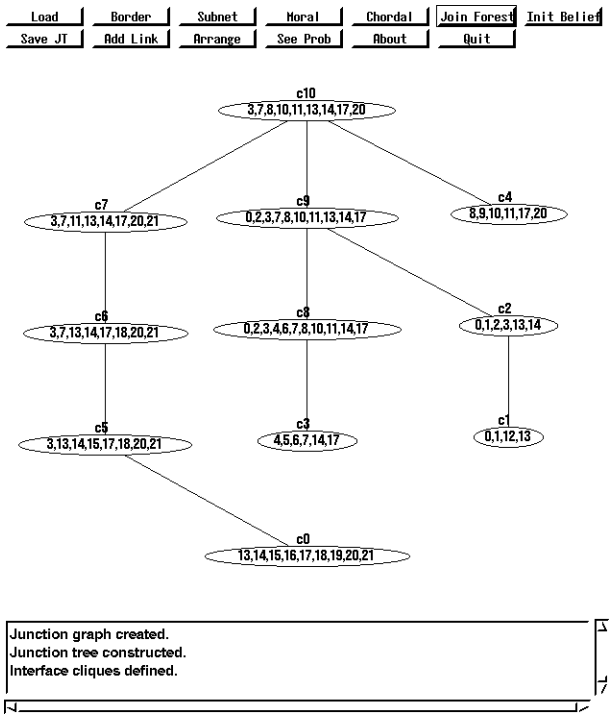

Fig. 4. The template of DBN for digital counter.

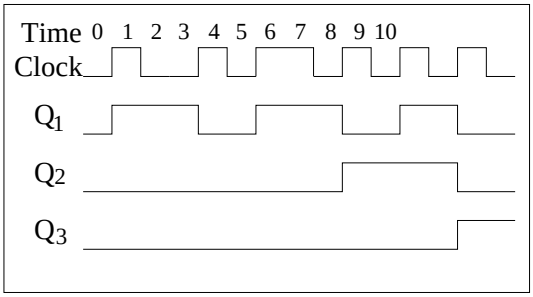

Fig. 5. Incorrect output due to breading down of DFF2 at $t = 4$.

and Q_1 are observed respectively, and so on. No other variables are observable after $t = 0$.

Figure 6 shows the belief at $t = 3$. Since no observation has been made except on clock, the inference engine has simulated the expected output of each DFF from $t = 0$ to $t = 3$ essentially based on their normal behavior. The first observation on Q_1 is made at $t = 4$ (not shown in figures due to space limit). Although Q_2 becomes abnormal and does not toggle at this interval, the observation on Q_1 does not reflect the problem yet.

Figure 7 shows the belief at $t = 5$. Since the observed value of Q_2 is in-

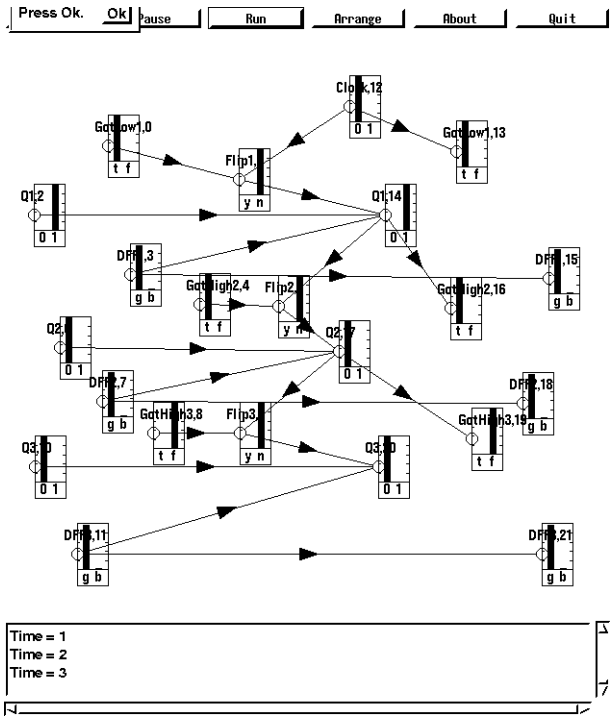

Fig. 6. The belief at $t = 3$.

consistent with the expected value 1, its abnormality is being suspected. Due to limited observation, DFF1 is also suspected. The suspicion on the abnormality of DFF1 is denied by subsequent observations. Hence at $t = 10$, the belief becomes $P(DFF1 = bad) = 0.05$, $P(DFF2 = bad) = 0.98$, and $P(DFF2 = bad) = 0.08$ (not shown in figures due to space). At this time, the monitor is fairly certain about the problem of the counter through tying together observations made across different time intervals.

7 Conclusions

In this work, we explore the stationary feature of problem domains to improve the efficiency of exact inference in DBNs. We propose the construction of a temporally invariant template of a DBN which can be reused at run time. This saves the run time computation associated with dynamic expansion and reduction by the DER method.

We show that once a slice of DBN is specified, the forward and backward interface form direct basis to select a minimum separator in the moral graph of the DBN. A subnet can then be defined from which the template is constructed. Unlike backward interface and other non-self sufficient separators, forward interface and other self sufficient separators allow local belief assignment using the subnet only. Thus self sufficient separators should be preferred as they simplify template construction.

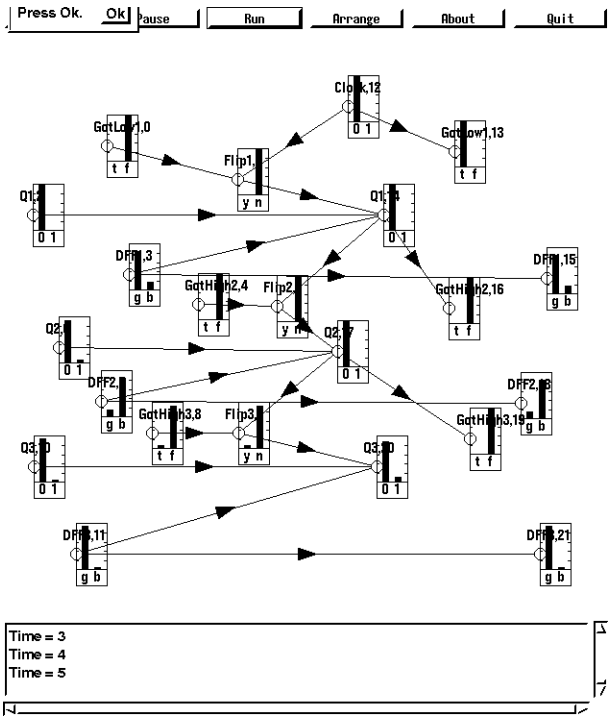

Fig. 7. The belief at $t = 5$.

Besides the property of self sufficiency, using a minimal separator appears to be a useful heuristics in order to reduce the size of total state space of the resultant template. Further experimental study is being conducted to test this heuristics.

Our approach can be extended to close-to-stationary domains. If the DBN can be expressed by a small number of distinct slices, several templates may be created for each distinct slice, one for each distinct preceding slice. The assumption of a constant number of active slices (Section 3) can also be lifted in the same way.

Our presentation has focused on inference that supports *estimation* (estimating the current state of some unobserved variables) and *forecast* (predicting the future state of the domain). The template constructed can also support *backward smoothing* (re-estimating the past state of some unobserved variables). The extension is straightforward.

Acknowledgement

This work is supported by the Research Grant OGP0155425 from NSERC.

Appendix: Graph-theoretic terminology

Let G be an undirected graph. The *adjacency* of a node x is the set of nodes adjacent to x. A set X of nodes in G is *complete* if each pair of nodes in X is adjacent. A set S of nodes in G is a *separator* if deleting S makes G disconnected. S is *minimum* if no node in S may be removed such that S is still a separator. A set C of nodes is a *clique* if C is complete and no superset of C is complete. G is *connected* if there is a path between every pair of nodes. G is *multiply* connected if there exists undirected cycles in G. A *chord* is a link connecting two nonadjacent nodes. G is *triangulated* if every cycle of length > 3 has a chord.

A node x in an undirected graph $G = (N, E)$ is *eliminated* if its adjacency is made *complete* by adding links (if necessary) before x and links incident to x are removed. Each link thus added is called a *fill-in*. Let ρ be the set of fill-ins added in eliminating all nodes in some order. Then the graph $G' = (N, E \cup \rho)$ is triangulated. Let T be a graph whose nodes are labeled by cliques of G' such that intersection of any two nodes are contained in every node on the path between them. Then T is a *junction tree* (JT) of G'. We shall call a node of T as a clique if no confusion is possible. Each link in T is labeled by the intersection of the two end nodes and is called a *sepset*.

Let D be a directed graph. For any arc (x, y) (from x to y), x is called the *tail* and y is called the *head* of the arc. The *family* of a node is the union of the node and its parent nodes. The *moral* graph of D is obtained by completing parents of each node and dropping the direction of each arc. Each link added is called a *moral link*. The process of obtaining the moral graph from D is called *moralization*.

References

1. S. Andreassen, R. Hovorka, J. Benn, K.G. Olesen, and E.R. Carson. A model-based approach to insulin adjustment. In *Proc. 3rd Conf. on Artificial Intelligence in Medicine*, pages 239–248. Springer-Verlag, 1991.
2. G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.
3. P. Dagum, A. Galper, and E. Horvitz. Dynamic network models for forecasting. In D. Dubois, M.P. Wellman, B. D'Ambrosio, and P. Smets, editors, *Proc. 8th Conf. on Uncertainty in Artificial Intelligence*, pages 41–48, Stanford, CA, 1992.
4. P. Dagum and M. Luby. Approximating probabilistic inference in Bayesian belief networks is NP-hard. *Artificial Intelligence*, 60(1):141–153, 1993.
5. T.L. Dean and K. Kanazawa. A model for reasoning about persistence and causation. *Computational Intelligence*, (5):142–150, 1989.
6. T.L. Dean and M.P. Wellman. *Planning and Control*. Morgan Kaufmann, 1991.
7. J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The batmobile: towards a bayesian automated taxi. In *Proc. Fourteenth International Joint Conf. on Artificial Intelligence*, pages 1878–1885, Montreal, Canada, 1995.
8. F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, (4):269–282, 1990.

9. U. Kjaerulff. A computational scheme for reasoning in dynamic probabilistic networks. In D. Dubois, M.P. Wellman, B. D'Ambrosio, and P. Smets, editors, *Proc. 8th Conf. on Uncertainty in Artificial Intelligence*, pages 121–129, Stanford, CA, 1992.
10. J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.
11. D.J. Rose, R.E. Tarjan, and G.S. Lueker. Algorithmic aspects of vertex elimination on graphs. *SIAM J. Computing*, 5:266–283, 1976.
12. W.X. Wen. Optimal decomposition of belief networks. In *Proc. 6th Conf. on Uncertainty in Artificial Intelligence*, pages 245–256, 1990.
13. Y. Xiang, D. Poole, and M. P. Beddoes. Multiply sectioned Bayesian networks and junction forests for large knowledge based systems. *Computational Intelligence*, 9(2):171–220, 1993.

This article was processed using the LaTeX macro package with LLNCS style