Modify Bayesian Network Structure with Inconsistent Constraints

Yi Sun and Yun Peng
Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD, 21250

Abstract

This paper presents a theoretical framework and related methods for integrating probabilistic knowledge represented as low dimensional distributions (also called constraints) into an existing Bayesian network (BN), even when these constraints are inconsistent with the structure of the BN due to dependencies among relevant variables in the constraints being absent in the BN. Within this framework, a method has been developed to identify structural inconsistencies. Methods have also been developed to overcome such inconsistencies by modifying the structure of the existing BN.

1 Introduction

Knowledge integration involves modifying the existing knowledge base according to newly discovered knowledge. This process can be difficult because pieces of new knowledge coming from different sources may conflict with each other or with the existing knowledge base. In this situation we say the pieces of new knowledge are *inconsistent*.

In this paper we focus on a specific kind of knowledge integration, i.e., the discrete probabilistic knowledge integration for uncertain knowledge, where the knowledge base is represented as a discrete joint probability distribution (JPD), and pieces of new knowledge are represented as lower dimensional distributions, which are also called *constraints*. We are especially interested in representing the knowledge base using a Bayesian network (BN) for its compact representation of interdependencies (Pearl 1988). Each node in a BN represents a variable in the knowledge base, and arcs between the nodes represent the interdependencies among variables. All the nodes and edges form a directed acyclic graph (DAG) of the BN, which we refer to as the structure of BN. A conditional probability table (CPT) is attached to each node to represent the strength of the interdependencies. Using BN introduces an additional challenge for integration, i.e., a constraint may conflict with the structure of the existing BN when the probabilistic dependencies among some of the variables in the constraint are absent in the BN. We call this kind of constraint *structural inconsistent constraint*.

When the structural inconsistency occurs, one can choose to 1) keep the structure of the BN and integrate as much as possible of the knowledge in the constraint that is consistent and ignore or reject the inconsistent part of the knowledge in the constraint; 2) modify the structure of the BN so that the knowledge in the constraint can be completely integrated into the BN; or 3) find a trade-off between the above two options.

Existing methods in this area mainly take the first option. The integration of pieces of new knowledge is carried out by only updating the parameters of the existing BN while keeping the network structure intact. The rationale behind this is that in many situations the structure of the existing BN represents more stable and reliable aspects of the domain knowledge, and therefore is desirable to not change frequently. In this paper, we explore the second option to completely integrate the structural inconsistent constraint. The motivation of doing this is that when new pieces of knowledge are more up-to-date or come from reliable sources and the new dependencies they bring should be accepted, it is necessary and beneficial to modify the structure of the existing BN so that the new constraints, including the new dependencies they contain, can be integrated into the BN in its entirety.

The rest of this paper is organized as follows. In Section 2 we introduce some background and formally define the problem we set to solve. In Section 3 we briefly review the existing works related to this problem. In Section 4 we explore how to identify structural inconsistencies between the constraints and the BN. In Section 5 we investigate how to overcome the identified structural inconsistencies during knowledge integration. Finally, we show the effectiveness of the proposed methods through experiments in Section 6 and conclude the paper in Section 7 with directions of future research.

2 Background and the Problem

We will follow the following conventions in this paper. To name a set of variables, we will use capital italic letters such as *X*, *Y*, *Z* and with superscripts such as X^i, Y^j, Z^k for subsets of variables. To name their instantiations, we will use lower case italic letters such as *x*, *y*, *z* correspondingly. To name an individual variable, we use capital italic letters such as *A*, *B*, *C*. To name their instantiations, we will use lower case italic letters such as *a*, *b*, *c* correspondingly. If the individual variable belongs to a set, we will use subscripts to indicate its position in the set, such as X_i for the i^{th} variable in set *X*, and x_i for its instantiation. *P*, *Q*, *R* are specifically used to indicate probability distributions, and bold **P**, **Q**, **R** are used for sets of distributions.

Knowledge integration refers to the process of incorporating new knowledge into an existing knowledge base. The probabilistic knowledge base can be represented using JPD $P(X)$, where *X* represents the set of variables in the domain. The new knowledge that needs to be integrated into $P(X)$ usually comes from more up-to-date or more specific observations for a certain perspective of the domain. It can be represented as a lower-dimensional probability distribution over a subset of X, which is called probabilistic constraint, or constraint for short. We use the probability distribution $R_j(Y^j)$ to denote the j^{th} piece of new knowledge or the j^{th} constraint which is a distribution over the set of variables Y^j, where $Y^j \subseteq X$. A set of *m* constraints can be represented as $\mathbf{R} = \{R_1(Y^1), ..., R_m(Y^m)\}$.

Definition 1 (Constraint Satisfaction) Let $P(X)$ be a JPD, and $R(Y)$ be a constraint, where $Y \subseteq X$. $P(X)$ is said to satisfy $R(Y)$ if $P(Y) = R(Y)$, i.e., the marginal distribution of $P(X)$ on variable set Y equals $R(Y)$.

Definition 2 (Consistent Constraints) If there is at least one JPD $P(X)$ that can satisfy all the constraints in **R**, i.e., $\forall R_j(Y^j) \in \mathbf{R}, P(Y^j) = R_j(Y^j)$ holds, then we say constraints in **R** are consistent.

Definition 3 (Inconsistent Constraints) If there does not exist a JPD that can satisfy all the constraints in **R**, then we say constraints in **R** are inconsistent.

The probabilistic knowledge integration problem is to modify $P(X)$ to satisfy all the constraints in **R**. Meanwhile, it is desirable to minimize the modification to $P(X)$ during the integration process, which can be measured using some metrics such as I-divergence (Csiszar 1975; Kullback and Leibler 1951; Vomlel 1999) or its weighted version I-aggregate (Vomlel 2003) defined afterwards.

Definition 4 (I-divergence) Given two probability distributions $P(X)$ and $Q(X)$, I-divergence from $P(X)$ to $Q(X)$ is:

$$I(P \| Q) = \begin{cases} \sum_{x \in X} P(x) \log \frac{P(x)}{Q(x)} & \text{if} \quad P << Q \\ +\infty & \text{otherwise} \end{cases} \tag{1}$$

where $P << Q$ means Q dominates P, i.e., $\{X \mid P(X) > 0\} \subseteq \{X' \mid Q(X') > 0\}$.

Definition 5 (I-aggregate) Let P be a JPD, $\mathbf{Q} = \{Q_1, ..., Q_m\}$ be a set of distributions, and ω_j be a nonnegative weight for Q_j. I-aggregate is the weighted sum of I-divergence for all of the distributions in **Q**, i.e.,

$$\psi(P, \mathbf{Q}) = \sum \omega_j \cdot I(P \| Q_j) \tag{2}$$

where $0 < \omega_j < 1$ and $\sum \omega_j = 1$.

BNs have been widely used for representing large uncertaint knowledge bases. For a given BN G, we use G_s to refer to the DAG, i.e., the network structure of G, and G_P to refer to the set of CPTs. A BN can then be represented as $G = (G_S, G_P)$, where $G_S = \{(X_i, \pi_i)\}$. Here π_i is the set of parents for X_i, and $G_P = \{P(X_i \mid \pi_i)\}$. The following are some graph definitions based on the DAG of the BN (Geiger, Verma and Pearl 1990).

Definition 6 (I-Map) Given a DAG G_s, and a JPD P which uses the same set of variables as G_s, if every independency implied by G_s holds in P, then we say G_s is an I-Map of P (Geiger, Verma and Pearl 1990). A complete DAG is an I-Map of any distribution.

Definition 7 (Minimal I-Map) The minimal I-Map of P is a DAG G_s that by removing any arc from G_s introduces independencies that do not hold in P (Geiger, Verma and Pearl 1990).

From the definition of a minimal I-Map we know that, to represent P, the BN structure G_s needs to be an I-Map of P, if it is not a minimal I-Map of P. That means G_s can have more dependencies than P, but not less (Geiger, Verma and Pearl 1990). This leads to our definition for structural inconsistent constraint.

Definition 8 (Structural Inconsistent Constraint) Given DAG G_s over a set of variables X, and a constraint R over a set of variables $Y \subseteq X$, if every independency implied by G_s involving variables in Y holds in R, then we say R is structurally consistent with G_s. Otherwise, we say R is structurally inconsistent with G_s, or we say R is a structural inconsistent constraint for G_s.

Structural inconsistent constraint (defined in Definition 8) and inconsistent constraints (defined in Definition 3) refer to different types of inconsistencies, and we refer them as Type I and Type II inconsistency, respectively, in the rest of this paper. Type I inconsistency in Definition 3 is defined for a *set* of constraints. It can arise whether or not the knowledge base is represented as a BN. Type II incon-

sistency in Definition 8 is defined for an *individual* constraint with respect to a particular BN. It can occur only when the knowledge base is represented as a BN, and some dependency relations in the constraint are absent in the BN.

Figure 2 below shows some examples of inconsistent constraints for the BN in Figure 1. Constraints in Figure 2(a) have Type I inconsistency because $R1(A) \neq R2(A)$. Constraint in Figure 2(b) has Type II inconsistency because it contradicts with the dependency relation in the BN structure of Figure 1, where B and C are independent given A, while $R3(B,C \mid A) \neq R3(B \mid A) \cdot R3(C \mid A)$.

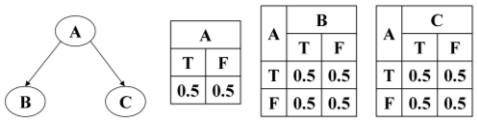

Figure 1: A 3 Node BN

A	B	R1(A,B)
T	T	0.2
T	F	0.4
F	T	0.2
F	F	0.2

A	C	R2(A,C)
T	T	0.2
T	F	0.2
F	T	0.4
F	F	0.2

(a) Constraints with Type I Inconsistency

A	B	C	R3(A,B,C)
T	T	T	0.10
T	T	F	0.15
T	F	T	0.15
T	F	F	0.10
F	T	T	0.10
F	T	F	0.15
F	F	T	0.15
F	F	F	0.10

(b) Constraint with Type II Inconsistency

Figure 2: Inconsistent Constraints

With the definition of Type I and Type II inconsistencies, the problem we set to solve is formally defined as follows:

Given BN $G = (G_S, G_P)$ with JPD $P(X)$, and a set of constraints $\mathbf{R} = \{R_1(Y^1), ..., R_m(Y^m)\}$ with at least one constraint having Type II inconsistency, construct a new BN $G' = (G_S', G_P')$ with probabilility distribution $P'(X)$ that meet the following conditions:
C1: Constraint satisfaction: $\forall R_j(Y^j) \in \mathbf{R}, P'(Y^j) = R_j(Y^j)$;
C2: Minimality: $I(P'(X) \,||\, P(X))$ is as small as possible.

3 Related Works

In this section we briefly review the existing works related to the problem. IPFP (Iterative Proportional Fitting Procedure) (Kruithof 1937) and virtual evidence method (Pearl 1990) are the two basis methods that support the other methods in this section.

IPFP is a mathematical procedure that iteratively modifies a JPD to satisfy a set of constraints while maintaining minimum I-divergence to the original distribution. Vomlel applied IPFP to knowledge integration problems with consistent constraints (Vomlel 1999). The core of IPFP is I-projection (Valtorta, Kim and Vomlel 2002). Specifically, for a set of constraints $\mathbf{R}$ and an initial JPD, the IPFP procedure is carried out by iteratively modifying the current JPD $Q_{k-1}(X)$ according to the following formula, using one constraint $R_j(Y^j)$ in $\mathbf{R}$ at a time:

$$Q_k(X) = Q_{k-1}(X) \cdot \frac{R_j(Y^j)}{Q_{k-1}(Y^j)}, \quad (3)$$

where $j = k \bmod m$, which determines the constraint used at iteration k, and m is the number of constraints in $\mathbf{R}$. Here, $Q_k(X)$ is said to be an I-projection of $Q_{k-1}(X)$ on the set of all JPDs that satisfy $R_j(Y^j)$, and as such, $Q_k(X)$ has the minimum I-divergence from $Q_{k-1}(X)$ among all the JPDs that satisfy $R_j(Y^j)$.

One limitation of IPFP-based integration methods is that it works on JPDs, not on BNs. To solve this problem, Peng and Ding proposed E-IPFP (Peng and Ding 2005) based on IPFP by adding a structure constraint:

$$Q_k(X) = \prod_{i=1}^{n} Q_{k-1}(X_i \mid \pi_i), \quad (4)$$

where $(X_i, \pi_i) \in G_s$. $Q_k(X)$ is the JPD of the BN whose CPTs are extracted from $Q_{k-1}(X)$ according to G_s (Peng et al. 2012). When constraints in $\mathbf{R}$ are consistent, E-IPFP will converge to a new BN which 1) has the same structure as the given BN, 2) satisfies all the constraints in $\mathbf{R}$, and 3) with changes to the given BN minimized.

Another limitation of IPFP is that it will not converge but oscillates when constraints are inconsistent (Csiszar 1975; Peng and Ding 2005; Vomlel 2003). To deal with inconsistent constraints (i.e., Type I), Vomlel proposed CC-IPFP (Vomlel 1999; 2003) and GEMA (Vomlel 2003) by extending IPFP. Later Peng et al. proposed SMOOTH (Zhang and Peng 2008) which has better performance than CC-IPFP and GEMA. SMOOTH circumvents the inconsistency by making bi-directional modifications during the integration process. When the knowledge base is represented as a BN, SMOOTH can be extended to E-IPFP-SMOOTH (Peng and Zhang 2010), which can deal with both Type I and Type II inconsistencies by modifying the constraints. This is acceptable when constraints are considered to contain some noise. But when new dependency relations are considered reliable, E-IPFP-SMOOTH will not be able to integrate this information into the revised BN since the structure of BN is not changed.

Next we briefly discuss virtual evidence method, not only because it plays an important role in BN belief update with uncertain evidences, but also because it serves as a basis for our proposed methods in Section 5.

Researchers have identified 3 kinds of evidences in BN belief update: hard evidence, virtual evidence, and soft evidence (Pan, Peng, and Ding 2006). Hard evidence specifies the particular state the variable is in, soft evidence specifies the probability distribution for the states the variable is in (Valtorta, Kim, and Vomlel 2002), and virtual evidence specifies the likelihood ratio for the uncertainty of the observations of the states the variable is in (Pearl 1990). For example, if one believes that the event represented by variable A occurs with probability p, and does not occur with probability $1-p$, then the likelihood ratio can be represented as $L(A) = p:(1-p)$, which does not necessarily need to be specific probabilities.

For belief update with virtual evidence, Pearl proposed the virtual evidence method (Pearl 1990). For virtual evidence on variable A with likelihood ratio $L(A)$, this method extends the given BN by creating a binary virtual node V as the child of A, with state v standing for the event that $A = a$ has occurred, and the CPT of V satisfying the following equation:

$$P(v \mid A = a) : P(v \mid A \neq a) \ = L(A) \ . \qquad (5)$$

It then treats v as a hard evidence by instantiating V to v. This will update the belief in the given BN, and the updated BN will satisfy the given virtual evidence. For belief update with soft evidence, Chan and Darwiche extended the virtual evidence method by first converting each soft evidence to a virtual evidence (Chan and Darwiche 2005). Let $P(X)$ be a distribution and $R(Y)$ be a soft evidence, where $Y \subseteq X$. We use $y_{(1)},...,y_{(l)}$ to represent all the possible instantiations of Y which form a mutually exclusive and exhaustive set of events. $R(Y)$ can then be converted to a virtual evidence with the following likelihood ratio:

$$L(Y) = \frac{R(y_{(1)})}{P(y_{(1)})} : \frac{R(y_{(2)})}{P(y_{(2)})} : ... : \frac{R(y_{(l)})}{P(y_{(l)})} \ . \qquad (6)$$

For multiple virtual evidences, the belief update for one virtual evidence will not affect the belief update for the other virtual evidences (Pan, Peng, and Ding 2006). However, this is not the case for multiple soft evidences. For BN belief update with multiple soft evidences, Peng et al. proposed BN-IPFP (Peng, Zhang and Pan 2010) which combines Pearl's virtual evidence method with IPFP. It is able to preserve marginal distributions specified in all soft evidences after converting them to virtual evidences. BN-IPFP converges iteratively, and the resulting BN can satisfy multiple soft evidences at the same time.

4 Identify Structural Inconsistencies

In this section we examine how to identify such structural inconsistencies for a given constraint and a BN. This is done by checking if every dependency relation in the constraint also holds in the BN.

First we discover dependency relations from the constraint and from the BN. We can use d-separation method (Pearl 1988) to discover dependency relations in the BN. This method tells whether every two variables are conditionally independent under another set of variables by looking at the connection type between these two variables in the BN.

We can use the independence test to find out dependency relations in the constraint. The zero-order (unconditional) independence test is to check whether $R(A,B) = R(A) \cdot R(B)$ holds for each pair of variables A and B in the variable set of R. Similarly, the j-order independence test for R is to check whether Equation 7 holds for each pair of variables A and B, where Z is a set that contains any j number of variables in the variable set of R other than A and B.

$$R(A,B \mid Z) = R(A \mid Z) \cdot R(B \mid Z) \qquad (7)$$

The following is the algorithm for identifying structural inconsistencies between the constraint and the BN.

Algorithm INCIDENT

Input: BN $G = (G_s, G_P)$ with ordering of nodes in G_s, and constraints $\mathbf{R} = \{R_1, R_2, ..., R_m\}$.

Output: Structural Consistent constraint set $\mathbf{R}^+$, structural inconsistent constraint set $\mathbf{R}^-$ and Dependency List DL.

Steps:

1. Create empty sets $\mathbf{R}^+$ and $\mathbf{R}^-$, and empty list DL;
2. Perform the following steps for each constraint R_i in $\mathbf{R}$:
   - 2.1 Create an empty list DL_i;
   - 2.2 For each pair of variables A and B in R_i, do from zero-order to $(|R_i| - 2)$ -order independence test on them, where $|R_i|$ is the number of variables in R_i. If the test fails, add $< R_i, A, B, Z >$ to DL_i;
   - 2.3 For each entry $< R_i, A, B, Z >$ in DL_i, use d-separation method to test whether A and B are independent given Z, i.e., $A \perp B \mid Z$. If the test fails, remove this entry from DL_i;
   - 2.4 If DL_i is empty, add R_i to $\mathbf{R}^+$. Otherwise, add R_i to $\mathbf{R}^-$, and merge DL_i into DL;
3. Return $\mathbf{R}^+$, $\mathbf{R}^-$, and DL.

5 Overcome Structural Inconsistencies

In this section we focus on modifying the structure of the existing BN to accommodate the identified structural inconsistencies in a way similar to adding virtual evidence node in Pearl's virtual evidence method. We can add one node in BN for each structural inconsistent constraint and make this node the child of all variables of that constraint. Then we set the CPT of the node with the likelihood ratio calculated from the constraint using (6), and set the state of the node to true. This guarantees the variables covered by the constraint to be dependent, which will overcome any structural inconsistencies for that constraint. This forms the core of our AddNode method, which can also be used with E-IPFP to integrate a set of constraints which may contain both structural consistent constraints and structural inconsistent constraints. The following is the algorithm which adds nodes for structural inconsistent constraints in $\mathbf{R}^-$, and updates BN with constraints in $\mathbf{R}^+$ using E-IPFP. The input constraints are applied iteratively during the process. When the algorithm converges, all constraints are satisfied (C1). Besides, the IPFP style computing makes the JPD of the final BN as close to the JPD of the original BN as possible (C2). We are working to formally prove these claims.

Algorithm AddNode+E-IPFP

Input: BN $G = (G_S, G_P)$ with ordering of nodes in G_S, and constraint set $\mathbf{R} = \{R_1, R_2, ..., R_m\}$.

Output: BN $G' = (G_S', G_P')$ that satisfies $\mathbf{R}$ with JPD as close to that of G as possible.

Steps:

1. Run INCIDENT to partition $\mathbf{R}$ into $\mathbf{R}^+$ and $\mathbf{R}^-$;
2. For each constraint R_i in $\mathbf{R}^-$, add a new node V_i to G_S with variables in R_i as its parents;
3. /* this step is the same as AddNode method */
   For each constraint R_i in $\mathbf{R}^-$,
   3.1 Calculate likelihood ratio for R_i using
   $$L(Y^i) = \frac{R_i}{P(Y^i)} = \frac{R_i(y_{(1)})}{P(y_{(1)})} : \frac{R_i(y_{(2)})}{P(y_{(2)})} : ... : \frac{R_i(y_{(l)})}{P(y_{(l)})},$$
   where Y^i is the variable set of R_i, and l is the number of distinct instantiations for Y^i;
   3.2 Construct CPT of V_i with likelihood ratio $L(Y^i)$;
   3.3 Set the state of V_i to true;
4. Apply one iteration of E-IPFP with $\mathbf{R}^+$ and the updated BN as input, i.e.,
   4.1 Do I-projection for the current probability distribution $P_{k-1}(X)$ on each constraint R_j in $\mathbf{R}^+$:
   $$P_k(X) = P_{k-1}(X) \cdot \frac{R_j(Y^j)}{P_{k-1}(Y^j)},$$
   where Y^j is the variable set of R_j, and X is the set of all variables in G;
   4.2 Form and apply the structure constraint:
   $$P_k(X) = \prod_{i=1}^{n} P_{k-1}(X_i \mid \pi_i),$$
   where $(X_i, \pi_i) \in G_s$;
5. Repeat step 3 and step 4 until the JPD of the updated BN does not change;
6. Return $G' = (G_S', G_P')$, where G_S' is the updated structure after adding nodes to G_S, and G_P' is the CPTs for the nodes in G_S'.

6 Experiments and Results

Our experiments are based on the BN shown in Figure 3, and constraint set $\mathbf{R} = \{R1, R2, R3\}$, where $R1(A, C, E) = (0.1368, 0.2232, 0.03, 0.03, 0.1672, 0.2728, 0.07, 0.07)$, $R2(B, C) = (0.6, 0.1, 0.2, 0.1)$ and $R3(C, D, E) = (0.228, 0.372, 0.076, 0.124, 0.04, 0.04, 0.06, 0.06)$.

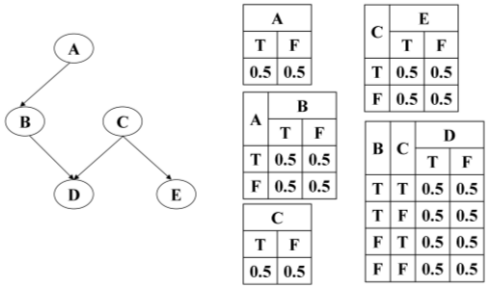

Figure 3: A 5 Node BN

First we run the INCIDENT algorithm to discover the structural inconsistencies. $\mathbf{R}^- = \{R1, R2\}$, $\mathbf{R}^+ = \{R3\}$ and DL= $\{< R1, A, C, \varnothing >, < R1, A, E, \varnothing >, < R1, A, C, \{E\} >, < R2, B, C, \varnothing >\}$ are returned as the output of INCIDENT. Then we run AddNode+E-IPFP to integrate constraints in $\mathbf{R}$. The BN structure has been updated as in Figure 4 after adding nodes for constraints in $\mathbf{R}^-$.

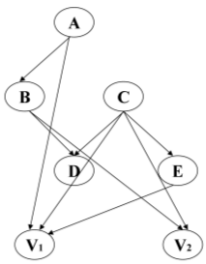

Figure 4: Modified BN Structure

Table 1 gives the performance comparison for different methods. From the table we can see that E-IPFP-SMOOTH does not add any extra nodes to the existing BN. But its I-aggregate is not 0. AddNode+E-IPFP adds extra nodes to the existing BN. But it has 0 I-aggregate. Table 1 includes experiment results for two more efficient variations of AddNode+E-IPFP. AddNode+Merge first merges the constraints in $\mathbf{R}^-$ into a single constraint using IPFP algorithm, and then adds a single node to the existing BN for the merged constraint. AddNode+Fatorization first decomposes each constraint in $\mathbf{R}$ into sub-constraints according to the independencies among its variables, and then merges the sub-constraints that are structural inconsistent with the existing BN, and at the end adds a single node to the existing BN for the merged constraint.

Table 1: Experiment Result with 5 Node BN

Method	Added Nodes	Iterations	Run Time	I-aggregate	I-divergence
E-IPFP-SMOOTH	0	149	195s	0.062	0.5
AddNode+E-IPFP	2	3	3.22s	0	0.65
AddNode+Merge	1	3	2.32s	0	0.65
AddNode+Factorization	1	3	2.27s	0	0.65

To further compare the performance of the different versions of our methods and get a sense of their scalability, we have conducted experiments with an artificially composed BN of 10 discrete variables. The input constraint set consists of 3 consistent constraints and 3 structural inconsistent constraints. The results are given in Table 2 below. AddNode+Merge and AddNode+Fatorization have better performance than AddNode+E-IPFP on large BN because these two methods add fewer nodes to the existing BN.

Table 2: Experiment Result with 10 Node BN

Method	Added Nodes	Iterations	Run Time	I-aggregate	I-divergence
AddNode+E-IPFP	3	1	102s	0	2.28
AddNode+Merge	1	1	15.8s	0	2.28
AddNode+Factorization	1	1	14.2s	0	2.28

7 Conclusions

In this paper we propose several methods to integrate structural inconsistent constraints into a BN. The methods first identify the structural inconsistencies between the constraint and the existing BN, and then overcome the identified structural inconsistencies by adding nodes to the existing BN in a way similar to the virtual evidence method. Experiments show that the proposed methods have an advantage in incorporating new structural information. We look forward to applying the framework and related methods to update a BN built for a demand-supply network with uncertain demand-reply relations. The information used to update this BN may require adding new demand-supply relations to the existing BN. This is important in supply chain management to better satisfy the needs of the customers. Our future work also includes improving the performance of our methods so they can be easily applied to industry problems modeled with large BNs.

References

[1] H. Chan and A. Darwiche. 2005. On the Revision of Probabilistic Beliefs using Uncertain Evidence. *Artificial Intelligence*, 67–90.

[2] I. Csiszar. 1975. I-divergence Geometry of Probability Distributions and Minimization Problems. *The Annuals of Probability*, 3(1): 146–158.

[3] D. Geiger, T Verma and J. Pearl. 1990. Identifying Independence in Bayesian Networks. *Networks*, 20: 507–534.

[4] R.Kruithof. 1937. Telefoonverkeersrekening. *De Ingenieur*, 52: 15–25.

[5] S. Kullback and R.A. Leibler. 1951. On Information and Sufficiency. *Ann. Math. Statis*, 22: 79–86.

[6] R. Pan, Y. Peng and Z. Ding. 2006. Belief Update in Bayesian Networks Using Uncertain Evidence. In *Proceedings of the IEEE International Conference on Tools with Artificial Intelligence*, 13–15.

[7] J. Pearl. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* San Mateo: Morgan Kaufman.

[8] J. Pearl. 1990. Jeffery's Rule, Passage of Experience, and Neo-Bayesianism. *Knowledge Representation and Defeasible Reasoning.* H.E. Kyburg, Jr., R.P. Loui, and G.N. Carlson, Eds. Boston: Kluwer Academic Publishers.

[9] Y. Peng and Z. Ding. 2005. Modifying Bayesian Networks by Probability Constraints. In *Proc. 21st Conference on Uncertainty in Artificial Intelligence.* Edinburgh.

[10] Y. Peng, Z. Ding, S. Zhang and R. Pan. 2012. Bayesian Network Revision with Probabilistic Constraints. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 317–337.

[11] Y. Peng and S. Zhang. 2010. Integrating Probability Constraints into Bayesian Nets. In *Proceedings of 9th European Conference on Artificial Intelligence (ECAI2010)*, Lisbon, Portugal.

[12] Y. Peng, S. Zhang and R. Pan. 2010. Bayesian Network Reasoning with Uncertainty Evidences. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 18(5): 539–564.

[13] M. Valtorta, Y. Kim and J. Vomlel. 2002. Soft Evidential Update for Probabilistic Multiagent Systems. *International Journal of Approximate Reasoning*, 29(1): 71–106.

[14] J. Vomlel. 1999. Methods of Probabilistic Knowledge Integration. Ph.D. diss., Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University.

[15] J. Vomlel. 2003. Integrating Inconsistent Data in a Probabilistic Model. *Journal of Applied Non-Classical Logics*, 14(3): 1–20.

[16] S. Zhang and Y. Peng. 2008. An Efficient Method for Probabilistic Knowledge Integration. In *Proceedings of The 20th IEEE International Conference on Tools with Artificial Intelligence (ICTAI-2008)*, Nov. 3–5. Dayton, Ohio.