12-2005

Integration of Probabilistic Graphic Models for Decision Support

Jiang C.

Poh K.

Tze-Yun LEONG
Singapore Management University, leongty@smu.edu.sg

Citation

Jiang C., Poh K., and Tze-Yun LEONG. Integration of Probabilistic Graphic Models for Decision Support. (2005). *Metacognition in computation: Papers from the 2005 AAAI Symposium*. 40-47. Research Collection School Of Information Systems.
Available at: https://ink.library.smu.edu.sg/sis_research/3022

Integration of Probabilistic Graphic Models for Decision Support

Chang-an JIANG, Kim-leng POH, Tze-yun LEONG
National University of Singapore
Medical Computing Lab, School of Computing, NUS, 117543, Singapore
Department of Industrial and Systems Engineering, NUS,119260, Singapore
{ChanganJiang, isepohkl}@nus.edu.sg leongty@comp.nus.edu.sg

Abstract

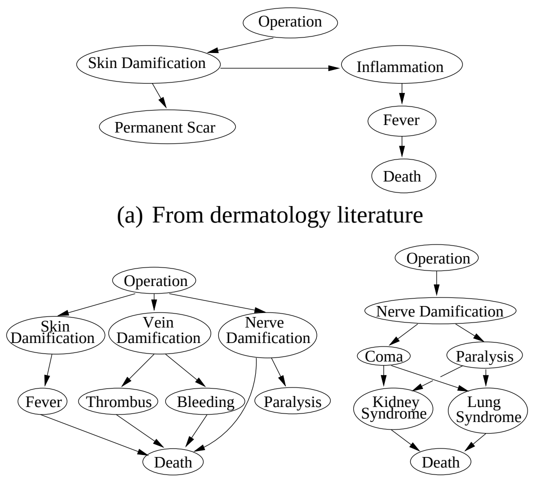
It is a frequently encountered problem that new knowledge arrived when making decisions in a dynamic world. Usually, domain experts cannot afford enough time and knowledge to effectively assess and combine both qualitative and quantitative information in these models. Existing approaches can solve only one of two tasks instead of both. We propose a four-step algorithm to integrate multiple probabilistic graphic models, which can effectively update existing models with newly acquired models. In this algorithm, the qualitative part of model integration is performed first, followed by the quantitative combination. We illustrate our method with an example of combining three models. We also identify the factors that may influence the complexity of the integrated model. Accordingly, we identify three factors that may influence the complexity of the integrated model. Accordingly, we present three heuristic methods of target variable ordering generation. Such methods show their feasibility through our experiments and are good in different situations. Finally, we provide some comments based on our experiments results.

Introduction

Bayesian network, a major type of probabilistic graphic models, is a powerful knowledge representation tool for abstracting uncertain information and decision problems. Over the last decades, research over graphical representation of knowledge receives many applications of modeling uncertainty.

In a rapidly changing world, different new fragments of knowledge or models may arrive when there is already an existing model. The problem of models integration is challenging. The different models to be integrated can differ in structure, or in parameters, even if they are obtained from the same data or experts from the same domain. This is due to the following reasons: (1) The sources of different models can be different (Druzdzel & van der Gaag 2000). (2) Models may be constructed with different graphic modeling techniques (Heckerman, Geiger, & Chickering 1994). They can be learned from data or elicited from domain experts.

A combined model is usually requested for the final decision or global view of a certain problem. Unfortunately, it could be a daunting task to get a united view over more than one model available towards the same issue for domain experts, as they cannot afford enough time and knowledge to both qualitative and quantitative combination of these models manually.



(a) From dermatology literature

(b) From surgery domain expert (c) From surgery domain expert

Figure 1: An example of model combination in medical domain

In medicine, for some complex medical decision problems, usually more than one expert is invited to provide their opinions, based on existing data or literature. These expert opinions, data or literature represent different knowledge sources. These knowledge sources may provide knowledge for the same issues. It is also quite often that different contributors are likely to have different views based on their expertise; therefore, different sets of factors (i.e., variables) will be considered.

Consider the following example: we assume that a surgeon *Jack* plans to do a head operation on his patient *Rose*. However, *Jack* is not confident of his knowledge on nerve damnification and skin damnification. In order to make a sound decision, *Jack* needs to acquire additional knowledge related to possible nerve damnification and skin damnification in a head operation. Therefore, he seeks help from dermatology literature and neurology data set.

This example case on a forthcoming head operation is shown in Figure 1. Three Bayesian networks are modeled from dermatology literature, a surgeon's domain expertise (i.e., *Jack*) and neurology data set respectively. The variables *operation* and *death* exist in all of the three networks. The first network and the second network have another two common variables—*skin damnification* and *fever*. The second network and the third network contain another two common variables—*nerve damnification* and *paralysis*. Although there are some common variables between any two networks, the structures are different. For example, there is a direct arc from skin damnification to fever in the second network, while there is no direct arc in the first network. In the second network, there is no link from variable *paralysis* to variable *death*, while there is a route from *paralysis* to *death* through *lung syndrome*. This example is a simplified version of real medical problems. In fact, real medical problems usually involve a large number of variables, complex relationships among the variables, and numerous parameters.

Some research (Clemen & Winkler 1999; Maynard-Reid & Chajewska 2001) has been done on combining probability distributions. Some merely address topology combination in BNs (Matzkevich & Abramson 1992; Joseph, Parmigiani, & Hasselblad 1998), which only two models can be combined at one time. Besides of shortcoming of unscalability, the resulting model can also influenced by the order of combination, if there are more than two models to be combined.

In this paper, we present an approach to solve both of qualitative and quantitative combination of an arbitrary number of probabilistic graphical models at a time. Our approach also provides a natural way and simple base for CPT combination.

Furthermore, we present three heuristic methods for automatic generation of target variable ordering for the resulting model.

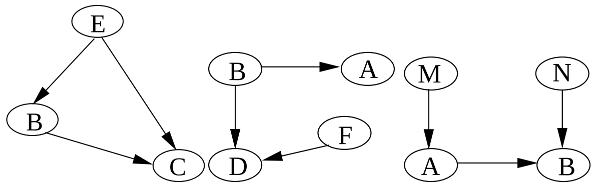
We make two intuitively reasonable assumptions for model combination: (1) Variables with same name model the same real world entity. (2) Variables with same name have to be over the same domains (in the discrete case, the same set of possible states).

Problem Formulation and Challenges

We assume a finite number of Bayesian networks $B_1, .., B_m$. $B_i = (V_i, \overrightarrow{E_i})$ where $i = 1, 2, ...m$, and $\overrightarrow{E} = (a, b)$ denote directed edges between every pair of nodes a and b within one probabilistic graphic model. The direction of edge is from a to b, which we denote $< a, b >$. These m Bayesian networks can satisfy $\phi \subseteq \bigcap_{i=1}^{m} V_i$ and $\phi \subseteq \bigcap_{i=1}^{m} \overrightarrow{E_i}$.

These available probabilistic graphic models to be combined are termed as *candidate Bayesian networks*. To combine the Bayesian networks, we aim at getting a single BN model. In general $E_{result} = \bigcup_{i=1}^{m} \overrightarrow{E_i}$ is true only in some special cases. An example of three different Bayesian networks is shown in Figure 2.

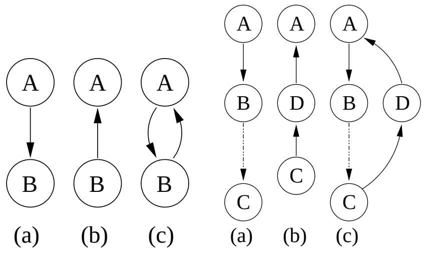
Each Bayesian network consists of two parts. The qualitative part that represents the structure of the network and dependency among variables; and the quantitative part that numerically represents the joint probability distribution over these variables.



(a) Candidate BN 1 (b) Candidate BN 2 (c) Candidate BN 3

Figure 2: Example of three candidate Bayesian networks

There are four major challenges in this task. The **first** challenge lies in the qualitative combination: how to avoid cycles after combination of multiple Bayesian networks. Direct combination of different models can result in cycle(s) in the resulting model. Figure 3 presents two examples of possible situations in *DAG* (Directed Acyclic Graph) that may incur a cycle in combination. We name these two situations as *direct conflict* cases and *indirect conflict* cases. The problem of avoiding a possible cycle can be solved using the arc reversal operation (Shachter 1984), which was applied in topology combination (Matzkevich & Abramson 1992) .



(a) Direct conflict (b) Indirect conflict

Figure 3: Conflict in DAG combination

The **second** challenge lies in the change of conditional independence relationships after combination. Adding an arc, it will break some independent relationships among variables. In other words, we may focus on minimize additional dependence relationships between nodes in the resulting Bayesian network. Set operation over conditional independence statements was used (Sagrado & Moral 2003) to solve the graph combination problem. Unfortunately, it is not easy to perform parameter combination based on this method.

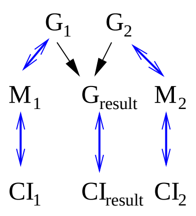
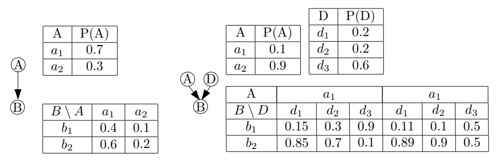


Figure 4: Probabilistic graphic models combination

Figure 4 shows an example case of combination of two probabilistic graphic models. M_1 and M_2 are the two candidate models to be combined. G_1 and G_2 are graphs that correspond to $M_1$and M_2 respectively. As M_1 implies G_1, and G_1 encodes the conditional independence CI_1 in M_1, M_1 is not only a valid probabilistic graphic model, but also a perfect map of the underlying dependency. Therefore, the problem that we are facing is to get the resulting G_{result} where the underlying CI_{result} breaks the least conditional independency from CI_1 and CI_2.

The **third** challenge concerns the quantitative computation. Different candidate Bayesian networks may have different structure, which means that the internal CPT (Conditional Probability Tables)s may be very different; not only in numbers, but also in the size of CPT. For example, the value of $P(A)$ in *Model* 1 is different from the value of $P(A)$ in *Model* 2. The CPT over node B in *Model* 1 is not only different from the CPT over node B of *Model* 2 in numbers, but also in CPT size, as shown in Figure 4. There is not a good way of combining CPTs in models yet.



A	P(A)
a_1	0.7
a_2	0.3

$B \setminus A$	a_1	a_2
b_1	0.4	0.1
b_2	0.6	0.2

(a) Model 1

A	P(A)
a_1	0.1
a_2	0.9

D	P(D)
d_1	0.2
d_2	0.2
d_3	0.6

A	a_1			a_1		
$B \setminus D$	d_1	d_2	d_3	d_1	d_2	d_3
b_1	0.15	0.3	0.9	0.11	0.1	0.5
b_2	0.85	0.7	0.1	0.89	0.9	0.5

(b) Model 2

Figure 5: CPT disagreement in two models

The **fourth** challenge in model integration is how to integrate more than two models at the same time. In some large Bayesian network model learning problems, knowledge engineering sometimes learn some small part of Bayesian networks and then combine them into a global Bayesian network. We can imagine a possible case that there are many small Bayesian networks to be combined. If all these models can be combined at a time, some manual work or time can be saved. Unfortunately, existing methods can only combine two models at a time. This is also a problem that we attempt to solve.

Bayesian Networks Combination

The proposed algorithm for integration of multiple Bayesian networks consists of four steps, as follows.

1. Reorganize original BN
2. Adjust variable ordering and edge direction
3. Save amended models as Intermediate Bayesian networks
4. Combine CPT

Step 1: Re-organize Bayesian networks

According to the chain rule factorization property of BN, a JPD (Joint Probability Distribution) can be factorized in more than one way. Each factorization indicates a different ordering of variables. When we change the ordering, we actually are changing the factorization of JPD. Therefore, we can always re-organize BN according to different partial ordering of variables, while maintaining the same JPD.

Step2: Adjust variable ordering to maintain DAG

Defn 1. Order Value. *Given a DAG $D = (V, E)$, the $Odervalue(v)$ of a node $v \in V$ in DAG D is defined as the longest path from a rooted node to node v.*

Defn 2. Variable Ordering. *λ is the sequence of $Ordervalue(v)$ for all node $v \in V$ in DAG D.*

Defn 3. Target Variable Ordering. *λ_{result} is the final variable ordering of the combined Bayesian network.*

Note that λ_{result} will not necessarily be the same as that of any candidate models, although it is possible.

Lemma Cycle can be avoided when every arc in the candidate Bayesian networks are from nodes with lower order value to nodes with higher order value.

Proof:

Given k Bayesian networks $B_{1,...,}B_k$, $\lambda_{1,...,}\lambda_k$ are variable ordering in $B_{1,...,}B_k$.

We assume there exists one arc ¡$s_{1,}s_2$ $>$ in B_i, $B_i \in \{B_1, ..., B_k\}$, in which

$$ordervalue(s_1) \geq ordervalue(s_2) \tag{1}$$

is satisfied.

According to the definition of ordervalue in Bayesian networks, ¡$s_{1,}s_2$ $>$ denotes an arc starts from s_1, and ends at s_2. Therefore we can get

$$ordervalue(s_2) = \max\{ordervalue(Pa(S_1))\}+1 \tag{2}$$

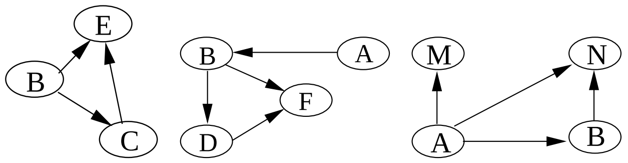
$$ordervalue(s_2) > ordervalue(s_1) \tag{3}$$

which is conflict with 1. Therefore, every arc in Bayesian networks is from nodes with lower *ordervalue* to nodes with higher *ordervalue*. □

Arc Reversal to Adjust Variable Ordering. Arc reversal is needed here so that variable ordering in these networks are consistent with target variable ordering with preservation of JPD, but with some structural changes (Howard & E 1981; Olmsted 1983). The three candidate Bayesian networks after arc reversal can been seen in Figure 6.

Node	CBN_1	CBN_2	CBN_3	*User Specified Target Ordering*
Ordervalue(A)		1	1	1
Ordervalue(B)	1	0	2	2
Ordervalue(C)	2			3
Ordervalue(D)		1		4
Ordervalue(E)	0			5
Ordervalue(F)		0		6
Ordervalue(M)			0	7
Ordervalue(N)			0	8

Table 1: Order values in candidate BNs and target ordering



(a) Reconstruct Candidate BN1 (b) Reconstruct Candidate BN2 (c) Reconstruct Candidate BN3

Figure 6: Arc reversal results over three candidate BNs

Step 3: Intermediate Bayesian Networks

We present a new concept of *Intermediate Bayesian Networks* in the procedure of combination.

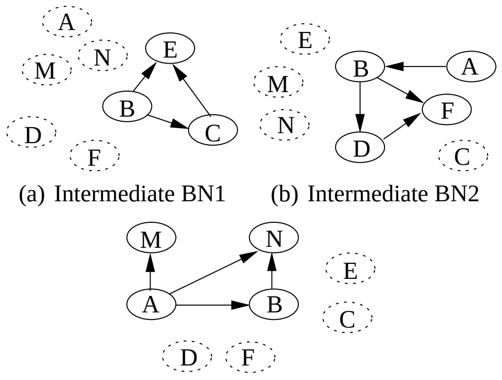
Defn 4. Intermediate Bayesian Networks. *Given k candidate Bayesian networks to be combined, $B_1, \cdots, B_k$. We make an identical copy from both qualitative part and quantitative part of these candidate Bayesian networks and save them as Intermediate Bayesian networks.*

The advantages of intermediate BN include 1) The structure and parameters of original input BN models can remain unchanged. 2) With intermediate BN, we may turn the two tasks of qualitative combination and quantitative combination into the single task of CPT combination among a set of intermediate BNs with isomorphic topology.

Virtual Nodes and Virtual Arcs. In order to get homogeneous structure for every intermediate BN, we present the concept of virtual nodes and virtual arcs. Figure 7 shows the example of virtual nodes.

Step 4: CPT Combination

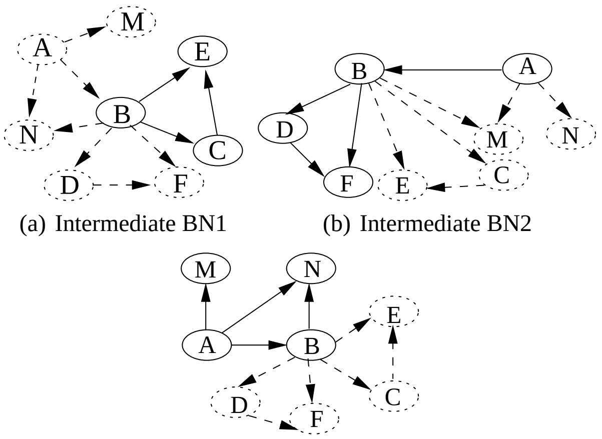
With the help of virtual nodes and virtual arcs, for candidate BNs to be combined, we can get intermediate BNs with same topology. An example is shown in Figure 7. In the CPT combination part, we suggest there are two solutions: (1) Using weighted combination, we can get standard CPT filled with point probability distributions in result BN. (2) Using interval combination, we can get Interval Bayesian Networks (Ha & Haddawy 1996).



(a) Intermediate BN1 (b) Intermediate BN2

(c) Intermediate BN3

Figure 7: Example of virtual nodes



(a) Intermediate BN1 (b) Intermediate BN2

(c) Intermediate BN3

Figure 8: Example of virtual arcs

Three Heuristic Methods of Target VariableOrdering Generation

We notice that there are some factors, which will influence the generation of target variable ordering: (1) Original order values of each variable; (2) Number of parents of each variable; (3) Size of each candidate Bayesian Networks. According to the above factors, we present three heuristic methods for target variable ordering generation.

Method 1: Target Ordering based on Original Order Valules

The following algorithm describes a method of automated target variable ordering generation–Order value based Target Variable Ordering Generation method. We also provide the proof of the correctness of this method.

Algorithm 1 Order value based Target Variable Ordering Generation in BNs Combination

```
Require: $B_1, \ldots, B_k$, $k \geq 2$ and $B_i = (V, E)$, $i = 1 \ldots k$
 1: for $i = 1$ to $k$ do
 2:     $\forall v$ in $B_i$, Store $OrderValue(v)$;
 3: end for
 4: $V_{result} = \cup_{i=1}^{k} V_i$ {push all nodes from candidate BNs into $V_{result}$};
 5: $NodesNum$ = —$V_{result}$—;
 6: initiate an array
 7: $AllNodes[NodeNum]$= $[NodeID, OrderValue, NewOrderValue]$;
 8: for $i = 1$ to $NodeNum$ do
 9:     for $j = 1$ to $k$ do
10:         v=AllNode(NumNode);
11:         $Sum_v$= $\sum_{j=1}^{k} OrderValue(v_k)$ ;{sum the node's ordervalue in all candidate BNs}
            NodeAppear[NodeNum]++;{count how many models that this node exist}
12:     end for
13: end for
14: for $i = 1$ to $NodeNum$ do
15:     $AverageOrder[NodeNum]$ = $\frac{Sum_v}{NodeAppear}$;
16: end for
17: sort $AverageOrder[NodeNum]$ according to $averagedordervalue$ and $nodeID$ {for two nodes with same average ordervalue, sort according to nodeID }
18: Assign $NewOrdervalue$ to each Node after sorting
```

Node	CBN_1	CBN_2	CBN_3	$Average$	Target Order
Ordervalue(A)		1	1	1	2*
Ordervalue(B)	1	0	2	$\frac{2}{3}$	1
Ordervalue(C)	2			2	3
Ordervalue(D)		1		1	2*
Ordervalue(E)	0			0	0
Ordervalue(F)		0			
Ordervalue(M)			0	0	0
Ordervalue(N)			0	0	0

Table 2: Example of target ordering based on original order value

Method 2: Target Ordering based on Number of Parents and Network Size

The key idea in the following algorithm is to generate target variable ordering according to linear computation over 1) the number of parent nodes of each variable in candidate Bayesian networks and 2) the number of nodes for each candidate Bayesian network.

Algorithm 2 Target Variable Ordering Generation based on Number of Parents and Network Size

```
Require: $B_1, \ldots, B_k$, $k \geq 2$ and $B_i = (V, E)$, $i = 1 \ldots k$
 1: for $i = 1$ to $k$ do
 2:     $\forall v$ in $B_i$,
        Store $NumParents_i(v)$;
        Store $NetSize_i(v)$; {$NetSize_i(v)$ denotes the size of network that node $v$ is in};
        $tempValue_i[v]$= $NumParents_i(NodeNum)*NetSize_i(NodeNum)$;
 3: end for
 4: $V_{result} = \cup_{i=1}^{k} V_i$ {push all nodes from candidate BNs into $V_{result}$};
 5: $NodesNum$ = —$V_{result}$—;
 6: initiate an array
 7: $AllNodes[NodeNum]$= $[NodeID, OrderValue, NewOrderValue]$;
 8: for $i = 1$ to $NodeNum$ do
 9:     $TargetOrder[NodeNum]$= $\sum tempValue_i[NodeNum]$;
10: end for
11: sort $NodeNum$ according to $TargetOrder[NodeNum]$and $nodeID$ {for two nodes with same TargetOrder[NodeNum], sort according to nodeID }
12: Assign $NewOrdervalue$ to each Node according to the position of each node after sorting
```

Node	$CBN1$	$CBN2$	$CBN3$	$WeighteSum$	TargetOrder
Num_Parent(A)		1	1	8	2*
Num_Parent(B)	1	0	2	11	3
Num_Parent(C)	2			6	1
Num_Parent(D)		2		8	2*
Num_Parent(E)	0			0	0
Num_Parent(F)		0		0	0
Num_Parent(M)			0	0	0
Num_Parent(N)			0	0	0
Num of Nodes	3	4	4		

Table 3: Example of target ordering based on num of parents and network size

Method 3: Target Ordering based on Edge Matrix

In this method, we consider the relative difference in order value between every pair of nodes in the candidate Bayesian networks. We construct the edge matrix, by storing the difference in order value for each variable in candidate Bayesian networks. Thus there are k edge matrix if there are k candidate Bayesian networks. To get the target variable ordering, we need to get a final edge matrix after computation over these k candidate Bayesian networks. According to the final edge matrix, we may get the relative difference in order value of between each pair of nodes in the resulting Bayesian networks.

Ending Node\Starting Node	*E*	*B*	*C*
E	0	-1	-2
B	1	0	-1
C	2	1	0

(a) Candidate BN_1

Ending Node\Starting Node	*A*	*B*	*D*	*F*
A	0	1	0	1
B	-1	0	-1	0
D	0	1	0	1
F	-1	0	-1	0

(b) Candidate BN_2

Ending Node\Starting Node	*A*	*B*	*M*	*N*
A	0	-1	1	1
B	1	0	2	1
M	-1	-2	0	1
N	-1	-1	-1	0

(c) Candidate BN_3

Table 4: Example of edge matrix of candidate BNs

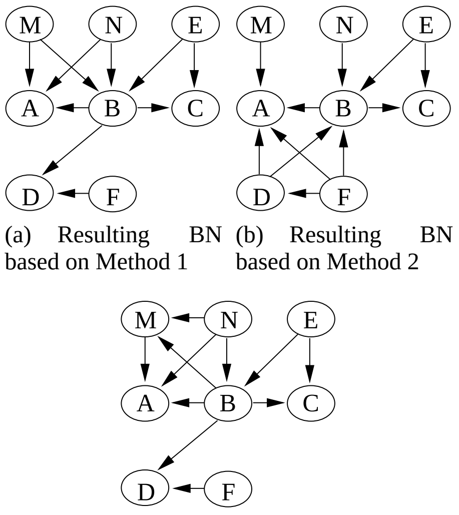
Ending Node\Starting Node	*A*	*B*	*C*	*D*	*E*	*F*	*M*	*N*
A	0	0^*	0	0	0	1	1	1
B		0	-1	-1	1	0	2	1
C			0	0	0	1	0	0
D				0	0	0	0	0
E					0	0	0	0
F						0	0	0
M							0	1
N								0

Table 5: Resulting edge matrix according to edge matrix based target ordering algorithm

System Implementation and Evaluation

We design and develop software architecture of PGMC System (Probabilistic Graphic Model Combination system). As shown in Figure 10, the PGMC system allows more than one probabilistic graphic model as inputs, and the output of the system is a resulting model. This system is developed in C++ under Windows environment, which is based on SMILE API and GeNIe (Decision System Lab, 2004).

In our experiments, we applied our system to over 30 Heart disease models (Tham, Heng, & Chin 2003), whose sizes are from 8 nodes to 13 nodes. There are 41 variables in total. These variables indicate either genotype or phenotype attributes for a single human subject. Part of the experiment results are shown in Tables 3 through 5. The combination time is counted in seconds, including time of combination procedure only, excluding the time of target variable ordering generation.



(a) Resulting BN based on Method 1 (b) Resulting BN based on Method 2

(c) Resulting BN based on Method 3

Figure 9: Example of Resulting BNs after Combination using three Methods

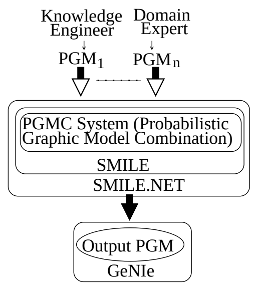


Figure 10: System overview

	Num_ArcReversed	Num_ArcAdded	Combination Time (Sec)
Method 1	6	5	1.953
Method 2	10	5	5.999
Method 3	6	3	1.813

Table 6: Comparison of 3 methods in three 6-node BN combination

	Num_ArcReversed	Num_ArcAdded	Combination Time (Sec)
Method 1	5	7	162.754
Method 2	8	9	350.143
Method 3	6	3	15.222

Table 7: Comparison of 3 methods in three 7-node BN combination

The experimental results indicate that different variable ordering has great impact on the resulting model. The network sizes of input models usually have influence on complexity of resulting model, but it is not a deterministic factor.

Discussion

One of our main contributions of our paper is extending existing research of combining probability distributions to the case of aggregating probabilistic graph models.

Our resulting model after combination is based on a certain target variable ordering. According to the target variable ordering, all arcs have to start from node with low order value, ends at node with high order value. In this way, no cycle will be generated in the procedure of combination and DAG structure can be maintained.

As previous research work only focus on either structure combination of Bayesian networks, or probability distribution. Solution to accomplish both tasks at one time is not available yet. Our approach can solve both structure combination and parameter combination for Bayesian networks.

In addition, the result of combination will not be influenced by the order of combination.

However, our models combination algorithm has its limitation, which results from the inherent property of arc reversal operation. Arc reversal often significantly increases the number of parents of the nodes that the two nodes in the arc are involved. Since CPT size increases exponentially with the number of parents, the resulting CPTs can become very large and require a prohibitive amount of computation to construct.

Conclusion

In this paper, we address the problem of integrating multiple probabilistic models. The main part of our research focuses on multiple Bayesian networks combination problem. We separate the task into two subtasks: qualitative combination and quantitative combination.

The qualitative combination of BNs is the first task. As BN can be reconstructed because JPD is factorizable with different partition of variables, a basic idea in our method is to get a target variable ordering for resulting BN so that the direction of arcs in the resulting BN are only allowed when it is from nodes with lower order value to nodes with higher order value. With target variable ordering, we utilize arc reversal operation to adjust order value of variables within one probabilistic model. In order to let the quantitative combination step can be clear and easy, we present the concept of intermediate BN, so that all modification steps, including arc reversal, filling of virtual nodes and virtual arcs, are performed over intermediate Bayesian networks and the structure and parameters of original Bayesian networks can be preserved. At last, we can reach consensus topology for each input BNs.

The target variable ordering can be specified by the user, for example, a domain expert. In case of absence of domain experts, the three heuristic methods of target variable ordering generation that we proposed, can be very helpful. However, the three methods are not guaranteed to yield optimal solution as it is a NP-hard problem (Matzkevich & Abramson 1993).

In the quantitative combination of Bayesian networks, we argue that the CPT in resulting Bayesian network after combination can be filled with either point probability distributions or interval probabilities, since exact probabilities are not always necessary.

The work in this paper that is designed for knowledge combination should be a general system that supports a wide spectrum of decision problems. Our approaches can be applied in various areas, such as stocks, business, air traffic control, medicine, military operation, etc. The research in this paper can also be applied in collaborative environments agents (e.g., robots or softbots) might develop their own understanding of the environment and want to combine their understanding with that of other agents.

Future work

It is possible to extend BN model combination to influence diagram combination, when we want to extend our objective from just knowledge representation to decision analysis. In an influence diagram, decision nodes and utility nodes are added, and those nodes inherited from Bayesian network are now named as chance nodes.

Acknowledgements

We would like to thank Prof. Peter Haddawy and Prof. Marek Druzdzel and all BiDE group members in National University of Singapore, for their valuable advice on this work.

References

Clemen, R. T., and Winkler, R. 1999. Combining probability distributions from experts in risk analysis. *Risk Analysis* 19:187–203.

Druzdzel, M. J., and van der Gaag, L. C. 2000. Building probabilistic networks: Where do the numbers come from. *IEEE Transactions on Knowledge and Data Engineering* 12(4):481–486.

Ha, V., and Haddawy, P. 1996. Theorectical foundations for abstraction-based probabilistic planning. In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, 291–298.

Heckerman, D.; Geiger, D.; and Chickering, D. M. 1994. Learning bayesian networks: The combination of knowledge and statistical data. In *KDD Workshop*, 85–96.

Howard, R. A., and E, M. J. 1981. *Influence Diagrams*, volume 2 of *Readings on the Principles and Applications of Decision Analysis*. Strategic Decisions Group, Menlo Park, CA. chapter The Principles and Applications of Decision Analysis, 719–762.

Joseph, L.; Parmigiani, G.; and Hasselblad, V. 1998. Combining expert judgment by hierarchical modeling: An application to physician staffing. *Management Science* 44:149–161.

Matzkevich, I., and Abramson, B. 1992. The topological fusion of bayes nets. In Dubois, D., and Wellman, M. P., eds., *Proceedings of the 8th Annual Conference on Uncertainty in Artificial Intelligence*, 152–158. Morgan Kaufmann.

Matzkevich, I., and Abramson, B. 1993. Some complexity considerations in the combination of belief networks. In *Proceeing of Uncertainty of Artificial Intelligence 1993*, 152–158.

Maynard-Reid, P., and Chajewska, U. 2001. Aggregating learned probabilistic beliefs. In *UAI*, 354–361.

Olmsted. 1983. *On representing and solving decision problems*. Ph.D. Dissertation, Department of Engineering-Economics Systems, Stanford University.

Sagrado, J. D., and Moral, S. 2003. Qualitative combination of bayesian networks. *International Journal of Intelligent Systems* 18(2):237 – 249.

Shachter, R. D. 1984. Evaluating influence diagrams. *Operations Research* 34(6):871–872.

Tham, C. K.; Heng, C. K.; and Chin, W. C. 2003. Predicting risk of coronary artery disease from dna microarray-based genotyping using neural networks and other statistical analysis tool. *Journal of Bioinformatics and Computational Biology* 1(3):521–531.