Practical Issues in Modeling Large Diagnostic Systems with Multiply Sectioned Bayesian Networks

Y. Xiang*, K.G. Olesen@ and F.V. Jensen@
*Department of Computer Science, University of Regina, Regina, Saskatchewan, Canada S4S 0A2
@Department of Computer Science, Aalborg University, DK-9220 Aalborg, Denmark

Abstract

As Bayesian networks become widely accepted as a normative formalism for diagnosis based on probabilistic knowledge, they are applied to increasingly larger problem domains. These large projects demand a systematic approach to handle the complexity in knowledge engineering. The needs include modularity in representation, distribution in computation, as well as coherence in inference. Multiply Sectioned Bayesian Networks (MSBNs) provide a distributed multiagent framework to address these needs.

According to the framework, a large system is partitioned into subsystems and represented as a set of related Bayesian subnets. To ensure exact inference, the partition of a large system into subsystems and the representation of subsystems must follow a set of technical constraints. How to satisfy these goals for a given system may not be obvious to a practitioner. In this paper, we address three practical modeling issues.

Keywords: diagnosis, multiagent systems, Bayesian networks

1 Introduction

Bayesian networks (BNs) (Pea88; Nea90; Jen96) provide a normative formalism for diagnosis based on probabilistic domain knowledge. In the past decade, researchers have studied how to model diagnostic problems using BNs (Hec90; DGH92; HBR95) and many algorithms have been proposed to perform inference in BNs (Pea88; Sha96; CGH97; Jen96). Most of these methods are based on a flat BN representation of the system to be diagnosed.

As BNs become widely accepted, they are applied to larger and more complex problem domains. Construction of BNs with several hundreds or even thousands of variables are being attempted, e.g., (ML96). Such large projects demand a systematic approach to handle the complexity in knowledge engineering. It is a parallel to the need for software engineering in computer science in general, and to the need for distributed artificial intelligence (BG88) and multiagent systems (WJ95) in the field of artificial intelligence. In particular, it is necessary to be able to decompose a large and complex domain into subdomains which can be individually represented and managed in a modular fashion. Distributed representation and inference computation should also be supported when the problem domain is naturally physically distributed, e.g, components in a large equipment. Each subdomain can then be treated not only as a unit of knowledge representation but also as a unit of computation, which we may abstract as an *agent*. Most importantly, the issue how to ensure exact inference while the representation and computation are modular and distributed must be resolved.

Multiply Sectioned Bayesian Networks (MSBNs) provide a framework to meet these needs. It was motivated in the development of a medical diagnostic system (XPE$^+$93) under the single agent paradigm, and was then extended to the multiagent paradigm (Xia96). An MSBN forms the core of a cooperative multiagent system for diagnosis of a large system. Each agent is equipped with private knowledge about a subsystem and acts autonomously and cooperatively with other agents.

To ensure exact inference, the partition of a large system into subsystems and the representation of subsystems must follow a set of technical constraints. To a practitioner, it is important to satisfy not only these constraints of correctness but also additional constraints arising from the practice. For example, the resultant representation must be computationally feasible with the given resource. In multiagent systems, verification of technical constraints must respect privacy of agents. How to achieve these goals for a given system may not be obvious. In this paper, we address three practical issues in these regards. For a more detailed illustration of application of MSBNs to equipment diagnosis, see (XG99). For the support of the MSBN framework to hierarchical modeling and object oriented modeling, see (Sri94; KP97).

We briefly review the basic theory of MSBN in Section 2. In Section 3, we address the issue how to model a system into a hypertree as required by the MSBN framework. In Section 4, we address the issue how to model the interface between subsystems to make the inference computation efficient. In Section 5, we present how to test the suitability of a domain decomposition without violating agent privacy.

2 Overview of MSBNs

An MSBN M is a collection of Bayesian subnets that together defines a BN. M represents probabilistic dependence of a *total universe* of variables partitioned

into multiple *subdomains*. Variables in each subdomain models a subsystem and is represented by a subnet.

Figure 1 (a) shows a digital circuit as an over-simplified example. Each component is modeled by a subnet as shown in (b).

To permit exact distributed inference, however, these subnets should satisfy certain conditions. The following terminology will be used to specify these conditions. A *graph* G is a pair (N, E), where N is a set of nodes and E is a set of links each of which is between two distinct nodes in N.

Definition 1 *Let $G_i = (N_i, E_i)$ $(i = 0, ..., n-1)$ be n graphs. The graph $G = (\cup_i N_i, \cup_i E_i)$ is the* `union` *of G_is, denoted by $G = \sqcup_i G_i$.*

One condition is that for each pair of subnets, the set of variables in one subnet should be independent of the set of variables in the other given the set of variables shared by both. It can be shown (Xia97) that this condition holds if and only if nodes shared by the two subnets form a *d-sepset*, as defined below:

Definition 2 *Let $D_i = (N_i, E_i)$ $(i = 0, 1)$ be two DAGs such that $D = D_0 \sqcup D_1$ is a DAG. The intersection $I = N_0 \cap N_1$ is a* `d-sepset` *between D_0 and D_1 if for every $x \in I$ with its parents π in D, either $\pi \subseteq N_0$ or $\pi \subseteq N_1$. Each $x \in I$ is called a* `d-sepnode`.

For example, the interface $\{f, g\}$between D_0 and D_1 in Figure 1 (b) is a d-sepset. In an MSBN, a non-d-sepnode occurs only once, but a d-sepnode has multiple occurrences, one in each subnet that shares it.

Just as the structure of a BN is a DAG, the structure of an MSBN is a multiply sectioned DAG (MSDAG) with a hypertree organization:

Definition 3 *A* `hypertree MSDAG` *$\mathcal{D} = \bigsqcup_i D_i$, where each D_i is a DAG, is a connected DAG constructible by the following procedure:*

Start with an empty graph (no node). Recursively add a DAG D_k (called a `hypernode`*) to the existing MSDAG $\bigsqcup_{i=0}^{k-1} D_i$ subject to the constraints:*
[d-sepset] For each D_j $(j < k)$, $I_{jk} = N_j \cap N_k$ is a d-sepset when the two DAGs are isolated.
[local covering] There exists D_i $(i < k)$ such that, for each D_j $(j < k; j \neq i)$, we have $I_{jk} \subseteq N_i$. For an arbitrarily chosen such D_i, I_{ik} is the `hyperlink` *between D_i and D_k which are said to be* `adjacent`.

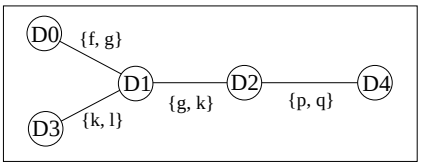

Figure 2: Hypertree organization of subnets.

The DAGs in Figure 1 (b) can be organized into the hypertree MSDAG in Figure 2, where each hypernode is labeled by a DAG and each hyperlink is labeled by a d-sepset. The semantics of the hypertree is that given a hyperlink (the value for the d-sepset), the two subtrees connected through the link are rendered irrelevant to (conditionally independent of) each other. Intuitively, if agents are organized into a hypertree MSDAG, then each agent can pass relevant information obtained locally to all other agents by communicating only with its hypertree neighbors. An MSBN is then defined as follows:

Definition 4 *An MSBN M is a triplet $M = (\mathcal{N}, \mathcal{D}, \mathcal{P})$. $\mathcal{N} = \bigcup_i N_i$ is the* `total universe` *where each N_i is a set of variables. $\mathcal{D} = \bigsqcup_i D_i$ (a hypertree MSDAG) is the* `structure` *where nodes of each DAG D_i are labeled by elements of N_i. Let x be a variable and $\pi(x)$ be all parents of x in $\mathcal{D}$. For each x, exactly one of its occurrences (in a D_i containing $\{x\} \cup \pi(x)$) is assigned $P(x|\pi(x))$, and each occurrence in other DAGs is assigned a constant table. $\mathcal{P} = \prod_i P_{D_i}$ is the* `jpd`*, where each P_{D_i} is the product of the probability tables associated with nodes in D_i. A triplet $S_i = (N_i, D_i, P_{D_i})$ is called a* `subnet` *of M.*

Definition 4 specifies the joint belief of all agents, which is a well defined probability distribution (the jpd). Furthermore, the jpd is consistent with the belief of each agent within its subdomain.

Inference in an MSBN is performed using a compiled representation (XPB93; Xia96). To inference effectively in each subnet, each D_i (hypernode) is converted into a tree structure. Conversion starts with *moralization*. A set of nodes is *complete* if they are pairwise connected. Moralization completes the parents of each node and then drops the direction of links. The resultant is called a *moral* graph. Figure 3 (a) shows a moral graph.

Next the moral graph is *triangulated* into a *chordal* graph. A *chord* is a link connecting two nonadjacent nodes. A graph is chordal if each cycle of length > 3 has a chord. Triangulation can be performed by node elimination. A node is *eliminated* if its adjacency is completed before it is removed with all incoming links. Links added during elimination are called *fill-ins*. Let the nodes of a graph $G = (N, E)$ be eliminated one by one with the set F of fill-ins. Then $(N, E \cup F)$ is chordal.

Triangulation in an MSBN can be performed by node elimination constrained by the hypertree organization (Xia98a). For the purpose of this paper, it suffices to say that the moral graph of each subnet must be eliminated relative to a d-sepset in order to preserve dependency in the compiled structure. That is, nodes in the d-sepset must be eliminated *after* elimination of all other nodes. Figure 3 (b) shows the chordal graph from M_0 obtained by eliminating all other nodes before the d-sepset $\{f, g\}$.

A maximal set of nodes that is complete is called a *clique*. For each subnet, a tree is then constructed where each node (called a *cluster*) is labeled by a clique of the chordal graph for the subnet. The clusters are so connected that the intersection (called *separator*) of

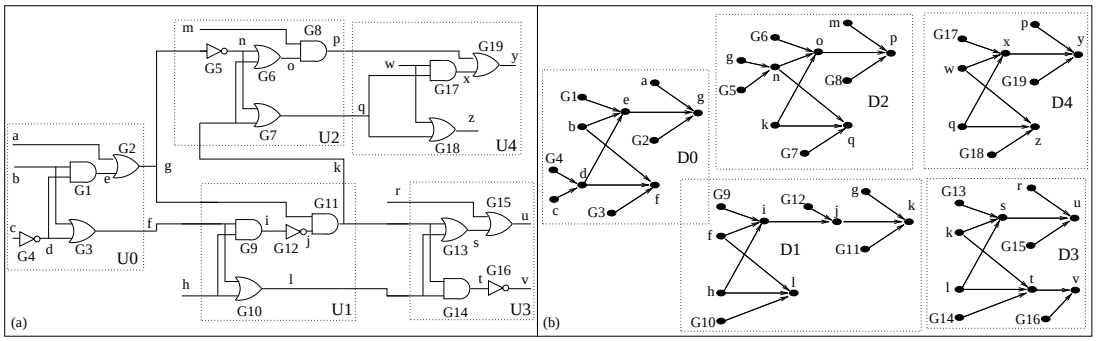

Figure 1: (a) A digital circuit partitioned into five components. Each component is treated as a subnet.

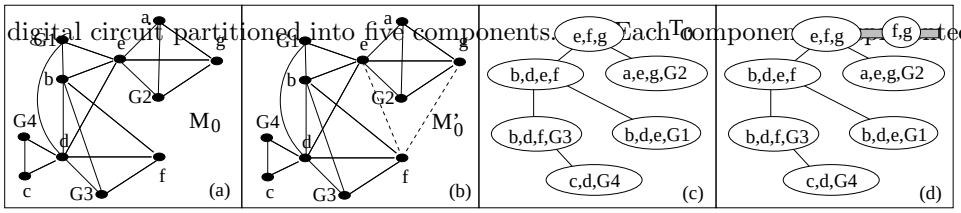

Figure 3: (a) The moral graph of D_0. (b) The chordal graph of M_0. (c) The junction tree from M_0'. (d) T_0 and its linkage tree with T_1.

any two clusters is contained in each cluster on their path. Such a cluster tree is called a junction tree (JT). An example is shown in Figure 3 (c). Now each DAG D_i is converted into a JT T_i over N_i.

To propagate information between subnets effectively, each d-sepset (hyperlink) is converted into a set of linkages defined below. This conversion essentially decomposes a potentially large d-sepset into a set of smaller subsets such that the probability table over the d-sepset can be represented compactly and communicated between subnets efficiently.

Definition 5 *Let I be the d-sepset between adjacent JTs T_a and T_b. A* `linkage tree` *L of T_a with respect to T_b is constructed as follows:*

Repeat the following until no variable can be removed:
(1) Remove a variable $x \notin I$ if x is contained in a single cluster C.
(2) If C becomes a subset of an adjacent cluster D after (1), union C into D.

Each cluster l in L is a `linkage`*. Define a cluster in T_a that contains l as its* `linkage host` *and break ties arbitrarily.*

Figure 3 (d) shows the linkage tree between T_0 and T_1 which is trivial in this case. More general cases are presented in Figures 8 and 9. The set of subnet JTs linked by linkage trees is collectively called a *linked junction forest* (LJF).

Diagnostic inference is performed using the LJF. Each agent/subnet acts autonomously. When evidence is available at the subsystem, it is entered to the JT and queries can be answered regarding the state of the subsystem. Such local inference is performed by different agents asynchronously in parallel. From time to time, agents can communicate (for details on communication, see (Xia96)). After each communication, answers to queries at each agent will be consistent with all evidence collected throughout the system.

3 How to model a system as a hypertree?

To ensure exact distributed inference, subsystems (subnets) in an MSBN must be organized into a hypertree. How do we satisfy this constraint when the natural system structure does not have an obvious hypertree structure?

We classify the potential faults in the system as *local* and *global*. A local fault has a direct impact on a small number of variables. For example, if a logic gate is faulty, it can directly affect only the correctness of its output. If a pipe is jammed, it can directly affect only the pressure at its two ends. On the other hand, a global fault has a direct impact on a large number of variables. For example, if a digital equipment is overheated, all ICs will have higher probability of failure. When the solar wind occurs, all radio transmissions will be jammed.

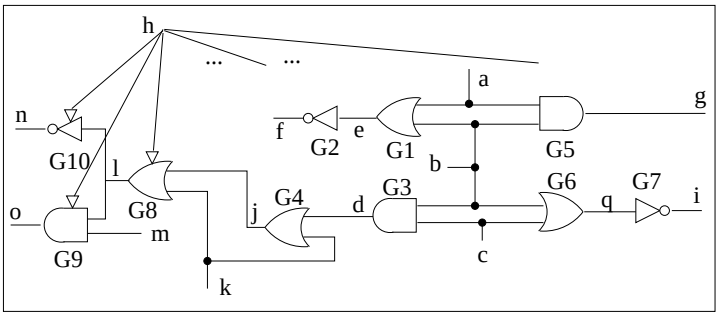

Figure 4: Part of a digital equipment. Outputs $\{i, g, n, o\}$ connect to other gates but no other paths exist between $\{i, g\}$ and $\{n, o\}$ except those that are shown. Variable h (overheating) affects each gate (details are partially shown).

Figure 4 shows part of an (over-simplified) digital

equipment. All variables, the states and inputs/outputs of gates, are local except the variable h of overheating is global.

To build a distributed diagnostic system, we partition devices of a large system into subsystems which we shall call *components*. For the partial equipment in Figure 4, if we ignore the global fault h, then the component partition in Figure 5 will have the structure of a hyperchain: $U_2 - U_0 - U_1$. Furthermore, the hypernodes in the chain has *loose* coupling: the interface between U_2 and U_0 is $\{j, k\}$, and the interface between U_0 and U_1 is $\{a, b, c\}$. Loose coupling translates to efficiency in distributed inference.

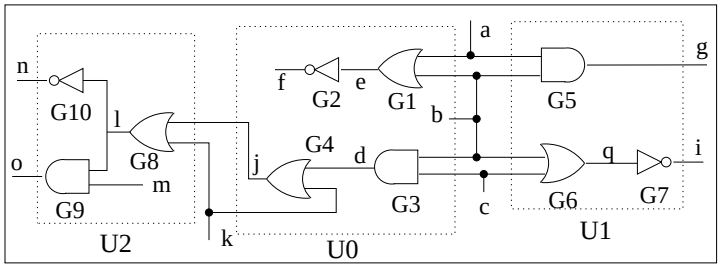

Figure 5: Three components U_i $(i = 0, 1, 2)$ (dotted box) in a digital equipment. Overheating is omitted.

However, when the global fault h is taken into account as shown in Figure 6, the nice chain structure appears to be destroyed.

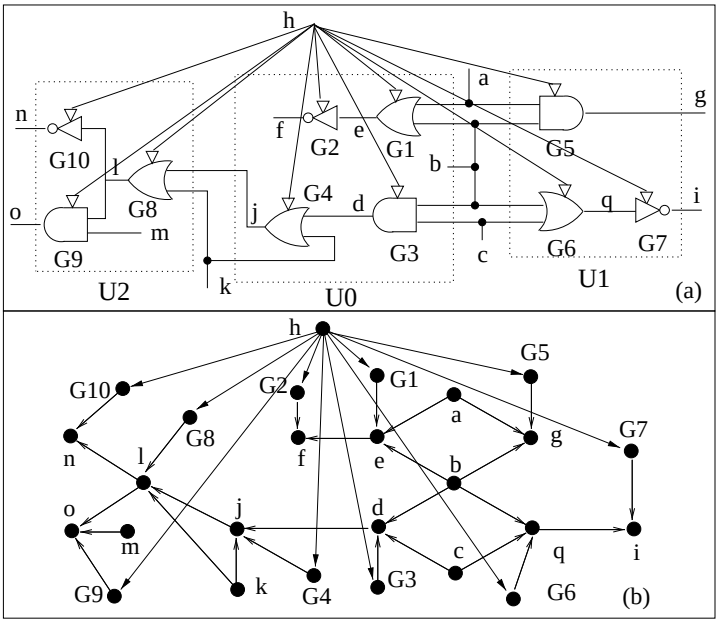

Figure 6: (a) Partial digital equipment with overheating restored. (b) The causal dependency structure as a DAG.

We indicate that the apparent *dense* coupling is only superficial. The essence of loose coupling is that a small subset of interface variables can render the two components independent. Since $\{j, k, h\}$ renders U_2 and U_0 independent and $\{a, b, c, h\}$ renders U_0 and U_1 independent, we can modify slightly the previous partition by adding h to each U_i and each d-sepset. Then the hyperchain $U_2 - U_0 - U_1$ will again be valid. The resultant subnet representation is shown in Figure 7.

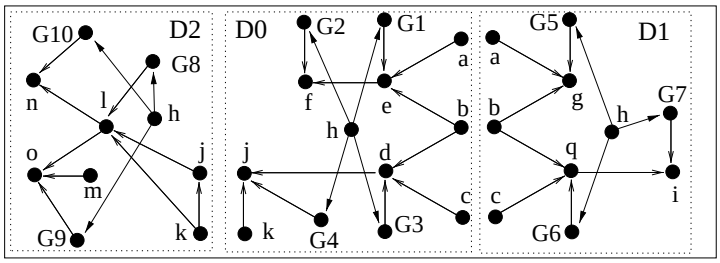

Figure 7: Subnets in an MSBN for the digital system.

In general, to partition a large system, we can first ignore the global faults and partition devices into components such that the hypertree constraint holds. Then we add the global faults into each component maintaining the hypertree structure. For each component, each global fault will be represented as a variable. It will be included in each d-sepset between subnets. The resultant representation will be a valid MSBN (assuming that other constraints are also satisfied).

In practice, faults can form a spectrum from local to global. For instance, the overheating variable h in the above example may affect the three components only but no other components. The above solution is still valid except that h will not be included in other components and hence not in the d-sepset among other subnets except between the three subnets in the figure.

Also note that the distinction between local and global faults depends on the size of the problem domain. A fault affecting 70 variables in a domain of 6000 variables should be modeled as a local fault, while a fault affecting 50 variables in a domain of 70 variables should be modeled as global.

It is clear that as the number of global faults increases, the size of each d-sepset and the complexity of inference will increase as well. Hence, the systems suitable for application of the MSBN framework are those that can be modeled with a small number of global faults. Otherwise, approximation in modeling appears necessary in order to apply distributed probabilistic reasoning. One such approximation would be to treat a global fault as a local one. For example, we may model the overheating in each component U_i by a variable h_i. This will *loosen* the coupling between subnets and hence makes inference more efficient. The price paid is that the actual logical constraint between h_i's cannot be exploited.

We indicate that the limitation to the number of global faults is *not* unique to the MSBN framework. Inference based on a flat BN is subject to the same limitation. This is not surprising as probabilistic inference in BNs is NP-hard in general (Coo90).

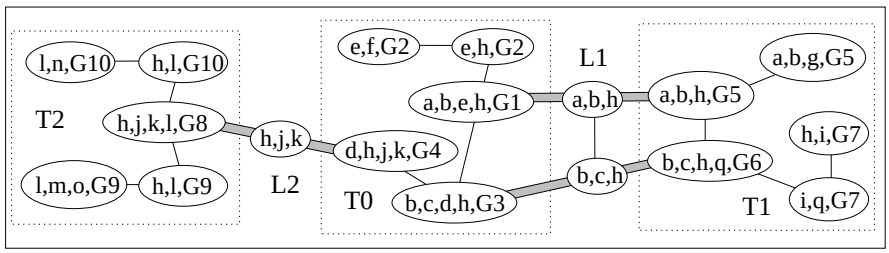

Figure 8: Part of the LJF of the MSBN for the digital system.

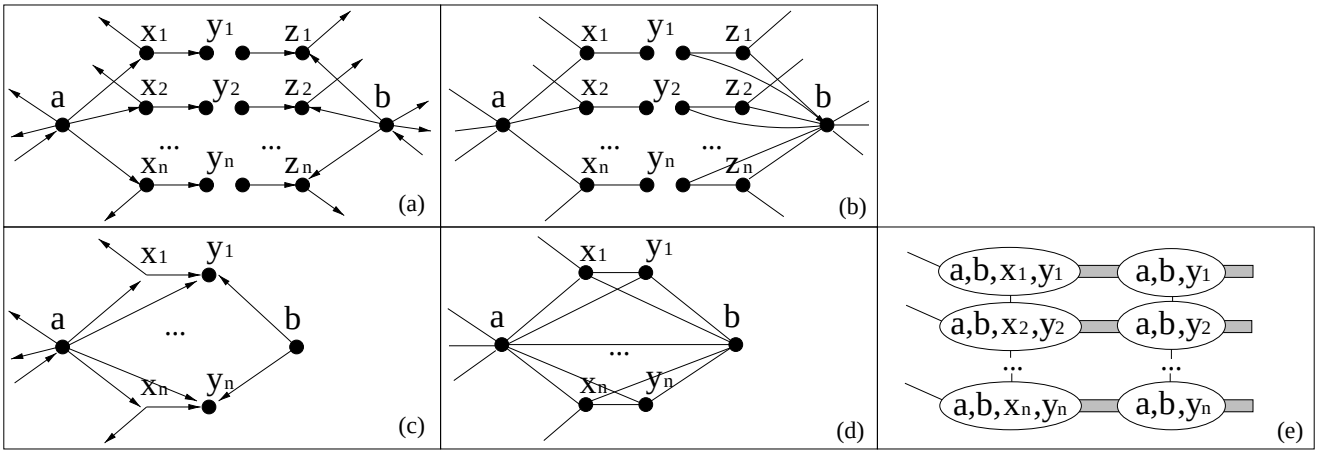

Figure 9: (a) Two subnets in an MSBN. (b) The moral graph. (c) Modified subnet. (d) The moral graph for subnet in (c). (e) The junction tree and the linkage tree.

4 How to reduce linkage tree state space?

The size of the d-sepset between a pair of subnets in an MSBN may be perceived as the ultimate factor that determines the efficiency of communication. Perhaps such perception originates from the observation that the size of a separator in the JT representation of a flat BN indeed determines the size of the message passed over the separator and affects directly the inference efficiency. However, this is only partially true for inference in an MSBN due to the linkage tree representation for each d-sepset.

For example, the partial MSBN in the above example is represented by the partial linked junction forest as shown in Figure 8. The linkage tree L_1 between JTs T_0 and T_1 has two cliques. Suppose each variable has four outcomes. Then the size of state space of L_1 is $4^3 + 4^3 = 128$, whereas the full state space of the d-sepset $\{a, b, c, h\}$ is $4^4 = 256$.

This illustrates that in an MSBN, the size of the state space of the linkage tree is a more direct factor for communication efficiency. In the following, we present a modeling technique for reducing this size.

Figure 9 (a) shows part of two adjacent subnets in an MSBN. The d-sepset is $\{y_1, ..., y_n\}$. Assume that there is no other paths between the nodes in (a) except those that are shown. After moralization, the moral graph looks as (b). Using constrained elimination for triangulation of the subnet in the left, we need to eliminate $a, x_1, ..., x_n$ before y_i $(i = 1, ..., n)$. As a consequence, the d-sepset will be completed (pairwise connected). The resultant linkage tree is trivial and has a state space of size $O(2^n)$.

In order to reduce the size of state space of the linkage tree, we *enlarge* the d-sepset by including variables a and b. The new subnet for the left part of (a) is shown in (c). Its moral graph is in (d). Using constrained elimination for triangulation, we need to eliminate $x_1, ..., x_n$ before $a, b, y_1, ..., y_n$. If we eliminate in the order $(x_1, ..., x_n, y_1, ..., y_n, a, b)$, no fill-in is added. The resultant partial JT for the subnet is shown in the left in (e) and the corresponding linkage tree is shown in the right. The linkage tree has reduced its size of state space from $O(2^n)$ to $O(2^3\ n)$. We can perform a similar change to the subnet in the right in (a) and obtain the same reduction.

In general, to use this technique, the variables added to the d-sepset should be chosen such that the new d-sepset partitions the variables in the subnet into conditionally independent groups. For example, x_is in the left subnet are rendered conditionally independent by the new d-sepset.

Note that the modification to the subnet as shown in (c) does not require assessment of a new set of probability distributions. For example, node y_1 is assigned $P(y_1|x_1)$ in the left subnet in (a). With the new subnet in the left of (c), there is *no* need to assess $P(y_1|x_1, a, b)$ and reassign y_1 this distribution. The reason is that our remodeling does *not* alter any dependence relations

among variables. We simply exploit some existing independence in the domain partition. As a consequence, the jpd of the new MSBN is identical to the previous jpd, and no node needs to change its previously assigned distribution.

To exploit this technique in multiagent systems, there is a small price to be paid. Agents have to reveal an additional piece of internal information, i.e., the variables added to the d-sepset. A negotiation mechanism is needed to identify these variables without revealing the entire internal structure of the agent. The negotiation mechanism can be activated by recognizing that the state space of a linkage tree is too large. In the above example, the agent with the left subnet can then use local heuristics to guess that adding a to the d-sepset may reduce the state space if the agent with the right subnet can do something similar. It can make a request to the agent with the right subnet for cooperation. We shall address the details elsewhere.

5 How to verify domain partition?

A system may be constructed from components supplied by different vendors (Xia98b). Each vendor may supply also a software agent for diagnosis of a component. The internal details of the component as well as the agent may be private to the agent (as representative of the vendor). To construct a multiagent MSBN for diagnosis of such a system, a system designer/integrator must ensure satisfaction of technical constraints. However, to protect the privacy of agents, the designer is not given the internal details of each agent. One constraint is that the union of subnet structures must be a DAG. This needs to be verified without revealing the (sub)DAG structure of each agent. The problem has been solved (Xia98b) with a distributed algorithm.

In this section, we address a different problem: verification of the suitability of a domain partition. According to the definition of MSDAG, whether a DAG union is a MSDAG can be verified by checking the d-sepset and local covering conditions. The verification is straightforward once a hypertree DAG union is specified. However, sometime it is desirable to know whether it is possible to build a MSDAG given a set of (sub)DAGs before the hypertree is given. Although the d-sepset condition can be tested pairwise, the local covering is a global condition and cannot be tested locally. Since the subDAG structure is private, the verification must be performed with only the information of the d-sepsets.

We shall call each variable in a d-sepset *public*. A variable not contained in any d-sepset is *private*. Each subnet S_i has a set of public variables denoted by Q_i, and a set of private variables denoted by V_i. We assume that the designer has the knowledge of all public variables but not for private variables.

The following theorem shows that if each hypernode in a MSDAG is labeled by its subdomain and each hyperlink is labeled by the d-sepset, then the resultant graph is a junction tree if and only if the graph union is built according to the local covering condition.

Theorem 6 *(Xia97) Let U_0,...,U_{k-1} each be a set of variables. Let G be a tree where each node is labeled by a U_i and each link is labeled by the nonempty intersection of its endpoints. Then the resultant graph is a junction tree if and only if G can be built according to the local covering condition.*

Using this relation, we test the suitability of a domain partition as follows: Based on the given knowledge, the designer can create an undirected graph G whose nodes are the set of all public variables. These nodes are connected such that each Q_i is complete. The following theorem suggests a simple test using G:

Theorem 7 *The set of subDAGs can be organized into a hypertree such that local covering is satisfied iff G is chordal.*

Proof:

Denote the set of nodes in each subDAG by U_i.

Suppose G is chordal. Then a junction tree T exists whose clusters are cliques of G. That is, each cluster of T is labeled by a Q_i. We construct a tree T' that is isomorphic to T. Each cluster of T' is, however, labeled by $Q_i \cup V_i = U_i$. For any two clusters in T', the intersection $U_i \cap U_j = Q_i \cap Q_j$ since V_i and V_j are private. Hence T' is also a junction tree. By Theorem 6, the set of subDAGs can be organized into a hypertree such that local covering is satisfied.

Next suppose the set of subDAGs can be organized into a hypertree with local covering. Then a junction tree T' exists such that each cluster is labeled by a U_i. From T', we construct a tree T isomorphic to T' with each cluster labeled by Q_i. Using the argument above, T is a junction tree and hence a corresponding undirected graph G is chordal. □

The Theorem 7 says that to determine if a particular domain partition is suitable for building an MSBN, it is necessary and sufficient to test whether the graph G is chordal. Hence the issue can be resolved with only the public knowledge about the agents.

6 Conclusions

Although the theory of MSBNs has been put forward a while ago, how to apply the theory in practice to built a normative diagnosis system may not be obvious to practitioners.

In this paper, we addressed three practical issues:

- How to model a system as a hypertree of subsystems when the natural system topology is not a hypertree?
- How to reduce the state space of the interface between subsystems such that the communication inference can be efficient?
- How to test the suitability of a domain partition without the private knowledge of each subdomain?

We hope that these results will help to meet the gap between the MSBN theory and diagnosis practitioners.

Acknowledgements

The support to the first author by Research Grant OGP0155425 from Natural Sciences and Engineering Research Council (NSERC) of Canada is acknowledged.

References

[BG88] A.H. Bond and L. Gasser, editors. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.

[CGH97] E. Castillo, J. Gutierrez, and A. Hadi. *Expert Systems and Probabilistic Network Models*. Springer, 1997.

[Coo90] G.F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393–405, 1990.

[DGH92] 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.

[HBR95] D. Heckerman, J.S. Breese, and K. Rommelse. Decision-theoretic troubleshooting. *Communications of the ACM*, pages 49–57, 1995.

[Hec90] D. Heckerman. *Probabilistic Similarity Networks*. PhD thesis, Stanford University, 1990.

[Jen96] F.V. Jensen. *An introduction to Bayesian networks*. UCL Press, 1996.

[KP97] D. Koller and A. Pfeffer. Object-oriented Bayesian networks. In D. Geiger and P.P. Shenoy, editors, *Proc. 13th Conf. on Uncertainty in Artificial Intelligence*, pages 302–313, Providence, Rhode Island, 1997.

[ML96] S.M. Mahoney and K.B. Laskey. Network engineering for complex belief networks. In *Proc. 12th Conf. on Uncertainty in Artificial Intelligence*, pages 389–396, 1996.

[Nea90] R.E. Neapolitan. *Probabilistic Reasoning in Expert Systems*. John Wiley and Sons, 1990.

[Pea88] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Sha96] G. Shafer. *Probabilistic Expert Systems*. Society for Industrial and Applied Mathematics, Philadelphia, 1996.

[Sri94] S. Srinivas. A probabilistic approach to hierarchical model-based diagnosis. In *Proc. 10th Conf. Uncertainty in Artificial Intelligence*, pages 538–545, Seattle, Washington, 1994.

[WJ95] M. Wooldridge and N.R. Jennings. Intelligent agents: theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[XG99] Y. Xiang and H. Geng. Distributed monitoring and diagnosis with multiply sectioned Bayesian networks. In *Proc. AAAI Spring symposium on AI in Equipment Service, Maintenance and Support*, 1999.

[Xia96] Y. Xiang. A probabilistic framework for cooperative multi-agent distributed interpretation and optimization of communication. *Artificial Intelligence*, 87(1-2):295–342, 1996.

[Xia97] Y. Xiang. Cooperative multiagent distributed interpretation: is multiply sectioned Bayesian networks necessary? Technical Report CS-97-05, Dec. 1997, University of Regina, 1997.

[Xia98a] Y. Xiang. Cooperative triangulation in MSBNs without revealing subnet structures. Technical Report CS-98-02, University of Regina, 1998. To appear in *Networks*.

[Xia98b] Y. Xiang. Verification of dag structures in cooperative belief network based multi-agent systems. *Networks*, 31:183–191, 1998.

[XPB93] 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.

[XPE+93] Y. Xiang, B. Pant, A. Eisen, M. P. Beddoes, and D. Poole. Multiply sectioned Bayesian networks for neuromuscular diagnosis. *Artificial Intelligence in Medicine*, 5:293–314, 1993.