Advance in Multiply Sectioned Bayesian Networks: Sensor Network Practitioners' Perspective

Y. Xiang
Dept. Computing and Information Science,
yxiang@cis.uoguelph.ca

K. Zhang
University of Guelph, Canada
zhangk@uoguelph.ca

Abstract

Multiply sectioned Bayesian networks provide a probabilistic framework for reasoning about uncertain domains in cooperative multiagent systems. Several advances have been made in recent years on modeling, compilation and inference under the framework. This paper links these advances together through a case study and presents them from the perspective of practitioners in intelligent sensor networks. We demonstrate how the framework can be applied to multisensor fusion and how intelligent sensor agents developed by independent vendors can be integrated into a coherent sensor fusion system.

1 Introduction

Built upon the success of Bayesian Networks (BN) [1], multiply sectioned Bayesian networks (MSBN) [2] provide a probabilistic framework for reasoning about uncertain domains in cooperative multiagent systems (MAS). Several advances have been made in recent years on modeling, compilation and inference under the framework, making the framework more suited to field application. Before a general technological framework can be turned into deployed applications, practitioners must well understand how theoretical intricacies map to practical reality. The contribution of this paper is to facilitate this process by linking together key technological steps involved in applying the MSBN framework through a case study in a laboratory setting.

Under the MSBN framework, a complex problem domain can be populated by a set of agents, each of which holds its partial perspective (a subnet over a subdomain). They reason autonomously as well as through limited communication and their beliefs are *exact* governed by Bayesian probability theory. The purpose of such reasoning is to determine what is the state of the domain so that agents can act accordingly. The domain of our case study is a moderately sized sensor network for monitoring a combinational digital system. We demonstrate how such a domain can be modeled as an MSBN-based MAS, how the MAS can be compiled into an efficient run-time representation, and how agents can cooperate to monitor the digital system and isolate faults (for repairing). The operations are demonstrated using a state of the art software tool kit, WebWeavr, developed by the first author. To serve its purpose, the paper will be kept as informal as possible, with pointers to references containing formal details.

2 Sensor Net for Digital System Monitoring

We consider monitoring a combinational digital system. It consists of remotely located components $U_0, ..., U_4$ supplied by five independent venders.

Each component has some external inputs, such as signal v_{54} in U_0. It may accept signals from other components, e.g., U_1 accepting signal b_0 from U_0, and may output signals to others, e.g., U_1 outputting signal c_0 to U_2.

Signals exchanged between components are labeled identically, e.g., $c_0, ..., c_9$ between U_1 and U_2. All signals are binary. Each digital device has a 0.01 probability to be faulty at any given time. A faulty NOT gate produces incorrect output 50% of time. The corresponding probabilities for AND and OR gates are 0.8 and 0.3, respectively. The digital system is used here as an example of any complex system made of multiple components, each of which can be further decomposed into simpler units, that together implement some functions, that may be electrical, mechanical, chemical, and so on.

To monitor such a system is to know whether, at any given time, it functions as intended and what units are faulty if it does not. A sensor network can be used to collect necessary information. We assume that each external input and the output of each gate (with some exception to be detailed later) can be observed through a sensor. Whether a gate is faulty cannot be observed and can only be inferred. These assumptions allow the case study to demonstrate the general nature of partial observability of practical sensor networks.

3 Integration of MSBN-based MAS

The digital system is monitored with an MAS. Each agent is responsible for one component and related sensors. The multiagent paradigm is well suited to the task: Sensor outputs related to one components are processed locally, reducing communication bandwidth and simplifying processing. Each agent is developed by the vendor of the component, protecting its know-how and removing the need for a centralized know-it-all expert. The core knowledge of an agent is a BN, called a *subnet*. The subnet consists of a set of discrete variables, called the *subdomain* of the agent, a directed acyclic graph (DAG), where each node corresponds to a variable and each arc corresponds to a causal dependence relation, and a probability distribution over the subdomain, specified by a set of conditional probability distributions (CPTs) one associated with each node x in the form of $P(x|\pi(x))$ where $\pi(x)$ is the parent nodes of x.

The subdomain V_0 of S_0 consists of two type of variables: *gate variable* and *sensor variable*. A gate variable represents the state of a digital gate: whether it is normal or faulty (denoted as $good$ and bad). For instance, vn_4 represents a NOT gate. A sensor variable represents the logical value of a signal perceived by a sensor (denoted as 0 and 1). For instance, vt_5 represents the sensed output of gate vn_4. We assume that sensors are reliable, although unreliable sensors can also be modeled with slightly increased complexity.

Suppose that a sixth independent vendor, called *Assembler*, assembles the five components into the final digital system. It also assembles the five corresponding agents into an MAS.

In the figure, the agent organization is drawn, that is a tree structure [2] (called a *hypertree*), where each node is labeled by the logic name of an agent. The organization defines to whom an agent can communicate *directly*.

As part of the organization specification, each agent is associated with a set of *public* variables. For instance, A_0 has public variables $b_0, ..., b_{10}$: signals exchanged between U_0 and U_1. Each public variable in an agent must be associated with at least another adjacent agent (otherwise, the variable is not really public). For each pair of adjacent agents, the two sets of public variables must have a non-empty intersection (otherwise, probabilistic information cannot be exchanged between them). If two non-adjacent agents A_i and A_j have a common public variable, then it must be a public variable in each agent along the hypertree pathway between A_i and A_j (otherwise, information from A_i on the variable cannot be communicated to A_j).

Integrator tool gives feedbacks to Assembler during organization specification until the above conditions are satisfied. Note that as Assembler, the vendor is in a unique position (responsible to interface components together) to negotiate with other vendors in deciding the public variables. The MAS is now *logically* specified and ready to be *physically* set up.

To do so, Assembler uses WebWeavr Binder tool. Binder has the access of MAS organization specification and waits for each agent to register by sending its physical address. After all agents have registered, Binder notifies each of them with the physical address of each adjacent agent on the hypertree as well as the set of common public variables, called *agent interface*. The MSBN-based MAS is now *integrated*, as each agent knows to whom it can communicate directly, how to reach thm, and what information should be exchanged during communication.

4 Model Verification

In addition to the hypertree agent organization, two other conditions are critical to exact reasoning. First, when agent subnets are viewed as a whole (by merging their public variables), it must be a DAG. This requirement maintains the causal interpretation of the dependence relations of the domain. Although each subnet is a DAG (as mentioned above), when multiple DAGs are merged together, it may be cyclic and violate causal interpretation. As each subnet is *private* (built by an independent vendor), the global DAG condition cannot be verified by physically merging individual subnets.

Furthermore, given the hypertree organization, public variables in an agent interface play the role of passing *all* relevant information from one side of the hypertree to another. This is possible only if each public variable x is a *d-sepnode*. That is, if we denote parent nodes of x by $\pi(x)$, taking into account all subnets that contain x, then there exists one subnet that contains $\pi(x)$. Again, because each subnet is *private*, the d-sepnode condition cannot be verified by physically merging individual subnets.

The verification tool DVerify in WebWeavr tool kit implements algorithms [3, 4] that verify both conditions without merging individual subnets. The verifications are performed by message passing among agents along the hypertree, initiated by any agent (referred to as the *root*). A message reveals only whether a given agent has any parent or child of a public node, but reveals neither the number of parents or children nor what they are. During verification, each agent executes a copy of DVerify

5 Agent Interface Enhancement

An MSBN-based MAS that has passed the above verification can support autonomous and exact multiagent probabilistic reasoning. It does not, however, ensure that the inference computation is efficient. An agent communicates with an adjacent agent by sending its subjective probability distribution over their interface. If the interface consists of m variables and each has k possible values, the message contains k^m values. For instance, the interface between A_1 and A_3 has 12 binary variables and a message between them then has a size of 4096.

To reduce the message size while supporting exact inference, it may be possible to factorize the distribution over the interface. For instance, if variables $e_0, ..., e_4$ are conditionally independent of $e_8, ..., e_{11}$ given e_5, e_6, e_7, then the message between A_1 and A_3 can be encoded into two distributions over $e_0, ..., e_7$ and $e_5, ..., e_{11}$ with a total size of 256+128 = 384.

What if there are no conditional independence relations within the interface or those that exist cannot reduce the message size sufficiently? A solution is to enhance the interface with additional variables that bring some conditional independence within the subdomain into the interface. Identifying these variables among a large number of alternatives is non-trivial and must take into account of both dependence relations within and across subnets. Instead of burdening the agent developers with the task, it may be better performed by a multiagent search.

To protect the privacy of subnets during enhancement, each agent vendor can specify variables in its subnet into three sets: private, public, and preferably private. The *private* set should be kept so absolutely. The *public* set is included in the initial agent interface. The *preferably private* set is initially private, but the agent is allowed to make some elements public if it believes that the disclosure improves inference efficience. The agent is required, however, to keep the disclosed subset as small as possible.

A suite of algorithms for multiagent interface enhancement has been developed [5]. Through multiagent search, each of the four agent interfaces are enhanced. For example, the interface between A_1 and A_3 (consisting of $e_0, ..., e_{11}$) is enhanced with additional variables $yd_{82}, yd_{101}, yd_{106}, w_{14}, wr_{16}, wr_{18}$. These variables bring several independence relations into the interface. For instance, e_0, e_1, e_2 are independent of e_3, e_4, e_5 given wr_{16}, yd_{106}. As the result of enhancement, the message size between each pair of adjacent agents is reduced significantly, as shown in Table 1, with the new message size to be as low as about 4% of the original (between A_3 and A_4).

Table 1. The message size between each pair of adjacent agents before and after interface enhancement.

Interface	$A_0 - A_1$	$A_1 - A_2$	$A_1 - A_3$	$A_3 - A_4$
Before	2048	1024	4096	4096
After	136	136	336	160

6 Compilation into Linked Cluster Trees

Inference computation in an MSBN-based MAS consists local inference at individual agents and communication among agents. Local inference involves updating the agent's belief (subjective probability distribution) over its subdomain based on local observations. Communication from one an agent involves passing its subjective probability distribution (the message) over an agent interface. The two computations are intertwined: A message for communication must be derived from the sending agent belief, and a message received must be absorbed into the receiving agent belief.

To allow both computations to be efficient, each agent compiles its subnet into a cluster tree, where variables are grouped into *clusters* with intersections of adjacent clusters referred to as *separators*. The cluster tree is so constructed such that the intersection of any two clusters is contained in every cluster on the path between them. Each cluster is associated with a probability distribution over its member variables. These cluster distributions are more efficient in space, yet uniquely define the agent's subjective probability distribution over its subdomain

To compute the message to an adjacent agent, an agent compiles its cluster tree into a reduced cluster tree, called a *linkage tree* [6]. It contains only variables in the interface between the pair of agents. Each cluster in the linkage tree is called a *linkage*. Each linkage has a corresponding cluster in the cluster tree, called its *host*, that contains the linkage. Each linkage is associated with a probability distribution that is derived from the distribution associated with its host. From these linkage distributions, the agent subjective probability distribution over the agent interface can be constructed. Although the distribution over the interface has a size of 32768, the inter-agent message made of linkage distributions has a total size of 136.

The above compilation effectively converts the MSBN-based knowledge representation into a set of linked cluster trees. It has been shown [8] that, if the MSBN is sparse, local inference and communication using linked cluster trees are efficient.

7 Sensor Net Monitoring and Fault Isolation

To monitor the digital system domain, each agent will collect sensor outputs and reason about the state of its subdomain autonomously. Less frequently, agents may choose to communicate in order to benefit from information in other agents. Through interleaving local inference and communication, agents can determine whether the digital system functions normally and isolate faults if not.

WebWeavr DMasMsbn tool supports agent sensing, inference and communication. We demonstrate digital system monitoring through the following scenario: AND gate wa_{130} in U_1 and OR gate $y0_{49}$ in U_3 are faulty and producing incorrect outputs. The incorrect outputs propagate through other gates and produce more incorrect signals throughout the system. To experiment in a laboratory setting, Scenario Simulator tool in WebWeavr is used to simulate the physical digital system and its associated physical sensor network and interact with the agents during monitoring. It simulates a set of external inputs to the the digital system, the faulty gates, and all other signals. Upon an agent's request, it sends the corresponding sensor value the agent as an observation.

To monitor the domain, each agent is assumed to have the bandwidth to observe as many sensors as about 5% of variables in its subdomain at one time. We assume that all signals are observable except the outputs of wa_{130} and $y0_{49}$. In the first round of observation, each agent observes the values of between 4 and 10 variables.

These local observations are not sufficient to detect any abnormality and none of the agent does after local inference. However, after one round of communication among agents, during which one message is passed from each agent to each adjacent agent, the pooling of information allows agents to detect abnormality. A_0 has $P(va_{44} = bad|obs) = 0.025$. A_1 has a number of gates suspected,

$$wn_{132}, wo_{124}, wo_{163}, wa_{126}, wa_{122}, wa_{139}, wa_{141}, wa_{130},$$

for instance, $P(wa_{130} = bad|obs) = 0.131$ Similarly, A_2 has $P(xa_{32} = bad|obs) = 0.132$, A_3 suspected

$$yn_{39}, yo_{43}, yo_{49}, yo_{15}, yo_{102}, yo_{121}, yo_{95}, ya_{105}, ya_{46},$$

and A_4 suspected $zn_{20}, zn_6, zo_{18}, zo_{61}, za_{59}, za_{13}, za_{56}$.

Alarmed, each agent makes more observations, subject to the bandwidth restriction. A_0 observes signals associated with the suspected gate va_{44}. Its output vd_{45} has been observed. Hence, its inputs v_{42} and v_{43} are observed and A_0 no longer suspects va_{44}.

A_1 observes the output of each suspected gate, except that of wa_{130}. After local inference, A_1 reduces its uncertainty and suspects only three gates:

$$wn_{128}, wn_{132}, wa_{130}.$$

A_2 observes two signals and no longer suspects xa_{32}. A_3 observes 8 signals and decides that $P(yo_{49} = bad|obs) = 0.504$ and $P(yo_{95} = bad|obs) = 0.504$. Given that the signal between the two, yr_{50}, is not observable, this is the best that A_3 can achieve. Perhaps, it will replace them in sequence and do a test in between. A_4 observes 6 signals and decides that its subdomain is normal.

As A_1 suspects three gates, it makes one more observation possible related to them: wt_{129}. After inference, it reduces the suspected gates to only wn_{132} and wa_{130}, which is the best that A_1 can achieve given the unobservability of the signal in between.

8 Conclusion

There are few multiagent frameworks for sensor net monitoring (space limit precludes a comprehensive reference). For example, Roos et al [9] proposed one based on logical consistency. They showed that establishing a global diagnosis under the framework is NP-Hard and therefore their protocol does not guarantee one. On the other hand, MSBN-based sensor net monitoring is efficient and guarantees globally consistent diagnosis, as long as the sensor net dependence structure (such as the one in the case study) is reasonably sparse. This contribution presents the framework intuitively (versus mathematically) through a sensor net monitoring case study and facilitates its application and deployment by practitioners.

Acknowledgements

The financial support from National Sciences and Engineering Research Council (NSERC) of Canada through Discovery Grant to the first author as well as that from National Science Foundation (NSF) of USA through a research contract with University of Massachusetts/Amherst are acknowledged.

References

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

[2] Y. Xiang and V. Lesser. On the role of multiply sectioned Bayesian networks to cooperative multiagent systems. *IEEE Trans. Systems, Man, and Cybernetics-Part A*, 33(4), 2003, 489-501.

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

[4] Y. Xiang and X. Chen. Interface verification for multiagent probabilistic inference. In J.A. Gamez, S. Moral, and A. Salmeron, editors, *Advances in Bayesian Networks*, (Berlin, Springer, 2004) 19-38.

[5] Y. Xiang and K. Zhang. Agent interface enhancement: Making multiagent graphical models accessible. In *(accepted to appear in) Proc. 5th Inter. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS'06)*, 2006.

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

[7] F.V. Jensen, S.L. Lauritzen, and K.G. Olesen. Bayesian updating in causal probabilistic networks by local computations. *Computational Statistics Quarterly*, (4), 1990, 269-282.

[8] Y. Xiang. *Probabilistic Reasoning in Multi-Agent Systems: A Graphical Models Approach* (Cambridge University Press, 2002).

[9] N. Roos, A.T. Teije, and C. Witteveen. A protocol for multi-agent diagnosis with spatially distributed knowledge. In *Proc. 2nd Inter. Joint Conf. on Autonomous Agents and Multiagent Systems*, Melbourne, 2003, 655-661.