PROBABILISTIC MODELING AND REASONING IN MULTIAGENT DECISION SYSTEMS

ZENG YIFENG

NATIONAL UNIVERSITY OF SINGAPORE

2005

PROBABILISTIC MODELING AND REASONING IN MULTIAGENT DECISION SYSTEMS

ZENG YIFENG

(M. ENG., Xia'men University, PRC)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2005

Acknowledgements

As I will soon get my PHD degree from the NUS, I would like to express my heartfelt gratitude to the many people who I am indebted to.

First and foremost, I would like to thank my supervisor, professor Poh Kim Leng. He has offered many fresh insights on how I should conduct my research work. Besides, he has also helped me in writing some comprehensive and well-motivated academic papers. I am grateful to his advice, encouragement and patience under his supervision.

I would also like to thank professor Leong Tze Yun. She has been supporting my research work and research activities since I joined the Biomedical Decision Engineering (BiDE) group four years ago. She has pointed out many mistakes in earlier versions of this dissertation, and given many valuable suggestions on the revision. I must also acknowledge professor Marek J. Druzdzel in University of Pittsburgh (U. S.), who has offered great advice on a part in this dissertation. He has been helping the building of my academic career.

My colleagues at the BiDE group, including Li Guoliang, Jiang Changan, Liu Jiang, Chen Qiongyu, Rohit, Yin Hongli, Ong Chenhui, Zhu Peng, Zhu Ailing, Xu Songsong, and Li Xiaoli, has all asked interesting questions in my presentation, and offered helpful comments on my research. I have enjoyed their company in our trips to meetings and conferences abroad.

My juniors, including Cao Yi, Wang Yang, Wu Xue, Guo Lei, and Wang Xiaoying, have been painfully reading the earlier versions of this dissertation. They has put much effort into the correction of confusing sentences, and given useful remarks on my research.

The members of the system modeling and analysis laboratory (SMAL), including Han Yongbin, Liu Na, Liu Guoquan, Zhou Runrun, Xiang Yanping, Lu Jinying, Bao Jie, and Aini, have spent a lot of time with me during my stay in Singapore. We have all got along very well. The lab technician, Tan Swee Lan, has provided an easy and convenient work space for us. I will memorize the happy time there for ever.

Last but certainly the most important, I owe a great debt to my family members: my wife Tang Jing, my father, my mother, and my brother. Their love and continual support on all levels of my life are priceless.

Table of Contents

[This page intentionally left blank]

Summary

Multiagent decision problems under uncertainty are complicated by large dimensions and agency features. New techniques for solving decision problems involving multiple agents are the focus of current research because existing approaches are unable to address such a large and complex decision problem and no effective methods can be utilized. To address a multiagent decision problem, I investigate probabilistic graphical model representation and evaluation methods as well as Bayesian learning algorithms. Bayesian learning algorithms help the construction of graphical decision models. The main challenging work is to solve a distributed decision problem involving multiple agents. In addition, learning a large Bayesian network structure from small data sets is a more complex task.

I proposed a new framework, including Multiply Sectioned Influence Diagrams (MSID) and Hyper Relevance Graph (HRG), to represent multiagent decision problems. This framework extends influence diagrams and considers properties of multiple agents. MSID is a probabilistic graphical decision model encoding agency features and is able to adapt to the changing world for its distributed design while HRG quantifies organizational relationships in multiagent systems. Then, I presented a symbolic method to verify a valid representation of MSID and HRG. This novel method exploits the algebraic property of probabilistic belief networks as well as the domain knowledge.

After that, I developed three evaluation algorithms to solve proposed decision models. The three evaluation algorithms are categorized into two groups: one is a direct approach that includes cooperative reduction algorithms and multiple evaluation networks; the other is an indirect approach based on rooted cluster tree algorithms. These algorithms designed in

a distributed fashion adopt some optimization strategies to ensure information consistency in the evaluation process. A case study on disease control involving multiple nations or communities in the medical domain was used to demonstrate the practical value of model representation and model evaluation algorithms. The results indicated that the new framework of MSID and HRG could represent a multiagent decision problem and the three evaluation algorithms are effective and efficient.

In addition, I investigated the issue of learning large Bayesian network structures in order to build a probabilistic decision model from data. Adopting the divide and conquer strategy, a novel learning algorithm, called block learning algorithm, was designed to learn a large network structure from a small data set. Instead of learning a whole network structure directly, the block learning algorithm learns individual blocks that constitute a final structure. Experimental results on two golden networks (ALARM and Hailfinder networks) showed that this new algorithm could be scaled up to learn a sizable network structure from a small data set and the algorithm is easily configured in the implementation. Hence the block learning algorithm provides a foundation to develop a unifying Bayesian learning framework.

All results show that my proposed methodologies could be used to solve multiagent decision problems. These methods could be generalized to solve many decision problems in practice such as the decision problem of disease control in the medical domain.

List of Tables

[This page intentionally left blank]

List of Figures

1 Introduction

Decision making in our daily lives often involves a group of persons who cooperate to achieve their goals. This decision problem can be modeled as a multiagent decision problem in which each agent acts cooperatively to achieve the best expected outcome in uncertain environments. The uncertainty, the dynamic nature of decision scenario and the unique attributes of multiple agents make it hard to solve a multiagent decision problem. Hence, it is worthwhile to investigate some effective and efficient methodologies to solve the problem.

1.1 Background and Motivation

A simple decision problem is often related to a person's scope of perception. In a large social network composed of many individuals, decisions are beyond any individual's scope and tend towards a group decision that is more valuable. Decision making in uncertain environments mainly concerns decision problems in which a number of agents are involved. Making a good decision in a multiagent system is particularly complicated when both the nature of decision scenario and the attributes of multiple agents have to be considered.

Research in decision analysis, artificial intelligence, operations research, and other disciplines has led to various techniques for analyzing, representing, and solving decision problems in uncertain environments. Most of these techniques make use of a

graphical probabilistic model, such as influence diagrams (Howard & Matheson 1984), limited memory influence diagrams (Lauritzen & Vomlelova 2001), unconstrained influence diagrams (Jensen & Vomlelova 2002), and sequential influence diagrams (Jensen *et al.* 2004). They provide a compact and informative representation for modeling decision problems in an uncertain setting. However, these techniques lack the ability to tackle multiagent decision problems because they are oriented to the single agent paradigm without considering the features of multiple agents.

Recently, achievements in the multiagent reasoning system have cast light on research about multiagent decision problems. Most work, such as Multiply Sectioned Bayesian Networks (MSBN, Xiang 2002), focuses on the communication and reasoning in multiagent systems. They have successfully developed a distributed and coherent framework for solving probabilistic inference problems in multiagent systems. This framework lays out a foundation for the research on the multiagent decision making.

The work on solving decision problems involving multiple agents benefits the building of intelligent decision systems. The construction of intelligent decision systems is always a burdensome task in a large knowledge domain. Existing approaches are not able to build such large decision systems in practice. Hence, a flexible framework with powerful evaluation algorithms is needed for an effective design of general methodologies for dealing with a large and complex knowledge domain. Case studies will show the practical value of my proposed techniques. On the other hand, a new learning algorithm is utilized to build a large probabilistic model from a data set, which enriches learning techniques that drive model construction.

1.2 The Multiagent Decision Problem

This work addresses multiagent decision problems in which agents reside in a distributed, but connected setting and they cooperate to make decisions on the basis of certain organizational relationships.

Some characteristics of this decision scenario are as follows: 1) Agents are distributed geographically or physically. Each agent is an independent entity in the world. It is not easy and reasonable to merge them into a single object. 2) Agents are cooperative. Although each agent is an independent entity, it still needs some cooperation for solving a certain decision problem. The cooperation is based on public information that they share. 3) Agents' privacy is protected. Although agents are in a cooperative setting, they intend to hold their privacy. 4) Agents' decisions and observations are interleaved; however, their interactions follow a sequential order. In a distributed environment, agents need some observations from their adjacent agents to support decision making. 5) Agent's organizational relationships exist. In a cooperative decision problem, an agent may need some information for its decision making while this information could only be obtained from its adjacent agents. Thus, a certain organizational relationship exists among these agents. Meanwhile, this kind of organizational relationship could be described by the relation between the information property and the supported decisions. 6) Agents seek their individual objectives while they expect a cooperative solution. In a distributed decision problem, every agent has its own goal since it is selfish. It wants to make the best decision on its own through the cooperation in which it could access some requisite information. For a cooperative solution globally, agents contribute by releasing honest and up to date information through which they expect to help the decision making in their adjacent agents. Hence, in this kind of decision scenario, what cooperative agents do concern is the shared

information. They are unwilling to compromise their own utility with the consideration of others' decisions.

Accordingly, a complex and large knowledge domain complicates the multiagent decision problem. The agency features, such as privacy and organizational relationships, make the decision problem more intractable although these features enrich the decision scenario.

1.3 The Application Domain

Medicine is a very rich domain for multiagent decision making. While the multiagent decision problems that I address are general, the application domain that I examine is focused on the policy design involving multiple communities or nations in medical decision making. Differing from medical decision making on diagnostic test and therapy planning (Leong 1994), the decision problem that I deal with is more related to policy design for disease control. The large domain with multiple decision entities, the uncertain information about disease and the intricate organizational relationships in the domain complicate a policy design process. Furthermore, decision making in a distributed and cooperative setting requires a trade-off among multiple objectives. Hence, the disease control involves both uncertain domain knowledge and the properties of multiple decision entities.

In the disease control domain, multiagent decision making will not only consider the uncertain environment, but also take into account the information exchange among the interacting units. The uncertain environment and the personal judgments comprise uncertain information in the domain. The complex relationships among associated decision entities determine the accessibility of public information and individual

objectives in collective actions. For instance, in Severe Acute Respiratory Syndrome (SARS) control (http://www.who.int/csr/sars/en/), multiple nations would share some information, like current status of SARS, and hold together aiming at alleviating the damage of SARS; although each nation has her own interest and private consideration.

1.4 Objectives and Methodologies

The goal of this thesis is to establish new methodologies for solving the multiagent decision problem, as well as develop novel techniques for learning large Bayesian network structures from a small data set. To achieve this goal, I carry out several stages as follows:

First of all, it is to build a new flexible framework. The main advantage of this decision-theoretic framework lies in its capability for representing a large knowledge domain in a distributed way. Furthermore, it adapts to a changing decision scenario by self-organizing its components. Hence, this adaptive framework should support large and complex decision systems in the changing world.

Then, it is to encode agency properties into a new representation. To personalize real decision making, this new framework is to be enriched with some properties of multiple agents. It not only describes the environment, but also reflects the characteristics of decision makers in a decision scenario. This agency approach must make probabilistic decision models more meaningful by strengthening their linkage with artificial intelligence concepts.

After that, some evaluation algorithms are to be developed to solve the model. Extended from basic methods for solving single agent based decision models, these

evaluation algorithms will be improved by overcoming some "bottleneck" issues of existing approaches. Its effectiveness and efficiency will be shown in practical case studies. Aiming at solving a large and complex decision model, these algorithms could improve the existing approaches and could be implemented.

Finally, a novel technique is to be proposed to learn large Bayesian network structures from a small data set. Adopting the divide and conquer strategy, the learning algorithm will solve a learning problem step by step. Some experiments are to be designed to show its learning ability. Armed with good strategies, this new learning algorithm is comparable to some typical learning algorithms and may be implemented in a commercial tool.

This study will address the issue of multiagent decision making under uncertainty with probabilistic graphical decision models. Hence, the explored area is confined to uncertainty in artificial intelligence and mostly concerns decision-theoretic systems. The existing techniques relevant to this work are influence diagrams, Bayesian networks and multiagent decision systems. This context will be illustrated in Chapter 2.

1.5 Contributions

The major contributions of this work are as follows:

Firstly, I have proposed a new probabilistic graphical model, as well as a relevance graph, to represent a multiagent decision problem under uncertainty. This framework is proposed for its ability to encode agency properties and for its capability to model a large and complex decision problem. It will facilitate decision modeling languages to solve a general class of decision problems.

Secondly, I have established a symbolic method to verify a probabilistic graphical decision model. By holding an algebraic view on the model, this approach breaks the traditional mold of graph-theoretic verification methods. These results will provide a unique insight into the research on probabilistic graphical decision models.

Thirdly, I have developed three evaluation algorithms for solving a decision model. Extended some basic evaluation algorithms for the single agent paradigm, these algorithms are shown to be effective and efficient. To demonstrate their utility, I formalize a case study in the disease control domain to highlight the capabilities and limitations of each approach. These results clearly illustrate evaluation strategies and will also contribute toward the design of adaptive solver systems.

Fourthly, I have presented a new algorithm on learning large Bayesian network structures from a small data set. Adopting the divide and conquer strategy, this learning algorithm has shown good performance in a series of experiments. A learning tool with an implementation of this novel algorithm will be put into practical use.

Finally, this research has provided insights into the representation, verification, and evaluation of multiagent decision problems. It also investigates the issue of learning Bayesian network structure. These methodologies can be generalized for addressing a class of general decision problems.

1.6 Overview of the Thesis

This chapter has given a concise introduction to some basic concepts in the field of decision analysis, reviewed some major work related to the topics addressed in this

dissertation, and roughly described the methodologies used and the overall contributions.

The rest of this thesis is organized as the following:

Chapter 2 introduces related work involving various graphical decision models and evaluation algorithms used in these representations. Most of the current work on Bayesian network structure learning is also covered.

Chapter 3 presents a graphical multiagent decision model to describe multiagent decision problems. The main characteristics of this new representation are highlighted and the model construction procedures are discussed in terms of a simple case.

Chapter 4 proposes a symbolic method to verify the decision model. The foundation and detailed operations in this approach are fully described. In addition, the complexity problem associated with this method is also analyzed and some measures are proposed to handle the verification failure problem.

Chapter 5 presents three evaluation algorithms to solve the new decision model proposed in Chapter 3. The comparison of these three algorithms shows their strengths on solving different graphical structures of the decision model.

Chapter 6 focuses on a decision problem in the medical domain. The whole solution procedures involving model representation, model verification, and model evaluation are described in detail.

Chapter 7 proposes a novel learning algorithm to learn a large Bayesian network structure from a small data set. Some experimental results and theoretical analysis demonstrate a good performance of this new learning approach. The new learning

method also benefits the building of probabilistic graphical models from non-context decision problems.

Chapter 8 summarizes this dissertation by discussing the contributions and limitations of the whole work. It also suggests some possible directions for future research.

[This page intentionally left blank]

2 Literature Review

This chapter briefly surveys some related work: Bayesian networks and multiply sectioned Bayesian networks, decision modeling with influence diagrams and multiagent influence diagrams, intelligent agent and multiagent decision making, and Bayesian network structure learning. The survey focuses on the major techniques on which this work is based and serves as a basis to a more detailed analysis on the capabilities and limitations of the existing approaches.

2.1 Bayesian Networks and Influence Diagrams

The concepts of Bayesian networks and influence diagrams are fundamental elements in the probabilistic modeling and reasoning. They provide basic ideas and techniques for the probabilistic expert systems and are to a large segment of the uncertainty in artificial intelligence (AI) community what resolution theorem proving is to the AI logic community.

2.1.1 Bayesian Networks and Multiply Sectioned Bayesian Networks

2.1.1.1 Bayesian Networks

A Bayesian network (Pearl 1988) is a compact representation of a joint probability distribution over a set of random variables. Formally, a discrete Bayesian network (BN) is a pair (G, P) consisting of a directed acyclic graph (DAG) G and a multiplicative factorization of the joint probability distribution P. Each node $x \in G$, which is called

chance node in a BN, corresponds to a discrete variable x framed with a conditional probability distribution $p(x|\pi(x))$ ($\pi(x)$: parents of variable x) that composes P over the domain $P = \prod_{x \in G} p(x|\pi(x))$. An arc between each pair of nodes indicates an influence or causal relationship between the corresponding variables. A Markov blanket of node $x \in G$ is composed of parents, children of node x and parents of children of node x.

Figure 2.1 shows an example of BN.

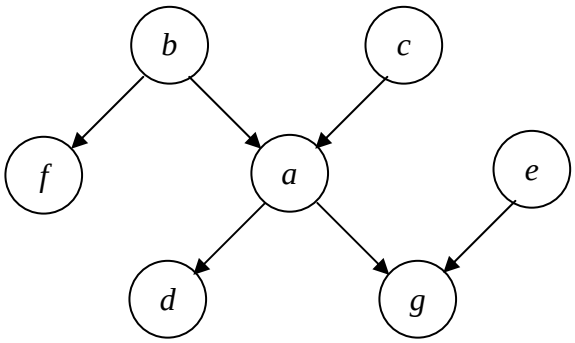


Figure 2.1: A BN

The BN consists of seven nodes $\{a, b, c, d, e, f, g\}$ and arcs between some of them. Each value of a node in the BN has one conditional probability distribution given a configuration of the values of its parents, such as node a with a conditional probability $p(a|b,c)$. A DAG structure and conditional probabilities in a BN define a unique factorization of a joint probability distribution in the domain. For example, in Figure 2.1, the BN gives the joint probability distribution as follows: $P(a,b,c,d,e,f,g) = p(b)p(c)p(f|b)p(a|b,c)p(e)p(d|a)p(g|a,e)$. The Markov blanket of node a includes nodes $\{b, c, d, g, e\}$.

One important concept in Bayesian networks is d-separation (Geiger & Pearl 1989). The d-separation encodes the independence relations specified by a DAG and follows the criterion below:

Definition 2.1: Let G be a directed acyclic graph. If X, Y, and Z are disjoint subsets of the nodes in G, then X and Z are said to be D-Separated given Y if there does not exist a trail between a node in X and a node in Z s.t.:

1. For every intermediate node w in a converging connection (head-to-head), either $w \in Y$ or w has a descendant in Y.

2. For every intermediate node w in a serial (head-to-tail) or diverging (tail-to-tail) connection, $w \notin Y$.

If X and Z are not D-Separated given Y, then we say the X and Z are D-Connected given Y. Each trail satisfying the conditions above is called active; otherwise, it is said to be blocked. For example, node c and node d are D-Separated given node a.

Probabilistic reasoning is one of the most important issues which should be considered in a Bayesian network framework; this task, however, has been proved to be NP-hard (Cooper 1990). In the past two decades, various methods have been proposed for inference in Bayesian networks. In general, these methods are divided into two groups: exact approaches and approximate approaches. The exact approach includes the junction tree method (Lauritzen & Spiegelhalter 1988; Jensen *et al.* 1990; Shafer 1996; Madsen & Jensen 1998), loop cutset conditioning method (Pearl 1988; Suermondt & Cooper 1991), direct factoring method (Li & Ambrosio 1994), variable elimination method (Dechter 1996) and so on.

Both the junction tree and loop cutset conditioning methods are based on the Kim and Pearl's message passing algorithm (Pearl 1988; Neapolitan 1990; Russell & Norvig 2003). The loop cutest conditioning method converts a general Bayesian network into multiple simpler polytrees. Each polytree performs the message passing algorithm resulting in a final combined answer. While the junction tree method transforms a general Bayesian network into one clustering tree with some graph operations such as moralization, triangulation, and so on, the propagation launches a message passing algorithm. The direct factoring and variable elimination methods view the inference as one combinatorial optimization problem. They target the query variables and marginalize (sum) out the rest of the variables one by one from the product of a small subset of probability distributions.

The approximate approach includes the search based inference (Henrion 1991; Poole 1993) and simulation based inference, like the logic sampling (Henrion 1988), likelihood sampling (Fung & Chang 1989; Shachter & Peot 1992), Gibbs sampling (Jensen 2001), self-importance sampling and heuristic-importance sampling (Shachter 1989), adaptive importance sampling (AIS-BN, Cheng & Druzdzel 2000), backward sampling (Fung & Favero 1994), and importance sampling using evidence pre-propagation (EPIS-BN, Yuan & Druzdzel 2003). The search based inference method approximates the posterior probability of query variables by summing a small subset of joint probability values that contains most of the probability mass. On the other hand, the simulation based methods use Monte Carlo sampling techniques to simulate a sufficient number of cases and compute the posterior probability from them. Other approximate inference methods have also been proposed. These include the state space abstraction (Wellman & Liu 1994), localized partial evaluation (Draper 1995), and removal of weak arcs (Kjærulff 1994).

In general, the exact inference methods provide precise results, but require a lot of computational costs. In practice, currently, the exact algorithms using junction tree are good enough for most small to medium sized networks, up to three dozens of nodes or even larger. However, the performance of the exact algorithms largely depends on the connectivity of networks. For large networks, or networks that are densely connected, approximate algorithms are preferred. As with exact inference methods, the approximate inference methods are also proved to be NP-hard within an arbitrary tolerance (Dagum & Luby 1993). If evidence being conditioned upon is not too unlikely, these approximate approaches converge fairly quickly. Currently, both AIS-BN and EPIS-BN have very good performance even with unlikely evidence. Accordingly, the approximate inference methods are good complements of the exact inference approaches for the propagation in Bayesian networks, especially for Bayesian networks with a large size.

To deal with some special cases, some extensions to Bayesian networks have been proposed. For example, the dynamic Bayesian networks (DBN, Nicholson 1992; Nicholson & Brady 1992; Russell & Norvig 2003), probabilistic temporal networks (Dean & Kanazawa 1989; Dean & Wellman 1991), dynamic causal probabilistic networks (Kjærulff 1997) and modifiable temporal belief networks (MTBN, Aliferis *et al.* 1995, 1997) model the change over time. These BNs, such as DBN and MTBN, are temporal extensions of BNs to facilitate normative temporal and casual modeling under uncertainty. They have a joint BN model encoding every time slice so that they could overcome the drawback of BNs which are not designed to model temporal relationships explicitly. Specifying some real problems, other extensions of Bayesian networks also appeared such as the probabilistic similarity networks (Heckerman 1990), hierarchical Bayesian networks (Srinivas 1994), object-oriented Bayesian

networks (OOBN, Koller & Pfeffer 1997) and probabilistic relational models (PRM, Koller & Pfeffer 1998; Koller 1999; Getoor 2001; Heckerman *et al.* 2004). The probabilistic similarity networks have many advantages in solving a large knowledge domain in which some variables are related to many mutually exclusive and exhaustive variables. The hierarchical Bayesian network models hierarchical knowledge in a tree structure so that the search space of models is reduced. The object-oriented Bayesian network uses Bayesian network fragments to describe the probabilistic relations between attributes of an objective in a large and complex domain. However, it is unable to model the uncertainty about structures. The probabilistic relational model evolves from OOBN and represents relationships between multiple instances of the same object class. It introduces uncertainty into database schema resulting in a combination of probabilistic reasoning and entity-relational schema in databases.

The above mentioned work is still around the probabilistic representation and propagation in the single agent paradigm, which leads to its failure in treating multiagent reasoning problems effectively.

2.1.1.2 Multiply Sectioned Bayesian Networks

Orienting towards the multiagent reasoning problem, the representation of multiply sectioned Bayesian networks (MSBN, Xiang *et al.* 1993; Xiang 2002) is considered as the milestone for solving the probabilistic reasoning in a multiagent system. It provides a coherent framework for probabilistic reasoning in cooperative multiagent distributed interpretation systems. It aims to solve a large and complex knowledge domain by dividing the domain into several subnets each of which is related with an intelligent agent. With a distributed fashion, an MSBN allows the privacy protection of intelligent

agents and the active communication in a multiagent system. Formally, the definition of an MSBN is given as follows (Xiang 2002).

Definition 2.2: An MSBN M is a triplet (N, G, P). $N = \bigcup_i N_i$ is the total universe where each N_i is a set of variables. $G = \bigcup_i G_i$ is a hypertree structure where nodes of each DAG G_i are labeled by elements of N_i. Let x be a variable and $\pi(x)$ be all parents of x in G. For each node x, exactly one of its occurrence (in a G_i containing $\{x\} \cup \pi(x)$) is assigned $P(x|\pi(x))$, and each occurrence in other DAGs is assigned a constant table. $P = \Pi_i P_{G_i}$ is a joint probability distribution, where each P_{G_i} is the product of the probability tables associated with nodes in G_i. A triplet $S_i = (N_i, G_i, P_{G_i})$ is called a subnet of M. Two distinct subnets S_i and S_j are said to be adjacent if G_i and G_j are adjacent in G.

It can be seen that an MSBN comprises a set of Bayesian networks that share some common nodes. Here, the common nodes compose an interface S between adjacent Bayesian networks associated to individual agents. One important property of the interface in an MSBN is stated as follows: the adjacent agents are independently conditioned on the observation of states in the interface which is the only channel for all their communication. Consequently, the definition of a d-sepset is followed to implicate the agent interface.

Definition 2.3: Let G be a directed graph such that a hypertree over G exists. A node x contained in more than one subgraph with its parents $\pi(x)$ in G is a d-sepnode if there exists one subgraph that contains $\pi(x)$. An interface S is a d-sepset if every $x \in S$ is a d-sepnode.

An example of an MSBN is shown in Figure 2.2. The MSBN is a DAG which comprises two local BNs, namely BN_0 and BN_1, each of which represents an individual agent's reasoning engine. Through common nodes $\{a,b,c\}$ coding their public information, these two agents communicate with each other to obtain a full and consistent reasoning in a multiagent system. In Figure 2.2, common nodes $\{a,b,c\}$ are also d-sepnodes since all of their parents reside in one local BN. For example, the parents of node b are nodes $\{d,a\}$ which are in BN_1, the only parent of node a is node d in BN_1 while the parents of node c are nodes $\{b,g\}$ residing in BN_0. Hence, these common nodes form a d-sepset between BN_0 and BN_1 in the MSBN.

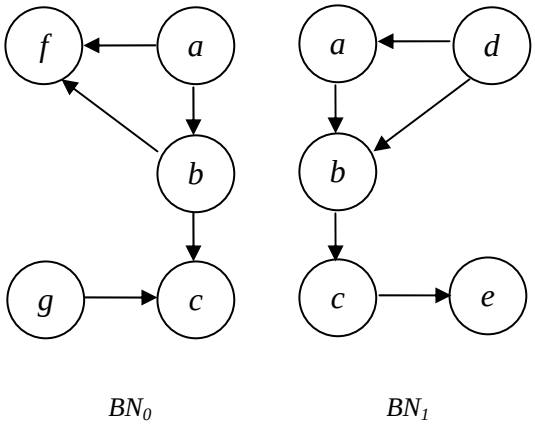


Figure 2.2: An MSBN

The definition of an MSBN addresses the issue of cooperative agents reasoning in a compact model. Considering some properties of intelligent agents, an MSBN shows a smart extension of the single agent based Bayesian Networks.

Once a multiagent MSBN is constructed, agents may perform probabilistic inference through coherent communication initiated by some observations. The inference methods in an MSBN are extensions of those for the single agent based Bayesian

networks. For example, the linked junction forest method (Xiang 1994) compiles each subnet into a junction tree, called a local junction tree, and converts each d-sepset into a junction tree, called a linkage tree. Then, the message passing algorithm is used in a junction tree for a general Bayesian network. Other propagation methods in an MSBN include the distributed forward sampling (Xiang 2002) extending the logic sampling (Henrion 1988), the distributed cutset conditioning (DCC, Xiang 2003) extending the loop cutset conditioning method in general Bayesian networks (Pearl 1988), the distributed Markov sampling (DMS, Xiang 2003) extending the Gibbs sampling (Jensen 1996), and so on.

With its distributed framework and efficient inference methods, an MSBN provides a good solution for a multiagent reasoning problem. On the other hand, it does not address the problem of decision making in a multiagent system. However, it provides a foundation to develop a representation of multiagent decision problems.

2.1.2 Influence Diagrams and Multiagent Influence Diagrams

2.1.2.1 Influence Diagrams

An influence diagram (Howard & Matheson 1984) is a graphical modeling language that represents the probabilistic inference and decision analysis model. An influence diagram describes the dependencies in decision analysis and specifies the states of information for which independencies can be assumed to exist. It can be considered as a Bayesian network augmented with decision nodes D and a value node v. Formally, an influence diagram is defined as follows (Zhang *et al.* 1994).

Definition 2.4: An influence diagram is a pair $I = (G, P)$ whose elements are defined as follows:

1. $G = (N, A)$ is a DAG such that $N \subseteq C \cup D \cup \{v\}$, where C and D are disjoint, and the following conditions are satisfied:

   (a) The value node v is a sink node which has no successors;

   (b) A directed path only consisting of all the decision nodes D exists in G;

   (c) Each decision node and its parents are parents to all of its subsequent decision nodes;

2. $P = \{P_i\}_{i \in C \cup D}$ is a collection of families P_i of conditional probability distributions $p(x \mid x_{\pi(i)})$, with one distribution for each configuration of $x_{\pi(i)}$.

It can be seen that an influence diagram is a two-layer representation with a qualitative level and a quantitative level. At the qualitative level, it is a directed acyclic graph G with three types of nodes: chance nodes C, decision nodes D and a value node v. At the quantitative level, a frame of numerical data P_i is associated with each node. At the same time, it is noticed that there are some constraints in the definition. Condition (b) implies a single decision maker should perform the decisions in a chronological order. Condition (c) is referred to as the no-forgetting constraint that information available at the time of one decision must be available at the time of all subsequent decisions. An influence diagram is always termed as a regular one when it satisfies all of the above constraints (Zhang 1994).

For example, one influence diagram is shown in Figure 2.3. The decision maker selects one alternative indicated in decision node d according to the evaluation of expected values corresponding to combinations of different outcomes in value node v.

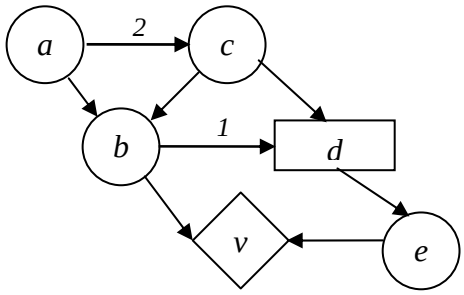


Figure 2.3: An Influence Diagram

In an influence diagram, an arc from a chance node to a decision node is called information arc which indicates chance nodes should be observed before the decision making. Simultaneously, the chance nodes are called observed nodes, denoted as the information set $I(D)$. An arc between chance nodes and value nodes or chance nodes is called influence arc which indicates chance nodes should affect their downstream nodes. Similarly, the descendants of decision node D, denoted as the set $Des(D)$, are affected when decisions are made. For example, in Figure 2.3, the arc *1* is an information arc while the arc *2* is an influence arc. The information set for decision d is $I(d) = \{b, c\}$ and $Des(d) = \{e, v\}$ depends on the outcome of decision d.

Currently, one relevant research issue is to determine requisite probability nodes RP_{d_i} and requisite observation nodes RO_{d_i} for decision node d_i in an influence diagram. Requisite probability nodes are those nodes for which conditional probability distributions might be required to compute the utilities of decision node d_i given other nodes. Requisite observation nodes are those observation nodes for which conditional probability might be needed to compute the utilities of decision node d_i given other nodes. Thus far, two approaches have appeared: one is the *Decision Bayes-ball* procedure (Shachter 1998, 1999) and the other is the *refined Decision Bayes-ball*

procedure (Nielsen 2001). Both approaches are based on simple techniques that stem from d-separation (Druzdzel & Suermondt 1994). The second procedure decides a minimum set of relevant value nodes for decision nodes beforehand so that the set of required nodes found in the procedure is more compact. Hence the basic *Decision Bayes-ball* procedure to construct the sets of requisite probability nodes RP_{d_i} and requisite observation nodes RO_{d_i} in an influence diagram with separable value nodes V and decision node $D = \{d_1, \cdots, d_i, \cdots, d_m\}$ is described here.

[Decision Bayes-ball]

FOR Iterate backwards for each earlier decision node $d_i \in D = \{d_1, \cdots, d_i, \cdots d_m\}$, $i = m, \cdots, 1$, **DO**

1. Let $V_i = \{V \cap \{Des(d_i) \setminus Des(d_{i+1})\}\}$ if $i < m$; otherwise $V_i = \{V \cap Des(d_i)\}$;
2. Run Bayes-ball algorithm on $X|Y$, where $X = \{V_i \cup RO_{d_{i+1}}\}$ if $i < m$, otherwise $X = V_i$, and $Y = \{d_i \cup I(d_i)\}$, in the influence diagram while ignoring any information arcs;

   FOR every $x \in X$ **DO**

   a) Visit x from a parent or child, or both;

   b) If $x \notin Y$ and the visit to x is from a child;

      i. If the top of x is not marked, then mark its top and visit each of its parents;

      ii. If the bottom of x is not marked, then mark its bottom and visit each of its children;

   c) If the visit to x is from a parent;

      i. If $x \in Y$ and the top of x is not marked, then mark its top and visit each of its parents;

ii. If $x \notin Y$ and the bottom of x is not marked, then mark its bottom and visit each of its children;

d) Mark x as visited and let $X = X \setminus x$.

END

3. If decision node d_i is not marked as visited then the decision is irrelevant to decision makers' value;
4. RP_{d_i} consists of all of the nodes marked on top starting with decision node d_m;
5. RO_{d_i} consists of all of the nodes in $I(d_i)$ marked as visited.

END

According to the analysis (Shachter 1998, 1999), the time computational complexity of *Decision Bayes-ball* algorithm is linear in the number of nodes and incident arcs into a node in an influence diagram.

In addition to the above research issues, evaluation algorithms for solving influence diagrams have been discussed in detail in much literature (Garcia & Druzdzel 2004). A basic and effective approach for solving a regular influence diagram is a reduction algorithm (Shachter 1986, 1988). The reduction algorithm simplifies the solving process by removing nodes from influence diagrams one by one. Some basic operations are involved in the reduction algorithms such as node removal and arc reversal. An improved reduction algorithm that avoids the operation of arc reversal was proposed in potential influence diagrams (Ndilikikesha 1994). It seems that these two algorithms are effective and easy to be implemented; however, determining an optimal elimination order in a reduction algorithm has been shown to be NP-hard (Cooper 1987).

Noticing the close relationships between Bayesian networks and influence diagrams, Shachter & Peot (1992) initiated research on an indirect method for solving influence diagrams through two steps. First, an influence diagram is transformed into a Bayesian network using the Cooper's transformation (Cooper 1988). Then, a probabilistic propagation is performed in the Bayesian network to obtain final decisions.

Two years later, Jensen *et al.* (1994) proposed the HUGIN architecture for solving influence diagrams that is based on the message passing in a strong junction tree. A strong junction tree representation of an influence diagram I is identified from its triangulated graph (Nielsen 2001):

1. Remove all informational arcs from I;
2. Moralize I and remove all value nodes resulting in the graph I^m;
3. Triangulate I^m by eliminating the variables with a strong elimination order.

A strong junction tree T is composed of the cliques identified from the triangulated graph. Of all these cliques in T, at least one distinguished clique R is called a strong root, such that for each pair (C_1, C_2) of adjacent cliques in T, it is true if C_1 is closer to R than C_2, there exists a partial order with nodes in the separator $C_1 \cap C_2$ preceding nodes in $C_2 \setminus C_1$.

After that, the strong junction tree is initialized by attaching each clique with the probability potential and the utility function from the influence diagram and activated to perform the message passing procedure. Message passing proceeds by absorbing messages that contain both a probability potential Φ_C and a utility function Ψ_C associated to the clique C from the leaves towards the strong root. A clique can pass

the message to its parent clique only if it has received the message from all of its children cliques.

Let C_i and C_j be neighbouring cliques with separator $S_{ij} = C_i \cap C_j$. Then absorption from C_i to C_j is involved with the following procedures: Firstly, to calculate the probability potential and the utility function passed through separator S_{ij} : $\Phi_{S_{ij}} = \underset{C_j \setminus S_{ij}}{\mathrm{M}} \Phi_{C_j}$ and $\Psi_{S_{ij}} = \underset{C_j \setminus S_{ij}}{\mathrm{M}} \Phi_{C_j} \cdot \Psi_{C_j}$ (where for chance variables, the symbol M equals to the sum operation; while for decision variables, it equals to the max operation); Secondly, to attach the clique C_i with the new Φ'_{C_i} and Ψ'_{C_j} : $\Phi'_{C_i} = \Phi_{C_i} \cdot \Phi_{S_{ij}}$ and $\Psi'_{C_j} = \Psi_{C_i} + \frac{\Psi_{S_{ij}}}{\Phi_{S_{ij}}}$.

Finally, the optimal policy for a decision variable can be obtained from the potentials associated with the closest clique to the strong root in T.

The strong junction tree algorithm for solving influence diagrams adopts the approach of message passing in Bayesian network inference when it transforms the decision model into a tree composed of cliques. Each clique is solved with the outcome of probabilities or utilities which are further transmitted and compose final policies.

Influence diagrams involve with only one decision maker in a symmetric decision problem. However, it provides a standard presentation to be extended to solve other types of decision problems, such as the Dynamic Influence Diagram (DID, Tatman & Shachter 1990), Valuation Bayesian Networks (VBS, Shenoy 1992), Multilevel Influence Diagrams (MLID, Wu & Poh, 1998), Time-ctitical Dynamic Influence Diagrams (TDID, Xiang & Poh, 1999), Limited Memory Influence Diagrams (LIMID,

Lauritzen & Vomlelova 2001), Unconstrained Influence Diagrams (UID, Jensen & Vomlelova 2002), and Sequential Influence Diagrams (SID, Jensen *et al.* 2004).

The above mentioned work has solved many kinds of decision problems. For instance, the DID models a temporal decision problem, the TDID solves a time-critical dynamic decision problem, and both UID and SID handle an asymmetric decision problem. However, all these model representations orient towards the single agent paradigm. This research work is insufficient to deal with the multiagent decision problem.

2.1.2.2 Decision Networks and Multiagent Influence Diagrams

To deal with decision problems involving with multiple agents, Zhang *et al.* (1994) proposed decision networks by lifting some constraints of influence diagrams. On representation, as with influence diagrams, decision networks still adopt traditional modeling strategies: describing the whole decision scenario in one model. Hence, the model size is intractable when the decision scenario becomes large and complex. On evaluation, one of the best evaluation algorithms, called *evalID* algorithm, was proposed in decision networks (Zhang 1998). Adopting the divide and conquer strategy and Bayesian network inference methods, the *evalID* algorithm partitions decision networks into several parts and solves each part individually with Bayesian network inference methods. Thus, it overcomes the "bottleneck" of evaluation approaches for solving decision problems with a large dimension. In all, a representation of decision networks without considering the properties of multiple agents still lacks the ability to describe a large and complex multiagent decision problem; however, the *evalID* evaluation algorithm is a good basis for developing efficient and effective evaluation algorithms for solving a general decision problem.

Multi-agent Influence Diagram (MAID, Koller & Milch 2001) is considered as a milestone regarding the research work on multiagent decision problems. It focuses on the representation of games and tries to find Nash Equilibria (Nash 1950) in the games with some strategies. A MAID extends the formalisms of Bayesian networks and influence diagrams to represent game problems involving multiple agents. To take advantage of independence structures in a MAID, a qualitative notion of strategic relevance is defined to find a global equilibrium through a series of relatively simple local computations. Since the goal of MAID is to represent and solve games involving multiple agents, its solution strategies specify only on game problem that is a subnet of decision problem. Furthermore, the MAID represents the whole decision scenario within one decision model so that it lacks the ability to handle a complex and larger problem domain..

Tracing the evolution of influence diagrams, it is found that some disadvantages still exist on the representation when the multiagent decision problems are to be solved. Two of these deficiencies are highlighted: one is that some properties of multiple agents, such as privacy and sociality, have not been considered; the other is that no effective and efficient methods have been proposed to cope with large and complex problem domains.

2.2 Intelligent Agents and Multiagent Decision Systems

With the development of computer network and distributed computing technology based on network, the research of intelligent agents and multiagent systems has been a new focus in the field of computer science (Wooldridge & Jennings 1995; Wiess 1999). An intelligent agent has the following attributes: 1) Reactivity: It can sense the

environment and act responsively; 2) Autonomy: It does not need human intervention; 3) Social and collaborative behavior: It can work with other agents or humans toward a common goal; 4) Inferential capability: It proactively seeks to meet its goals, including analyzing its environment; 5) Temporal continuity: Its identity and state persist over long periods; 6) Adaptivity: It can learn and improve with experience. In addition, one of the most outstanding characters residing in an intelligent agent is the privacy. Agents have the intention to protect their own privacy and only disclose the necessary information under some agreements. This feature does match some phenomena in the practical world. For example, an individual computer system or others, which could be considered as an agent, is often designed by different vendors who are not willing to release full details of their system design. Consequently, it is challenging work to make workable integrated systems without knowing details of individual systems.

A single intelligent agent only displays an individual intelligence, which is not a focus of artificial intelligence. One of the primary goals in artificial intelligence is to analyze, represent, and control the behaviors of intelligent agents, including belief reasoning and decision making, in a dynamic and uncertain environment. Not only does it focus on the individual agent's intelligence but also on the multiple agents' intelligence. Thus, research on multiagent systems is more promising. A multiagent system (Wiess 1999) can be viewed as an agent organization (by analogy with human organization) or as an artificial society or organization. It has, like a human organization evolving in a certain environment, goals to achieve and agents operate together to achieve these goals. Meanwhile, the individual agents are self-organized with certain types of relationships. For more complete characterizations and formalizations, five relationships are identified (Zambonelli *et al.* 2001): 1) Control: It identifies the authority structures with a system; 2) Peer: It identifies agents of equal status; 3)

Benevolence: It identifies agents with shared interests; 4) Dependency: It identifies the ways in which one agent may rely on another; 5) Ownership: It delimits organizational boundaries. Hence, with the bonding of a certain organizational relationship, multiple agents would cooperate to make decisions in a multiagent system, which is called multiagent decision systems.

The traditional research on multiagent decision systems focuses on the Cooperative Distributed Problem Solving (CDPS) which studies how loosely-coupled networks of problems solvers can work together to solve problems that are beyond their individual capabilities (Durfee *et al.* 1989a, 1989b). The main issues to be addressed in CDPS include the task sharing, result sharing, and coordination. Some progress has been achieved such as contract net for task sharing (Smith 1977, 1980a, 1980b; Smith & Davis 1980), coordination through partial global planning (Durfee 1988, 1996). Meanwhile, a large amount of research efforts have been invested in multiagent decision systems, such as the protocol or mechanism design in agents' interactions and agents' communication languages (Wooldridge 2002). The classic application is in distributed sensing (Lesser & Erman 1980; Durfee 1988). For example, Lesser's well-known Distributed Vehicle Monitoring Testbed (DVMT) provides a ground for many of today's multiagent system development techniques. Other applications consist of multiagent information retrieval systems (Wellman *et al.* 1996), policy modeling by multiagent simulation (Downing *et al.* 2001) and so on.

One of the most important elements in multiagent decision systems is the interaction among multiple agents which have their own sphere of influence (Jennings 2000). In most of current literature, the study of multiagent encounters is oriented towards game theory (Von Neumann & Morgenstern 1947). In general, Nash equilibrium is sought in

order to guide what agents should do in any given scenario. Usually, two strategies s_1 and s_2 are said to be in a Nash equilibrium between agent i and agent j if : 1) under the assumption that agent i plays s_1, agent j can do not better than play s_2; and 2) under the assumption that agent j plays s_2, agent i can do not better than play s_1. Hence, neither agent has any incentive to deviate from Nash equilibrium. Without doubt, much research effort goes to the topic of searching for Nash equilibrium in multiagent interactions. An insightful summary could be found in (Mckelvey & McLennan1996; Von Stengel 2002). Concerning my research topics, I care more about solving games with probabilistic graphical models, such as MAID. The approach is called graphical models for games.

The classical work on graphical models for games is on game trees (Fudenberg & Tirole 1991). A game tree represents agents' actions within internal nodes and utility value of outcomes within terminal nodes along with each branch. Hence, a game tree has the curse of dimensions and obscures certain important structure that is often present in real world game scenarios. To overcome these disadvantages of a game tree, La Mura's work on Expected Utility Networks (EUNs, La Mura & Shoham 1999) and Game networks (G nets, La Mura 2000) incorporate both the probabilistic and utility independence in a multiagent game setting. La Mura defined a notion of strategic independence, and used it to break up the game into separate components so that it facilitates an economical method for computing all equilibria in the game. On another aspect, a variety of algorithms for identifying equilibria in a game have also appeared. The TreeNash algorithm (Kearns *et al.* 2001a, 2001b) exploits the locality of interaction that always exists in complex multiagent games described compactly in a graphical structure. It also assumes that each agent's reward function depends on the actions of a subset of agents rather than all other agents' actions. The multiagent

algorithms (Vickrey & Koller 2002) use variable elimination methods to solve graphical games.Here, a kind of gradient ascent algorithm was also proposed to determine the equilibria profile. However, the running times of these algorithms depend on the tree-width of the graph and these algorithms require multiple interactions and no bound is currently known on the number of interactions required. Recently, the continuous method (Blum *et al.* 2003) exploits game structures and follows a trajectory of equilibria of perturbed games until the equilibrium of the original game is found. A simple method (Ryan *et al.* 2004) formulates game problems as a feasibility program and adopts some approaches, such as the general back-tracking algorithm for solving Constraint Satisfaction Problem (CSP, Dechter 2003), to search all equilibria in a game.

All of the above research work on multiagent decision systems is of a game-theoretic orientation. It intends to seek a kind of equilibrium among multiple agents and to help analyze their interactions with the aim to design a suitable strategy or protocol in multiagent systems.

2.3 Learning Bayesian Network Structure from Data

Influence Diagrams can be considered as Bayesian networks consisting of only chance nodes, with the addition of decision nodes and value nodes. The construction of influence diagrams is to identify influence or information relationships between any pair of chance nodes, decision nodes and value nodes. In general, determining influence relationships between chance nodes is one of the most difficult tasks in the probabilistic graphical model construction. Hence, building Bayesian networks from a

domain becomes a popular research focus in the past two decades. It indirectly helps the construction of influence diagrams or other graphical decision models.

A Bayesian network that is made up of structures and parameters can be built either through domain knowledge or from data. The first approach is called Bayesian network construction from domain knowledge while the second one is called Bayesian network learning from data. Constructing Bayesian networks from domain knowledge is a little subjective because of experts' judgment, which often results in inconsistent networks. Moreover, it is difficult to elicit dependence relationships and variable probabilities from domain experts. Consequently, much effort has been made to devise some engines for learning Bayesian networks from data.

In general, approaches for learning Bayesian networks are categorized according to two cases: 1) whether the structure is known or unknown; and 2) whether the data set is complete or incomplete. When the structure is known, the problem becomes a parameter learning problem; otherwise, the problem becomes a structure learning problem when the structure is unknown beforehand (Neapolitan 2004). It is also possible to learn structures and parameters together from data. However, learning structures is much more difficult than learning parameters. An incomplete data set complicates the learning problem while a complete data set alleviates the learning difficulty. The problem I address leans towards the issue of learning Bayesian network structures from a complete data set also called learning Bayesian network structures from data.

In this section, I would like to briefly review some typical techniques for learning Bayesian network structures. I will discuss some basic learning methods which provide

foundations to various learning algorithms, and some advanced learning methods which are well designed recently.

2.3.1 Basic Learning Methods

The basic learning approaches which appeared in the last decade provide basic ideas for learning Bayesian networks. Based on these concepts various learning approaches have been developed. The task of learning Bayesian network structures is to find an accurate structure that best fits the observed data. This task is hard because the number of possible networks in the search space is super-exponential in the number of nodes in Bayesian networks. The learning becomes intractable when the size of Bayesian networks increases. For example, 10 variables in Bayesian networks result in 4.2×10^{18} possible searched structures (Glymour & Cooper 1999). Thus, a lot of heuristic search approaches have been proposed to speed up the search in the structure space by finding a local optimal structure.

The methods for learning structures are generally classified into two groups: scoring based search and constraint based search. In the first approach, the algorithms view the learning as an optimization problem. They try to find a structure that can best explain dependence relationships among attributes in the data set based on some scoring criteria, such as the Bayesian scoring method (Cooper & Herskovits 1992; Madigan & Raftery 1994; Buntine 1994; Buntine 1996; Geiger & Heckerman 1995; Heckerman 1995; Heckerman 1996; Ramoni & Sebastiani 1997; Friedman & Koller 2000), minimum description length method (Bouckaert 1993; Suzuki 1993; Lam & Bacchus 1994; Friedman & Goldszmidt 1996; Suzuki 1996; Tian 2000) or entropy based method (Herskovits 1991; Steck 2000; Rebane & Pearl 1987; Herskovits & Cooper 1990; De Campos 1998). For example, the Bayesian score (Cooper & Herskovits 1992)

for measuring the learned Bayesian network G is its posterior probability given the database D : $p(G^h|D) = P(G^h, D) / P(D)$ where G^h denotes the hypothesis of the Bayesian network structure. Later, the BDe metric (Bayesian metric with Dirichlet priors and equivalence) (Heckerman *et al.* 1995) evolves from the search for the network with the largest posterior probability given Dirichlet priors over network structures and parameters.

The typical scoring based algorithms include the K2 algorithm (Cooper & Herskovits 1992), the HGC algorithm (Heckerman *et al.* 1994), the WKD algorithm (Wallance *et al.* 1996) and some heuristic algorithms such as genetic algorithm and evolutionary programming (Larranaga *et al.* 1996; Larrañaga *et al.* 1996; Myers *et al.* 1999; Wong *et al.* 1999), simulated annealing (Chickering *et al.* 1995), tabu search (Bouckaert 1995; Muntenau & Cau 2000) and ant colony optimization (de Campos *et al.* 2002). Since the search space in the scoring based approach is always large, local search methods are imposed in some common algorithms (Buntine 1991; Chickering *et al.* 1995; Heckerman *et al.* 1995; de Campos *et al.* 2003). The main idea of local search based methods is to decide the "neighbor" structure of each node. A common definition of "neighbor" refers to all structures that can be generated from the current structures by adding, deleting or reversing a single arc, subject to the acyclicity constraint. For instance, the K2 algorithm (Cooper & Herskovits 1992) first initializes nodes with no parents, but with a prior order; then incrementally adds parents to increase the score of the resulting structure. When no addition of a single parent can increase the score, it stops adding parent nodes. On another aspect, the heuristic algorithm, like genetic algorithm (Larranaga *et al.* 1996), starts from a network with or without node ordering. Then it performs some operations with mutation or crossover operators, until the score does not increase with these operations. Another typical learning algorithm is greedy

search (Chickering 2002a). Greedy search starts at a specific point (an initial structure) in the structure space, considers all the nearest neighbors, and moves to the neighbor that has the highest score. If no neighbor has a higher score than the current point (i.e. a local maximum is reached), the algorithm is terminated. In Chickering's work (Chickering 1996), it shows that greedy search with random restarts can produce better models than a heuristic algorithm. Furthermore, Chickering (Chickering 1996; Chickering 2002b) provides a theoretic justification of the greedy search method and exploits the concept of equivalence classes of Bayesian network structures to facilitate the search process. Two classes of Bayesian network structures are said to be equivalent if their distributions represented by their corresponding graphical structures are equal (Chickering 2002b).

In the constraint based approach, the algorithms try to infer the structure by identifying dependencies from data through some conditional independency (CI) tests. In fact, the CI tests are statistical tests on the data set. In order to use the results to reconstruct the structure, some assumptions have to be made. The assumptions of causal sufficiency and causal Markov condition ensure the reconstruction of causal models given data, such as detecting the existence of edges between nodes and orienting their directions. The assumption of faithfulness provides the justification of a constraint based approach for recovering Bayesian networks from data (Spirtes *et al.* 1993). The typical methods include the Wermuth-Lauritzen algorithm (Wermuth & Lauritzen 1983), boundary directed acyclic graph algorithm (Pearl 1988), the SGS algorithm (Spirtes *et al.* 1990), the IC algorithm (Pearl & Verma 1991), the PC algorithm (Spirtes *et al.* 1993). Most of these proposed learning algorithms, such as the IC and PC algorithms, find dependency structures from data by conducting CI tests to identify and/or orient arcs between each pair of nodes. Hence, their complexity, like the complexity of the SGS

algorithm, increases exponentially with the number of variables of the domain. The complexity of the PC algorithm, which is considered as more efficient and popularly used in many experiments, also increases exponentially with the degree of any node in Bayesian networks (Spirtes *et al.* 1993).

There are also some hybrid algorithms that use a combination of constraint based and scoring based approaches (Singh & Valtorta 1993; Singh & Valtorta 1995; Spirtes & Meek 1995; Dash & Druzdzel 1999; Acid & De Campos 2000; Acid & De Campos 2001; De Campos *et al.* 2003). For instance, a clever method in Dash and Druzdzel's work (Dash & Druzdzel 1999) uses a constraint based method to search a space of equivalent networks; then borrows the Bayesian score to evaluate the candidate models.

It can be seen that much progress has been made on learning Bayesian network structures. However, most of these approaches do not aim at learning large Bayesian networks with hundreds or thousands of variables. Furthermore, when the size of the data set is quite small, the structure learning problem becomes more intractable. This is especially true for the constraint based methods, where the reliability of CI tests decreases when there is insufficient data. Here, the methods discussed above are ranked as basic learning methods. Hence the challenging work of learning large Bayesian networks from a small data set disables most of basic learning methods.

2.3.2 Advanced Learning Methods

The advanced learning approaches appear in recent years. They comprise much current insightful research work on the edge of learning techniques. Some of them adopt well-designed strategies and focus on the challenging work of learning a large Bayesian network (from a small dataset), such as the Sparse Candidate algorithms (SC, Friedman *et al.* 1999), the Three Phases Dependency Analysis algorithm (TPDA,

Cheng *et al.* 2002), the Max-min Bayesian networks (MMBN, Tsamardinos *et al.* 2003) and module networks (Segal *et al.* 2003).

The SC algorithm (Friedman *et al.* 1999) learns possible parents of each variable to recover the whole network. However, in an unknown structure, it is difficult to set the possible number of parent variables. Hence, many iterative learning processes have to be performed to obtain a final optimal network. In the same year, an incremental learning algorithm (Castelo & Siebes 1999) was proposed to learn a large Bayesian network. This algorithm first groups several clusters of nodes in the network; then recovers the whole network incrementally. However, it is only effective for a sparse Bayesian structure and it is hard to set the cluster size.

The TPDA algorithm (Cheng *et al.* 2002) is a typical constraint based learning algorithm that is based on information theory completely. It divides the learning process into three phases: drafting, thickening and thinning. In the first two steps it outputs a graph that catches all dependence between two nodes based on the criteria of mutual information. In the third step, it identifies some conditional independence relationships for each pair of nodes and removes the corresponding arcs to recover a final structure. The TPDA algorithm has been successfully implemented in the learning tool of Belief Network Power Constructor (BNPC, Cheng *et al.* 2002). The SC and TPDA algorithms are able to learn a medium Bayesian network efficiently and may have the potential ability to learn a large Bayesian network.

Two more promising algorithms for solving the challenging work are max-min Bayesian networks and learning module networks. The MMBN algorithm learns the Markov blanket of each node to recover the whole network. Each local structure in Bayesian networks is discovered in a relatively large sample size in comparison with

the size of a local structure, which makes the learned result more robust. However, the MMBN algorithm is suitable for learning a sparse structure in which a large number of nodes are prohibited in the Markov blanket. Otherwise, this algorithm is always not quite efficient when a large local network contains numerous nodes from the Markov blanket of each node. The other algorithm in learning module networks emphasizes a set of variables that display some common features in the data set. Then, it clumps these variables into some modules and learns each module individually. After this it combines local modules to recover the final network. To ensure global optimization for the combination, it is unavoidable that some constraints exist in this algorithm. For example, it requires that all nodes in the same module must have the same parents.

Recently an inclusion-driven learning approach (Castelo & Kocka 2003) has been proposed to learn Bayesian networks from a large dataset with 10,000 cases or so. This new method utilizes partial orders encoded in conditional independencies, called inclusion order, and searches the space of equivalent classes of Bayesian networks. The idea is unique; however, it does not claim the ability to learn a large Bayesian network from a small data set.

It can be seen that the challenging work is still unsolved although research on the topic of learning Bayesian network structures in the past two decades has achieved much progress. It is intractable for basic methods to learn large Bayesian network structures. On the other hand, some deficiencies still exist in advanced approaches for learning large Bayesian network structures from a small data set.

2.4 Summary

Many researchers have exerted their effort on the topics that I will investigate in this dissertation. They have proposed different representations to better capture characteristics of decision problems, and developed various kinds of approaches for learning Bayesian network structures from data. Based on this research work, I have a solid foundation to study my research topics and carry out continuous research work.

This chapter gives a concise review of a variety of topics related to decision making, multiagent decision systems and Bayesian network structure learning. Some terms and concepts are introduced for a better understanding of the discussion in subsequent chapters of this dissertation.

[This page intentionally left blank]

3 Model Representation

This chapter provides the formal definition of Multiply Sectioned Influence Diagrams (MSID) and Hyper Relevance Graph (HRG), followed by an intuitive description, and then makes use of an example to illustrate the utilization of MSID and HRG in representing a multiagent decision problem, that is a distributed decision problem involving multiple agents. This chapter introduces the important properties of MSID and HRG, and describes the model construction process as well.

3.1 Agency and Influence Diagrams

A good representation or model is a channel which should convey full information about a problem domain in an intuitive and logical manner. It is required to capture the characters of domain scenario. In some decision domains, agency is implicated in some graphical decision models such as the mature representation language of influence diagrams. It can be inferred that some common abstract foundations should exist between agency and graphical decision models although they have been studied separately in two parallel fields for a long time.

Intelligent agent has been a key concept in both AI and the main stream of computer science. The agent-based technology plays an important role in software engineering. In the past decades, the theory and application of agents have been well developed. As for agency theory, many models or architectures for characterizing agents have been

proposed. From an ideally theoretical and more practical perspective, the Belief-Desire-Intention (BDI) agent model is widely accepted and implemented in many fields (Rao & Georgeff 1995; Wooldridge & Jennings 1995; Wooldridge 2002).

In the field of decision science, some concepts, such as rational decision making and normative decision systems, are always discussed. Influence diagram has been a compact graphical model for representing decision problems under uncertainty. It guides how to arrive at an optimal decision with respect to the preference of a decision maker and the states of an uncertain environment. Accordingly, it is possible and reasonable to model agents as influence diagrams in order to solve decision problems in the agent-based system. In fact, the logical framework residing in both BDI agent and influence diagrams does match with each other because both of them are based on the essence of rational persons and want to solve decision problems with a rational attitude.

A BDI model shows the information attitude (Belief) and pro-attitude (Desire and Intention) of an agent. In detail, beliefs are related to the information on which an agent thinks about the world it occupies. This information not only includes the agent's knowledge on what is the world, but also includes its attitude on what happens in the world. Intentions can be seen as states of affairs that an agent has committed to bring about. States of affairs are updated with agents' actions or performances. Desires or goals are objectives that an agent wants to realize while objectives reflect agents' preferences.

In an influence diagram, chance nodes describe states of the environment which intelligent agents are involved in. Hence, it is related to agents' knowledge and represents agents' beliefs. Decision nodes provide decision options and indicate the

sequential actions that an agent will perform. Finally, value nodes represent the criterion against which different outcomes are evaluated. Thus utility nodes represent agents' preferences and determine the final optimal path that an agent should follow.

The above description shows that there are some corresponding implications between variables or nodes (Henceforth, I shall make no distinction between variables and nodes.) in influence diagrams and properties of intelligent agents. It shows again that an influence diagram is a desirable alternative to model an agent-based decision problem under uncertainty.

3.2 Multiply Sectioned Influence Diagrams and Hyper Relevance Graph

Influence diagram is a well-designed probabilistic graphical model for representing an uncertain decision problem. However, it is only for a single agent paradigm so that it does not have a powerful ability for representing a distributed decision problem involving multiple agents, also called a multiagent decision problem, such as the following complex decision problem in the medical domain.

[Medical Domain]

Severe Acute Respiratory Syndrome (SARS) is a serious infectious disease that could potentially develop into an epidemic or even an endemic. Its outbreak causes unexpected loss everywhere in the world. Its uncertain and various sources have frustrated the medical community and policy makers. Beyond all doubt, it needs the collaborative effort of multiple nations concerning their own local as well as global benefits.

This is one decision problem of policy design for controlling the SARS to decrease both social and economical loss in the world. The decision problem involves many nations or communities, even those without the outbreak of the SARS. These involved nations are collectively seeking good solutions to control the SARS in the whole community in order to avoid more loss. They would share some valuable information such as some available SARS reports. However, to protect privacy, the involved nations could not release all of their own information in the cooperation. Summarily, an individual privacy protection nation seeks its best decision while it has a full, correct and consistent observation in a situation. They cooperatively solve a decision problem in a distributed way. This kind of decision problems is what I call multiagent decision problems. Intuitively, a multiagent decision problem always relates to a large knowledge domain since it concerns a complex decision problem involving multiple agents. The potential approach for solving multiagent decision problems should extend traditional graphical decision models as well as exploit features of multiple agents.

The aforementioned decision scenario could not be solved using traditional methods of influence diagrams and extensions of the representation which have been reviewed in Chapter 2, such as the DID, VBS, MLID, TDID, LIMID, UID, SID, and decision networks. Two major reasons are stated. Firstly, those formalisms emphasize the single agent based decision problem. Thus they do not consider cooperation among multiple agents and properties of multiple agents such as agents' privacy and organizational relationships. Secondly, those graphical models are not scalable. The multiagent decision problem is a large and complex decision problem so that it is hard to concisely represent a large number of elements and their relationships within a single decision model.

The most relevant work to my topic is the representation of MSBN. As an extended model from Bayesian networks, an MSBN is a set of local Bayesian networks connecting through the linkage of d-sepset. In my work, the concept of d-sepset is extended to represent an intersection among adjacent subnets such as local Bayesian networks, and not just between each pair of subnets. Furthermore, for a concise representation of intricate interactions among adjacent agents, an irreducible d-sepset is defined. An irreducible d-sepset is a set of d-sepnodes which encode the necessary information. The necessary information includes the requisite information and the supporting information for agents' decision making. The requisite information is encoded either in requisite probability nodes or in requisite observation nodes and is required for agents' decision making. Both of these required nodes belong to d-sepsets. The supporting information, also encoded in d-sepnodes, is not required for agents' decision making; however, this information supports their decision making. For example, an agent may give the complementary information to its adjacent agents who cannot access this information by themselves. Hence an irreducible d-sepset not only provides a concise representation, but also indicates the most economical information for supporting decision making in multiagent decision problems.

The representation of a multiagent decision problem requires a new graphical decision model extending from the basic decision models such as influence diagrams. On the other hand, an MSBN provides a coherent framework for multiple agents reasoning although it does not address the decision making problem. This work provides a foundation to define an MSID based on both the influence diagrams and the MSBN formalism.

3.2.1 Multiply Sectioned Influence Diagrams (MSID)

Definition 3.1: An MSID I is a set of influence diagrams I_j such that each diagram I_j represents an agent j and the shared chance nodes among adjacent diagrams $I_i, I_j, \cdots, I_k$ comprise an irreducible d-sepset $S_{ij\cdots k}$.

For instance, for two agents i and j, the MSID is denoted by $I = I_i \cup I_j$. The d-sepset between influence diagram I_i and influence diagram I_j is denoted by S_{ij} $(i \neq j)$. It can be seen that the MSID $I = \bigcup_j I_j$ involves two concepts: an agent-based influence diagram I_j and an irreducible d-sepset $S_{ij\cdots k}$.

In brief, an MSID is a set of local influence diagrams that reflect an individual agent's knowledge, actions and preference. Agents communicate with each other through a d-sepset that indicates public information. Except for the shared information in the d-sepset, other information is protected in each local influence diagram. The shared information indicates agents' beliefs on their environment. It may be affected by agents' actions. Sometimes the information can also exert its influence on agents' behavior. To make a consistent representation in an MSID, Bayesian networks representing agents that are only information collecting entities without actions or behaviors are called degenerated influence diagrams. Consequently, an MSID is a hybrid probabilistic graphical model that could be a combination of Bayesian networks and influence diagrams. In this sense, an MSBN is a special case of MSID in which all agents do not make decisions and are represented by local Bayesian networks.

The definition of MSID implies that an MSID should comply with three constraints: DAG structure, d-sepset of agent interface and irreducible d-sepset. The first constraint

prohibits any directed cycle in an MSID. This is a graphical structure requirement in probabilistic decision models following logical thinking in decision analysis (Robert & Terry 2001). An MSID is a sizable decision model characterizing a large and complex knowledge domain. Multiple agents have consistent thinking including causal relationships on the same observation. Although their observations and decisions are interleaved with each other, a sequential order exists in their interaction. The second constraint requires that nodes shared by local influence diagrams should be d-sepnodes, which means that all parents of a public node should be included in the same local influence diagram. This constraint protects agent's privacy and indicates that an agent can decide its actions using local information only when the information in the d-sepset is known. The final constraint is to ensure the compactness of MSID, and to remove any redundant information. The information, including the requisite information and the supporting information, encoded in a d-sepset between two agents, is just what the two agents are willing to share. Unnecessary information would complicate a model representation and confound the necessary information.

Besides the qualitative properties of MSID on DAG structure, an MSID also represents a set of joint probability distributions encoded in an individual influence diagram. This is to say $P = \prod_j P_j = \prod_j (\prod_{x \in C_j \cup D_j} P(x|\pi(x)))$ where $\prod_{x \in C_j \cup D_j} P(x|\pi(x))$ is a collection of conditional probability distributions for node x (belonging to either the set of decision nodes D_j or the set of chance nodes C_j) given its parents $\pi(x)$. For node x in d-sepset, exactly one of its occurrences (in an I_j containing $\{x\} \cup \pi(x)$) is assigned $\prod_{x \in I_j} P(x|\pi(x))$ while each occurrence in other local influence diagrams is assigned a uniform potential,

such as 1. In this way, an MSID is able to ensure a consistent representation of joint probability distributions.

In the aforementioned SARS decision problem, each nation or community can be considered as an individual agent that is modeled as a local influence diagram. Figure 3.1 shows the MSID that represents this multiagent decision problem concerning the SARS control. The MSID is defined as $I = \bigcup_{j=1,2,3} I_j$. In this MSID, there are three local influence diagrams $(I_j, j = 1,2,3)$ that represent three agents $(A_j, j = 1,2,3)$ respectively (Agent A_1: Nation 1; Agent A_2: Nation 2 and Agent A_3: Nation 3). Each local influence diagram describes an individual agent's knowledge that formalizes an agent's judgments on the situation. The three agents share some public information represented as grey color nodes $\{a,b,c\}$ such as the WHO (World Health Organization) report on the SARS, status of transmission customers and the SARS report from a certain nation (not all involved nations would like to release this information). The information $\{a,b,c\}$ indicates agents' common beliefs on the same observation. Among this information, the information $\{a,b\}$ is not affected by agents' decisions while the information c is affected by agent A_1's decision d_2 . Furthermore, their privacy, like the SARS report from its own nation and states of hospital facilities (some nations have to hide these information for their own benefit), is protected in local influence diagrams such as g in A_1, k in A_2, and l in A_3. The actions of agents are represented as decision nodes in corresponding local influence diagrams such as temperature checking at entries (d_1 in I_1), home quarantine policy (d_2 in I_1), experiments of SARS virus (d_1 in I_2) and overseas tour policy (d_1 in I_3). The benefit for each agent is defined as utility functions represented by utility nodes in the MSID. In Figure 3.1, some agents can make decisions independently with the known information in the d-

sepset. For example, with the known information c between I_1 and I_3, A_3 will make the decision d_1 without considering any effect of A_1's actions.

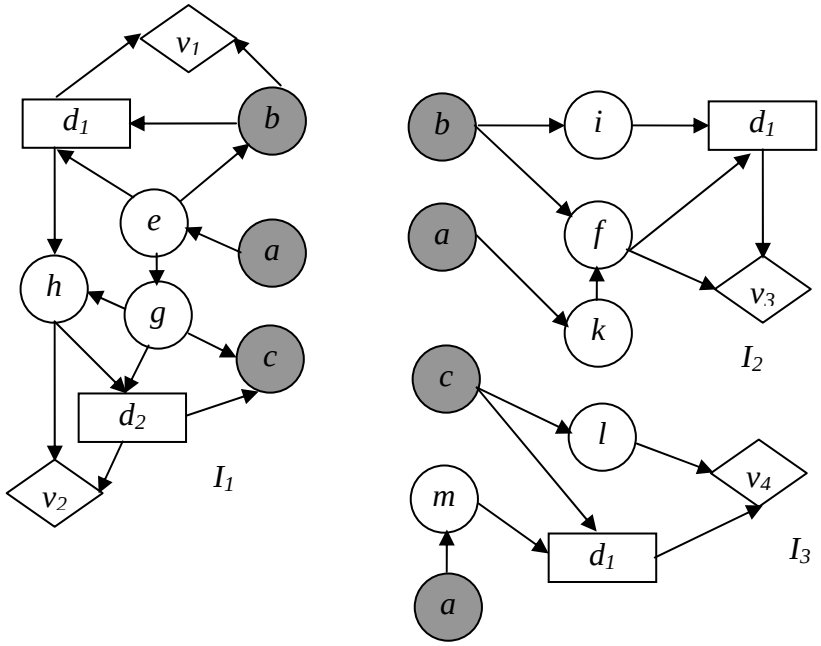


Figure 3.1: An MSID for the SARS Control

3.2.2 Hyper Relevance Graph (HRG)

An MSID has the ability of representing a multiagent decision problem concerning the six characters of this decision scenario I discussed in the previous chapter. However, it does not explicitly describe the property of organizational relationships among multiple agents such as the required information for decision making. Moreover, in an uncertain and dynamic environment, agents have to be regrouped to adapt to a new surrounding. In this process, it is the organizational relationship that guides multiple agents to construct an updated multiagent system. In Zambonelli's work (Zambonelli *et al.* 2001), five types of organizational relationships (*Control, Peer*, *Benevolence*, *Dependency* and *Ownership*) between agents were discussed. However, from the viewpoint of decision making and information flow in a multiagent system, these

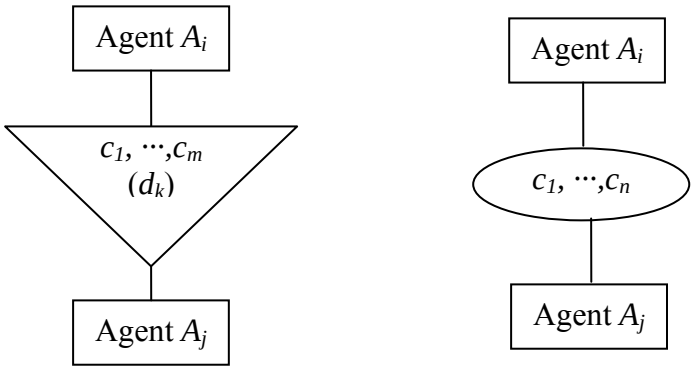
relationships can be classified into only two types: *Control* and *Communication*. Hence, the concept of Hyper Relevance Graph (HRG) is introduced to represent the information implicated in the relationship.

Definition 3.2: A Hyper Relevance Graph (HRG) H is composed of three types of nodes: rectangular, triangular and oval nodes, and arcs between them. A rectangular node denotes an individual agent A_j associated with local influence diagram I_j in an MSID. A triangular node denotes the shared information C in d-sepset S that is required for an agent's decision D. An oval node denotes the shared information C in d-sepset S that is not required by any agent's decision D.

With the elements in an HRG, two kinds of basic relevance graphs, called Control Relevance Graph and Communication Relevance Graph, can be built as shown in Figure 3.2. Each is associated with a function that implies an organizational relationship in a multiagent system.

1. Control Relevance Graph is associated with a function $\mathrm{Re}\,q(A_i, A_j, d_k) = \{c_1, \cdots, c_m\}$, which indicates that the set of requisite information $\{c_1, \cdots, c_m\}$ agent A_i provides is required for agent A_j's decision d_k. It signifies the *Control* relationship.

2. Communication Relevance Graph is associated with a function $Sup(A_i, A_j) = \{c_1, \cdots, c_n\}$ which indicates that the set of supporting information $\{c_1, \cdots, c_n\}$ flowing between agent A_i and agent A_j supports their decision making; however, this information is not required for any of their decisions. It signifies the *Communication* relationship.



a) Control Relevance Graph

b) Communication Relevance Graph

Figure 3.2: Two Basic Relevance Graphs

In a control relevance graph, the set of requisite information $\{c_1, \cdots, c_m\}$ flows from agent A_i to agent A_j by the direction of the triangular node in Figure 3.2(a). This information is required for the optimal decision making in agent A_j. It is a subset of both requisite observation nodes (*RO*) and requisite probability nodes (*RP*) for decision d_k in influence diagram I_j. The requisite observation nodes for decision d_k are those which can be observed before decision d_k that might be worth observing so as to evaluate the decision problem starting with decision d_k; while, the requisite probability nodes for decision d_k are those for which conditional probability distributions might be needed to evaluate the decision problem starting with decision d_k. Hence, in a control relevance graph, it is said that agent A_i controls agent A_j's decision d_k with the information $\{c_1, \cdots, c_m\}$; but not vice versa. In a communication relevance graph, the supporting information $\{c_1, \cdots, c_n\}$ indicated in the oval node implies common beliefs between agent A_i and agent A_j; however, this information is not required for their decision making.

Hence, an HRG can be driven and built based on the structural representation of an

MSID and the organizational relationships in the problem domain. For example, the HRG based on the MSID in Figure 3.1 is shown in Figure 3.3. The HRG in Figure 3.3 clearly shows that agent A_1 controls agent A_3's decision d_1 with the information c, representing that the nation 1's SARS report affects overseas tour policies in nation 3. Furthermore, the information c is affected by agent A_1's decision d_2. Thus it is said that agent A_1's decision d_2 exerts an influence on agent A_3's decision d_1. The HRG also shows that agent A_2 controls agent A_1's decision d_1 with the information b, representing that temperature checking decisions at nation 1's border crossing are affected by the status of citizens from nation 2. In this case, agent A_2 affects agent A_1's decision d_1 by its judgment that is not affected by agent A_2's decision d_1. Besides the public information $\{b,c\}$, these nations share the public information a that conveys the indication of the WHO report on the SARS. However, this information is not the requisite one for agents' decision making. In other words, the HRG gives a full picture of the relationships among the three nations modeled as agents A_1, A_2, and A_3.

The HRG in Figure 3.3 indicates that multiple nations are able to arrive at a good decision on the SARS control with some cooperation, even without a central control from the WHO.

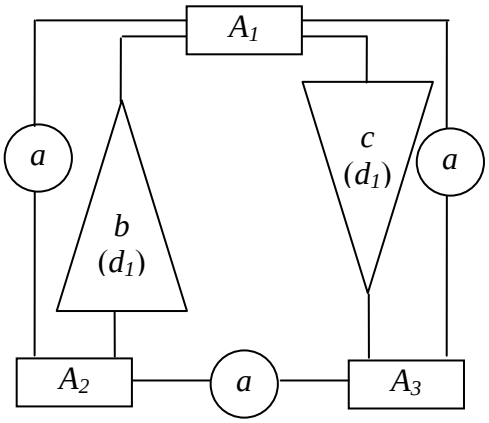


Figure 3.3: The HRG for the MSID in Figure 3.1

3.3 Model Construction

Model construction is a model refinement process with the aim to characterize decision scenarios in a compact and accurate model. It requires some guidelines to facilitate the building of a reliable and exact model.

3.3.1 MSID and HRG

An MSID represents both the knowledge of the environment and the properties of intelligent agents while an HRG characterizes the organizational relationships in the multiagent system. Evolving over time, intelligent agents are regrouped according to new relationships represented in the HRG. Consequently, the MSID should be rebuilt based on the updated HRG.

Triangular and oval nodes in an HRG have the exact chance nodes that can be obtained from a d-sepset in an MSID. Elements in the HRG could be driven from a well-built MSID through either *Decision Bayes-ball* or *refined Decision Bayes-ball* procedure. These two procedures could obtain the required chance nodes, including requisite

observation nodes and requisite probability nodes, for corresponding decision nodes. These required chance nodes and the corresponding decision nodes are elements in triangular nodes of the HRG. Except for these chance nodes, other nodes in the d-sepset belong to elements in oval nodes. Extracted from a knowledge domain, the organizational relationships related with two functions are confirmed in the HRG. On another aspect, when an HRG is reorganized because of dynamic organizational relationships in a multiagent system, it will lead to a reconstruction of the MSID. For instance, it is to refine (decreasing or increasing) chance nodes in the d-sepset and to reorient their relevance (adding or removing arcs) to decision nodes. Accordingly, the MSID and HRG are regulated with each other to arrive at an elaborate framework.

A compact MSID requires that a d-sepset should be irreducible, which depends on the relationship between intelligent agents. The information flowing through the d-sepset is just what intelligent agents want according to their organizational relationships. The necessary information is composed of the requisite information and the supporting information which indicate the control relationship and the communication relationship between agents respectively. An HRG is the exact model which could encompass all the implications concerning the irreducibility of d-sepset. Hence it is able to help the constraint (irreducibility of d-sepset) checking and the reconstruction of an MSID.

3.3.2 Modeling Process

The MSID, together with the HRG, provides the basic elements to represent a large, complex and distributed decision problem involving multiple agents. These two models could be built from a certain decision domain in a simultaneous or a sequential way.

1) Simultaneous Approach: It is to build the MSID and HRG from domain knowledge at the same time as shown in Figure 3.4 (a). In this way, both of the initial models are built from decision scenarios directly. It is inevitable that the model construction process incorporates much subjective consideration from domain experts. After that, the two models go through the verification process and do a further refinement. Finally, the refined MSID and HRG is output when they are considered and satisfied by domain experts. The final model should be valid and represent the real decision scenario accurately.

2) Sequential Approach: It is to build the MSID and HRG from domain knowledge in a sequential way as shown in Figure 3.4 (b). An MSID is constructed directly from the domain knowledge while an HRG is produced based on the built MSID. In this way, it avoids much inconsistency between the initial two models. Finally, after model verification and further refinement, the satisfied models of the MSID and HRG are generated.

It can be seen that model construction is an iterative refinement process that requires both an objective step of model verification and a subjective step of model elicitation from domain knowledge. In Chapter 4, model verification will be discussed in length. Since model elicitation from domain knowledge largely depends on the subjective thinking on the domain, it is hard to figure out a formal elicitation approach. However, if a model is elicited from data in the domain, there exist many mature methods. I will focus on this topic in Chapter 7.

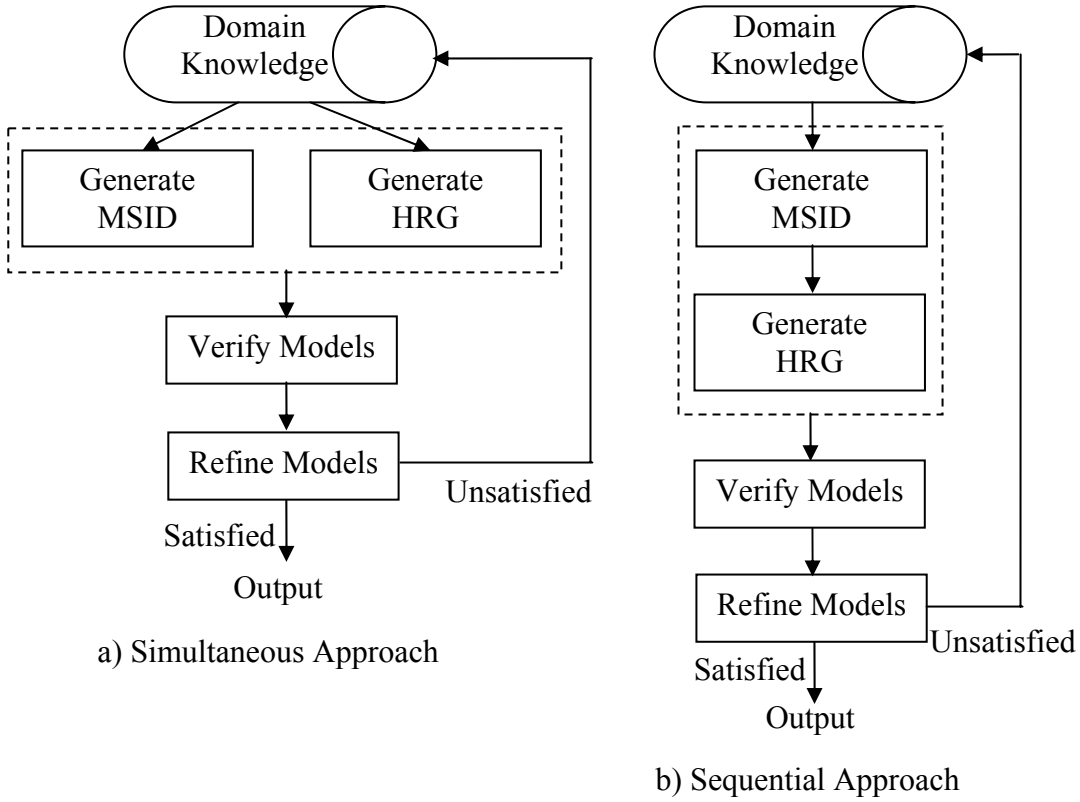


Figure 3.4: Modeling Approaches

3.4 An Application

Multiagent decision problems are rather common and essential in our daily lives. They are dominant in some specific problem domains such as military defense, air control, medicine and so on. The MSID and HRG together provide a flexible, compact and distributed framework for representing these kinds of decision problem. Besides the aforementioned medical domain from which I have developed the corresponding decision models, the maritime security domain also attracts my attention. In this section, I will investigate one decision scenario in this domain and build decision models of MSID and HRG that will be referred to in Chapter 4.

3.4.1 Case Description

A multiagent decision problem could be formulated as a typical decision scenario in the maritime security domain:

[Maritime Security Domain]

The issue of coast safety has long been a focus in the maritime security domain. The task of an information collection center on the land is to ensure the coast safety and ships' normal activities along the coastline as well as to monitor the status of entry into a harbor. In a long and broad coast, all kinds of ships may leave and enter the coast at any time. For example, a navy ship may go on its rounds on patrols or to invigilate reported incidents in the sea, and a commercial ship may be carrying tourists for sightseeing or other pleasure activities. As they share a common space, these ships have to cooperate with each other in order to achieve their respective missions and goals. An information collection center conveys some public information, such as the weather information and the report on the condition and status of sea, to a navy ship and a commercial ship. A navy ship may report back to headquarters after it has observed any unusual happenings out there, such as the abnormal status of sea and the unexpected performance of a commercial ship. In the follow-up, the navy ship may decide to intercept the ship if the navy ship is in good function and in a favorable position corresponding to this ship. A commercial ship, on the other hand, must focus on its navigation and take appropriate actions in reaction to the conditions of sea. For example, a commercial ship may decide to seek help from the authorities when it feels that its position is being endangered concerning the sea condition. This action

transmits the information of its behavior to a navy ship. Also, the commercial ship may adjust its work type, such as changing a tour line, when it does a full evaluation of its mechanism function. While all these activities are going on, public information should be accessible to them when operating in this area.

The above is an example encoded with a distributed decision problem involving multiple agents (ships). There are two kinds of ships, namely a navy ship and a commercial ship, and an information collection center. All of them share some common information such as the weather information. Meanwhile, a commercial ship shares the information of its performance with a navy ship. However, for privacy protection, both a commercial ship and a navy ship would not release publicly the information of their function and their position in the sea. On another aspect, agents have their individual objectives, but their decisions and observations are interleaved within multiple agents. For instance, a commercial ship needs the information of the sea condition to make the decision on seeking help; and a navy ship needs to be informed of the performance of a commercial ship before it takes the action of doing a report. Accordingly this is a typical multiagent decision problem. Decision models of MSID and HRG have the ability and capability to represent this decision problem.

3.4.2 Model Formulation

According to the above decision scenario in the maritime security domain, the Information Collection Center (*ICC*), Navy Ship (*NS*) and Commercial Ship (*CS*) are considered as an individual agent, and organized into a multiagent system. Each of them is described as an (degenerated) influence diagram (I_1, I_2 and I_3) respectively in an MSID. Some of the variables representing uncertainty and decisions are tabulated as shown in Table 3.1.

After variables are identified in the problem description, models of MSID and HRG can be built simultaneously as shown in Figures 3.5 and 3.6 respectively. As I discussed earlier, the HRG can be obtained in two ways. One is to produce the HRG according to the implication of MSID. After that, the HRG is verified with the domain knowledge. The other method is to produce the HRG with regard to the problem description, then to verify it with respect to a constructed MSID. Here I adopt the second approach and will discuss its verification in Chapter 4.

Table 3.1: Variable Identification of Agents *ICC*, *NS* and *CS*

Information Collection Center in the land (*ICC*)

a) Uncertainty

Uncertainty	Weather (***w***)	Condition of Sea (***c***)	Status of Entry (***e***)	Status of Sea (***s***)	Other Information (***i***)
Outcomes	Good	Good	Blocked	Normal	Authentic
	Bad	Bad	Unblocked	Abnormal	Inauthentic

Navy Ship (*NS*)

a) Uncertainty

Uncertainty	Weather (***w***)	Condition of Region (***cr***)	Position of Self (***ps₂***)	Performance of CS (***pp***)	Function (***f₂***)
Outcomes	Good	Good	Favorable	Expected	Good
	Bad	Bad	Unfavorable	Unexpected	Bad
Uncertainty	Status of Sea (***s***)				
Outcomes	Normal				
	Abnormal				

b) Decision

Decision	Intervene (***in***)	Report (***re***)
Alternatives	Yes	Yes
	No	No

c) Utility Value

Utility	***v₂***	
Attributes	*in*	*re*
	f_2	*pp*

Commercial Ship (*CS*)

a) Uncertainty

Uncertainty	Weather (***w***)	Condition of Sea (***c***)	Position of Self (***ps₁***)	Behavior of Self (***pp***)	Function (***f₁***)
Outcomes	Good	Normal	Danger	Normal	Good
	Bad	Abnormal	Safety	Abnormal	Bad

b) Decision

Decision	Work Type (***wt***)	Ask Help (***ah***)
Alternatives	Old	Yes
	New	No

c) Utility Value

Utility	***v₁***	
Attributes	*c*	f_1
	wt	

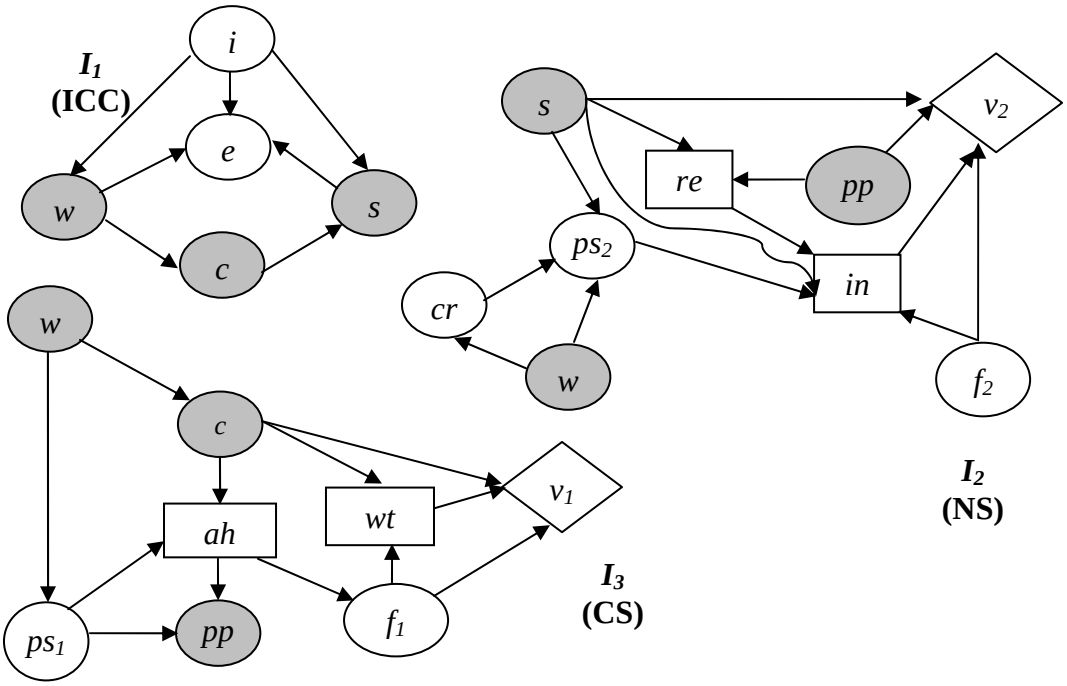


Figure 3.5: An MSID for Agents *ICC*, *CS* and *NS*

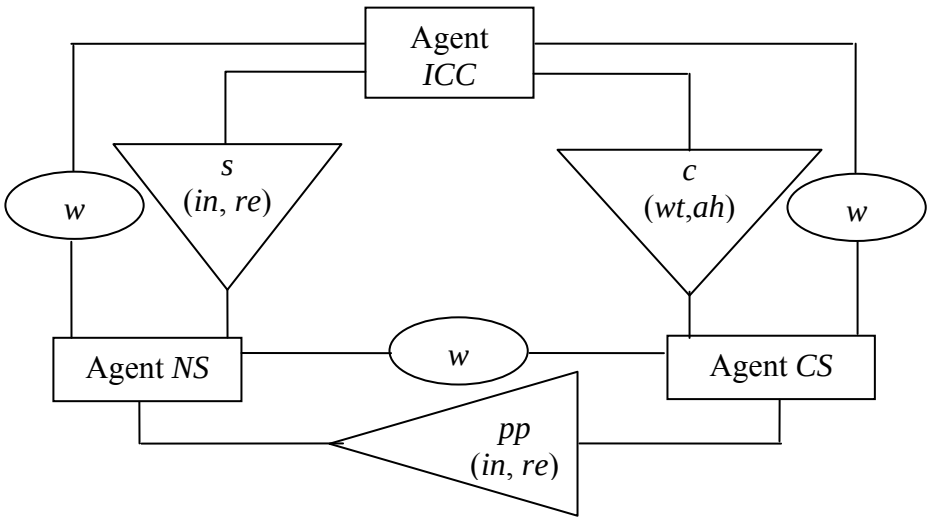


Figure 3.6: An HRG for Agents *ICC*, *CS* and *NS*

In Figure 3.5, agent *ICC* is modeled as a degenerated influence diagram I_1 in the MSID. It provides public information to agents *NS* and *CS*, and does not make any decision. Agents *NS* and *CS* take some actions based on their observations. Agents' privacy, like agents' functions and position, are protected in corresponding local influence diagrams. From the HRG in Figure 3.6, it is noticed that agents *ICC*, *NS* and *CS* share the public information w. This public information affects agents' observations; however, this information does not control agents' decisions. On the other hand, some information from agent *ICC* controls decisions of agents *NS* and *CS*. For example, agent *ICC* controls agent *NS*'s decisions $\{in, re\}$ with the information s and controls agent *CS*'s decisions $\{ah, wt\}$ with the information c. At the same time, it is noticed that the information $\{c, s\}$ is not affected by agent *ICC*'s decisions because agent *ICC* is only an information collection center. In contrast, agent *CS* controls agent *NS*'s decisions $\{re, in\}$ with the information pp while this information is affected by agent *CS*'s decision ah. It indicates that agent *CS*'s decision ah exerts an influence on agent *NS*'s decisions $\{re, in\}$, which just matches the domain knowledge. Thus it can be seen that the HRG explicitly describes the organization relationships among multiple agents through quantifying the information support for decision making.

From the representation of MSID and HRG, it seems that this framework is flexible and scalable since the model is designed in a distributed manner. It could solve a decision problem in the changing world. For example, in the maritime security domain, any new ship that enters the invigilated region could be modeled as a new local influence diagram that would be added into the existing MSID without damaging other components. On the other hand, currently, the model representation of MSID lacks the

ability to handle game problems since it requires a sequential order between decisions and observations among multiple agents.

3.5 Summary

Multiagent decision problems are difficult to be addressed by current representation languages in probabilistic graphical models so that new methodologies need to be proposed. This chapter provides a formal definition of the MSID and HRG which could be utilized to represent a distributed decision problem involving multiple agents in an uncertain environment. At the same time, this chapter describes many opportunities of real world applications such as the policy design for avoiding more loss due to the SARS in 2003 and ship models in the maritime security domain.

The content in this chapter serves as a basis for the discussion in the following chapters.

[This page intentionally left blank]

4 Model Verification

Multiply Sectioned Influence Diagrams (MSID), together with Hyper Relevance Graph (HRG), is a distributed and cooperative probabilistic graphical model to represent multiagent decision problems. As shown in the definitions in Chapter 3, the representations of MSID and HRG should satisfy three constraints: DAG structure, d-sepset of agent interface and irreducible d-sepset. Thus model construction involves one important process of model verification which will be illustrated in this chapter.

4.1 The Introduction

Model verification is an old but not obsolete topic in the engineering area and is an essential process in practical applications (Cousot 2005). The aim of model verification is to ensure a valid and reliable model. In general, model validation and verification can be conducted through expert evaluation, data evaluation, and some model consistency checking, such as finding and resolving conflicts by forming a consistency matrix. In expert systems or knowledge based systems, model verification always depends on both the knowledge of domain experts and the skills of model builders. It is desirable that a valid and reliable model be generated from the problem domain and this model can characterize the domain knowledge accurately and conformably.

In Chapter 3, I have proposed graphical decision models of Multiply Sectioned Influence Diagrams (MSID) and Hyper Relevance Graph (HRG). An MSID represents a decision problem involving multiple agents in a distributed and flexible fashion while an HRG encodes the organizational relationships in the multiagent system. From the definition, it can be seen that the MSID and HRG should obey the three constraints such as the DAG structure, d-sepset of agent interface, and irreducible d-sepset.

Both distributed decision making and compact model representation require that the MSID and HRG observe a set of constraints. These constraints need to be verified before model evaluation with the aim to avoid "garbage-in-garbage-out". In addition, the verification process is a cooperative task for multiple agents whose knowledge is encoded into local influence diagrams individually. Since agents are autonomous and built by different vendors, agents' privacy should be protected. Hence, verification of these constraints raises a challenge.

In the field of decision analysis, a traditional approach to verify a graphical model involves a detailed study on its graphical structure, like a graphical verification on DAG in an MSBN (Xiang 1998; Xiang & Chen 2002). The method requires a good knowledge on graph theory that always frustrates the novices. To relieve a knowledge repertory with graph theory, a symbolic method is proposed to deal with the model verification. The method considers the verification in an algebraic view that characterizes a factorization joint probability in an MSID. Furthermore, to enrich model verification, the issue of verification failure is investigated with some useful comments on model correction.

4.2 Foundation of Symbolic Verification

Basically, like influence diagrams, an MSID represents decision problems from two points of view. One is a graphical structure G, which is a DAG; the other is a Joint Probability Distribution (JPD) P, which is a set of multiplication of the factorization product of conditional probabilities of nodes given their parents, like a recursive factorization in Bayesian networks (Pearl 1998; Shachter *et al.* 1990; Wong & Wu 2002) ,

i.e. $P = \prod_j P_j = \prod_j (\prod_{x \in C_j \cup D_j} P(x|\pi(x)))$

where C_j and D_j are sets of chance and decision nodes in local influence diagrams I_j.

Essentially, the two viewpoints on an MSID display the same knowledge about a problem domain and are consistent with each other. Taking advantage of this situation, it is reasonable and possible to verify the structure by its corresponding part, like JPD. Hence a new approach, called symbolic verification, deserves to be studied.

From a graphical perspective, an MSID is a set of local DAG structures (for influence diagrams) that defines a DAG globally; however, from a symbolic view, it is a set of individual JPD P_{I_j} associated with local influence diagrams. Furthermore, with the form of conditional probability, an individual JPD P_{I_j} can be rewritten as

$$P_{I_i} = \prod_{j=1}^{n} p(x_j|\pi(x_j)) = \prod_{j=1}^{n} \frac{p(x_j, \pi(x_j))}{p(\pi(x_j))} \quad (4.1)$$

where x is a chance node or a decision node in local influence diagrams.

It is evident that the denominator of last term in Equation (4.1) implies parents of nodes while the numerator includes a family of nodes. Collectively, they provide us with the parental information on nodes in local influence diagrams, which is consistent with the structural information. Conveniently, Equation (4.1) is called the algebraic description of local influence diagrams which provides a foundation to a symbolic verification.

4.3 Symbolic Verification of DAG structure

The first constraint, DAG structure, means that there should be no directed cyclic paths in an MSID. The verification task is to ensure a global DAG structure in an MSID although local influence diagrams are identified as a true DAG. The symbolic method takes some basic operations based on the algebraic description of local models to test the DAG structure in an MSID,

Theorem 4.1: An MSID without value nodes has the same DAG property as the original MSID.

Proof. An MSID includes three types of nodes: chance nodes, decision nodes and value nodes. Since a value node is a sink node without outgoing arcs in an MSID, it cannot belong to any part of a directed cyclic path. Removing the value node ensures the DAG property in an MSID. ▌

With Theorem 4.1, in the process of DAG structure verification, value nodes in an MSID can be removed safely. After removing value nodes and considering decision nodes as chance nodes in an MSID, the MSID has the same structure as an MSBN

which is a set of local Bayesian networks. Here, a symbolic verification method, instead of graphical methods, will be carried out in an MSBN.

4.3.1 Basic Concepts

Verification of DAG structure involves a global testing of directed cyclic paths in an MSBN. It is clear that if a directed cyclic path exists in an MSBN, the path must include d-sepnodes shared by adjacent subnets. This path is called a D-Cycle and is defined as follows:

Definition 4.1: A D-Cycle (DC) is a cyclic directed path that includes at least one d-sepnode globally.

Simarly, a directed path in a local subnet is called a D-Path (DP). It is likely to be a part of a D-Cycle. The formal definition is as follows:

Definition 4.2: A D-Path (DP) is a directed path in a local subnet in which both the source and sink nodes are d-sepnodes.

In each subnet, concerning the relationships between pairs of d-sepnodes, two types of d-sepnodes can be classified as follows.

Definition 4.3: A D-Head (DH) set is composed of d-sepnodes that have out-going arcs in I_i. I use the notation $DH = \{\overleftarrow{a}, \overleftarrow{c} \cdots, \overleftarrow{y}\}$ to denote the D-Head set.

Definition 4.4: A D-Tail (DT) set is composed of d-sepnodes that have incident arcs in I_i. I use the notation $DT = \{\underline{b}, \underline{d} \cdots, \underline{z}\}$ to denote the D-Tail set.

When one d-sepnode belongs to both a DH and a DT, it is called a symmetric node. The corresponding nodes in the DH and the DT are called a *symmetric type* of this d-

sepnode, denoted by $\bar{m}$ and $\underline{m}$. Thus, in each subnet, elements from a DH to a DT respectively compose one DP. These DPs are formed into a set, denoted as

$$DPs=\left\langle (\bar{a}\bullet\underline{b})\otimes(\bar{c}\bullet\underline{d})\otimes\cdots\otimes(\bar{y}\bullet\underline{z})\middle|\bar{a},\bar{c},\cdots,\bar{y}\in DH,\underline{b},\underline{d},\cdots,\underline{z}\in DT\right\rangle$$

where the two operators $\bullet$ and $\otimes$ are defined as follows:

Denotation 4.1: Operator $\bullet$ denotes that there **must be** a DP between an element in a DH and an element in a DT.

Denotation 4.2: Operator $\otimes$ denotes that there **may be** another DP whose nodes are from different DPs.

Operator $\bullet$ has a higher priority compared with operator $\otimes$ and is denoted by $\bullet\succ\otimes$.

4.3.2 *DPs* with Algebraic Description

From Equation (4.1), the JPD in each subnet can be expressed as follows:

$$P_{I_i}=\prod_{j=1}^{n}\frac{p(x_j,\pi(x_j))}{p(\pi(x_j))}. \tag{4.2}$$

Obviously, the fact that $p(\pi(x_j))$ equals to 1 implies node x_j is a root node in I_i.

Based on the definition of operators, firstly, some possible DPs could be described in each subnet. From Equation (4.1), $p(x_j|\pi(x_j))$ indicates that there is a directed path from $\pi(x_j)$ to x_j. After that, based on Equation (4.2) and some operations (defined later), DPs could be obtained for each subnet.

The example network as shown in Figure 4.1 is used to illustrate some operations. This example has the same structure as that in Xiang's work (Xiang 1998) and its DAG structure has been well discussed and analyzed by graphical methods.

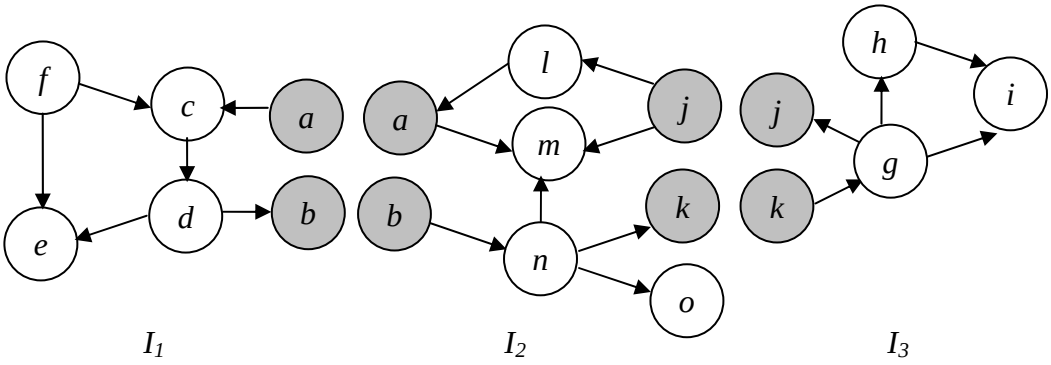


Figure 4.1: An Example Network

Considering the three DAGs (I_1, I_2 and I_3) in Figure 4.1, each I_i is a subnet in an MSBN. Using Equation (4.2), the JPD of each subnet is written as follows:

$$P_{I_1} = \frac{p(a)}{1} \times \frac{p(c,a,f)}{p(a,f)} \times \frac{p(f)}{1} \times \frac{p(e,f,d)}{p(f,d)} \times \frac{p(d,c)}{p(c)} \times \frac{p(b,d)}{p(d)} \tag{4.3}$$

$$P_{I_2} = \frac{p(j)}{1} \times \frac{p(l,j)}{p(j)} \times \frac{p(a,l)}{p(l)} \times \frac{p(m,j,a,n)}{p(j,a,n)} \times \frac{p(n,b)}{p(b)} \times \frac{p(b)}{1} \times \frac{p(o,n)}{p(n)} \times \frac{p(k,n)}{p(n)} \tag{4.4}$$

$$P_{I_3} = \frac{p(j,g)}{p(g)} \times \frac{p(k)}{1} \times \frac{p(g,k)}{p(k)} \times \frac{p(h,g)}{p(g)} \times \frac{p(i,h,g)}{p(h,g)} \tag{4.5}$$

The DH and DT sets for each subnet I_i are as follows:

$DH(I_1) = \{\bar{a}\},\ DT(I_1) = \{\underline{b}\};\ DH(I_2) = \{\bar{a}, \bar{b}, \bar{j}\},\ DT(I_2) = \{\underline{a}, \underline{k}\};\ DH(I_3) = \{\bar{k}\},\ DT(I_3) = \{\underline{j}\}.$

These DH and DT sets can be identified based on an algebraic description of each subnet. It involves the following operations:

Cancel Node: Nodes The operation of node canceling goes on when a node finds itself in other terms.

Identify *DH*: For each d-sepnode in the denominator of each term in Equation (4.2), the d-sepnode is in a DH if it can be cancelled by the same node in the numerator of the same term; otherwise, it is not in a DH. For example, in Equation (4.3), the d-sepnode a in the $p(a, f)$ can be cancelled by a in the $p(c, a, f)$. Hence the d-sepnode a is in a DH.

In fact, if only the d-sepnode is included in the denominator, the d-sepnode should be in a DH.

Identify *DT*: For each d-sepnode in the numerator of each term in Equation (4.2) whose denominator does not equal to 1, the d-sepnode is in a DT if it cannot be cancelled by the same node in the denominator of the same term; otherwise, it is not in a DT. For example, in Equation (4.3), the d-sepnode b in the $p(b, d)$ is not cancelled by the $p(d)$. Hence the d-sepnode b is in a DT.

It should be noted that during the DH identifying operation, the final result is a union of all identifications. For example, in Equation (4.4), although d-sepnode a in the numerator $p(m, j, a, n)$ can be cancelled by the $p(j, a, n)$ in the term $\frac{p(m, j, a, n)}{p(j, a, n)}$, it is still in a DT because it cannot be cancelled in the term $\frac{p(a, l)}{p(l)}$. It is seen that $\bar{a}$ and $\underline{a}$ are symmetric types of d-sepnode a. A d-sepnode in the numerator of the term whose denominator equals to 1 must be in a DH. Hence, a DP will begin with these DH nodes.

A DP in each subnet can be identified through the above denotations and the following operations. For example, Equation (4.3) for subnet I_1 could be rewritten as follows.

$$P_{I_1}=\frac{p(a)}{1}\times\frac{p(c,a,f)}{p(a,f)}\times\frac{p(f)}{1}\times\frac{p(e,f,d)}{p(f,d)}\times\frac{p(d,c)}{p(c)}\times\frac{p(b,d)}{p(d)}=\frac{p(a)}{1}\times\frac{p(c,a,f)}{p(a,f)}\times\frac{p(d,c)}{p(c)}\times\frac{p(b,d)}{p(d)}\times\frac{p(f)}{1}\times\frac{p(e,f,d)}{p(f,d)}$$

Beginning with the DH node a the operations will be described as follows:

$$\frac{p(a)}{1}\cdot\!\cdot\frac{p(c,a,f)}{p(a,f)\uparrow}\cdot\!\cdot\frac{p(d,c)}{p(c)\uparrow}\cdot\!\cdot\frac{p(b,d)}{p(d)\uparrow} \tag{4.6}$$

Operation 4.1: Let symbol ·∙. denote that a node in the **numerator** of a term tries to find itself in the **denominator** of another term.

It is clear that the first node subjected to operation 4.1 must be a DH node.

Operation 4.2: Let symbol ↑ denote that a node in the **denominator** tries to cancel the same node in the **numerator** and lets its children find themselves in the next operation ·∙·

For example, in the second term of Equation (4.6), nodes a and f will cancel node a and node f in the denominator and only node c remains. Hence, node c will perform the operation ·∙. in the third term and so on.

Hence, a selected DH node performs these two operations alternately until it meets a DT node in the **numerator** and the operation cannot be performed any more. For example, in Equation (4.6), the DH node a meets the DT node b and no operation can be performed any more.

The termination of these operations produces a DP, such as the DP in Equation (4.6): $DPs=\langle(\bar{a}\bullet\underline{b})\rangle$.

So far, for I_1, I_2 and I_3, DPs formed based on Equations (4.3 ~ 4.5) are as follows:

$$I_1:\ DPs_1=\langle(\bar{a}\bullet\underline{b})\rangle \tag{4.7}$$

$$I_2:\ DPs_2 = \left\langle (\bar{j} \bullet \underline{a}) \otimes (\bar{b} \bullet \underline{k}) \right\rangle \tag{4.8}$$

$$I_3:\ DPs_3 = \left\langle (\bar{k} \bullet \underline{j}) \right\rangle \tag{4.9}$$

4.3.3 Find *DC*

The DAG verification is to ensure that there is no *DC* in an MSBN. Hence all *DPs* from local subnets should be integrated to test whether there is a *DC* globally. An integration of *DPs* composes some *DC* Candidates (*DCC*). The steps are illustrated as follows.

Firstly, integrate *DPs* from potential shared subnets and form each corresponding *DCC* with a *DP* which only includes their shared d-sepnodes. For example, the combination of Equations (4.7-4.9) generates one *DCC*.

$$DCC = \left\langle (\bar{a} \bullet \underline{b}) \right\rangle \otimes \left\langle (\bar{j} \bullet \underline{a}) \otimes (\bar{b} \bullet \underline{k}) \right\rangle \otimes \left\langle (\bar{k} \bullet \underline{j}) \right\rangle \tag{4.10}$$

Secondly, find a *DC* by absorbing symmetric nodes in both sides of operator ⊗ such as $(\bar{a} \bullet \underline{b}) \otimes (\bar{j} \bullet \underline{a}) \equiv (\underline{b} \bullet \bar{a}) \otimes (\underline{a} \bullet \bar{j})$. Hence $\bar{a}$ and $\underline{a}$ will be absorbed. Moreover, the action of absorbing is not limited by the number of symmetric nodes. It means that one *DH* node can absorb more than one *DT* node and vice versa. If all the nodes in a *DCC* are absorbed completely, there is at least one *DC* in the MSBN; otherwise no *DC* exists.

Finally, verify a DAG structure. If there is no *DC* in the MSBN, the overall structure of the MSBN is a DAG; otherwise it is not a DAG structure.

Theorem 4.2: A *DC* exists in an MSBN if and only if nodes in all *DCCs* are absorbed completely.

Proof. If a DC exists in an MSBN, each node in DCC must be a symmetric node. They are on both sides of operator $\otimes$ and will be absorbed. Thus all the nodes in DCC must and will be absorbed totally. Conversely, if all nodes in a DCC are absorbed completely, it indicates that every node has two symmetric types on both sides of operator $\otimes$ and each type of this node is connected with another node by operator $\bullet$. According to the priority of the operator $\bullet \succ \otimes$, there must be a DP among these relative nodes, and these DPs are formed into a DC. For instance, in Equation (4.10), although in the second term $\left\{(\bar{j} \bullet \underline{a}) \otimes (\bar{b} \bullet \underline{k})\right\}$ it is unknown whether nodes a and b are connected; however, the connection is confirmed from the first term $\left\{(\bar{a} \bullet \underline{b})\right\}$. Consequently, all the nodes are connected to form a DC in an MSBN. ▌

For the aforementioned example, nodes in Equation (4.10) are absorbed completely. Hence there must be a DC in the MSBN and a DAG structure of the MSBN is not held. This conclusion can be verified through the graphical structure in Figure 4.1.

4.3.4 Complexity Analysis

Based on the operations described above, a symbolic verification of DAG structure in an MSID can be described in the following steps:

[Symbolic DAG Verification]

1) Convert an MSID into an MSBN;
2) Describe each subnet with the algebraic form;
3) Identify DH and DT;
4) Find DP and build DPs;

5) Integrate DPs into $DCCs$;

6) Find DC in each DCC;

7) Verify the DAG structure.

To analyze the complexity of the symbolic verification, some notations are assumed as follows (Xiang 1998):

m: the maximum number of nodes in each subnet;
t: the maximum cardinality of node adjacency in each subnet;
k: the maximum number of d-sepnodes shared by subnets;
n: the total number of subnets in an MSBN.

It is observed that $o(k)$ d-sepnodes need to be identified in each subnet. Hence, in Step 3, a total of $o(nk)$ d-sepnodes are identified in an MSBN. For each d-sepnode, $o(mt)$ nodes in each subnet have to be searched during the process of finding DP. Thus the complexity of Step 4 is $o(nkmt)$. The most time-consuming part resides in Step 5 where 2^n $DCCs$ may be formed. Fortunately, in practice, the number of subnets in an MSBN is not so large. In Step 6, each node in a DCC will search $o(nk)$ nodes during the operation of absorbing. Hence the complexity of Step 6 is $o(n^2k^2)$.

The algorithm will be executed by the cooperation of agents associated with subnets. In order to protect the privacy of each agent, each agent performs Steps 2 ~ 4 individually. Then, they will provide only DPs for further verification. Hence the last two steps are executed by a coordinator agent that may be a computer or a human. In this way, the algorithm can be performed in a distributed and centralized mode and protects the privacy of the agents.

4.3.5 Dealing with Verification Failure

Verification fails when there are some DCs in an MSID. In this case, it is expected to identify the violated subnets with the aim of correcting the structure. Based on the symbolic approach, the problem of verification failure is investigated in this section.

The verification of DAG structure depends on the DC test in each DCC. This means that the verification fails once there is a DC in any DCC. Thus it is the subnets whose DPs comprise a DCC that cause the failure. Furthermore, these subnets can be identified by tracing back in Step 5 during the verification process. For example, the DAG structure is not ensured in Figure 4.1. One DC has been found. At the same time, in Step 6, it is known that a DCC is shared by I_1, I_2 and I_3. Hence it is the combined structure of I_1, I_2 and I_3 that leads to the failure of verification. The related subnets need to be modified for the DAG structure.

In Step 5, it is known that the larger the set of DPs for each subnet, the larger the set of $DCCs$ and the more likely the failure can happen. Hence, the subnet with the largest set of DPs has more chances to cause failure. Thus the corresponding subnets that share the $DCCs$ must be checked and the structure should be corrected. Based on this point, it is expected to decrease the cardinality of the set of DT or DH in each subnet, which will decrease the cardinality of the set of DPs.

4.4 Symbolic Verification of Agent Interface

The second constraint, d-sepset of agent interface, requires that every node in the agent interface be a d-sepnode, which indicates that all of its parents should be included in at least one local influence diagram.

Theorem 4.3: An MSID without value nodes has the same d-sepset property as the original MSID.

Proof. Since a value node is a sink node without outgoing arcs in an MSID, it cannot become parents of public nodes in an agent interface. Removing it ensures the d-sepset property in an MSID. ▌

In addition, only chance nodes in an MSID are included in the agent interface. Thus, when considering decision nodes as chance nodes in an MSID, the method of verification of d-sepset is similar to that in an MSBN. Here, based on an algebraic description of local influence diagrams, a symbolic method is proposed to deal with the verification and the issue of verification failure is also investigated.

4.4.1 Process of Symbolic Verification

Testing d-sepnode in an agent interface involves two main processes: one is to identify parents of a common (or public) node; the other is to find and judge whether there is a local subset that includes all the parents of this common node.

As the agents' privacy must be protected, only the information of common nodes is shared. To illustrate my approach, an example is shown in Figure 4.2. Each subnet I_j is a local BN (subnet) in the MSBN after removing value nodes and transforming decision nodes into chance nodes in an MSID.

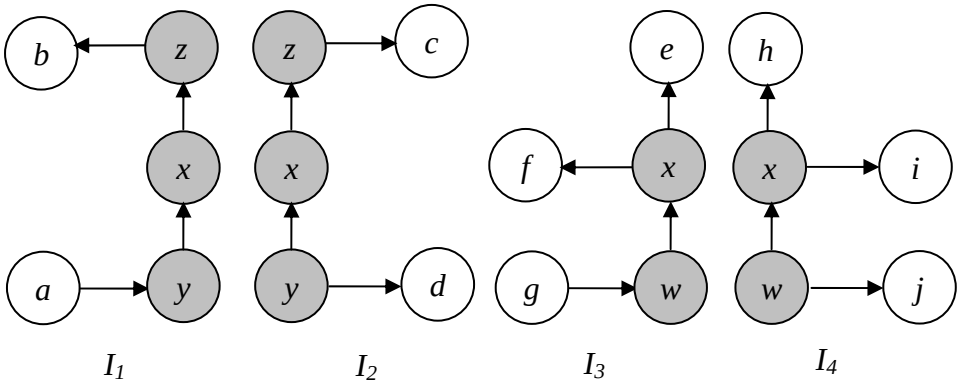


Figure 4.2: Another Example Network

Step 1: (Algebraic Description) By Equation (4.2), each subnet can be described as follows:

$$P_{I_1} = p(b|z)\times p(z|x)\times p(x|y)\times p(y|a)\times p(a) = \frac{p(bz)}{p(z)}\times\frac{p(zx)}{p(x)}\times\frac{p(xy)}{p(y)}\times\frac{p(ya)}{p(a)}\times\frac{p(a)}{1} \tag{4.11}$$

$$P_{I_2} = p(c|z)\times p(z|x)\times p(x|y)\times p(y)\times p(d|y) = \frac{p(cz)}{p(z)}\times\frac{p(zx)}{p(x)}\times\frac{p(xy)}{p(y)}\times\frac{p(y)}{p(1)}\times\frac{p(dy)}{p(y)} \tag{4.12}$$

$$P_{I_3} = p(g)\times p(w|g)\times p(x|w)\times p(f|x)\times p(e|x) = \frac{p(g)}{1}\times\frac{p(wg)}{p(g)}\times\frac{p(xw)}{p(w)}\times\frac{p(fx)}{p(x)}\times\frac{p(ex)}{p(x)} \tag{4.13}$$

$$P_{I_4} = p(h|x)\times p(x|w)\times p(i|x)\times p(w)\times p(j|w) = \frac{p(hx)}{p(x)}\times\frac{p(xw)}{p(w)}\times\frac{p(ix)}{p(x)}\times\frac{p(w)}{1}\times\frac{p(jw)}{p(w)} \tag{4.14}$$

Step 2: (Identify Parent Set) In the term $\frac{p(x_j,\pi(x_j))}{p(\pi(x_j))}$, for each subnet, if a public node x_j in the numerator cannot be cancelled in the denominator, nodes in the denominator of this term must be the parents of x_j that are elements of $SPS(x_j)$ (Sub-Parent Set). A PS (Parent Set) of each public node x_j is an integration of $SPS(x_j)$ for node x_j in each local subnet, denoted as $PS(x_j)=\bigcup_i SPS_{I_i}(x_j)$.

For example, for the local subnet I_1, public nodes are nodes $\{x,y,z\}$ between I_1 and I_2. Hence, based on Equation (4.11), parents of node x are identified as follows.

In the term $\frac{p(zx)}{p(x)}$, node x in the numerator $p(zx)$ can be cancelled by node x in the denominator $p(x)$, but in another term $\frac{p(xy)}{p(y)}$, node x in the numerator $p(xy)$ cannot be cancelled by node x in the denominator $p(y)$. On all accounts, the sub-parent set is $SPS_{I_1}(x) = \{y\}$ in the local subnet I_1.

The same operation required by node x is executed on other local sets based on Equations (4.11 ~ 4.14). The SPS for node x is as follows: $SPS_{I_2}(x) = \{y\}$; $SPS_{I_3}(x) = \{w\}$; $SPS_{I_4}(x) = \{w\}$.

By the operator of union, the PS for node x in the MSBN is $\{y, w\}$, denoted as $PS(x) = \{y, w\}$.

For the sake of the privacy protection of each agent, an SPS should not present details about the private nodes in local subnets, including the name of the private nodes. During the process, the notation ε_{ij} is used to implicate the j th private node in I_i. In this way, the SPS that the agent provides does not refer to the details about its privacy.

Step 3: (D-Testing) In Step 2, a PS of each public node is identified; in this step, it is to find whether there is a local subnet that includes all the parents of a public node. It means that judging the condition of a d-sepnode depends on whether there is an SPS_{I_i} equal to a PS.

Proposition 4.1: If an agent interface between adjacent agents is a d-sepset, for each public node in the agent interface, there is no fewer than one SPS_{I_i} that equals to a PS.

The proof is trivial.

Proposition 4.2: If there are more than one $SPS_{I_i}(x_j)$ including node ε_{ij}, then the node x_j must not be a d-sepnode.

Proof. Assume one local DAG includes a node ε_{ij} , $\varepsilon_{ij} \in SPS_{I_i}(x_j)$ and one of its adjacent DAG includes another node ε_{kj}, $\varepsilon_{kj} \in SPS_{I_k}(x_j)$. As it is known that any agent does not include private nodes of other agents, $\varepsilon_{ij} \notin SPS_{I_m}(x_j), m \neq i$ and $\varepsilon_{kj} \notin SPS_{I_m}(x_j), m \neq k$, no local DAG including both ε_{ij} and ε_{kj} can be found. Thus the difference set between any SPS_{I_i} and a PS will not be empty. The condition of d-sepnode is not obeyed. ▌

The operation difference between two sets is employed to make a comparison between an SPS_{I_i} and a PS. The empty set of the difference between any SPS_{I_i} and a PS implies that they are equal and contain the same elements.

For example, the PS of node x is $\{y, w\}$. However, the SPS_{I_i} of node x are as follows: $SPS_{I_1} = \{y\}$, $SPS_{I_2} = \{y\}$, $SPS_{I_3} = \{w\}$ and $SPS_{I_4} = \{w\}$. There is no empty set between any SPS_{I_i} and the PS. Hence node x is not a d-sepnode, which causes the agent interface to be an invalid d-sepset.

4.4.2 Complexity Analysis and Further Discussion

The complete agent interface verification process in an MSID is performed symbolically in the following steps:

[Symbolic Verification of Agent Interface]

1) Remove value nodes and transform decision nodes into chance nodes in an MSID;

2) Describe each subnet in an MSBN with the algebraic form;

3) Identify a PS of every public node in an agent interface;

4) Test the condition of d-sepset.

To analyze the complexity of the symbolic verification process, the following notations (Xiang & Chen 2002) are assumed:

s : the maximum number of adjacent agents to any agent;

t : the maximum cardinality of nodes in a local subnet (DAG);

k : the maximum number of nodes in an agent interface;

n : the total number of agents.

In the third step of the algorithm, $o(k)$ nodes in the agent interface must be checked on $o(t)$ nodes in a local DAG. At the same time, $o(n)$ agents should perform the process. Hence the complexity is $o(ntk)$. In the last step, there are $o(k)$ PSs that need to be checked with $o(s)$ $SPSs$. Consequently, the complexity is $o(nsk)$.

From the complexity analysis, it seems that the main factors having a large effect on the algorithm complexity are variables t (the maximum cardinality of nodes in a local DAG) and n (the total number of agents). For a large and complex domain, their product is just the same as the sum of nodes in an MSBN after an MSID has been pre-processed. Hence, the point is how to partition a large domain into several small local subnets, which gives a hint for further research on the topic of partitioning a large domain. At least, a reasonable method should consider the effect on the algorithm complexity.

For the same reason, an individual agent will provide only the SPS of its own to a coordinator agent that will deal with the D-Testing operation and make a final decision. The introduction of a coordinator agent that may be a computer or a human being is necessary for privacy protection because, except the coordination agent, all other agents have no idea about a PS. At the same time, the introduction is helpful for the dealing with verification failure.

4.4.3 Dealing with Verification Failure

When the verification of d-sepset fails through a symbolic verification in D-Testing, there should be no empty difference set between any SPS and a PS. One measure is proposed to fix this problem.

Firstly, find the largest cardinality of $SPS_{I_k}(x_j)$, named as $LSPS_{I_k}(x_j)$. It is known that no fewer than one $SPS_{I_k}(x_j)$ equals to a PS if the condition of d-sepnode is satisfied. One way to correct the verification failure is to let a more promising $SPS_{I_k}(x_j)$, such as $LSPS_{I_k}(x_j)$, be equal to the PS.

Secondly, identify elements in the difference set between $LSPS_{I_k}(x_j)$ and a PS. If the element ε is identified, only the local subnet including the element ε should be corrected; otherwise, other subnets that share the element should be referred to. However, the problem of correcting an MSID structure to satisfy the d-sepset condition is related to other constraints of the MSID. It must be considered in an overall way.

4.5 Pairwise Verification of Irreducibility of D-sepset

The irreducibility of d-sepset is to ensure the compactness of MSID and HRG, and to remove any redundant information in this model representation. The irreducibility of d-sepset depends on organizational relationships among agents and information flowing through the d-sepset. The information should be composed of the requisite information and the supporting information (for agents' decision making) which indicate the control and communication relationships among agents respectively. An HRG is just an effective framework that can be used to test this constraint. Hence, the verification of two basic relevance graphs in an HRG is discussed as follows.

Control Relevance Graph: Figure 3.2 (a) shows the *Control* relationship between agent A_i and agent A_j denoted by the function $\mathrm{Re}\,q(A_i, A_j, d_k) = \{c_1, \cdots, c_m\}$. To verify the irreducibility of a d-sepset amounts to testing whether the information (encoded in) $\{c_1, \cdots, c_m\}$ agent A_i provides belongs to the required information for agent A_j's decision d_k.

It is known that the procedures *Decision Bayes-ball* and refined *Decision Bayes-ball* could be used to obtain the required information for any decision node in an influence diagram. Through these procedures, the information $\{c_1, \cdots, c_m\}$ required for agent A_j's decision d_k can be obtained. Assuming *Control* is a unique relationship between agent A_i and agent A_j and the d-sepset is S_{ij}; thus, $S_{ij} \subseteq \bigcup_k \mathrm{Re}\,q(A_i, A_j, d_k) \Leftrightarrow \textit{Irreducibi lity}$.

Communication Relevance Graph: Figure 3.2 (b) shows the *Communication* relationship between agent A_i and agent A_j denoted by the function $Sup(A_i, A_j) = \{c_1, \cdots, c_n\}$. The irreducibility of d-sepset means that the information passing through a d-sepset is the supporting information that both

agent A_i and agent A_j could access; however, this information is not required for their decision making. The verification depends on the domain knowledge. For example, domain expert could tell what kind of information is unavailable to one agent while its adjacent agents could provide such kind of information. Assuming *Communication* is the unique relationship between agent A_i and agent A_j, and the d-sepset is denoted as S_{ij}; therefore, $S_{ij} \subseteq Sup(A_i, A_j) \Leftrightarrow Irreducibility$.

Mixed relationships (both control and communication) between agent A_i and agent A_j exist with an integrated verification as follows.

$$S_{ij} \subseteq (((\bigcup_k \mathrm{Re}\, q(A_i, A_j, d_k)) \cup (\bigcup_k \mathrm{Re}\, q(A_j, A_i, d_k))) \cup Sup(A_i, A_j)) \Leftrightarrow Irreducibi\ lity \quad (4.15)$$

For example, given the MSID in Figure 3.5, the d-sepsets are $S_{12} = \{s, w\}$, $S_{23} = \{w, pp\}$ and $S_{13} = \{w, c\}$.From the procedure of *Decision Bayes-ball*, nodes $\{s, c\}$ are required for decisions of agents *NS* and *CS* respectively. The cases are characterized as $\mathrm{Re}\, q(A_{ICC}, A_{CS}, ah) = c$, $\mathrm{Re}\, q(A_{ICC}, A_{CS}, wt) = c$, $\mathrm{Re}\, q(A_{CS}, A_{NS}, in) = pp$, $\mathrm{Re}\, q(A_{CS}, A_{NS}, re) = pp$, $\mathrm{Re}\, q(A_{ICC}, A_{NS}, in) = s$, and $\mathrm{Re}\, q(A_{ICC}, A_{NS}, re) = s$. Assuming the irreducibility of d-sepset and according to the Equation (4.15), it follows that $Sup(A_i, A_j) = w$ where $i, j = ICC, NS, CS$ and $i \neq j$. Based on this analysis, the HRG can be produced as shown in Figure 3.6.

At this time, the HRG could be evaluated by domain experts whether it realistically reflects the actual knowledge domain. If it does, the irreducibility of d-sepset is verified; otherwise, the HRG will be modified. The modeling process is iterative until both the HRG and the MSID are acceptable.

The complexity of verification process lies in the procedure *Decision Bayes-ball* that is analyzed in detail in some literature (Shachter 1998; Nielsen 2001). It will not be discussed here. One feature of this verification process is a combination of both MSID and HRG which makes full use of knowledge domain. It verifies the fact that a valid and meaningful representation is essentially related with the problem domain.

4.6 Summary

Verification of a graphical decision model is essential in the cycle of decision analysis. The representation of MSID and HRG describes a multiagent decision problem in a distributed and cooperative framework. Its definition is accompanied by three constraints: DAG structure, d-sepset of agent interface and irreducible d-sepset.

In this Chapter, I proposed a symbolic method for the verification of DAG structure and d-sepset of agent interface in an MSID. It utilizes an algebraic description in an MSID. The symbolic method is cooperative while protecting agents' privacy in the verification process. My method has the ability to provide essential information when verification fails (i.e., a global cycle is detected) to correct the offending parts of the network. Furthermore, it provides an alternative view, so called an algebraic angle, on the research of an MSID. For the verification of irreducible d-sepset, a pairwise approach is designed with the consideration of the MSID structure and the HRG implication. It ensures that the decision model could represent a real decision problem accurately.

5 Model Evaluation

The aim of decision analysis is to provide optimal policies to decision makers based on an evaluation of valid and accurate models which fully represent the corresponding decision scenario. The output of solving decision models largely depends on the effectiveness and efficiency of evaluation algorithms which have been designed to decode models. In the previous chapters, valid decision models of MSID and HRG have been developed to represent multiagent decision problems. This chapter will discuss three evaluation algorithms in detail to solve these proposed decision models.

5.1 The Introduction

Multiagent decision problems refer to a kind of distributed decision problems involving multiple agents in an uncertain environment. Within such a decision problem, there are intricate interactions between observations and decisions of multiple agents. For example, one agent's observations may have an influence on other agents' decisions which in turn may affect more of other agents' decisions. Moreover, agents' features, such as their privacy and organizational relationships, should not be lost in their interactions. An individual agent seeks the best decision while it has a full, correct and consistent observation in the situation through the communication with its adjacent agents. As one of the characters of a multiagent decision problem indicates agents seek their individual objectives while they expect a cooperative solution.

Solving a multiagent decision problem is to find out individual objectives of agents in a distributed setting cooperatively. An individual agent has its own goal and wants to make the best decision after obtaining a full and consistent observation from its adjacent agents. The motivation of their cooperation is to access the full and consistent information in order to maximize their own utility as well as to release the relevant and honest information in order to help their partners' decisions. Hence, the core of this cooperation is the accessibility of the full, consistent and honest public information. This decides that potential approaches for solving multiagent decision problems should be a cooperative method in a distributed setting.

In Chapter 3, the representation of MSID and HRG has been proposed to model multiagent decision problems. An MSID is a distributed graphical decision model which connects local influence diagrams together through d-sepsets while an HRG explicitly represents the information support for decision making in the MSID. Model evaluation is to solve an MSID to produce optimal decisions with the consideration of the implication of an HRG.

In this chapter, three evaluation algorithms, called a cooperative reduction algorithm in Section 5.2, a distributed *evalID* algorithm in Section 5.3 and one indirect evaluation algorithm in Section 5.4, are proposed to solve an MSID.

5.2 Cooperative Reduction Algorithms

A direct method for solving influence diagrams adopts some basic operations, such as node removal and arc reversal, to reduce a full decision model according to a node or variable elimination sequence (Olmsted 1983; Shachter 1986, 1988). A potential approach for solving an MSID could extend the basic reduction algorithm. Since an

MSID is a set of local influence diagrams connected by d-sepnodes, a cooperative reduction algorithm shall be constructed to coordinate the evaluation of local influence diagrams through a cooperative evaluation on d-sepnodes. In this way, local influence diagrams are ensured to have a consistent and full evaluation with each other in the process of solving an MSID. Hence, a reduction algorithm for solving an MSID is said to be cooperative if d-sepnodes could be evaluated simultaneously.

To formalize a cooperative reduction algorithm, the relevant issues, such as model transformation and node elimination sequence, should be investigated in an MSID. In this section, firstly, the concept of a legal transformation is defined in an MSID. Then, a local elimination sequence in local influence diagrams and a global elimination sequence for d-sepnodes are investigated respectively. A global elimination sequence is the cornerstone of a cooperative reduction algorithm. The validity and efficiency of an elimination sequence are discussed in detail. Finally, a cooperative reduction algorithm is further studied based on different data representations of an MSID.

5.2.1 Legal Transformation

The basic operations, such as node removal and arc reversal, transform an original MSID into a series of interim MSIDs in a cooperative reduction algorithm till the MSID is solved finally. To ensure no loss of relevant information, a legal transformation is defined as the one that satisfies both the semantic level and the strategic level. The semantic level is satisfied when an interim MSID is a new MSID that follows three constraints: DAG structure, d-sepset of agent interface and irreducibility of d-sepset. The strategic level is satisfied when the optimal strategies for a final MSID, together with the associated partial strategies (recording the decision

functions upon removing a decision node), could comprise the optimal strategies for the original MSID.

Model transformation through basic operations involves an update of both graphical structures and associated data. Both of them have an effect on the transformation process, even on a specific transformation operation. For the graphical structure, a certain elimination sequence could be identified and it guides the transformation process. For the associated data, different data representations in a decision model could have different operations in a reduction algorithm. Recently, two kinds of data representations, namely conditional probability tables c (Howard & Matheson 1984) and potential representation Ψ (Ndilikikesha 1994), allow the joint probability distribution of an MSID to be defined as follows.

$$P = \prod_j P_{I_j} = \prod_j \prod_i c(x_{ji} | \pi(x_{ji})) \text{ or } P = \prod_j P_{I_j} = \prod_j \prod_i \Psi(x_{ji})$$

where $\pi(x_{ji})$ are parents of node x_{ji} in local influence diagrams I_j.

For instance, the Ψ representation could avoid the operation of arc reversal in a reduction algorithm when compared with the c representation. This could help to design different evaluation strategies in a cooperative reduction algorithm for solving an MSID. For convenience, the evaluation of an MSID based on the c representation is called C-Evaluation while the evaluation based on the Ψ representation is called P-Evaluation. Hence, a legal model transformation has to be verified with respect to the C-Evaluation and the P-Evaluation respectively.

5.2.2 Local and Global Elimination Sequence

5.2.2.1 Foundation

The goal of evaluating influence diagrams is to find an optimal strategy δ_i for each decision node d_i and to maximize an expected utility function μ. The computation is based on the following expression:

$$\delta_{d_i}(c_1, d_2, \cdots, c_{i-1}) = \arg\max_{d_i} \sum_{c_i} \cdots \max_{d_n} \sum_{c_n} P(c_i, \cdots, c_n | c_1, \cdots, c_{i-1}, d_1, \cdots, d_n)\mu$$

where μ is the utility function specified by value node v; c_i and d_i are chance nodes and decision nodes respectively in influence diagrams.

To obtain optimal strategies, variables are eliminated by the operator in a certain order in the computation. A legal elimination sequence is acceptable if all nodes are eliminated by following a partial order of chance nodes in information sets and decision nodes (Nielsen & Jensen 1999). However, determining an optimal elimination sequence has been shown to be NP hard (Cooper 1987). Instead of providing such an elimination sequence, I investigate the issue on the efficiency and validity of an elimination sequence in both local influence diagrams and a global MSID. An elimination sequence is said to be legal if it is acceptable. On the other hand, an elimination sequence is said to be efficient if it could avoid some expensive operations, such as arc reversal, as much as possible.

The idea of identifying an elimination sequence in influence diagrams was mentioned in some previous work (Shachter 1986; Rege & Agogino 1988); however, it has not been formalized and highlighted in a reduction algorithm. In reality, a good elimination sequence could avoid some unnecessarily expensive operations, such as

arc reversal, to save the computational cost. For solving an MSID, a cooperative reduction algorithm should emphasize the development of an efficient and legal elimination sequence.

5.2.2.2 Elimination Sequence

Terminology: Consider (local) influence diagrams $I(C, D, v)$ [1] where there are n nodes including chance nodes C, decision nodes D and value node v.

t: Time to represent the latest influence diagram that has all updated data, new evidences or observations;

$< x, y >$: A directed arc from node x to node y, where $x \in (C \cup D)$, $y \in (C \cup D \cup \{v\})$. Node x is called a direct predecessor of node y, denoted by $x = \Pr e(y)$, while node y is a direct successor of node x, denoted by $y = Suc(x)$;

$Num(\Pr e(x))$, $Num(Suc(x))$: The number of direct predecessors and successors of node x respectively;

(x, y): A directed path from node x to node y, where $x \in (C \cup D)$, $y \in (C \cup D \cup \{v\})$;

$I(y) = \{x | < x, y >\}$, $y \in D$: An information set for decision node y;

$C(y) = \{x | < x, y >\}$, $y \in C \cup \{v\}$: An influence set for chance node or value node y;

$f(x)$: An index function for node x. Each node x is associated with a unique integer number $i \in \{1, \cdots, n\}$ in the node list, denoted by $f(x) = i$. It is the order of node x in a node list;

[1] To avoid complex denotations, subscripts specifying local influence diagrams are omitted. Since a regular influence diagram is considered, only one value node exists.

R1: Rule 1: Set $f(c_k) < f(c_l)$ if there exists an arc $< c_k, c_l >$;

R2: Rule 2: Set $f(c_k) < f(c_l)$ if $Num(Suc(c_k)) > Num(Suc(c_l))$;

$NL(t)$: A node list at time t. A node list is generated in an ascending order of $f(x)$;

$ES(t)$: A relative elimination sequence at time t. It is generated by sorting nodes (excluding value node) in a descending order with index function $f(x)$.

$ES(t, x)$: An order for node x in an elimination sequence $ES(t)$ at time t. It follows $ES(t, x) = n - f(x)$.

$BES(t, x)$: A base order for node x in an elimination sequence $ES(t)$ at time t. It is the earliest order in which node x can be removed.

$ES(t, x)$ and $BES(t, x)$ are always abbreviated as $ES(x)$ and $BES(x)$ without considering time t.

The process GSL (Generate Sequences Locally) to generate an efficient legal elimination sequence in local influence diagrams is divided into the following steps.

Step 1: Pick up decision nodes $d_i \in D$ and value node v to compose an initial node list $NL(t)$ denoted by $\{d_1, \cdots, d_i, \cdots d_m, v\}$ in which $f(d_1) < \cdots < f(d_m) < f(v)$ exists;

Step 2: Identify the information set $I(d_i)$ for a decision node $d_i \in D$ and the influence set $C(v)$ for the value node v. (1) When node $x \in C(v) \bigcap (\bigcap_i I(d_i))$, $f(x) < \min(f(d_i))$ is set; (2) When node $x \in C(v)$ but $x \notin I(d_i)$, $f(d_i) < f(x) < f(v)$ is set. Here node x is categorized into $C(v)$; (3) When node $x \in \bigcap_i I(d_i)$ but $x \notin C(v)$, $f(x) < \min(f(d_i))$ is set;

(4) When nodes $x, y \in I(d_i)$ or $x, y \in C(v)$, two alternatives are available. The first alternative is to follow *R1* and *R2*. The second one is to set either $f(x) < f(y)$ or $f(y) < f(x)$. After that, the node list $NL(t)$ is updated as $\{I(d_1), d_1, \cdots, I(d_i), d_i, \cdots, I(d_m), d_m, C(v), v\}$ after inserting $I(d_i)$. Finally, the $BES(t, x)$ is generated for all nodes as follows;

$$BES(t,x) = \begin{cases} Max(f(d \cup I(d))) - f(x) + 1, x \in d \cup I(d); \\ n, x = v; \\ 0, otherwise; \end{cases}$$

Step 3: Identify the influence set $C(c_i)$ for chance node $c_i \in I(d_i) \cup C(v)$ where $d_i \in D$. (1) When node $x \in \bigcap_i C(c_i)$ where $c_i \in C(v)$, $f(x) < \min(f(c_i))$ is set; (2) When node $x \in \bigcap_i C(c_i)$ where $c_i \in I(d_i)$ but $c_i \notin C(v)$, $\max(f(d_j)) < f(x) < \min(f(Suc(x)))$ is set; (3) When nodes $x, y \in C(c_i)$, *R1* and *R2* are followed; otherwise, either $f(x) < f(y)$ or $f(y) < f(x)$ is set arbitrarily. After that, the node list $NL(t)$ is updated until all nodes c_i are exhausted;

Step 4: Repeat step 3 until all chance nodes $c_i \in C(c_i)$ are exhausted;

Step 5: Generate an elimination sequence $ES(t)$.

The basic idea of the process GSL is based on decision windows in influence diagrams. In Step 1, decision nodes are identified based on a chronological order in influence diagrams. After that, in Step 2, it deals with nodes in both an information set and an influence set. The first case ensures a valid operation imposed on nodes that precede a value node. The second case indicates a removal of a node that is near the value node. The third case is based on the chronological order of associated decision nodes. The

fourth case considers nodes in the same information set or influence set. The rule *R1* indicates the removal of a node that is near the value node. The rule *R2* avoids an operation imposed on a node with multiple successors so that a large amount of computation cost is saved. After Step 2, the earliest elimination order for a node, namely a base order, could be driven. In Step 3, it considers all nodes in an influence set. The first case removes a node near the value node. The second case ensures subsequent valid operations imposed on the decision node. The third case considers nodes in the same influence set. The two rules *R1* and *R2* are followed with the same aim. Step 4 enters a loop and searches all the nodes in an influence set. Finally, in Step 5, an elimination sequence is produced according to the index function.

Lemma 5.1: If regular influence diagrams are pruned (by removing barren nodes), the process GSL to generate an elimination sequence always terminates.

Proof. Without barren nodes and with a finite number of nodes in a regular influence diagram that has a directed acyclic graphical structure, the process GSL terminates after searching all chance nodes in the fourth step. ▌

Theorem 5.1: The elimination sequence generated in the process GSL is legal.

Proof. Step 1 and Step 2 deal with the elimination sequence of decision node $d_i \in D$ and chance node $x \in I(d_i)$. In the second step, all the nodes in the information set are assigned a lower index function than their corresponding decision nodes. In this way, the elimination sequence order satisfies the partial order of chance nodes in the information set and decision nodes. Thus it is acceptable. ▌

Theorem 5.2: The elimination sequence generated in the process GSL is efficient.

Proof. It will be discussed in the C-Evaluation and the P-Evaluation respectively. ▌

5.2.3 Global Elimination Sequence

A cooperative reduction algorithm is a cooperative evaluation of local influence diagrams in an MSID. The cooperation is valid only if there is a global elimination sequence for d-sepnodes since an MSID connects local graphs through the linkage of d-sepset. In the previous section, a legal elimination sequence $ES_j(t)$ ($j = 1 \cdots m$) is produced through the process GSL in each local influence diagram I_j. Each local elimination sequence could generate a specific elimination sequence locally for d-sepnodes. It is unknown whether a global elimination sequence for d-sepnodes does exist in an MSID since different elimination sequences for d-sepnodes always happen in local influence diagrams.

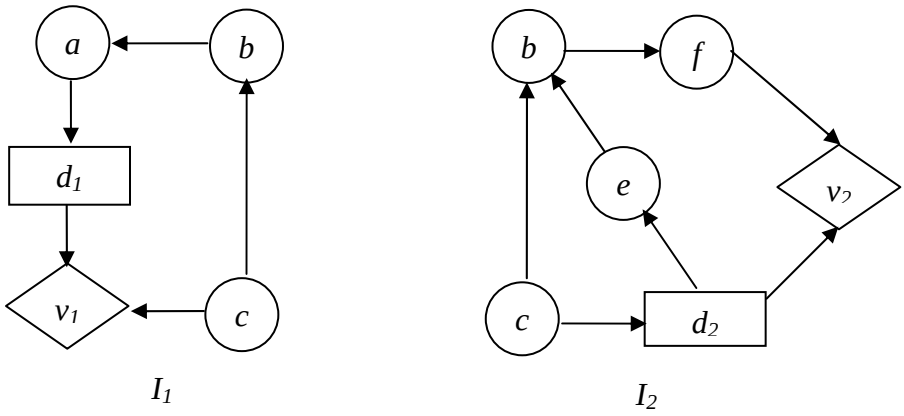


Figure 5.1: An MSID of I_1 and I_2

For instance, in Figure 5.1, the MSID adopts the c representation. The two local influence diagrams I_1 and I_2 share d-sepnodes $\{b, c\}$. Through the process GSL, either $ES_1(t) = \{c, d_1, a, b, v_1\}$ or $ES_1(t) = \{c, d_1, b, a, v_1\}$ is obtained in I_1; and $ES_2(t) = \{f, b, e, d_2, c, v_2\}$ is obtained in I_2. Both elimination sequences in I_1 indicate that node c should be removed after node b. However, node c has to be removed

before node b in I_2 based on $ES_2(t) = \{f, b, e, d_2, c, v_2\}$. Consequently, a cooperative reduction algorithm does not exist since there is no global elimination sequence for d-sepnodes. This problem motivates my investigation on a global elimination sequence for d-sepnodes in an MSID.

5.2.3.1 Elimination Sequence for D-sepnodes

A final elimination sequence in local influence diagrams is always not unique since flexible elimination orders may be assigned to nodes in the process GSL . It is difficult to determine an absolute elimination order for each node. However, in the first two steps of the process GSL , a legal elimination sequence should be generated for both decision nodes and chance nodes in information sets. This legal elimination sequence outputs a base elimination order $BES(x)$ which is the foundation of a final elimination sequence for all nodes. Concerning the base elimination order $BES(x)$, nodes with a zero value have a flexible relative elimination order while nodes with a non-zero value have a fixed relative elimination order in a final elimination sequence. Hence a global elimination sequence for d-sepnodes should be developed according to the base order. Furthermore, only those d-sepnodes with a non-zero base elimination order should be considered since they may constrain their elimination sequence among local influence diagrams.

Assuming that d-sepnodes, $\{x_{j1}, \cdots, x_{jn}\}, j = 1, \cdots, m$, are considered in local influence diagrams I_j , the global elimination order can be generated in the process GER (Generate Elimination Order) below.

Step 1: Divide d-sepnodes $\{x_{j1}, \cdots, x_{jn}\}$ into two sets according to a base elimination order $BES(x)$. One set, defined as a flexible set $\{x_{jk}, \cdots, x_{jl}\}$, includes nodes having

$BES(x) = 0$; the other set, defined as a fixed set $\{x_{jr}, \cdots, x_{js}\}$, includes nodes having $BES(x) > 0$. If a fixed set is empty, the process GER terminates. Any global elimination sequence could be an output;

Step 2: Sort the fixed set $\{x_{jr}, \cdots, x_{js}\}$ in an ascending order of $BES(t, x)$ to obtain a new set $DS_j = \{x_{jr} \prec x_{j(r+1)} \prec \cdots \prec x_{js}\}$;

Step 3: Collect all $DS_j = \{x_{jr} \prec x_{j(r+1)} \prec \cdots \prec x_{js}\}, j = 1 \cdots m$, to compose a rough elimination graph as shown in Figure 5.2;

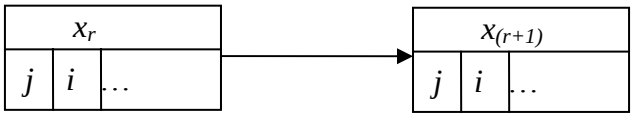


Figure 5.2: Rough Elimination Graph

In Figure 5.2, a large rectangular node indicates a node (in the above long frame) shared by some local influence diagrams (indicated in the below small frame). An arrow is triggered when its tail nodes are removed. For example, in Figure 5.2, nodes x_r and $x_{(r+1)}$ are shared by local influence diagrams I_j and I_i. Following the direction of the arc in Figure 5.2, it indicates that node x_r must be removed before node $x_{(r+1)}$.

In a rough elimination graph, a node is to be removed if and only if all of its incident arcs are triggered. An arc is triggered when its tail node, such as node x_r, is removed. (Similarly, node $x_{(r+1)}$ is called head node.) This trigger rule helps us identify a global elimination sequence in Step 4.

Step 4: Identify a global elimination sequence for d-sepnodes in a rough elimination graph built in Step 3.

Step 4.1: Test the existence of a global elimination sequence for d-sepnodes. This is to confirm the possibility of a cooperative reduction algorithm for solving an MSID. The method is to check directed cycles in a rough elimination graph. If there is a directed cycle in a rough elimination graph, a global elimination sequence does not exist. The process *GER* terminates here. Thus there is no cooperative reduction algorithm.

Step 4.2: Build a global elimination graph for d-sepnodes. Firstly, detect a pair of nodes that are connected with multiple directed paths. These paths exist since, after the test in Step 4.1, an elimination graph that has no directed cycle remains. Secondly, build a global elimination graph. This graph is built when a directed arc between the detected pair-wise nodes is removed.

Step 4.3: Obtain a global elimination sequence for d-sepnodes. A global elimination sequence is identified through searching multiple paths from source nodes to sink nodes in a pruned elimination sequence graph. All nodes in a directed path from a source node to a sink node compose an "island" i of an elimination sequence, denoted as $GES_i = \{x_r \prec x_{(r+1)} \prec \cdots \prec x_s\}$. In general, a global elimination sequence is made out of several islands of elimination sequences because multiple directed paths always exist between a source node and a sink node as well as multiple source nodes and sink nodes always exist in a final elimination graph. Hence a global elimination sequence is produced as follows $GES = \bigcup_i GES_i$, where i is the island number.

The process *GER* depends on the pruning on an elimination graph to obtain a global elimination sequence. In the first three steps, a rough elimination graph only describes d-sepnodes which have a non-zero base elimination order since they coordinate the evaluation of local influence diagrams. The case that all d-sepnodes have a flexible relative elimination order in their associated local influence diagrams could output any global elimination sequence which may refer to any local elimination sequence obtained in the process *GSL* . This output global elimination sequence would not make any local elimination sequence invalid. In Step 4.1, a unique elimination sequence for d-sepnodes would not cause any directed cycle in a rough elimination graph. Hence, testing the existence of a global elimination sequence is equivalent to checking directed cycles. After the existence of a global elimination sequence is confirmed, the approach of pruning arcs is utilized to build a global elimination graph in Step 4.2. A directed arc is removed between a pair of nodes if there are multiple paths connecting these two nodes in a rough elimination graph. Since no operation happens with the arc between this pair of nodes, the arc removal is to ensure a head node to be triggered at a later time when other paths are followed. In Step 4.3, multiple paths in an elimination graph are considered. Every path shows a cooperation of local influence diagrams to remove d-sepnodes. Finally, a global elimination sequence *GER* is generated.

Theorem 5.3: A global elimination sequence generated in the process *GER* allows a cooperative reduction algorithm.

Proof. The first step could generate any global elimination sequence which allows a cooperative evaluation of local influence diagrams. Otherwise, a global elimination sequence may or may not exist. In the fourth step, the possibility of a global elimination sequence is tested to confirm a cooperative reduction algorithm. Hence a

global elimination sequence GES which is obtained through some operations in a global elimination graph is valid in the process GER. This allows a cooperative reduction algorithm for solving an MSID. ▌

Suppose there are three local influence diagrams, namely I_1, I_2 and I_3, in an MSID. Each of them has d-sepnodes with an ascending order on $BES(x)$ such as $DS_1 = \{b_1 \prec a_1 \prec e_1 \prec f_1\}$, $DS_2 = \{b_2 \prec a_2 \prec c_2 \prec d_2\}$ and $DS_3 = \{c_3 \prec d_3 \prec e_3 \prec f_3\}$. Firstly, a rough elimination graph is built in Figure 5.3. In Figure 5.3, d-sepnode e is removed after triggering arcs $< a, e >$ and $< d, e >$. This requires the removal of node a (between I_1 and I_2) and node d (between I_2 and I_3).

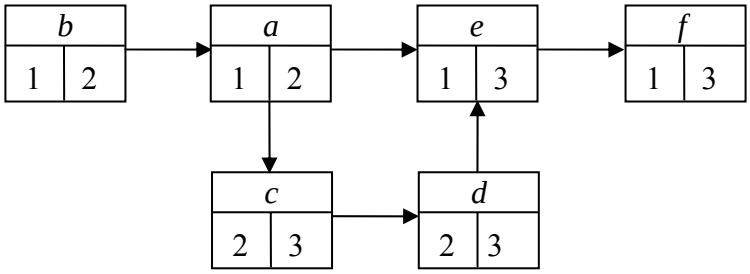


Figure 5.3: Rough Elimination Graph for the Three Local Influence Diagrams

After that, directed cycles are checked in the rough elimination graph. In Figure 5.3, it is confirmed that there is no directed cycle. Hence it is possible to obtain a global elimination sequence in a cooperative reduction algorithm. In Figure 5.3, two directed ways (an arc $< a, e >$ and a path (a, e)) are detected between node a and node e. Following the method in Step 4.2, the directed arc $< a, e >$ is deleted and the graph is pruned as shown in Figure 5.4.

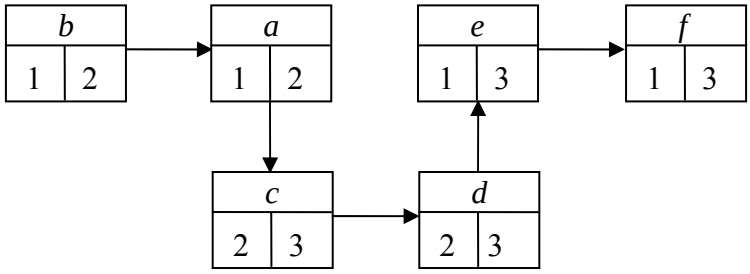


Figure 5.4: Global Elimination Graph

In the global elimination graph, there is only one source node b and one sink node f. In Step 4.3, only one island of elimination sequence is found. Hence a final global elimination sequence is generated as $GES = \{b \prec a \prec c \prec d \prec e \prec f\}$. In fact, according to a global elimination graph, a cooperation order coordinating the evaluation of local influence diagrams can be determined when the sequence of d-sepnodes is followed in a path. For example, in Figure 5.4, the cooperation between I_2 and I_3 follows the cooperation between I_1 and I_2.

5.2.3.2 Discussions

The process GER produces a global elimination sequence with the consideration of all local elimination sequences in an MSID. Only with a global elimination sequence, a cooperative reduction algorithm is valid; otherwise, reduction algorithms have to be carried out separately in local influence diagrams without the cooperation. To distinguish these two cases, two types of algorithm implementations, namely *on-line algorithm* and *off-line algorithm*, are defined.

Definition 5.1: *On-line algorithm* is implemented when reduction algorithms are carried out in a cooperation of local influence diagrams through d-sepset with the timely information.

Definition 5.2: *Off-line algorithm* is implemented when reduction algorithms are carried out in local influence diagrams individually without any cooperation after the d-sepset is initialized.

An on-line algorithm is a real implementation of a cooperative reduction algorithm. One of the advantages behind an on-line algorithm is the ability to access the up-to-date information globally in a distributed system. However, an on-line algorithm requires the possibility of a cooperative reduction algorithm. Furthermore, it is costly to connect all local systems and to coordinate reduction algorithms in local influence diagrams. In contrast, an off-line algorithm is free to perform reduction algorithms locally without the cooperation. It does not require a cooperative reduction algorithm. Thus the latest information in d-sepnodes cannot be accessed since local influence diagrams are kept alone and evaluated separately in an off-line algorithm. As for the computational memory, an off-line algorithm has to keep more copies of d-sepnodes while an on-line algorithm operates on only one copy. In summary, an on-line algorithm is suitable for a distributed system which is sensitive to any new information or evidence while an off-line algorithm is suitable for a resource-constrained system.

Solving an MSID requires a cooperative reduction algorithm so that local influence diagrams could be evaluated with the full and consistent information on d-sepnodes. It is the intention that agents could seek their goal while accessing the information to support their own decision making as well as releasing the information to help their adjacent agents' decision making. Hence an on-line algorithm is a desirable implementation.

5.2.4 C-Evaluation and P-Evaluation

A cooperative reduction algorithm involves a series of model transformations in an MSID when some basic operations, such as node removal and arc reversal, are imposed on local influence diagrams. A legal transformation demands that the transformation satisfy both the strategic and the semantic level. Solving an MSID is to find out the individual optimal decision policy for each agent which has obtained the full and consistent information from its adjacent agents. In a cooperative reduction algorithm, a legal transformation is satisfied automatically in the strategic level since each local influence diagram adopts the traditional reduction algorithm and d-sepnodes are evaluated in the cooperation with its adjacent diagrams. As for the semantic level, the validity of a legal transformation has to be investigated in this section.

The semantic level concerns a graphical structure of interim models when some basic operations are imposed on an MSID. It requires that an interim MSID be a valid model which satisfies three constraints: DAG structure, d-sepset and irreducibility of d-sepset. Since different data representations in a decision model have different operations which drive a model transformation, a legal transformation should be studied in the C-Evaluation and the P-Evaluation respectively.

5.2.4.1 C-Evaluation

The C-Evaluation for solving an MSID adopts the c representation of joint probability distribution in an MSID. That is to say, $P = \prod_j P_{I_j} = \prod_j \prod_i c(x_{ji} | \pi(x_{ji}))$. The C-Evaluation adopts four basic operations in regular influence diagrams (Shachter 1986; Tatman & Shachter 1990). Before going on, the term of a node accessibility is defined as follows.

Definition 5.3: Accessibility on node x from node y indicates that node x is reachable from node y through a directed path. Accessibility increasing on node x from node y indicates that the first directed path between them is just built. Accessibility decreasing on node x from node y indicates that some of the directed paths between them are blocked or removed.

The C-Evaluation in regular influence diagrams consists of four basic operations.

1) Barren node removal: It is to remove a barren node without any operation on other nodes, which does not change the accessibility on other nodes.

2) Chance node removal: Any chance node x whose successor $Suc(x)$ is a single chance node or value node can be removed. Its successor $Suc(x)$ inherits all predecessors of chance nodes $\Pr e(x)$. The accessibility on node x decreases while the accessibility on other nodes does not change.

3) Decision node removal: A decision node can be removed if its only successor is a value node and the information set of this decision node belongs to the influence set of the value node. The value node inherits no predecessor of the decision node. The accessibility on the value node decreases.

4) Arc reversal: The arc $< x, y >$ between two chance nodes x and y can be reversed if no other path between them is found. Both nodes inherit each other's predecessors. The accessibility on node y decreases while the accessibility on node x increases.

The four basic operations reduce local influence diagrams step by step according to a local elimination sequence. Various elimination sequences lead to different

combinations of basic operations. An efficient elimination sequence does not require an expensive operation such as arc reversal in the C-Evaluation. Now, Theorem 5.2 is revisited here.

Theorem 5.4: The elimination sequence generated in the process GSL is efficient in the C-Evaluation.

Proof. The most intricate and expensive operation is the arc reversal in the C-Evaluation. In the process GSL , *Rule 2* decreases the possibility of multiple successors on a certain node while *Rule 1* specifies the removal of nodes which are near the value node. This increases the chance that a value node becomes one of $Suc(x)$ in all operations. Hence the operation of arc reversal is avoided as much as possible. In conclusion, an elimination sequence generated in the process GSL is efficient in the C-Evaluation.▌

The four basic operations ensure that a legal transformation of local influence diagrams is satisfied in the semantic level. However, it is unknown whether the semantic level is satisfied when the transformation is imposed on the whole MSID.

For example, in Figure 5.5, an arc between node b and node c has to be reversed in order to remove node b in I_1. After the arc reversal, DAG is ensured in I_1; however, it is violated in the resulting MSID. In Figure 5.6, there are two directed cycles such as $e \to b \to g \to f \to k \to l \to e$ and $e \to c \to b \to g \to f \to k \to l \to e$.

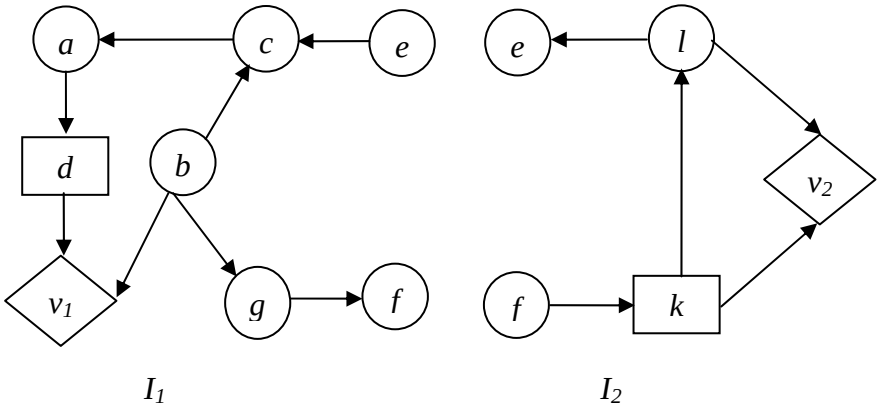


Figure 5.5: An MSID before Arc Reversal

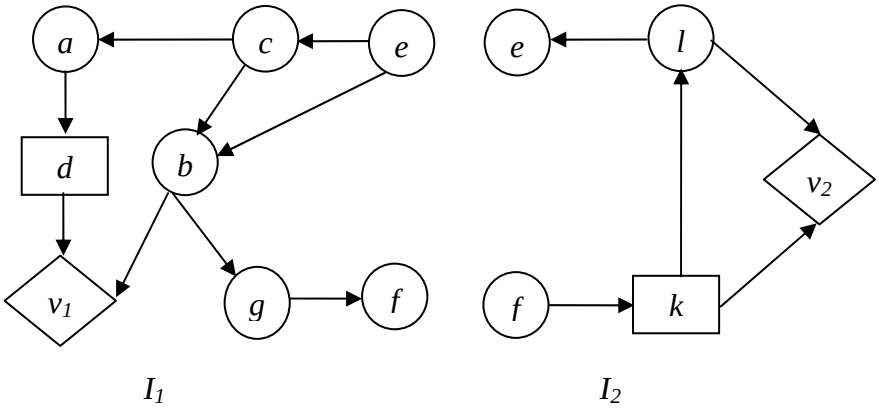


Figure 5.6: An MSID after Arc Reversal

The problem raises my investigation into the issue of legal transformations (in the semantic level) based on the four basic operations. After a transformation, a new MSID must follow three constraints: DAG structure, d-sepset of agent interface and irreducibility of d-sepset. Hence the four basic operations will be studied in detail concerning their effect on the holding of these constraints in an MSID.

Irreducibility of d-sepset: As a transformation is a reduction approach, it simplifies an MSID by removing one node at a time. Thus the constraint of irreducible d-sepset is satisfied when any of the four basic operations is imposed.

d-sepset of the agent interface: This constraint requires that all parents of a public node should be included in the same local influence diagram. It is not expected that any of the basic operations would introduce new parents for public nodes. The operations of barren node removal, decision node removal and chance node (with a value node as a single successor) removal do not introduce new parents for public nodes. As for chance node (with chance nodes as its single successor) removal, it is possible to introduce new parents to a public node if the public node is a successor of the removed nodes. However, *Rule 1* in the process *GSL* avoids this operation. As for the operation of arc reversal, new parents are introduced into the related nodes. Thus the constraint can't be ensured if the operation is imposed on public nodes. Fortunately, this constraint can be relaxed when no cause-effect is considered in an MSID similar to that in an MSBN (Xiang 2002).

DAG structure: The definition of the accessibility indicates that an increasing accessibility on nodes (expect value nodes) leads to more chances of producing new directed cycles. Based on the accessibility analysis on basic operations, an increasing accessibility occurs in the operation of arc reversal. Hence, the constraint is not satisfied in an MSID although a DAG structure is ensured in local influence diagrams after this operation.

Thus, Theorem 5.5 is obtained.

Theorem 5.5: With an operation of arc reversal, a legal transformation in an MSID is not ensured in the C-Evaluation.

5.2.4.2 P-Evaluation

The P-Evaluation for solving an MSID adopts the p representation of joint probability distribution. That is to say, $P = \prod_j P_{I_j} = \prod_j \prod_i \Psi(x_{ji})$. In the P-Evaluation, there are three basic operations, namely barren node absorption, chance node absorption and decision node removal (Ndilikikesha 1994).

1) Barren node absorption : A barren decision node is removed simply without other operations. A barren chance node is removed with an operation: add (as necessary) an arc from another of its predecessors to the one with the highest number $f(x)$.

2) Chance node absorptions: A chance node x is removed with the following two operations:

   a. Add an arc from every direct predecessor $\Pr e(x)$ to every direct successor $Suc(x)$;

   b. Add an arc from other successors $Suc(x)$ to the successor $y \in Suc(x)$ with the highest $f(y)$. If node j receives an arc from another node k, add an arc from every direct predecessor of k to j. The accessibility on node y increases;

3) Decision node removal: A decision node whose successor is a single value node can be removed directly if all other predecessors of the value node are also direct predecessors of this decision node.

Similar to the efficiency analysis of an elimination sequence in the C-Evaluation, Theorem 5.2 is revisited here following the above three operations in the P-Evaluation.

Theorem 5.6: An elimination sequence generated in the process GSL is efficient in the P-Evaluation.

Proof. Generally, it is always assumed that influence diagrams are evaluated after pruning all barren nodes. Hence barren node absorption does not cost much. The most computation consuming operation is chance node absorption in the P-Evaluation. Similar to the proof for Theorem 5.4, *Rule 1* and *Rule 2* produce few numbers of $Suc(x)$, and try to let a value node become one of $Suc(x)$. Thus the operation of adding arcs is seldom applied and becomes a more economical operation when the process GSL is followed to produce the elimination sequence.▌

Comparing with the C-Evaluation having the four operations, the P-Evaluation requires no operation of arc reversal. It is this operation that causes a violation of the DAG structure constraint in an MSID when the C-Evaluation is used. In the P-Evaluation, a possible violation of the DAG structure constraint is the operation of chance node absorption. The possibility to increase the accessibility on chance nodes is avoided if both *Rule 1* and *Rule 2* are followed in the process GSL. With these two rules, the accessibility increasing always happens on value nodes. However, this leads to no chance of violation on a DAG structure since a value node is a kind of sink node. Hence, Theorem 5.7 follows.

Theorem 5.7: A transformation in an MSID is legal in the P-Evaluation.

Proof. The constraint of d-sepset irreducibility is followed since the transformation is a reduction method. The second constraint of d-sepset for agent interface and the third

constraint of DAG structure are obeyed if *Rule 1* and *Rule 2* are followed in the transformation as discussed above. Thus the transformation is legal.▌

In conclusion, an elimination sequence generated in the process *GSL* is efficient in both the C-Evaluation and the P-Evaluation. On another aspect, a legal transformation is ensured in the P-Evaluation while it does not hold in the C-Evaluation. The reason is due to the fact that the P-Evaluation avoids the basic operation of arc reversal in a reduction algorithm.

A cooperation reduction algorithm requires a legal transformation imposed on an MSID. The C-Evaluation always causes an invalid transformation that violates the constraint of DAG structure in an MSID. It is not expected to adopt the C-Evaluation in a cooperation reduction algorithm. The P-Evaluation is preferred since it not only ensures a legal transformation, but also provides economical computation in the data updating. On another aspect, both the C-Evaluation and the P-Evaluation could be adopted in an off-line algorithm since it does not require a legal transformation. An on-line algorithm could only choose the P-Evaluation. However, economical computation in the P-Evaluation relieves large resources consumed in an on-line algorithm. Consequently, the choice of C-Evaluation or P-Evaluation helps a good design of a cooperation reduction algorithm in an MSID.

5.2.5 Summary

Many issues deserve a serious study for solving an MSID with a cooperative reduction algorithm. Extending traditional reduction algorithms in influence diagrams, a cooperative reduction algorithm is proposed and developed in this section. It stands on a legal and efficient elimination sequence for both local influence diagrams and d-

sepnodes. It also involves a legal transformation discussed in both the C-Evaluation and the P-Evaluation.

In summary, a flow chart for solving MSID is described in Figure 5.7. It is a distributed and cooperative reduction algorithm. Firstly, after initializing an MSID, elimination sequences for each local influence diagram are identified through the process *GSL* . Secondly, in the process *GER* , a global elimination sequence is checked in the MSID. If a global elimination sequence exists in the MSID, a cooperative reduction algorithm is applied; otherwise, the MSID has to be solved with the implementation of an off-line algorithm. Finally, either the C-Evaluation or the P-Evaluation is selected to solve the MSID. It is also worthwhile to mention here that during the cooperation, an intelligent agent is in charge of its corresponding local influence diagrams, and only discloses information on the d-sepset thus protecting the agent's privacy.

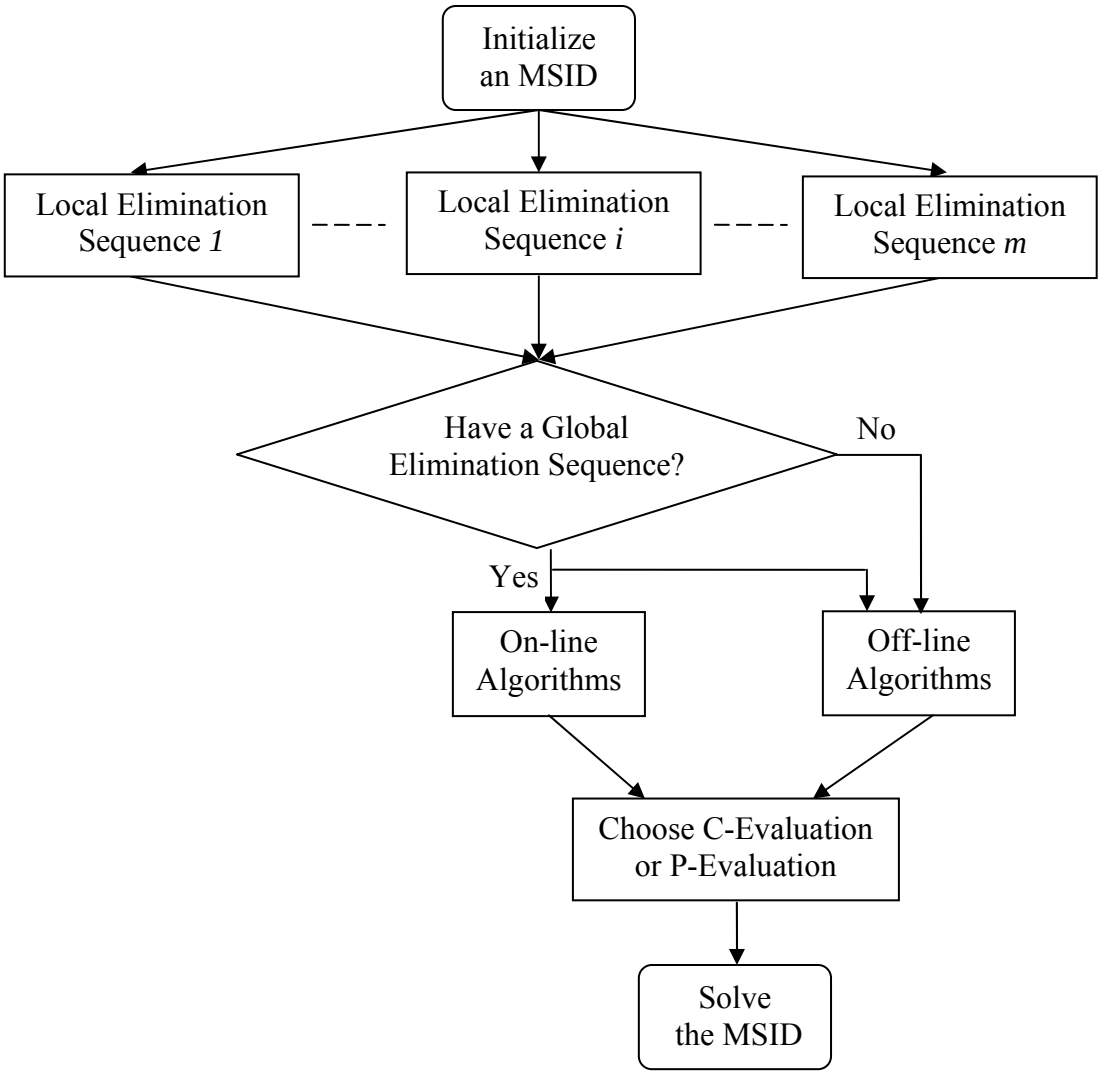


Figure 5.7: Flow Chart for Cooperative Reduction Algorithms

5.3 Distributed *evalID* Algorithm

The basic methods for solving an MSID are extensions of evaluation algorithms on single agent-based influence diagrams such as the reduction algorithm (Shachter 1986). The *evalID* algorithm is presently considered as one of the most efficient algorithms, especially in decision networks (Zhang 1998). Hence it may facilitate the design of evaluation algorithms for solving an MSID.

In this section, a distributed *evalID* algorithm is proposed to solve an MSID. A distributed *evalID* algorithm is well developed based on a framework of multiple evaluation networks which coordinates an evaluation of local influence diagrams. Multiple evaluation networks are composed of a set of evaluation networks which formulate basic ideas of the *evalID* algorithm for solving decision networks.

5.3.1 Evaluation Network

Before the concept of an evaluation network is described, the *evalID* algorithm is revisited in decision networks. The aim of the *evalID* algorithm is to solve decision networks by decomposing whole networks. Adopting the divide and conquer strategy, firstly, the *evalID* algorithm divides decision networks into two parts: a tail T and a body B. The tail only consists of relative nodes (in T_c and T_v which will be defined later.) with respect to the tail decision node d (the last decision in a decision sequence in decision networks). Hence, the tail can be solved with ease resulting in optimal rules for the tail decision node. Secondly, the *evalID* algorithm attaches an artificial value node u to the body. The value node u carries an evaluation function e_{T} from the optimal solution of the tail decision node. ($\pi_{d,r}$ denotes parents of decision node d and M_v denotes the maximum value of value node v.)

$$e_{\mathrm{T}}(\pi_{d,r}, d) = \frac{\sum_{v \in V_2} P_{\mathrm{T}_v}(v = 1, \pi_{d,r}, d) M_v}{P_{\mathrm{T}_c}(\pi_{d,r}) / |\Omega_d|}$$

where T_c is obtained from a tail T by pruning nodes outside of an ancestral set of $\pi_{d,r}$, T_v is obtained from a tail T by pruning nodes outside of an ancestral set of $\pi_{d,r}$ and $\{d, v\}$, and Ω_d is the number of possible values with respect to a decision node d. Thirdly, the *evalID* algorithm considers the identified body (in the first step) as new

decision networks and solves the body with the same methods applied to the original decision networks. Thus, a new tail and a new body are produced. Finally, the *evalID* algorithm reduces the original network into a body which has a single decision node. The final body is called the value network and is solved by the Cooper's transformation (Cooper 1988) which transforms decision nodes and value nodes into chances nodes. Accordingly, the final optimal decision rules are composed of decision rules for every tail node and the final decision node in the value network. The optimal decision value comes from the expected values for decision nodes in the corresponding tails and the value network.

In the *evalID* algorithm, decision networks are divided into several tails with respect to every decision node (a tail decision node) besides the final value network. Take the example (Jensen *et al.* 1994; Zhang 1998) in Figure 5.8. After identifying the tail $\mathrm{T}(d_i)$ for each decision node d_i step by step, the decision networks are divided into four BNs (BN_i, $i = 1,2,3,4$) and one value network (BN_0). (Decision nodes and value nodes except those in value networks are transformed into chance nodes c_m and v_m respectively through the Cooper's transformation; while u_m indicates the evaluation function e_{T} for the preceding tail decision node.) After the value network is transformed into a BN, there are five BNs connected with each other as shown in Figure 5.9. Hence, if the *evalID* algorithm is followed, the decision networks in Figure 5.8 could be solved when every BN is solved individually in the sequence (from BN_4 to BN_0). Now, the structure in Figure 5.9 is utilized to help the formulation of an evaluation network.

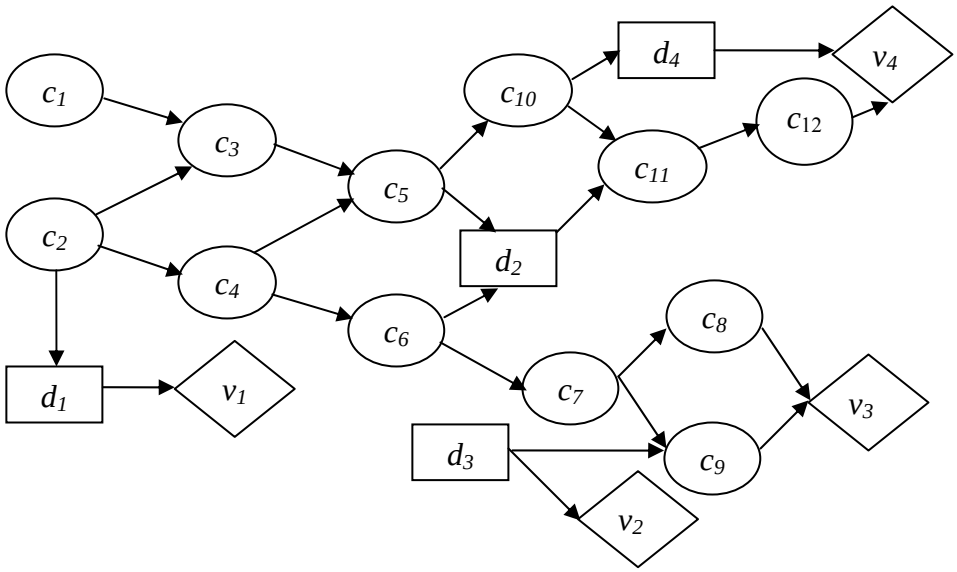


Figure 5.8: Decision Networks

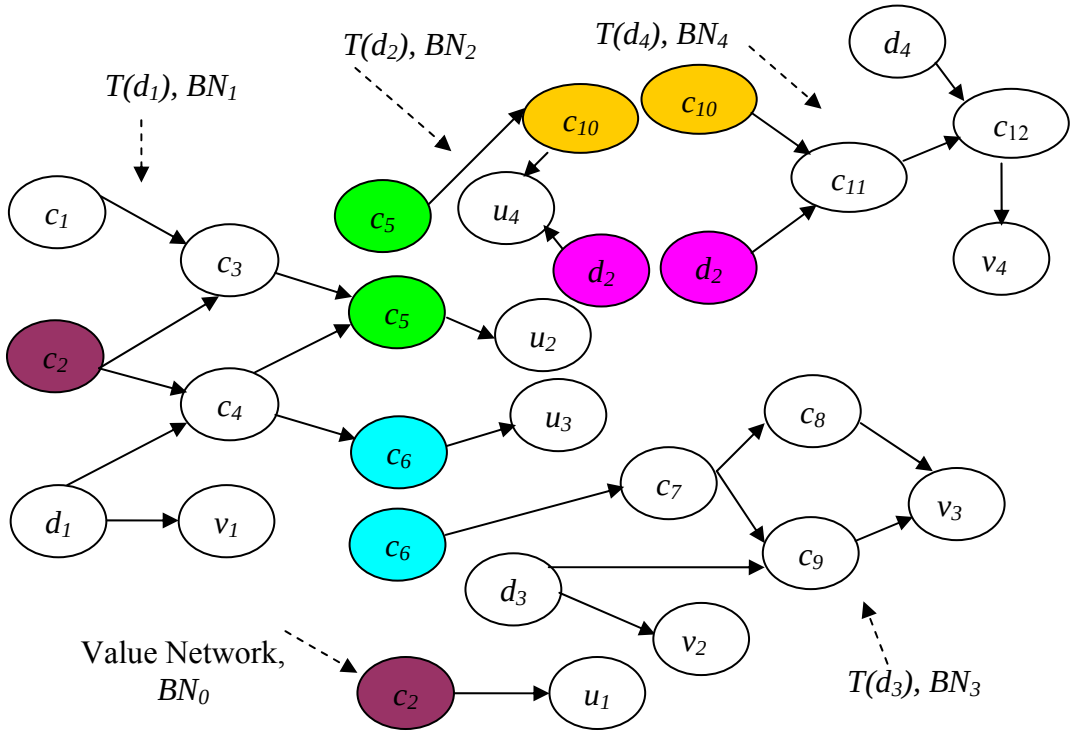


Figure 5.9: Tails (Corresponding BNs) in Decision Networks

Intuitively, the network in Figure 5.9 composed of five BNs (BN_i, $i = 0,1,2,3,4$) has an MSBN structure (here, an MSBN is considered in a graphical structure, not in a

hypertree level) (Xiang 2002). In fact, this is not a coincidence as demonstrated by the following proposition.

Proposition 5.1: A structure of networks (if T_c exists for every tail node d_j) composed of tails and a value network is the same as the graphical structure of an MSBN.

Proof. A graphical structure of an MSBN should follow two constraints: DAG structure and d-sepnode. In the *evalID* algorithm, value nodes (denoted as u_m) are added into the final networks. However, this operation could not violate the constraint of DAG structure since a value node is a sink node. Furthermore, the final networks composed of tails and value networks are obtained when some arcs are removed in the procedure of identifying tails and bodies in the original decision networks. Thus the final network should satisfy the DAG structure.

It is found that public nodes shared between tails $\mathrm{T}(d_i)$ and $\mathrm{T}(d_{i-1})$ ($i \geq 1$) are nodes in T_c belonging to $\mathrm{T}(d_{i-1})$. Besides, all parents of nodes in T_c of $\mathrm{T}(d_{i-1})$ belong to $\mathrm{T}(d_i)$. Hence, parents of these public nodes are included in the tail $\mathrm{T}(d_i)$ which is the body with respect to the tail $\mathrm{T}(d_{i-1})$. Also, a value network is the body of the tail $\mathrm{T}(d_1)$ (Assuming that d_i succeeds d_j if $i > j$.). Thus the second constraint is followed. ▌

Furthermore, it is noticed that the evaluation function e_{T} formulated with the variables T_{u_m} (T_{v_m}) and T_c has an additive form associated with artificial utility variables (u_m) and utility variables (v_m). This motivates the formulation of an evaluation network. For example, the evaluation network for decision networks discussed in Figure 5.8 and Figure 5.9 is constructed as shown in Figure 5.10.

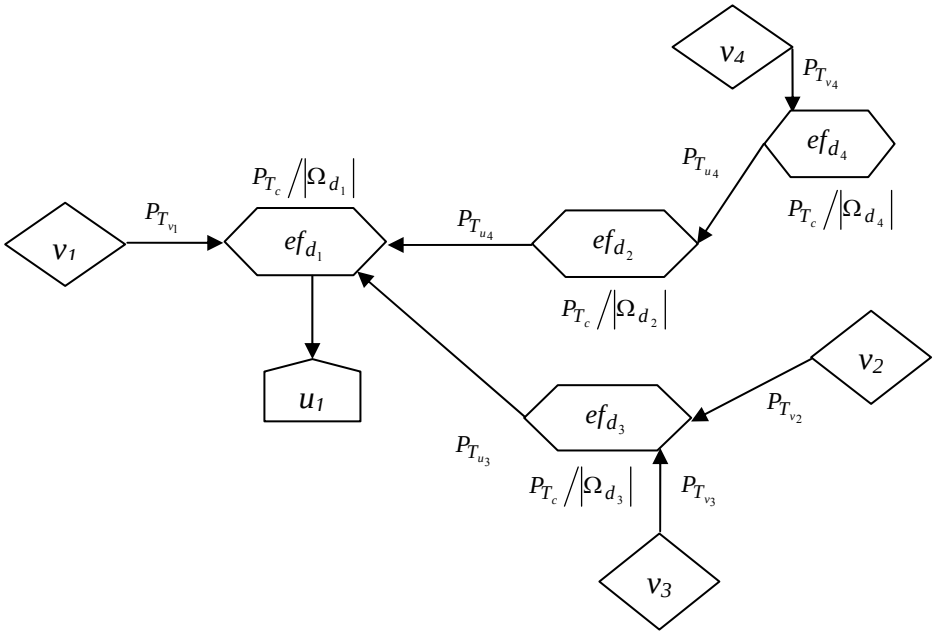


Figure 5.10: Evaluation Networks

An Evaluation Network (EN) is composed of three types of nodes: diamond, pentagon and hexagon nodes, and arcs between them. A diamond node denotes original value nodes in decision networks. A pentagon node denotes the final value network in decision networks, called sink node. (Since only one value network exists after the *evalID* algorithm is imposed on decision networks.) A hexagon node denotes tails in decision networks, called tail node. An arc from a value node to a tail node is called a utility arc while an arc between tail nodes is called an evaluation arc. (To avoid redundant denotations, nodes and variables are interchangeable in the representation.)

Based on utility and evaluation functions, an EN represents decision networks at two levels: quantitative level and qualitative level. At the qualitative level, an arc indicates the dependency between utility variables and evaluation functions in the tail node. Its direction tells the decision sequence that decides the computation of evaluation functions. Tracing directed paths in an EN results in a partial decision order. The

number of incident arcs into nodes in an EN indicates the number of items in the additive form of an evaluation function. A sink node terminates with the maximum expected value. For example, in Figure 5.10, the evaluation function in the tail node ef_{d_1} depends on the utility variable v_1 and the evaluation function ef_{d_2}. The partial decision order follows $d_1 \prec d_2 \prec d_4$.

At the quantitative level, a utility node is associated with a utility function and the output is the maximum value of the utility function. The tail node is framed with data on the ratio of the joint probability of T_c and the number of decision alternatives. It sums out the maximum value of an evaluation function after combining all maximum utility values and evaluation functions (from incident arcs). The sink node indicates a final computation in an EN. The arc incident into the tail node ef_{d_j} carries the joint probability of required nodes (with respect to d_j) to compute u_j (in tail node ef_{d_j}). For example, in Figure 5.10, the tail node ef_{d_1} keeps the data of $P_{T_c} / |\Omega_{d_1}|$ and receives the maximum utility values M_{v_1}, in addition to the maximum value M_{u_2} and M_{u_3} from the evaluation function on the tails ef_{d_2} and ef_{d_3}, respectively. At the same time, the tail node obtains the joint probability of $P_{T_{v_1}}$, $P_{T_{u_4}}$ and $P_{T_{u_3}}$ attached to three incident arcs individually. Hence, the output is the sum of some items as in the following equation.

$$e_{T(d_1)}(c_2, d_1) = \frac{P_{T_{v1}}(v_1 = 1, c_2, d_1) M_{v_1} + P_{T_{u2}}(u_2 = 1, c_2, d_1) M_{u_2} + P_{T_{v3}}(u_3 = 1, c_2, d_1) M_{u_3}}{P_{T_c}(c_2) / |\Omega_{d_1}|}$$

It can be seen that an EN clearly describes the components of the expected value in decision networks and the required computation in every component. An EN displays the dependency of the utility value with respect to decision nodes, which allows a

distributed evaluation algorithm in decision networks. An EN can be described as the extended form as shown in Figure 5.10 as well as other forms without the representation of numerical values.

5.3.2 Multiple Evaluation Networks

The definition of an HRG includes two functions: $\mathrm{Re}q$ and Sup. Nodes included in the functions connect local influence diagrams to form an MSID. Nodes in $\mathrm{Re}q$ consists of RO (requisite observation nodes) and RP (requisite probability nodes) that can be identified by *Decision Bayes-ball* or refined *Decision Bayes-ball* procedure[2]; while nodes in Sup can be removed safely without affecting the optimal solutions since they have no contribution to the evaluation on any decision node. Hence, the following proposition is obtained.

Proposition 5.2: No nodes in Sup are included in an EN.

In addition, tracing the process of obtaining tails and bodies in decision networks and the procedure *Decision Bayes-ball*, it is found that they are based on the same concept of d-separation in a probabilistic graph (Pearl 1988, Shachter 1998). Both procedures have the same goal to find RO and RP. Thus, the following Lemmas are obtained:

Lemma 5.2: Nodes in T are composed of required nodes, including RO and RP, obtained in the procedure *Decision Bayes-ball*.

Lemma 5.3: Nodes in $\pi_{d,r}$ are requisite observation nodes RO obtained in the procedure *Decision Bayes-ball*.

[2] In the following parts, *Decision Bayes-ball* is used to denote these two procedures.

Lemma 5.4: Nodes in T_v or T_u are required nodes, including *RO* and *RP*, obtained for the utility variables v and u in the procedure *Decision Bayes-ball*.

Hence both *RO* and *RP* connect local ENs with respect to local influence diagrams in an MSID. Both of them are involved in the computation of evaluation functions; however, they are not required to be evaluated simultaneously. In fact, the following proposition 5.3 can be proven.

Proposition 5.3: The evaluation of *RO* for decision node d_i is involved in the computation of tail node $ef_{d_{(i-1)}}$.

Proof. The computation of e_{T} could be analyzed as follows since $\pi_{d,r}$ are root nodes in T_v.

$$e_{\mathrm{T}}(\pi_{d,r},d)=\frac{\sum_{v\in V_2}P_{\mathrm{T}_v}(v=1,\pi_{d,r},d)M_v}{P_{\mathrm{T}_c}(\pi_{d,r})/|\Omega_d|}=\frac{P_{T_V}(\pi_{d,r})\times\sum_{v\in V_2}P_{\mathrm{T}_v/T_c}(v=1,\pi_{d,r},d)M_v}{P_{\mathrm{T}_c}(\pi_{d,r})/|\Omega_d|}$$

(T_v/T_c denotes node subtraction in two graphs.)

Thus the evaluation of *RP* is required in ef_{d_i} while the evaluation of *RO* is counted in the later $ef_{d_{(i-1)}}$. ▌

Consequently, the combination of ENs through public nodes, namely *RO* and *RP*, provides an efficient and distributed framework to solve an MSID. The framework is called Multiple Evaluation Networks (MEN) based on the algorithm *evalID*. Arcs from *RO* or *RP* to tail nodes in a local EN indicate that they are involved in the computation of an evaluation function.

For example, in Figure 5.11, it is assumed that RO and RP are public nodes among EN_1, EN_2 and EN_3. Each evaluation network is associated with a local influence diagram in an MSID. With the consideration of required nodes with respect to decision nodes, RO and RP are classified in an individual EN as follows: RO_{1,d_i} and RP_{1,d_i} are required nodes for decision node d_i in EN_1, RO_{2,d_k} and RP_{2,d_k} are required nodes for decision node d_k in EN_2 while RO_{3,d_j} and RP_{3,d_j} are required nodes for decision node d_j in EN_3. According to the MEN, the sequence for solving public nodes could also be defined. (In Figure 5.11, a dotted arc into a tail node indicates other tail nodes exist in the local EN.)

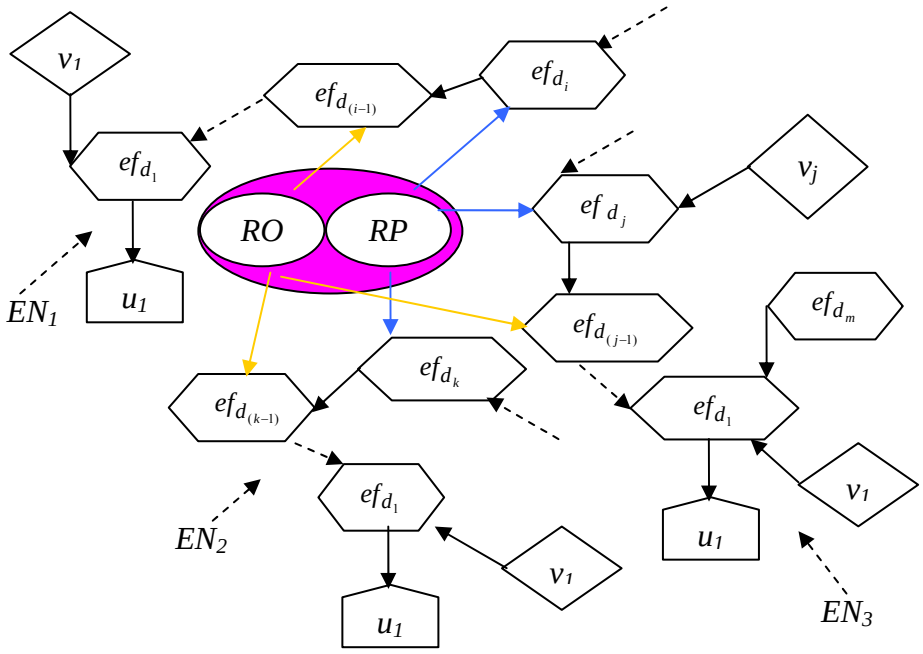


Figure 5.11: Multiple Evaluation Networks (MEN)

5.3.3 Distributed *evalID* Algorithms

Global optimal solutions for an MSID consist of global optimal strategies and global maximum expected values. In a multiagent decision problem, an agent is concerned

with its own utility and would not compromise its utility with the consideration of other agents' decisions. However, it wants the cooperation with the aim to make full use of available information to make the best decision for its own and to release some useful information in order to help other agents' decisions. Hence global optimal solutions depend on local optimal solutions in local influence diagrams and cooperative evaluation of public nodes. Agents expect to access the full and consistent information in a cooperative evaluation. Since the *evalID* algorithm ensures a local optimal solution in local influence diagrams, what is of concern is the evaluation of d-sepnodes in an MSID.

D-sepsets are channels through which agents hold good communication. Both *RO* and *RP* nodes in d-sepsets would affect a cooperative evaluation of local influence diagrams in an MSID. To ensure consistent information in a cooperative evaluation, both *RO* and *RP* nodes should join the evaluation and be removed simultaneously when solving an MSID. As shown above, an MEN indicates an appropriate sequence in the evaluation of *RO* and *RP* if the directions of arcs in a local EN are followed. Thus, every local EN in an MEN could be evaluated using the *evalID* algorithm individually while both *RO* and *RP* nodes could be evaluated and removed simultaneously from the MEN. Combining all the above discussion, a distributed algorithm *evalID* based on an MEN is formulated as follows

[Distributed *evalID* Algorithm]

1) Update all information in an MSID;
2) Build an EN for local influence diagrams in an MSID;
3) Build an MEN with the implication of an HRG;
4) Obtain a global evaluation sequence for *RO* and *RP* nodes based on the MEN;

5) Solve the MEN and produce optimal solutions.

In this algorithm, the first step is to update all numeric values associated with nodes in an MSID as well as the graphical structure. The updated MSID represents the latest decision scenario accurately. Then, for an individual local influence diagram, a corresponding EN is built through identifying tails and bodies in the subnet. After that, in the third step, all ENs are combined together to form an MEN through the linkage of d-sepnodes. Nodes associated with the Req function in the HRG are kept in the MEN and are classified into two types: RO and RP. Nodes associated with the Sup function are removed safely. Consequently, only RO and RP nodes in the d-sepset join the evaluation process They should be removed simultaneously to hold consistent information in a cooperative evaluation of local influence diagrams. Hence the sequence in which RO and RP nodes are evaluated is very important since it coordinates the evaluation of local ENs in the MEN. Since both RO and RP nodes are required for decision nodes associated with tail nodes in an EN, following the directions of arcs in the EN will generate a partial order for decision nodes locally. This partial order indicates the sequence for evaluating RO and RP nodes. Hence, in the fourth step, after obtaining a partial order for tail nodes in local ENs, a global evaluation sequence for RO and RP nodes could be produced. This sequence will guide the evaluation process in the MEN in the final step.

For model evaluation in an MSID, RO, together with RP, is vital to obtain a global optimal solution. Only if they are evaluated among local influence diagrams at the same time, will the full and consistent information associated with them be ensured in the whole evaluation process. The distributed *evalID* algorithm which is well designed above is to achieve this aim. Thus, Theorem 5.8 is obtained.

Theorem 5.8: A distributed *evalID* algorithm generates global optimal solutions in an MSID.

5.4 Indirect Evaluation Algorithm

In Sections 5.2 and 5.3, a cooperative reduction algorithm and a distributed *evalID* algorithm have been discussed in detail to solve an MSID. A cooperative reduction algorithm is one of the basic evaluation algorithms for solving an MSID. It directly solves influence diagrams with some basic operations such as node removal and arc reversal. On the other hand, a distributed *evalID* algorithm is an advanced evaluation algorithm for solving an MSID. It extends the *evalID* algorithm in decision networks. Both of these two algorithms belong to direct methods. They are a kind of reduction algorithms which remove elements from an MSID directly for solving the MSID.

The principle of direct methods for solving an MSID is easy to be understood and formulated in an intuitive way. However, it needs some efforts to design a good evaluation process. As we know, the indirect method for solving influence diagrams (Jensen *et al.* 1994) is an alternative approach in the model evaluation. It first transforms an influence diagram into a rooted cluster tree (Jensen *et al.* 1994); then, it solves the rooted cluster tree. Furthermore, as the process of transforming an MSID is similar to that in an MSBN (Xiang 1996), I would like to discuss an indirect method for solving an MSID in this section. Some relevant strategies in this indirect method will be highlighted.

5.4.1 Algorithm Design

Like solving influence diagrams, indirect evaluation algorithms to solve an MSID follow two steps: transforming an MSID into a multiple rooted cluster tree composed of a set of interconnected rooted cluster trees (Shachter 1999), and solving the multiple rooted cluster tree.

1) Transforming an MSID: The methods to transform an MSID into a multiple rooted cluster tree are combinations of evaluation algorithms in both influence diagrams and MSBN (Jensen *et al.* 1994; Shachter 1999; Xiang 2002). Firstly, a modified MSID is being built. For each local influence diagram, requisite observation nodes are identified using *Decision Bayes-ball* procedure. Then, arcs are added between identified required nodes and their corresponding decision nodes. Finally, a multiple rooted cluster tree is built in a cooperative algorithm (Xiang 2002). The basic idea of this algorithm is to transform local influence diagrams into a local rooted cluster tree as well as to transform d-sepsets into linkage trees. The linkage trees connect the transformed local rooted cluster trees together into a multiple rooted cluster tree. In this process, an index is introduced to show a clique sequence in the local rooted cluster tree.

2) Solving a multiple rooted cluster tree: A multiple rooted cluster tree is a tree structure for an MSID. It is a set of local rooted cluster trees connecting with the linkage trees. Thus, evaluating an MSID amounts to solving these local rooted cluster trees as well as the linkage trees. The local rooted cluster tree is evaluated following a partial decision order in a local influence diagram. The linkage tree among local rooted cluster trees coordinates the computation of adjacent cliques to make sure that d-sepnodes are evaluated at the same time.

Hence, d-sepnodes in a linkage tree could be removed simultaneously. This has to involve a relative elimination order for d-sepnodes in an MSID. The relative elimination order could be obtained by considering the clique index in local rooted cluster trees globally. A good relative index on cliques allows the linkage trees to be evaluated cooperatively among adjacent local rooted cluster trees.

In summary, an MSID is solved in a distributed way following the above two steps. In this work, I do not focus on the process of developing a multiple rooted cluster tree since it could be found in detail in much literature (Jensen *et al.* 1994; Shachter 1999; Xiang 2002). I emphasize more on the solving of linkage trees in a multiple rooted cluster tree, which is illustrated in the following case.

5.4.2 Evaluation of SARS Control Situation

Take the example in Figure 3.1. The three influence diagrams I_1, I_2 and I_3 are transformed into three local rooted cluster trees T_1, T_2 and T_3 respectively. The d-sepsets are transformed into linkage trees. For the MSID in Figure 3.1, the d-sepset transformation is simple because each chance node in the d-sepset composes one linkage tree. These local rooted cluster trees and linkage trees compose a multiple rooted cluster tree in Figure 5.12.

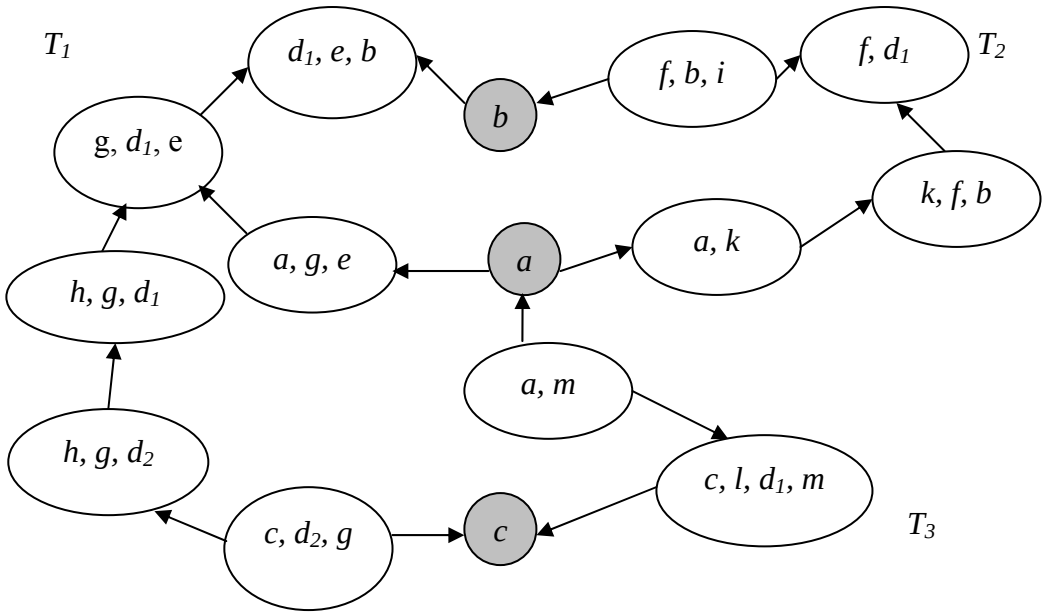


Figure 5.12: A Multiple Rooted Cluster Tree
(Nodes without color denote cliques and nodes with grey color denote linkage trees.)

The d-sepnodes in the linkage tree adjust the evaluation of cliques in adjacent local cluster rooted trees since they have to be removed simultaneously. The removal of d-sepnodes must follow an elimination order in a local cluster rooted tree. For example, in T_1, nodes $\{a,c\}$ have to be removed before node b although there is no fixed elimination order for nodes $\{a,c\}$. However, in T_3, the removal of node a precedes the removal of node c. Hence a legal elimination order for d-sepnodes in the computation of the multiple rooted cluster tree exists: node a is removed before the node c which should be removed before node b. It is also noticed that the arc associated with the linkage tree including node c is from T_1 to T_3 in Figure 5.12. This follows with the implication of the HRG in Figure 3.3. It indicates that agent A_1 controls agent A_3's decision d_1 by the information c. For the linkage tree among T_1, T_2 and T_3, the arc

direction depends on an elimination order for d-sepnodes if this order exists; otherwise, it is optional.

Following the above procedures, the MSID for the SARS model is solved by evaluating local influence diagrams individually and combining all local optimal solutions. The consistent information, implicated in the d-sepset in an MSID and restricted in organizational relationships in an HRG, is guaranteed throughout the evaluation process. If the shared information is inconsistent, the decision making is prone to be wrong. For example, with reference to the MSID in Figure 3.1, if Nation 1 deliberately hides its SARS report (represented by node c), decision d_1 in Nation 3 will definitely be wrong. Consequently, decisions in the whole community will lead to quite a large loss, which has been the fact in the SARS control in 2003. Accordingly, a true model of MSID and HRG not only provides good decisions for policy makers, but also allows a large number of simulations to report some alerts in order to avoid a destructive loss.

5.5 Comparison on the Three Evaluation Algorithms

Solving a multiagent decision problem is to seek the best decision for multiple agents in a cooperative and privacy protection setting. Every agent aims to obtain an optimal solution while it cooperates with its adjacent agents. An agent, without disclosing its privacy, wants to access the full and consistent public information in order to maximize the decision value. Here, the global optimal solution of the multiagent decision problem is defined as the combination of best decision solutions from multiple agents which have full and consistent observations.

An MSID represents a multiagent decision problem with a set of local influence diagrams. These local influence diagrams are connected with d-sepsets. Through the information in the d-sepset, local influence diagrams could affect their adjacent ones. Hence, in the evaluation process, some relevant strategies should be adopted to avoid the incomplete and inconsistent information in the d-sepset. To achieve this goal, d-sepnodes should follow a uniform elimination order in an evaluation algorithm so that they could be evaluated and removed from an MSID simultaneously. Through this way, it is believed that an agent has obtained a full, correct and consistent observation so that each agent's benefit is maximized in a distributed and unpredictable situation.

Following the above principles residing in evaluation algorithms for solving an MSID, the three evaluation algorithms, namely cooperative reduction algorithm, distributed *evalID* algorithm and indirect evaluation algorithm, have been discussed in the previous sections. In general, all of these three evaluation algorithms can be utilized to solve an MSID. However, every method has its own advantages. For example, a distributed *evalID* algorithm could deal with a general MSID while a cooperative reduction algorithm could only deal with a regular MSID in which local influence diagrams are regular. A cooperative reduction algorithm is easily understood and is suitable to solve an MSID with no large dimension; however, a good formulation needs much effort. Moreover, both an indirect evaluation algorithm and a distributed *evalID* algorithm could make full use of Bayesian inference algorithms in the evaluation process. Accordingly, it is hoped that some strategies be designed with the aim to select an appropriate evaluation algorithm for solving an MSID.

On the other hand, the three evaluation algorithms are special forms for the solving of an MSID by removing nodes from the MSID step by step. In fact, they have a common

interest to decide an elimination order for nodes in the MSID. Hence, the three evaluation algorithms could be generalized into simple procedures for identifying the elimination order for nodes. They could be used to solve general graphical decision models.

5.6 Summary

This chapter described three evaluation algorithms, namely cooperative reduction algorithm, distributed *evalID* algorithm, and indirect evaluation algorithm, for solving an MSID. All of these three evaluation algorithms extend those for solving influence diagrams or decision networks. A cooperative reduction algorithm extends basic reduction algorithms in influence diagrams. A distributed *evalID* algorithm is based on the *evalID* algorithm in decision networks while an indirect evaluation algorithm originates from the junction tree method for solving influence diagrams. The three algorithms are analyzed theoretically as well as illustrated in some simple examples.

[This page intentionally left blank]

6 Case Study

This chapter illustrates a comprehensive case study – policy design for avoiding more loss due to the SARS in 2003 - which extends the example in the medical domain in Chapter 3. This work puts together my proposed methodologies of Multiply Sectioned Influence Diagrams (MSID) and Hyper Relevance Graph (HRG) which consist of model representation, model verification and model evaluation discussed in Chapters 3, 4, and 5, respectively. It shows how these methods can be utilized to address a complex practical problem.

6.1 Decision Scenario

Policy design for controlling disease spread is one of the important issues in the medical domain. For example, currently, the disease of bird flu seems to be widespread in the world and is endangering human beings without notice. The involved or uninvolved nations (communities) have been trying to design some policies to avoid the further spread of the disease. Although they are separated geographically, they want to cooperate with each other to alleviate the loss through a full exchange of some useful and consistent information. Hence, this is a kind of multiagent decision problems. Besides, the control on the Severe Acute Respiratory Syndrome (SARS) case is another practical decision problem in the medical domain. In Chapter 3, I have briefly described the decision scenario. Now, this case is extended here and discussed in detail.

The SARS is a serious infectious disease that could potentially develop into an epidemic or even an endemic. Its outbreak causes unexpected loss everywhere in the world. Its uncertain and various sources have frustrated the medical community and policy makers. Beyond all doubt, it needs the collaborative effort of multiple nations concerned with their own local as well as global benefits. Assume that there are several nations or communities ($A_1 \sim A_5$) which are involved in a decision scheme for the SARS control.

Nation A_1 concentrates on the SARS control in an airport and plans to build quarantine centers around the airport. It is concerned with decisions of changing the airline schedule denoted by[3] d_{11}, having the SARS screening in the airport d_{12} and building quarantine centers d_{13}. The relevant uncertain variables are as follows: (1) The SARS situation indicated in the WHO's report a; (2) Permission from airlines b_1; (3) Sign of the SARS with the fever symptom c_1; (4) The total number of customers in the airport d; (5) Health condition of overseas customers h; (6) A_1's hospital facilities f; (7) Loss without controlling the SARS in the airport g_1. Nation A_1 is concerned with the benefit-cost of building quarantine centers around the airport v_1.

Nation A_2 puts its effort to control and prevent the SARS in the society. Hence, it needs to investigate some decisions such as adopting the home quarantine measure d_{21}, arranging more ambulances around communities d_{22}, and building special hospitals d_{23}. The involved uncertain variables are as follows: (1) The SARS situation indicated in the WHO's report a; (2) The SARS situation

[3] Subsequently, in this case, the words "denoted by" will be omitted before the symbol which is the variable representing the front sentence.

indicated in nation A_2's report d_2; (3) The SARS distribution c_2 in nation A_2; (4) The SARS situation indicated b in the official report of nation A_2; (5) The SARS virus spread e_2; (6) The loss without controlling the SARS f_2; (7) Nation A_2's hospital facilities g. Nation A_2 is concerned with the benefit-cost of building special hospitals v_2.

Nation A_3 intends to avoid the input of the SARS virus and considers the citizens' health. Thus it has a plan d_{31} whether to reschedule its tour groups to nation A_2 in which there is an outbreak of the SARS. The decision involves the following uncertain variables: (1) The SARS situation indicated in the WHO's report a; (2) The SARS situation indicated in nation A_2's report b; (3) The SARS situation indicated in nation A_3's report a_3; (4) Possibility of catching the SARS when traveling to nation A_2 c_3; (5) Loss for travelers when catching the SARS d_3; (6) Economic loss for tour groups e_3; (7) The number of travelers to nation A_2 f_3. Hence nation A_3 is concerned with the benefit-cost of a tour in nation A_2 v_3.

Nation A_4 is a neighbor of nation A_1 and has a wide affiliation with nation A_1. For example, there are many people traveling between nation A_1 and nation A_4 for business or education. Hence, the joint effort from both nation A_4 and nation A_1 contributes a lot to the SARS control. It is a benefit that both nations could release some relevant information if new measures are taken. Here, nation A_4 investigates some foreign affairs with A_1 such as controlling the quota of travelers d_{41} and requiring medical examinations from travelers d_{42}. Many uncertain variables exist as follows: (1) The SARS situation indicated in nation

A_1's report a_1 (depends on nation A_4's evaluation); (2) The SARS situation indicated in nation A_4's report b_4; (3) Profit for applicants to nation A_1 c_4; (4) Loss for travelers with the SARS g_4; (5) Possibility of identifying the SARS through the health checking e_4; (6) Cost of the health checking f_4; (7) Fluctuation of travelers to nation A_1 after controlling the quota of travelers d; (8) Health status of travelers to nation A_1 h. Nation A_4 considers the benefit-cost for applicants to nation A_1 v_4.

Community A_5 is one of the international organizations that would like to offer some assistances to the SARS-affected nations (A_1 and A_2) such as distributing the medicine d_{51}, sending out rescuers d_{52} and developing the SARS vaccine d_{53}. It has to consider many uncertain variables as follows: (1) The SARS situation indicated in the WHO's report a; (2) The SARS situation indicated in nation A_1's report c_5 (depends on community A_5's evaluation); (3) The SARS situation indicated in nation A_2's report b; (4) Nation A_1's hospital facilities f; (5) Nation A_2's hospital facilities g; (6) Effect of the distributed medicine d_5; (7) Effect of the SARS vaccine e_5; (8) Risk of rescuers h_5; (9) Passion of volunteers i_5; (10) Benefit of local patients j_5. Community A_5 has to consider the benefit-cost of both rescuer assignment v_{51} and vaccine development v_{52}.

6.2 Model Formulation

Clearly, the decision scenario above is a typical multiagent decision problem. The involved organizations $A_1 \sim A_5$ are interrelated with each other and share some common information. Each nation or community is considered as an agent $A_1 \sim A_5$

respectively and is modeled as a local influence diagram in an MSID. Hence, there are five local influence diagrams $I_1 \sim I_5$ in the MSID as shown in Figure 6.1. (Chance nodes with grey color indicate public nodes among local influence diagrams.) The public information is encoded in d-sepsets in the MSID and exerts its influence on the decision making while agents' privacy is protected in local influence diagrams.

Also, their organizational relationships can be represented in an HRG. On the basis of the MSID in Figure 6.1, the corresponding HRG can be built in Figure 6.2. The HRG characterizes organizational relationships among the involved nations. The HRG shows that the international organization A_5 has strong and complicated relationships with other nations since it makes its decisions based on the input information from adjacent entities. For example, A_2 controls A_5's decision d_{52} with the required information $\{b, g\}$. It is noticed that the information g depends on A_2's decision d_{21} while the information b is A_2's judgment which is not affected by A_2's decisions. Hence, A_2's decision d_{21} may affect A_5's decision d_{53}. For the adjacent agents A_1 and A_5, the information f controls both A_1's decision d_{13} and A_5's decision d_{51} so that the authentic information benefits their decisions. A_4 supports A_1 with the information d while controls A_1's decision d_{12} with the information $\{h\}$. Finally, it is noticed that the information a is a key element that connects most agents. It may be the required information for some agents' decisions such as A_5's decision d_{51} and A_2's decision d_{23}. On the other hand, the information a may be the only public information shared among adjacent agents. For example, it can be accessed by agents A_1 and A_3; however, it is not needed for the computation of their decisions.

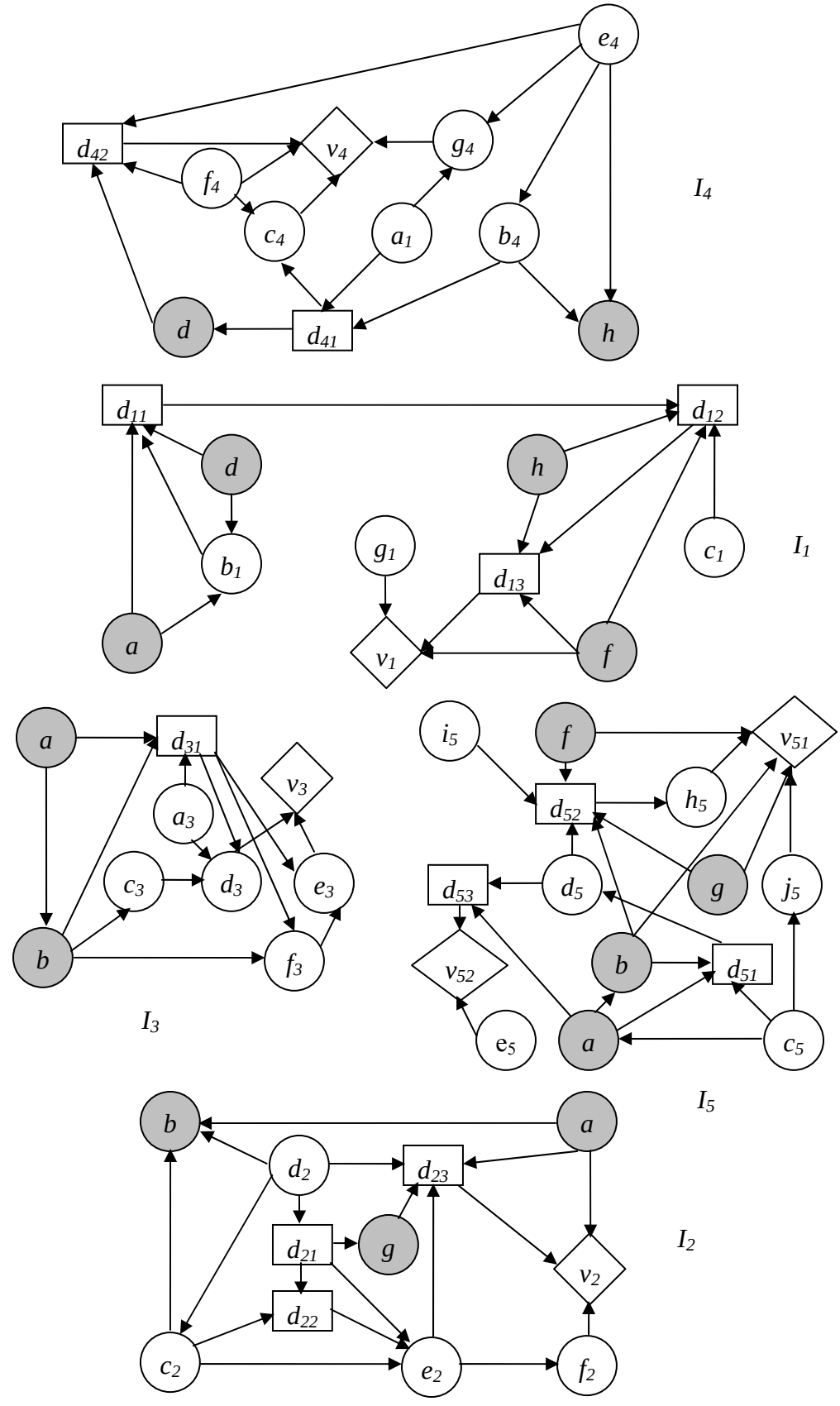


Figure 6.1: The MSID

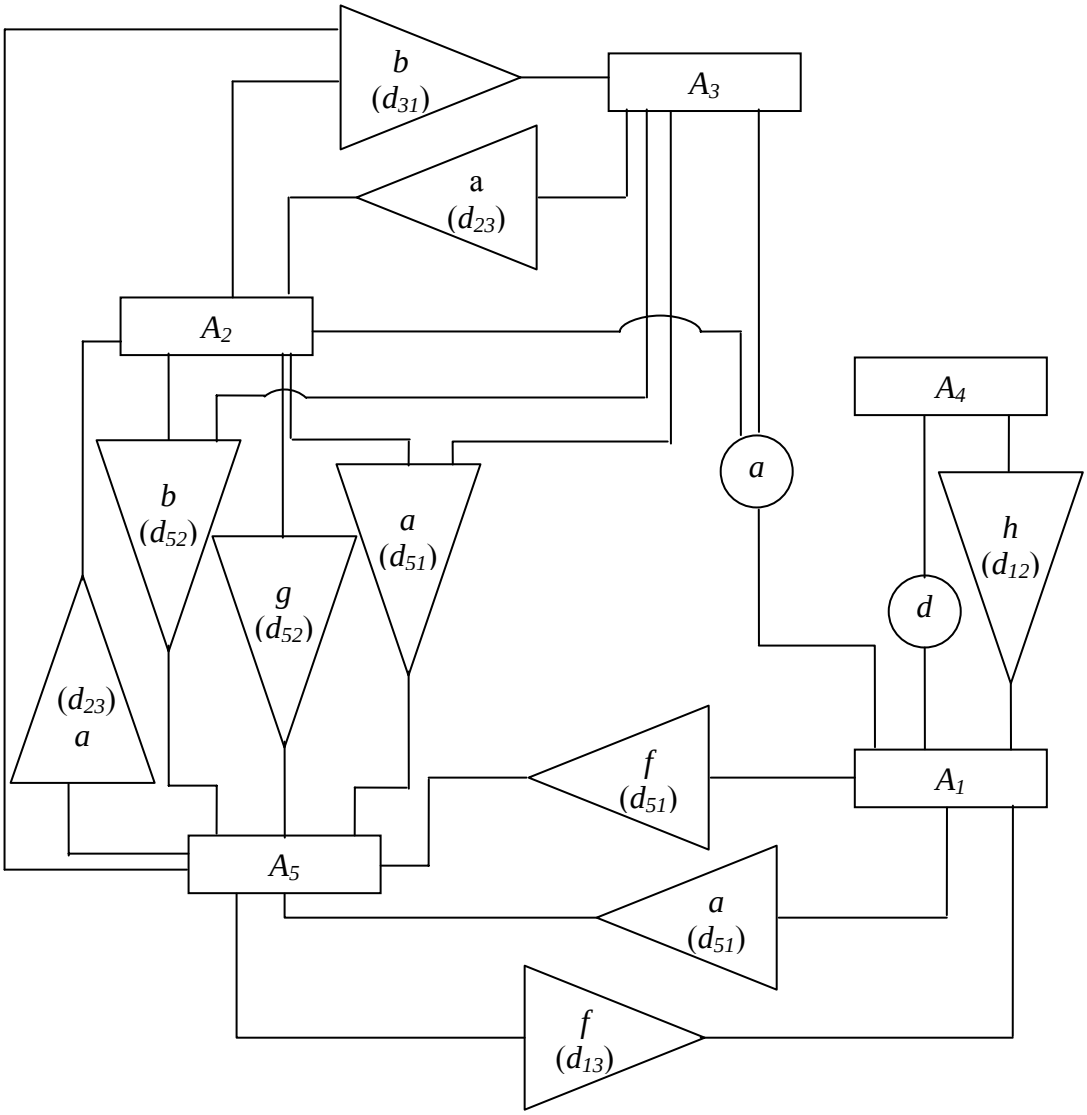


Figure 6.2: The HRG for the MSID in Figure 6.1

Concerning the information *a* that indicates the SARS status in the WHO's report, agents may have different judgments on this observation. In this case, an interval probability could be utilized to unify their beliefs. It allows the involved nations to make a good decision with the consideration of the WHO's report.

This case study shows that graphical decision models of MSID and HRG can represent a large and complex decision problem involving multiple agents. Furthermore, the

decision models are reusable and scalable. Any of the existing local influence diagrams can be removed at any time without affecting other components. At the same time, new components can be added into the existing model through connecting them with relevant parts. For example, in the MSID shown in Figure 6.1, I_4 can be removed without changing the structures of I_1, I_2, I_3 and I_5. The nation that recently involves the SARS may join the global control group easily only if a corresponding local influence diagram is put into the existing MSID.

6.3 Model Verification

Model verification amounts to checking whether the three constraints of MSID and HRG are obeyed: DAG structures, d-sepset of agent interface and irreducible d-sepset. I adopt the symbolic methods in Chapter 4 to verify the MSID and HRG in Figure 6.1 and Figure 6.2 respectively.

6.3.1 Verification of DAG Structures

The task of verifying a DAG structure is to ensure that there is no directed cycle in the MSID as shown in Figure 6.1. According to Theorem 4.1, firstly, value nodes can be removed from local influence diagrams. After that, the algebraic description for each local influence diagram $I_1 \sim I_5$ in the MSID is obtained as follows:

$$p_{I_1} = \frac{p(a)}{1} \times \frac{p(b_1, d, a)}{p(d, a)} \times \frac{p(d_{11}, d, a, b_1)}{p(d, a, b_1)} \times \frac{p(d)}{1} \times \frac{p(d_{12}, d_{11}, h, f, c_1)}{p(d_{11}, h, f, c_1)} \times \frac{p(h)}{1} \times \frac{p(c_1)}{1} \times \frac{p(f)}{1} \times \frac{p(g_1)}{1} \times \frac{p(d_{13}, h, d_{12}, f)}{p(h, d_{12}, f)} \tag{6.1}$$

$$p_{I_2} = \frac{p(a)}{1} \times \frac{p(d_{23}, a, d_2, e_2, g)}{p(a, d_2, e_2, g)} \times \frac{p(f_2, e_2)}{p(e_2)} \times \frac{p(e_2, d_{21}, d_{22}, c_2)}{p(d_{21}, d_{22}, c_2)} \times \frac{p(c_2, d_2)}{p(d_2)} \times \frac{p(d_2)}{1} \times \frac{p(d_{21}, d_2)}{p(d_2)} \times \frac{p(b, a, d_2, c_2)}{p(a, d_2, c_2)} \times \frac{p(d_{22}, d_{21}, c_2)}{p(d_{21}, c_2)} \times \frac{p(g)}{1} \quad (6.2)$$

$$p_{I_3} = \frac{p(a)}{1} \times \frac{p(b, a)}{p(a)} \times \frac{p(d_{31}, a, b, a_3)}{p(a, b, a_3)} \times \frac{p(c_3, b)}{p(b)} \times \frac{p(f_3, b, d_{31})}{p(b, d_{31})} \times \frac{p(e_3, f_3, d_{31})}{p(f_3, d_{31})} \times \frac{p(a_3)}{1} \times \frac{p(d_3, a_3, c_3, d_{31})}{p(a_3, c_3, d_{31})} \quad (6.3)$$

$$p_{I_4} = \frac{p(a_4)}{1} \times \frac{p(g_4, a_4, e_4)}{p(a_4, e_4)} \times \frac{p(e_4)}{1} \times \frac{p(b_4, e_4)}{p(e_4)} \times \frac{p(h, b_4, e_4)}{p(b_4, e_4)} \times \frac{p(d_{41}, a_4, b_4)}{p(a_4, b_4)} \times \frac{p(d, d_{41})}{p(d_{41})} \times \frac{p(c_4, f_4, d_{41})}{p(f_4, d_{41})} \times \frac{p(f_4)}{1} \times \frac{p(d_{42}, e_4, d, f_4)}{p(e_4, d, f_4)} \quad (6.4)$$

$$p_{I_5} = \frac{p(i_5)}{1} \times \frac{p(d_{52}, b, i_5, f, d_5, g)}{p(b, i_5, f, d_5, g)} \times \frac{p(f)}{1} \times \frac{p(h_5, d_{52})}{p(d_{52})} \times \frac{p(d_5, d_{51})}{p(d_{51})} \times \frac{p(d_{53}, d_5, a)}{p(d_5, a)} \times \frac{p(e_5)}{1} \times \frac{p(b, a)}{p(a)} \times \frac{p(a, c_5)}{p(c_5)} \times \frac{p(d_{51}, b, a, c_5)}{p(b, a, c_5)} \times \frac{p(c_5)}{1} \times \frac{p(j_5, c_5)}{p(c_5)} \times \frac{p(g)}{1} \quad (6.5)$$

Based on the above algebraic description, the sets of *DH* and *DT* for each local influence diagram are identified as shown in Table 6.1.

Table 6.1: *DH* and *DT*

I_j	$DH(I_j)$	$DT(I_j)$
I_1	$DH(I_1) = \{\overline{f}, \overline{h}, \overline{d}, \overline{a}\}$	$DT(I_1) = \Phi$
I_2	$DH(I_2) = \{\overline{a}, \overline{g}\}$	$DT(I_2) = \{\underline{g}, \underline{b}\}$
I_3	$DH(I_3) = \{\overline{a}, \overline{b}\}$	$DT(I_3) = \{\underline{b}\}$
I_4	$DH(I_4) = \{\overline{d}\}$	$DT(I_4) = \{\underline{d}, \underline{h}\}$
I_5	$DH(I_5) = \{\overline{f}, \overline{g}, \overline{a}, \overline{b}\}$	$DT(I_5) = \{\underline{b}, \underline{a}\}$

Then, the two operations 4.1 and 4.2 in Chapter 4 are used to find *DPs* for local influence diagrams. Only one *DP* is found in I_3. That is $DP_{s_3} = \langle \overline{a} \bullet \underline{b} \rangle$. Similarly, the

path appears in I_2 and I_5. In this case, no directed cycle would be formed. Hence, it is verified that the MSID in Figure 6.1 follows the constraint of DAG structure.

6.3.2 Verification of D-sepset

Verification of d-sepset is to check whether all common nodes are d-sepnodes in the MSID as shown in Figure 6.1. It is equivalent to verifying whether all the parents of every common node belong to at least one local influence diagram in the MSID.

According to the algebraic descriptions (Equations 6.1~6.5), first, the *SPS* for each common node is obtained in Table 6.2.

Table 6.2: *SPS* for Common Nodes

Common node	SPS_{I_j}
a	$SPS_{I_1}(a)=\Phi$, $SPS_{I_2}(a)=\Phi$, $SPS_{I_3}(a)=\Phi$, $SPS_{I_5}(a)=\{c_5\}$
b	$SPS_{I_2}(b)=\{a,d_2,c_2\}$, $SPS_{I_3}(b)=\{a\}$ $SPS_{I_5}(b)=\{a\}$
d	$SPS_{I_1}(d)=\Phi$, $SPS_{I_4}(d)=\{d_{41}\}$
g	$SPS_{I_2}(g)=\{d_{21}\}$, $SPS_{I_5}(g)=\Phi$
h	$SPS_{I_1}(h)=\Phi$, $SPS_{I_4}(h)=\{b_4,e_4\}$
f	$SPS_{I_1}(f)=\Phi$, $SPS_{I_5}(f)=\Phi$

After that, the union operation is used to get the *PS* for each common node as shown in Table 6.3.

Table 6.3: *PS* for Common Nodes

Common Nodes	*PS*
a	$PS(a) = \{c_5\}$
b	$PS(b) = \{a, d_2, c_2\}$
d	$PS(d) = \{d_{41}\}$
g	$PS(g) = \{d_{21}\}$
h	$PS(h) = \{b_4, e_4\}$
f	$PS(f) = \Phi$

Finally, the step of D-Testing compares *SPS* in Table 6.2 with *PS* in Table 6.3 to get Table 6.4. It can be seen that the d-sepset of agent interface in the MSID is verified successfully.

Table 6.4: Final Results

Common Node	**Results**	**D-sepnode?**
a	$PS(a) = SPS_{I_5}(a) = \{c_5\}$	YES
b	$PS(b) = SPS_{I_2}(b) = \{a, d_2, c_2\}$	YES
d	$PS(d) = SPS_{I_4}(d) = \{d_{41}\}$	YES
g	$PS(g) = SPS_{I_2}(g) = \{d_{21}\}$	YES
h	$PS(h) = SPS_{I_4}(h) = \{b_4, e_4\}$	YES
f	$PS(f) = SPS_{I_1}(f) = \Phi$ or $PS(f) = SPS_{I_5}(f) = \Phi$	YES

6.3.3 Verification of Irreducibility

The irreducibility has to concern with whether both MSID and HRG are the exact representation of domain knowledge. In fact, in the case of Figure 6.1 and Figure 6.2,

the decision models were built in a sequential way. Under this case, the HRG in Figure 6.2 is driven from the MSID in Figure 6.1. Hence, at this stage, the domain knowledge would help to verify the accurate representation of decision models. This is a bit subjective since the verification process largely depends on domain experts and model builders. Thus, I focus on the pairwise verification of the MSID and HRG in this section.

The main task of pairwise verification is to judge the property of d-sepnodes in the MSID and to compare the results with the HRG. Through the procedure (*refined*) *Decision Bayes-ball*, d-sepnodes are identified as to whether they are required chance nodes for decision nodes in local influence diagrams. Then, the correct representation of the HRG is evaluated. The whole results are shown in Table 6.5.

Table 6.5: Pairwise Verification

Pair Local Influence Diagrams I_i and I_j	S_{ij}	$\mathrm{Re}\,q(A_i, A_j, d_k) = \{c_1, \cdots, c_m\}$ **and** $Sup(A_i, A_j) = \{c_1, \cdots, c_n\}$
I_1 and I_2	$S_{12} = \{a\}$	$Sup(A_1, A_2) = \{a\}$
I_1 and I_3	$S_{13} = \{a\}$	$Sup(A_1, A_3) = \{a\}$
I_1 and I_5	$S_{15} = \{a, f\}$	$\mathrm{Re}\,q(A_1, A_5, d_{51}) = \{f\}$, $\mathrm{Re}\,q(A_1, A_5, d_{51}) = \{a\}$ and $\mathrm{Re}\,q(A_5, A_1, d_{13}) = \{f\}$
I_1 and I_4	$S_{14} = \{d, h\}$	$\mathrm{Re}\,q(A_4, A_1, d_{12}) = \{h\}$ and $Sup(A_1, A_4) = \{d\}$
I_2 and I_3	$S_{23} = \{a, b\}$	$\mathrm{Re}\,q(A_3, A_2, d_{23}) = \{a\}$ and $\mathrm{Re}\,q(A_2, A_3, d_{31}) = \{b\}$
I_2 and I_5	$S_{25} = \{a, b, g\}$	$\mathrm{Re}\,q(A_5, A_2, d_{23}) = \{a\}$, $\mathrm{Re}\,q(A_2, A_5, d_{52}) = \{b, g\}$, and $\mathrm{Re}\,q(A_2, A_5, d_{51}) = \{a\}$
I_3 and I_5	$S_{35} = \{a, b\}$	$\mathrm{Re}\,q(A_5, A_3, d_{31}) = \{b\}$, $\mathrm{Re}\,q(A_3, A_5, d_{52}) = \{b\}$, and $\mathrm{Re}\,q(A_3, A_5, d_{51}) = \{a\}$

From Table 6.5, it is verified that each pair of local influence diagrams satisfies the equation: $S_{ij} \subseteq (((\bigcup_k \mathrm{Re}\, q(A_i, A_j, d_k)) \cup (\bigcup_k \mathrm{Re}\, q(A_j, A_i, d_k))) \cup Sup(A_i, A_j)))$. Hence, the constraint of irreducible d-sepset is followed.

So far, the verification of the MSID and HRG in Figure 6.1 and Figure 6.2 is completed. However, the output of valid decision models could be further refined by domain experts and model engineers.

6.4 Model Evaluation

Model evaluation is to solve decision models in order to obtain optimal decisions for decision makers. In Chapter 5, three evaluation algorithms, namely cooperative reduction algorithm, distributed *evalID* algorithm, and indirect evaluation algorithm, have been proposed to solve an MSID. The core of these evaluation algorithms is to design various strategies in order to remove d-sepnodes simultaneously. Through this way, it is ensured that the consistent and updated information, especially in the d-sepset, is involved in solving an MSID. Hence, building a framework which guides the process of removing d-sepnodes from the MSID is vital. On another aspect, as discussed in Chapter 5, these three evaluation algorithms have their own advantages for solving an MSID. In fact, this phenomenon depends on the corresponding structure of local influence diagrams in an MSID. It requires some strategies to choose a suitable evaluation algorithm for solving a local influence diagram. This topic will be covered in my future work. In this section, a hybrid evaluation algorithm is designed to solve the MSID in Figure 6.1. It is composed of the three evaluation algorithms in Chapter 5. Each evaluation algorithm is selected to solve a local influence diagram by considering the graphical structure of each local influence diagram. The hybrid evaluation

algorithm is designed as shown in Table 6.6. In this hybrid evaluation algorithm, I emphasize the elimination sequence for evaluating d-sepnodes in the MSID. The aim of this work is to obtain a global elimination sequence for d-sepnodes. With a valid elimination sequence for d-sepnodes, local influence diagrams could be evaluated cooperatively. Consequently, the whole MSID could be solved in a distributed fashion without incurring any inconsistent computation in the evaluation process. To design the hybrid evaluation algorithm, first, each local influence diagram is analyzed by the selected evaluation algorithm. The analysis generates a local elimination sequence for d-sepnodes. After that, a global elimination sequence for d-sepnodes is built after adjusting some conflicting local elimination sequences. Finally, local influence diagrams are evaluated by the selected evaluation algorithm. Meanwhile, the global elimination sequence is followed when d-sepnodes are evaluated.

Table 6.6: Components in the Hybrid Evaluation Algorithm

Local Influence Diagram	**Evaluation Algorithm**
I_1	Reduction Algorithm
I_2	Reduction Algorithm
I_3	Rooted Cluster Tree
I_4	Rooted Cluster Tree
I_5	*evalID* algorithm

6.4.1 Solve I_1

The local influence diagram I_1 in the MSID in Figure 6.1 is a regular one and has a sparse structure, which motivates the adoption of a basic reduction algorithm. Thus it

is easy to use the *GSL* process to obtain the local elimination sequence for all nodes in I_1: $ES_1(t) = \{g_1, d_{13}, d_{12}, f, h, c_1, d_{11}, b_1, d, a\}$ [4].

6.4.2 Solve I_2

The local influence diagram I_2 in the MSID in Figure 6.1 is also a regular one and has a sparse structure. Thus the *GSL* process is utilized to obtain the local elimination sequence in I_2: $ES_2(t) = \{b, f_2, d_{23}, g, a, e_2, d_{22}, c_2, d_{21}, d_2\}$.

6.4.3 Solve I_3

The local influence diagram I_3 in the MSID in Figure 6.1 is composed of a large number of chance nodes and one decision node. It is convenient to use an indirect evaluation algorithm to solve I_3 since it is easy to transform the local influence diagram into the rooted cluster tree as shown in Figure 6.3.

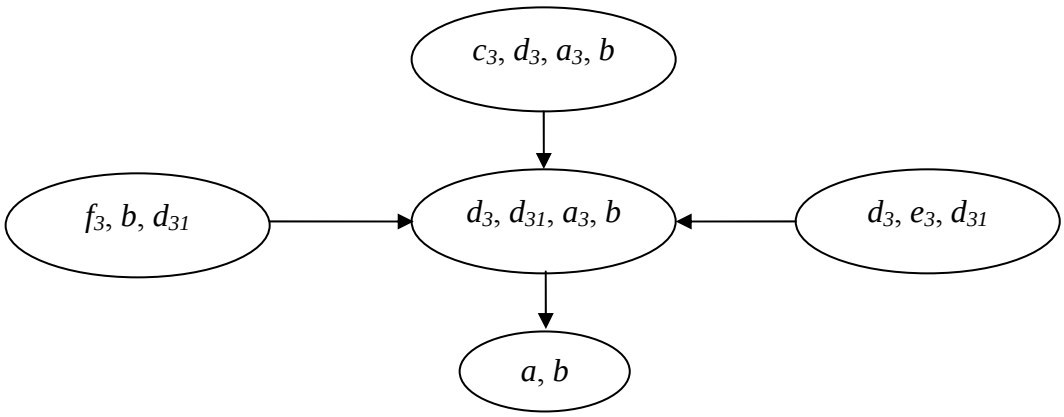


Figure 6.3: Rooted Cluster Tree for I_3

From Figure 6.3, one local elimination sequence for nodes in I_3 is obtained as follows:

$ES_3(t) = \{f_3, e_3, c_3, d_3, d_{31}, a_3, a, b\}$.

[4] Here, value node is neglected since it is always the last one to be removed.

6.4.4 Solve I_4

The local influence diagram I_4 in the MSID in Figure 6.1 seems to have a complex structure. Here, the indirect evaluation method is utilized. After the transformation, the rooted cluster tree associated with I_4 is shown in Figure 6.4.

From Figure 6.4, one local elimination sequence for nodes in I_4 is obtained as follows:

$ES_4(t) = \{h, c_4, g_4, d_{42}, f_4, d, e_4, d_{41}, a_4, b_4\}$.

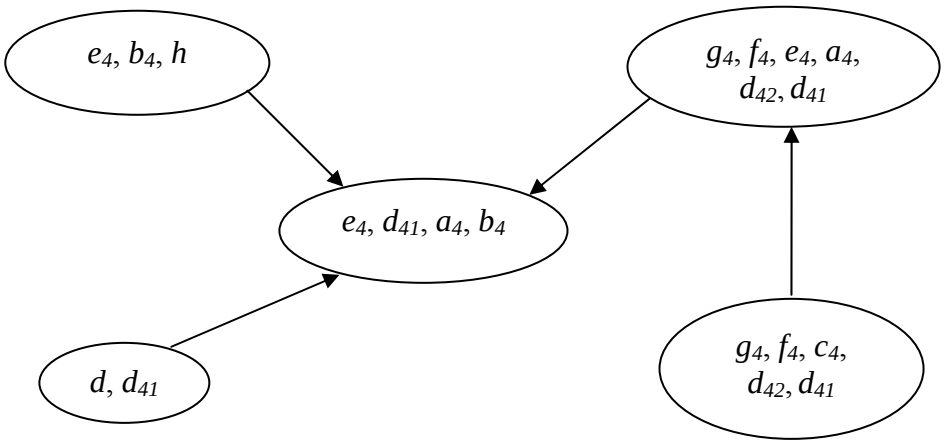


Figure 6.4: Rooted Cluster Tree for I_4

6.4.5 Solve I_5

The local influence diagram I_5 in the MSID in Figure 6.1 is not a regular one and has a dense structure. It is hoped that using the *evalID* algorithm could relieve the computation task. Here, the evaluation network for I_5 is developed as shown in Figure 6.5.

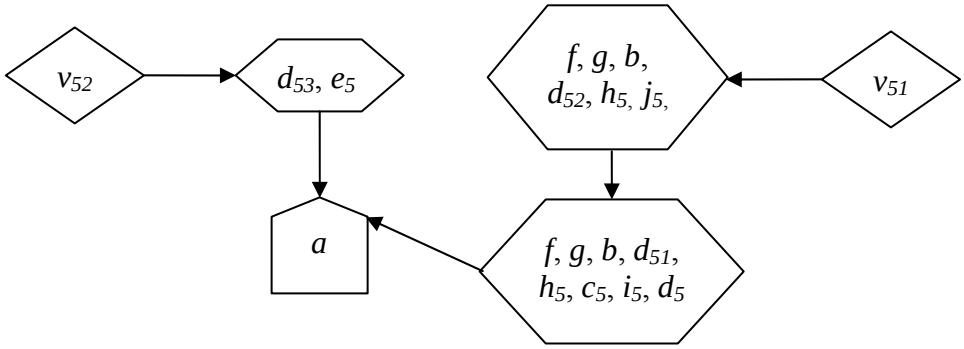


Figure 6.5: Evaluation Network for I_5

From the evaluation network in Figure 6.5, a partial order to remove nodes in I_5 could be obtained as follows: $ES_5(t) = \{(e_5, d_{53}), j_5, d_{52}, (f, g, b, d_{51}, h_5, i_5, c_5, d_5), a\}$ [5]. The absolute elimination sequence depends on what kind of inference engine is utilized to evaluate tail nodes which compose the evaluation network. However, $ES_5(t) = \{(e_5, d_{53}), j_5, d_{52}, (f, g, b, d_{51}, h_5, i_5, c_5, d_5), a\}$ provides a base elimination order to all nodes in I_5.

6.4.6 Solve the MSID

The challenging work in solving an MSID is to obtain a valid global elimination sequence for d-sepnodes because a unique local elimination order may not exist. A valid elimination sequence allows a cooperative evaluation of local influence diagrams in an MSID. In the cooperative evaluation process, the d-sepnodes are removed simultaneously so that the consistency is ensured in the solving of the MSID.

Based on the analysis in Sections 6.4.1 – 6.4.5, the elimination sequence for each local influence diagram in the MSID in Figure 6.1 has been built. It is repeated in Table 6.7.

[5] Elimination order of elements in the bracket could be exchanged.

Table 6.7: Elimination Sequence in Local Influence Diagrams

I_j	**Elimination Sequence**
I_1	$ES_1(t) = \{g_1, d_{13}, d_{12}, f, h, c_1, d_{11}, b_1, d, a\}$
I_2	$ES_2(t) = \{b, f_2, d_{23}, g, a, e_2, d_{22}, c_2, d_{21}, d_2\}$
I_3	$ES_3(t) = \{f_3, e_3, c_3, d_3, d_{31}, a_3, a, b\}$
I_4	$ES_4(t) = \{h, c_4, g_4, d_{42}, f_4, d, e_4, d_{41}, a_4, b_4\}$
I_5	$ES_5(t) = \{(e_{53}, d_{53}), j_5, d_{52}, (f, g, b, d_{51}, h_5, i_5, c_5, d_5), a\}$

Since an elimination sequence for d-sepnodes coordinates the evaluating of local influence diagrams in an MSID, a conflicting elimination order is not allowed. Hence, only the elimination order for d-sepnodes $ESDS_i(t)$ is shown in Table 6.8.

Table 6.8: Elimination Sequence for D-sepnodes

I_j	**Elimination Sequence for D-sepnodes**
I_1	$ESDS_1(t) = \{f, h, d, a\}$
I_2	$ESDS_2(t) = \{b, g, a\}$
I_3	$ESDS_3(t) = \{a, b\}$
I_4	$ESDS_4(t) = \{h, d\}$
I_5	$ESDS_5(t) = \{(f, g, b), a\}$

From the elimination sequence for d-sepnodes shown in Table 6.8, it seems that only the elimination order for d-sepnodes $\{b, a\}$ conflicts among local influence diagrams I_2, I_3 and I_5. Node b is needed to be removed after node a in I_3 while it has to be removed before node a in both I_2 and I_5. However, from the rooted cluster tree in Figure 6.3, the elimination order for nodes $\{b, a\}$ could be adjusted as $ESDS_3(t) = \{b, a\}$. Thus no conflicting elimination sequence for d-sepnodes exists in all local influence

diagrams. Consequently a global elimination sequence for d-sepnodes GES is elicited as follows: $GES = \{f, h, d, b, g, a\}$. Following this elimination sequence, the d-sepnodes could be removed simultaneously from local influence diagrams in the MSID. Hence local influence diagrams are evaluated cooperatively. On the other hand, other (private) nodes in local influence diagrams could be removed independently while they follow the local elimination sequence $ES_j(t)$ in each local influence diagram. In summary, the MSID in Figure 6.1 is solved in a distributed way and the evaluation process is coordinated through the global elimination sequence of d-sepnodes.

6.5 Summary

This chapter studies the SARS case in the medical domain. All of my proposed methodologies on MSID and HRG are revisited and used to solve the meaningful and complex decision problem. This work fully demonstrates that the proposed methodologies are significant and applicable in solving practical problems.

[This page intentionally left blank]

7 Block Learning Bayesian Network Structures from Data

In the preceding chapters, I mainly discuss the methodologies for solving multiagent decision making problems, which consist of model representation, model verification and model evaluation. This work can be utilized to solve a large and distributed decision problem involving multiple agents. However, building a decision model, such as the Multiply Sectioned Influence Diagrams (MSID), depends largely on domain knowledge which is elicited from the decision scenario. Hence this elicitation process consists of some subjective factors and is always called model construction. On another aspect, in the research field of normative decision systems, Bayesian networks are basic elements of decision models. For example, influence diagrams augment Bayesian networks by decision nodes and value nodes. Thus extensive research work on Bayesian networks appears in a large amount of literature, which includes Bayesian model learning, Bayesian propagation and so on. In this chapter, I will emphasize on the topic of Bayesian network learning, which is to construct Bayesian networks from data.

7.1 The Challenge

Bayesian network has been an important concept in normative decision systems. It compactly represents probabilistic knowledge and explicitly spells out dependencies among involved variables. Moreover, Bayesian reasoning provides a probabilistic approach to inference, prediction and planning. Hence, many domains such as medical

informatics domain (Friedman *et al.* 2002; Galan *et al.* 2002) and military domain (Sanguk & Gmytrasiewicz 1998), have adopted Bayesian networks as knowledge representation and inference engine in decision systems. Despite these successful applications, the construction of Bayesian networks is a piece of arduous and tedious work many system engineers have to face. One technique that may resolve this difficulty is to learn Bayesian networks from data obtained in the relevant domain. It includes parameter learning and structure learning in Bayesian networks (Heckerman *et al.* 1994). However, it is more difficult to learn structures than it is to learn parameters. My current work focuses on structure learning.

Various techniques for learning Bayesian network structures from data have appeared in the past decades such as the PC algorithm (Spirtes *et al.* 1993, 2000), Sparse Candidate algorithms (SC) (Friedman *et al.* 1999), Three Phases Dependency Analysis (TPDA) algorithm (Cheng *et al.* 2002) and Max-min Bayesian networks (MMBN) (Tsamardinos *et al.* 2003). These approaches have largely relieved the laborious task for building a precise Bayesian structure. However, recently some practical issues disable the applications of the existing learning techniques. For example, in gene expression data in the biomedical informatics domain, enormous variables are involved and available data is insufficient. Hence, the state-of-art learning algorithms will run into computational and statistical problems concerning the following two aspects. Since learning Bayesian network structure is a time-consuming process, the learning process deteriorates with a growing network size, such as memory running out in the computational process. Also, since some learning approaches like constraint based learning algorithms (Neapolitan 2004), depend on statistical tests to detect independencies, insufficient data weakens the reliability of the tests, leading to an

inaccurate learned structure. As such, the task of learning large Bayesian network structures from small data sets provides a motivation for this research.

7.2 Block Learning Algorithm

As investigated in Chapter 2, the existing learning algorithms have partly addressed the challenging work of learning large Bayesian network structures from a small data set. However, different research views on this learning problem could also motivate some simple and applicable methods to solve this challenging work.

The constraint based approach learns a network structure by using some statistical hypothesis tests in order to detect dependencies or (conditional) independencies among variables or attributes in a data set. The approach is more sensitive to failures in conditional independence (CI) tests when there is insufficient data. At the same time, a huge dimension of variables in a data set leads to an increasing order in the statistical tests and makes the learning task intractable. A naïve idea to cope with this problem is to decompose the learning task into several stages. Thus, a block learning algorithm is proposed to illustrate this idea. The novel algorithm includes several procedures as follows: Generating the Maximum Spanning Tree (GMST), Identifying the Blocks and Markov Blankets of Overlaps (IBMB), Learning the Overlaps (LO), Learning the Blocks (LB), and Combining the Blocks (CB). These procedures are executed sequentially. In the block learning algorithm, a maximum spanning tree is built in the first procedure. Based on the maximum spanning tree, several blocks and their overlap nodes are found in the second procedure. Then, V-structures are identified in the learned structures of the overlaps in the third procedure. After that, each block is

learned with the constraint of V-structures related to the block nodes. Finally, the learned blocks are combined together into the whole Bayesian networks.

7.2.1 Generate Maximum Spanning Tree

Bayesian network structures exhibit the dependency among variables in a data set. A strong dependency among variables always gathers them into one local density graph. In other words, these tightly linked variables are potential variables that will be enclosed in the same block. Hence, an initial graph has to be generated so as to identify such blocks. The graph must be able to characterize a strong dependency for every pair of variables. One of the mature methods to build this graph has been developed in Chow and Liu's work (Chow & Liu 1968). The algorithm uses the mutual information to construct a tree called the Maximum Spanning Tree (MST). Amongst all the tree-shaped models, the algorithm finds the model that maximizes the likelihood of the data.

Consider a finite set $\chi = \{x_1, \cdots, x_n\}$ of discrete random variables and a training data set $D = \{x^1, \cdots, x^N\}$, a Bayesian network B that matches D best is to be found. The procedure to Generate MST (GMST) is formulated in Figure 7.1.

Procedure GMST

Input: A data set $D = \{x^1, \cdots, x^N\}$
Output: MST M

1. Load the data set D
2. Build M based on the mutual information

Figure 7.1: GMST Procedure

In the procedure GMST, an MST is built after computing the mutual information

between every pair of variables. For n variables, the tree is developed with $O(n^2)$ steps using weight comparisons so that expensive CI tests are avoided.

7.2.2 Identify Blocks and Markov Blankets of Overlaps

The MST developed in the procedure GMST is just the graph in which blocks can be identified. A block is composed of variables that have a strong dependency with each other. The variable with a large cardinality of connectivity, called block center S_i, will absorb its neighbors and leaf nodes connected to these neighbors to compose a block. The procedure of Identifying Blocks (IB) is formulated in Figure 7.2.

Procedure IB

Input: A graph M

Output: Blocks B_i

1. Initialize an individual block B_i $(i = 1 \cdots n)$ as one block center S_i with its family $Fam(S_i)$ in M
2. Merge blocks B_i and B_j $(i \neq j)$ that have the same cardinality of connectivity and share the number of nodes larger than $ComNum1$
3. Search leaf nodes connected to those nodes in $Fam(S_i)$ and enclose them into block B_i
4. Merge block B_i and B_j $(i \neq j)$ that share the number of nodes larger than $ComNum2$
5. Finalize Blocks B_i $(i = 1 \cdots r)$

Figure 7.2: Procedure of Identifying Blocks

In the initialization phase, there are n blocks each of which centers around every node in M. Each block includes both a center node S_i and its family $Fam(S_i)$ which is adjacent to the center node. The number of the shared nodes between a pair of blocks, called $ComNum1$ or $ComNum2$, determines the criteria to merge these rough blocks. In general, their values are decided according to the connectivity of the M structure. Usually, the value of $ComNum2$ is larger than the value of $ComNum1$ because the blocks formulated in the later stage always have more nodes than the initial blocks. After the merging and absorbing operations, the blocks composed of the majority of variables become the final blocks that divide M into r pieces. Every final block also centers around one variable with a large cardinality of connectivity. It can be seen that no CI test is involved in this procedure. Assuming that k is the maximum cardinality of node adjacency in M, the procedure requires at most $O(nk)$, $O(n^2)$, $O(n^2)$, $O(nk)$, and $O(n^2)$ basic operations in Step 1- 5, respectively. In the worst case ($k = n$), the complexity of each step is at most $O(n^2)$.

Blocks are overlapped with each other to compose the whole network. Generally, the number of nodes in the overlaps is much smaller than that in the blocks. Hence, learning overlap structures is more operational and reliable. Moreover, I am concerned with the dependencies between the overlaps and the blocks, because the overlap structure will be utilized to combine the learned blocks in the last phase. The overlap O_{ij} includes common nodes between block B_i and block B_j. A novel idea is to learn the Markov blanket of those nodes in the overlaps. From the learned Markov blanket, some dependencies between the overlaps and the blocks could be detected. The procedure of Identifying the Markov Blanket of nodes in the overlaps (IMB) is formulated in Figure 7.3.

Procedure IMB

Input: Blocks B_i $(i = 1 \cdots r)$
Output: Overlaps O_{ij}, Markov Blankets of Overlaps MB_{ij}

1. Identify O_{ij} between blocks B_i and B_j $(i \neq j)$
2. Search all nodes within two lengths away from nodes in O_{ij} and pull them into MB_{ij}

Figure 7.3: Procedure of Identifying Overlaps and Markov Blankets

The nodes found in Step 2 in each set of the procedure IMB do not exactly equal those nodes in a real Markov blanket of nodes in the overlaps. By comparison, the resulting MB_{ij} includes more nodes some of which do not belong to the real Markov blanket of the overlap nodes. Assuming that t is the maximum number of nodes in an overlap, Steps 1 and 2 require at most $O(r^2)$ and $O(tr^2k^2)$ basic operations respectively.

To illustrate the procedures IB and IMB, it is assumed that there is one MST in Figure 7.4 and the two parameters $ComNum1$ or $ComNum2$ are assigned with the values of 1 and 2 respectively. In the initialization phase of IB, there exist 14 blocks ($B_i, i = 1, \cdots, 14$) corresponding to each individual node $\{a, b, \cdots, l, m, n\}$ as block center S_i. For example, the block B_1 centers on the node $S_1 = a$ whose family consists of the nodes $\{a, g, h, b, e\}$ and the connectivity cardinality of this block is 4. Among these blocks, the two blocks B_1 and B_3 centering on nodes a and c have the largest connectivity cardinality of 4; however, they do not share any node. The two blocks B_2 and B_4 centering on nodes b and d respectively have the connectivity cardinality of 2 and they share two nodes $\{b, d\}$. Hence, in the second step in the procedure IB, the two blocks B_2 and B_4 could be merged into the block $B_2 = \{a, b, c, d\}$ centering on the node b. In this step, only these two blocks B_2 and

B_4 are merged. In the third step, the block B_1 absorbs the leaf nodes f and i since they are connected to nodes e and h respectively in its family. The block B_2 also absorbs the nodes $\{g,k\}$. Similarly, the block B_3 absorbs the nodes $\{l,m\}$. Then, in the fourth step, many blocks are merged into a larger one. For example, the blocks $B_6 \sim B_9$ are merged into the block B_1 and the blocks $B_{11} \sim B_{14}$ are merged into the block B_3. Hence, after the fourth step, there exist three blocks: B_1 is with family nodes $\{a,g,h,b,e,f,i\}$ centering on the node $S_1 = a$, B_2 is with family nodes $\{b,a,c,d,g,k\}$ centering on the node $S_2 = b$ and B_3 is with family nodes $\{c,d,k,j,l,m,n\}$ centering on the node $S_3 = c$. Finally, the blocks B_2 and B_1 are merged into the block B_1 since they share 3 nodes $\{b,a,g\}$. Hence, the output of the procedure IB has two blocks: B_1 is with family nodes $\{a,g,k,h,b,e,f,i,d,c\}$ centering on the node $S_1 = a$ and B_3 is with family nodes $\{c,d,k,n,j,m,l\}$ centering on the node $S_3 = c$.

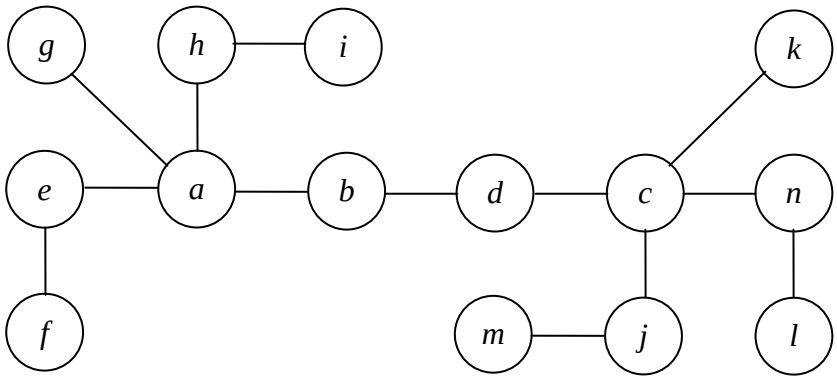


Figure 7.4: An MST

In the procedure IMB, the two blocks B_1 and B_3 are the input. In the first step, the overlap $O_{13} = \{c,d,k\}$ is identified. Finally, a rough Markov blanket of these overlap nodes is searched and denoted as $MB_{13} = \{a,b,d,c,j,m,k,n,l\}$. It can be seen that the Markov blanket

even has a larger capacity than the block B_3. Hence, some improvement could be made in this step. However, the approach adopted in the second step is a simple one to identify a rough Markov blanket of overlap nodes.

7.2.3 Learn Overlaps

Overlaps connect adjacent blocks to compose a whole network and the overlap structures can be obtained through learning their Markov blankets. Any of the existing learning algorithms, called ALG1, can be utilized to identify the structure of the Markov blanket. For example, the PC algorithm, the GS algorithm (Margatitis 203) and the IAMB algorithm (Tsamardinos 2003) are able to learn MB_{ij}. The resulting Markov blanket provides a foundation to identify some V-structures associated with overlap nodes. The procedure of Learning Overlaps (LO) is formulated in Figure 7.5.

Procedure LO

Input: A data set $D = \{x^1, \cdots, x^N\}$ and MB_{ij}
Output: V-structure of O_{ij} : VS(O_{ij})

1. Load the data set D and learn MB_{ij} using ALG1
2. Identify the V-structure of O_{ij} from the learned MB_{ij}
3. Produce VS(O_{ij})

Figure 7.5: Procedure of Learning Overlaps

The robust V-structures relevant to O_{ij} avoid error spread in learning blocks when VS(O_{ij}) is set as constraints in the learning process. It can be seen that a large amount of computations happen in the process of learning MB_{ij}. Assuming that the PC algorithm is used in this procedure, Step 1 requires $O(r^2(tk^2)^k)$ CI tests while Step 2 requires at most

$O(tkr^2)$ basic operations. If the PC algorithm is terminated without orienting edges in the final phase (Spirtes *et al.* 1993), Step 2 can be avoided.

7.2.4 Learn Blocks and Combine Blocks

Learning Bayesian network structures with some constraints, like a partial order for some variables, speeds up the learning process because some CI tests may be avoided. In the procedure LO, VS(O_{ij}) is obtained and is set as the constraints when learning the corresponding blocks. Here the size of the blocks B_i, denoted by m_i, may be expanded to include the variables in VS(O_{ij}) from all of its adjacent blocks B_j. Clearly, any of the existing learning algorithms, denoted as ALG2, can be used to learn blocks in the procedure of Learning Blocks (LB). Subsequently, the whole network structure is recovered after combining the final learned block in the procedure of Combining Blocks (CB). These two procedures are formulated in Figure 7.6 and Figure 7.7 respectively.

Procedure LB

Input: A data set $D = \{x^1, \cdots, x^N\}$, B_i and VS(O_{ij})
Output: Learned $B_i\ (i = 1 \cdots r)$

1. Load the data set D and learn B_i using the ALG2 with constraints VS(O_{ij})
2. Produce the learned $B_i\ (i = 1 \cdots r)$

Figure 7.6: Procedure of Learning Blocks

Procedure CB

Input: Learned B_i and VS(O_{ij}) ($i, j = 1 \cdots r$ and $i \neq j$)
Output: B

1. Identify λ_i, λ_j for the nodes in O_{ij} between blocks B_i and B_j
2. For the two blocks B_i and B_j ($i \neq j$): if $\lambda_i = \lambda_j$, combine B_i and B_j ; otherwise if $m_i < m_j$, combine B_i and B_j following λ_i
3. Produce B

Figure 7.7: Procedure of Combining Blocks

The procedure CB is relevant to the research work on the combination of Bayesian networks. This issue on Bayesian network structure combination has been investigated in much research work (Mckelvey & McLennan 1996; Joseph *et al.* 1998). Currently, Jiang's work (Jiang *et al.* 2005) proposes some heuristic methods to combine several Bayesian networks simultaneously. Most of these existing methods try to output a valid Bayesian network structure which should be a directed acyclic graph. However, these methods are not guaranteed to yield an optimal solution as it is a NP-hard problem (Mckelvey & McLennan 1996).

One of the most important tasks in the procedure CB is to build the final B which disallows directed cycles. Hence, the procedure CB is concerned with a valid method for combining the learned blocks. The basic idea in my approach is to make overlap nodes follow one uniform partial order before the combination operation is carried out. A directed acyclic graph does not allow opposite orders in a set of nodes, even in a pair of nodes. Thus, in this way, a valid Bayesian network could be output after the combination.

Assume that the blocks B_i and B_j ($i \neq j$) linked by O_{ij} are ready to be combined into a new large block B_{ij}, and a partial order for nodes in O_{ij} is identified in the block B_i and the block B_j individually, denoted by λ_i and λ_j. The following proposition can be proved.

Proposition 7.1: The fact that λ_i equals to λ_j ensures no directed cycle in B_{ij} composed of B_i and B_j ($i \neq j$).

Proof. A directed cycle in B_{ij} must involve at least two nodes in O_{ij}. Hence, there is no uniform order for those involved nodes in B_{ij}. For example, a directed cycle contains two nodes x_1 and x_2 that should respect different orders: $x_1 \to x_2$ in λ_i and $x_2 \to x_1$ in λ_j, or vice versa. A uniform order for those involved nodes ensures no directed cycle in B_{ij}. ▌

In the case that the reversal orders are obtained in λ_i and λ_j for the variables in O_{ij}, they have to be forced to follow one uniform order in the block B_i that has a smaller size m_i, assuming $m_i < m_j$. The reason lies in the consideration of statistical tests in a data set. For the same cases or instances, the fewer the variables, the more reliable the dependencies among nodes are tested. Figure 7.7 clearly shows this strategy. In fact, when overlap nodes O_{ij} in the block B_j are forced to follow the uniform order associated with overlap nodes O_{ij} in the block B_i, the structure of the corresponding nodes in a directed path along overlap nodes O_{ij} has to be reconstructed to an equivalent one in the block B_j. In this process, it involves the operation of arc reversal to get the equivalent structure of B_j concerning the uniform order of overlap nodes O_{ij}.

This discussion is beyond the scope of this thesis. Details could be found in Jiang's work (Jiang *et al.* 2005). However, in a series of experiments in Section 4, overlap nodes always have uniform partial orders in adjacent blocks. Hence the combination procedure is not hard in practice.

The procedures LB and CB conclude the block learning algorithm. Assuming that the PC algorithm is adopted in LB and m is the maximum number of nodes in a block, the procedure LB requires $O(r(m)^k)$ CI tests while the procedure CB only requires at most $O(r^2tkm)$ basic operations.

7.3 Experimental Results

The aim of the following experiments is to demonstrate the ability of the block learning algorithm for learning a large Bayesian network structure from a small data set as well as its capability for encompassing other learning techniques. I evaluated the block learning algorithm on two networks as benchmarks: the ALARM network (Beinlich *et al.* 1989) and the Hailfinder network (Edwards 1998). Several versions of these two networks have appeared in the current literature; the version I adopted in this experiment can be accessed with the link: http://www.cs.huji.ac.il/labs/compbio/Repository/. The data set was generated from the script in http://www.cs.huji.ac.il/labs/compbio/Repository/networks.html. To utilize the huge resource of BNT in http://www.ai.mit.edu/~murphyk/Software/BNT/bnt.html, all procedures of the block learning algorithm were implemented with MATLAB, except for ALG1 in LO and ALG2 in LB. The implementation of ALG1 and ALG2 can make use of the existing learning tools. All these experiments were conducted on a Pentium 2.6 GH_Z PC with 512 MB of RAM running under Windows XP.

7.3.1 Experiments on the Hailfinder Network

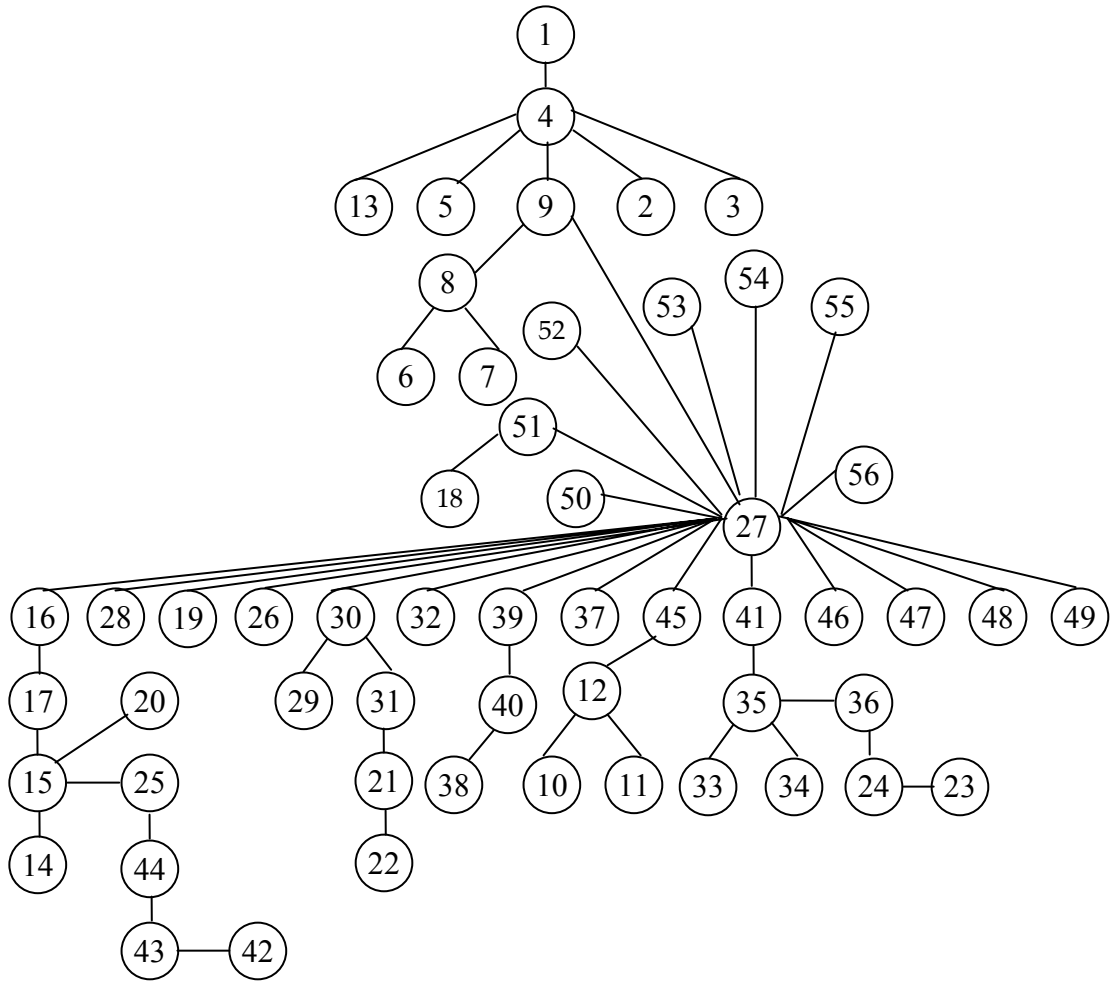
The Hailfinder network (Edwards 1998) is a normative system that forecasts severe summer hail in northeastern Colorado. It consists of 56 nodes and 66 arcs. The number of node states in the Hailfinder network is up to 11. For the large size of network structure, much research work has adopted it as a benchmark for algorithm evaluation. From the available resources in the literature, the tool of BNPC (Believe Networks Power Constructor (BNPC) implements the TPDA algorithm.), winner of 2001 KDD cup, shows quite good learned results on the Hailfinder network. However, it did not display any ability for addressing the learning problem when insufficient data exists.

To show the ability of the block learning algorithm for learning a large Bayesian network structure from a small data set, I designed a series of experiments to compare the block learning algorithm with the TPDA algorithm implemented in BNPC. Five sample sizes were generated: 8K[6] cases, 5K cases, 1K cases, 0.3K cases and 0.1K cases. The procedures GMST, IBMB and CB in the block learning algorithm were executed in MATLAB while the procedures LO and LB were run in BNPC with the default parameters, such as the threshold with 1 time of default value. The values 2 and 3 are assigned to the parameters *ComNum*1 and *ComNum*2 respectively in order to identify blocks in the procedure IB. In the procedure LO, V-structures were identified from the Markov blanket of overlaps that was learned in BNPC. Finally, BNPC was also used to learn the blocks. Additionally, to compare the block learning algorithm with the TPDA algorithm, I directly used BNPC to learn the whole Hailfinder network from the four samples. The settings in BNPC were all default.

To illustrate the mechanism of the block learning algorithm, I traced all procedures

[6] 1K =1000 Cases

throughout one experiment based on 0.1K cases. For example, in the process of learning the Hailfinder network from this dataset, the MST was generated in Figure 7.8. Compared with the node distribution in the original Hailfinder network, the MST produced in the procedure GMST was the appropriate graph which could be utilized to identify the blocks. Based on the MST in Figure 7.8, seven blocks ($B_i, i = 1 \cdots 7$) were identified in the procedure IB. The block centers ($S_i, i = 1 \cdots 7$) and the block variables were found in Table 7.1.



1: N0_7muVerMo, 2: SubjVertMo, 3: QGVertMotion, 4: CombVerMo,
5: AreaMeso_ALS, 6: SatContMoist, 7: RaoContMoist, 8: CombMoisture,
9: AreaMoDryAir, 10: VISCloudCov, 11: IRCloudCover, 12: CombClouds,
13: CldShadeOth, 14: AMInstabMt, 15: InsInMt, 16: WndHodograph,
17: OutflowFrMt, 18: MorningBound, 19: Boundaries, 20: CldShadeConv,
21: CompPlFcst, 22: CapChange, 23: LoLevMoistAd, 24: InsChange,
25: MountainFcst, 26: Date, 27: Scenario, 28: ScenRelAMCIN,
29: MorningCIN, 30: AMCINInScen, 31: CapInScen, 32: ScenRelAMIns,
33: LIfr12ZDENSd, 34: AMDewptCalPl, 35: AMInsWliScen, 36: InsSclInScen,
37: ScenRel3_4, 38: LatestCIN, 39: LLIW, 40: CurPropConv
41: ScnRelPlFcst, 42: PlainsFcst, 43: N34StarFcst, 44: R5Fcst,
45: Dewpoints, 46: LowLLapse, 47: MeanRH 48: MidLLapse,
49: MvmtFeatures, 50: RHRatio, 51: SfcWndShfDis 52: SynForcng,
53: TempDis, 54: WindAloft, 55: WindFieldMt, 56: WindFieldPln

Figure 7.8: The MST for the Hailfinder Network

Table 7.1: Blocks, Centers and Block Elements (Hailfinder Network on 0.1K Cases)

B_i	S_i	Block Elements						
B_1	31	21	22	30				
B_2	44	25	42	43				
B_3	12	10	11	45				
B_4	15	14	16	17	20	25	44	
B_5	35	23	24	33	34	36	41	
B_6	4	1	2	3	5	9	13	
B_7	27	4	6	7	8	9	12	16
		17	18	19	29	26	28	30
		31	32	33	35	37	38	39
		40	41	45	46	47	48	49
		50	51	52	53	54	55	56

As shown in Table 7.1, these blocks included the exact nodes that had strong dependency so that their dependency structures would not be damaged when the network was learned locally. After the overlaps and the blocks were learned respectively in the procedures LO and LB, the final Hailfinder network was recovered when all the learned blocks were combined together in the procedure CB.

Similar procedures in the block learning algorithm were executed in other experiments for learning the Hailfinder network from data set with 8K, 5K, 1K, and 0.3K cases respectively. At the same time, to evaluate the TPDA algorithm, BNPC is directly used to learn the whole network from the five data sets. Both the block learning algorithm and the TPDA algorithm are run in 10 different data sets with each size. The structures learned by these two algorithms in each round were compared with the original Hailfinder network. One set of comparison results for every data set is shown in Table

7.2.

Table 7.2: Comparison 1 of BL and TPDA Algorithms

Cases	***ALG***	***NB***	***EE***	***ME***
8K	**TPDA**	N/A	10	18
	BL	8	8	16
5K	**TPDA**	N/A	8	16
	BL	7	8	17
1K	**TPDA**	N/A	33	25
	BL	8	24	25
0.3K	**TPDA**	N/A	31	30
	BL	7	28	26
0.1K	**TPDA**	N/A	33	34
	BL	7	30	29

BL**:** Block Learning Algorithm; ***ALG*****:** ALGorithm;
NB**:** Number of Blocks; ***EE*****:** Extra Edges; ***ME*****:** Missing Edges.

In Table 7.2 results show that the block learning algorithm is able to discover more accurate structures than the TPDA algorithm. In the structure learned by the block learning algorithm, more correct edges are detected and more error edges are removed. For example, in Table 7.2, for the data set with 8K cases, 16 edges are lost and only 8 extra edges exist in the structure learned by the block learning algorithm. However, 18 edges are missed and 10 extra edges are wrongly identified in the structure learned by the TPDA algorithm. The advantages of the block learning algorithm are more noticeable when only a small data set is available. In Table 7.2, for the data set with 0.1K cases, the performance of both learning algorithms deteriorates. However, the block learning algorithm still produces more precise structures than the TPDA algorithm. It can be seen that 29 correct edges are missed and 30 extra edges are

wrongly added in the structure learned by the block learning algorithm. On the other hand, 34 correct edges are lost and 33 extra edges are wrongly identified in the structure recovered by the TPDA algorithm. Hence, the block learning algorithm shows a good ability for learning a large Bayesian network structure from a small data set compared with the TPDA algorithm. (Here, I want to mention that the experimental results are based on the datasets that were produced in the experiment. The datasets may be quite different from those in other publications (Cheng *et al.* 2002). The two algorithms are compared when they were utilized to learn the same dataset generated in each round.)

To do a further comparison, the Euclidean distance d of the sensitivity and specificity from the perfect score 1 was used to judge the algorithm performance (Tsamardinos *et al.* 2003).

$$d = \sqrt{(1 - sensitivity)^2 + (1 - specificity)^2}$$

where the sensitivity of the algorithm is the ratio of correctly identified edges (undirected arcs) over the total number of edges in the original network while the specificity is the ratio of edges correctly identified as not belonging in the graph over the true number of edges not present in the original network. Moreover, to evaluate the robustness of the block learning algorithm different data sets with the same size are explored 10 rounds in the experiment. In this comparison, the average μ of sensitivity, specificity and distance, as well as their standard deviation δ, are reported as shown in Table 7.3.

Table 7.3: Comparison 2 of BL and TPDA Algorithms

Cases	***Sens.(%)***				***Spec. (%)***				***Dist.***			
	TPDA		***BL***		***TPDA***		***BL***		***TPDA***		***BL***	
	μ	δ	μ	δ	μ	δ	μ	δ	μ	δ	μ	δ
8K	72.24	1.61	75.16	1.28	99.32	0.08	99.42	0.06	0.26	0.02	0.24	0.01
5K	75.00	0.80	76.62	1.06	99.39	0.06	99.46	0.05	0.26	0.01	0.24	0.01
1K	63.64	1.81	63.64	1.33	97.76	0.10	98.37	0.10	0.36	0.02	0.36	0.05
0.3K	55.30	1.25	61.36	1.06	97.93	0.12	98.13	0.09	0.45	0.01	0.39	0.02
0.1K	49.24	1.64	56.06	1.24	97.80	0.05	97.96	0.05	0.51	0.02	0.44	0.01

***Sens.*:** Sensitivity; ***Spec.*:** Specificity; ***Dist.*:** Distance

In Table 7.3, the results show that the average sensitivities of the block learning algorithm are 75.16% and 76.62% for learning the Hailfinder network structures from the data set with 8K and 5K cases respectively. It decreases to 56.06% for the data set with 0.1K cases. In comparison, the sensitivity of the TPDA algorithm decreases significantly from 72.24% to 49.24% when the sample size shrinks from 8K cases to 0.1K cases. As shown in Table 7.3, the average specificity of these two algorithms is close to 100% because there are a large number of potential edges in the Hailfinder network. Even in this case, the block learning algorithm still has a higher specificity than the TPDA algorithm.

The criterion of the distance d is a combination measure to evaluate the gap between the learned structure and the original network structure. In Table 7.3, it shows that the block learning algorithm does not have much larger average distance than the TPDA algorithm when both of them are utilized to learn the Hailfinder network from large data sets such as the 8K and 5K cases. The average distance of both learning algorithms is almost the same for the data set with 1K, 5K and 8K cases. However, for

the small data set with 0.1K cases, the average distance of the block learning algorithm is only 0.44 while the distance of the TPDA algorithm is 0.51. On the other hand, the standard deviation of the distance of the TPDA algorithm seems a little worse than that of the block learning algorithm when they are used in learning structures from small data sets. For example, it is only 0.01 for the block learning algorithm while it is 0.02 for the TPDA learning algorithm in the 0.1K cases. Hence, all the experimental results could demonstrate that the block learning algorithm is able to learn a large Bayesian network structure from a small data set. Moreover, the block learning algorithm could output more reliable results.

7.3.2 Experiments on the ALARM Network

The ALARM network (Beinlich *et al.* 1989) is a popular Bayesian network in the medical domain. It consists of 37 nodes and 46 arcs. The number of node states is up to 4. The ALARM network has been widely utilized as a benchmark to evaluate state-of-the-art learning algorithms in the literature.

To demonstrate the capability of the block learning algorithm for encompassing other learning techniques, the PC algorithm was adopted in the learning procedures of the block learning algorithm. In the procedures LO and LB, learning algorithms ALG1 and ALG2 were set as the PC algorithm to learn overlaps and blocks. Hence, in the experiment on the ALARM network, the procedures GMST, IBMB and CB in the block learning algorithm were executed in MATLAB while the procedures LO and LB were run in the tool of HUGIN (http://www.hugin.com/) that has successfully implemented the PC algorithm. The PC algorithm was also used to learn the ALARM network directly so that I could evaluate their performance according to the learned structures.

To compare the block learning algorithm with the PC algorithm, I designed the experiment to show the learned structure on the data set with different sizes. The data sets with 0.1K cases, 0.3K cases, 1K cases, 5K cases, and 8K cases were generated respectively for the ALARM network. The settings in HUGIN were all default. Each learning algorithm is run 10 times individually on different data sets with the same size. One set of the learned results is chosen as shown in Table 7.4.

Table 7.4: Comparison 1 of BL and PC Algorithms

Cases	***ALG***	***NB***	***EA***	***RA***	***MA***
8K	**PC**	N/A	0	1	4
	BL	6	0	1	4
5K	**PC**	N/A	0	1	4
	BL	6	0	1	4
1K	**PC**	N/A	2	2	6
	BL	8	1	2	4
0.3K	**PC**	N/A	2	4	14
	BL	6	1	2	10
0.1K	**PC**	N/A	4	6	20
	BL	6	3	4	14

EA: Extra Arcs; ***RA***: Reversed Arcs; ***MA***: Missing Arcs.

From Table 7.4, it is noticed that there is no difference between the structures learned by the block learning algorithm and the PC algorithm for the data sets with 8K and 5K cases. Both algorithms miss 4 correct arcs and wrongly orient 1 arc in the learned structure in comparison with the original ALARM network. This is due to the fact that the learned structure is almost near the margin of the original ALARM network. It is difficult for both algorithms to break through the learning bottleneck. In this case, the block learning algorithm has the only merit that the computation time decreases

sharply (I will discuss this in the next section). On the other hand, for the small data set with 0.1K cases, the block learning algorithm retains its advantages on more precise structures that were learned. For instance, the block learning algorithm only misses 14 correct arcs and wrongly adds 3 arcs. However, the PC algorithm loses 20 correct arcs and adds four more arcs. Moreover, 6 arcs are reversed in the PC algorithm while only 4 arcs are misdirected in the block learning algorithm.

To further investigate the performance of both algorithms, I provide the analysis on the average μ of sensitivity, specificity and distance, and their corresponding standard deviation δ. The results are shown in Table 7.5.

Table 7.5: Comparison 2 of BL and PC Algorithms

Cases	***Sens.(%)***				***Spec. (%)***				***Dist.***			
	PC		***BL***		***PC***		***BL***		***PC***		***BL***	
	μ	δ	μ	δ	μ	δ	μ	δ	μ	δ	μ	δ
8K	91.30	0.92	91.30	0.92	100.00	0.08	100.00	0.08	0.09	0.01	0.09	0.01
5K	86.30	1.45	91.30	1.45	100.00	0.08	100.00	0.07	0.09	0.01	0.09	0.01
1K	88.96	1.12	91.30	1.12	99.68	0.10	99.84	0.11	0.13	0.01	0.09	0.01
0.3K	67.39	1.60	78.26	1.47	99.68	0.10	99.84	0.08	0.33	0.02	0.22	0.01
0.1K	56.52	2.00	67.39	1.47	99.19	0.13	99.52	0.09	0.43	0.02	0.32	0.01

Table 7.5 shows that the average sensitivity of the block learning algorithm remains the same (91.30%) for different data sets with 8K, 5K, and 1K cases. However, the average sensitivity of the PC algorithm decreases sharply from 91.30% down to 88.96% when the sample size shrinks. When the sample size shrinks to 0.1K the average sensitivity of the PC algorithm is only 56.52% while the average sensitivity of the block learning algorithm still catches the value 67.39%. On the other hand, the

average specificity of both algorithms is close to 100.00% despite different sample sizes. This is because enormous potential edges exist in the ALARM network.

On another aspect, the average distance of the PC algorithm increases from 0.09 to 0.43 intuitively when the sample size decreases from 8K to 0.1K. However, the average distance of the block learning algorithm only rises from 0.09 to 0.32 in spite of the shrinking cases. Thus, it shows that the performance of the PC algorithm deteriorates with decreasing sample size while the block learning algorithm still retains a good result. Also, it is noticed that the block learning algorithm has more reliable results when learning networks from small datasets. For example, the standard deviation of the block learning algorithm is 0.01 for both sample sizes with 0.3K cases and 0.1K cases while there is a larger value 0.02 for the PC algorithm when it learns the structure from these two different small sample sizes. Thus the block learning algorithm has more reliable learned results.

Hence, the experiment on the ALARM network repeats in showing the ability of the block learning algorithm for learning a large network from a small data set. At the same time, it also verifies that the block learning algorithm is able to encompass other learning techniques so that it is easy to be configured.

7.4 Theoretical Discussion

The experimental results demonstrate two advantages of the block learning algorithm: 1) the ability of learning a large Bayesian network structure from a small data set; 2) the capability of encompassing other learning techniques. In fact, this phenomenon can be analyzed theoretically.

Firstly, the intractable learning task is decomposed in an effective way so that it is easy to handle the structure learning problem. Instead of learning the whole network, the block learning algorithm learns individual blocks that have a much smaller size. It is known that the typical learning algorithm, such as the PC algorithm, learns Bayesian network structures with the complexity of $O(n^k)$ while the block learning algorithm only requires $O(r(m)^k)$ where $n \approx rm$. It seems that these two learning algorithms have the same algorithm complexity because of $O(r(m)^k) = O(n^k / r^{k-1})$. However, the divide and conquer strategy allows the block learning algorithm to be executed in an economical time. Figure 7.9 shows this phenomenon intuitively. Here, I assume that both k and r equal to 6 although k may be larger and r increases with the expanding size of the network in practice. In Figure 7.9, the number of variables in the Bayesian network indicates the size of the network structure. Figure 7.9 clearly shows that the complexity of the PC algorithm rises abruptly with the growing size of a network structure. However, there is little change in the complexity of the block learning algorithm when the size of a network structure increases. Hence, the block learning algorithm is scalable to learn a sizable Bayesian network structure.

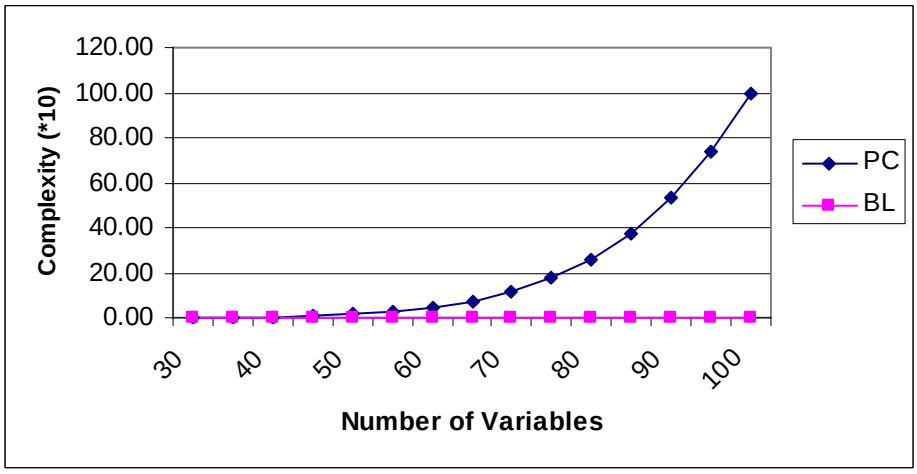


Figure 7.9: Complexity Comparison of BL and PC Algorithms

Secondly, the localization strategy allows each variable to be estimated on a relatively large sample size so that the block learning algorithm is able to provide good results when they are used to address the learning problem with a small data set. In a large dimension domain with insufficient data, the confounding information always disallows the removal of spurious correlations among variables. Since the block learning algorithm learns blocks which always include a small number of variables, it will easily discover the dependency among variables. Thus the learned results are more reliable.

Thirdly, the localization strategy also allows learning blocks independently. In the learning procedures of the block learning algorithm, various learning techniques can be utilized to configure ALG1 in LO and ALG2 in LB. Furthermore, the block learning algorithm can adopt suitable learning algorithms to learn the corresponding overlaps or blocks. Hence, the block learning algorithm is able to encompass other learning techniques without damaging the learning quality.

Finally, the block learning algorithm overcomes the main deficiency in the constraint based approach. In the constraint based approach, a small error on CI tests will subsequently exert a bad influence on the learning process globally, which causes a poor structure. In comparison with other constraint based approaches, the block learning algorithm confines possible errors in the separate blocks so as to avoid the spread of structure errors in the whole network. Furthermore, V-structures, which are the most reliable structures that can be discovered, are kept and set as constraints for learning blocks in the block learning algorithm. Accordingly, the block learning algorithm prohibits the spread of structure errors while it forwards the benefit of accurate structures in the learning process. These effective strategies ensure the good performance of the block learning algorithm.

7.5 Further Discussion

Current algorithms for learning Bayesian network structures involve enormous paradigm nomenclatures and implementation considerations in the literature. A novice is always confused by the core ideas behind these algorithms. In fact, some of the ideas are translations of each other; some are extensions while others are combinations of the existing ideas. Hence, a unifying learning framework will provide a uniform view on various learning techniques and will facilitate the wide applications of learning tools.

The block learning algorithm enriches various techniques for learning Bayesian network structure. With outstanding features, it not only addresses the challenging work of learning a large network structure from a small data set, but also provides a foundation to generate a unifying learning framework. In this section, I will investigate the issue of building a unifying learning framework on the basis of the block learning algorithm.

Firstly, the block learning algorithm has a natural learning granularity. In the past decades, various learning algorithms composed of basic learning approaches and advanced learning approaches have appeared to cope with a variety of learning problems. Although the context in which these learning techniques exist is quite different, the proposed learning algorithms have unconsciously been following a principal idea for decreasing the learning dimensionality. In the last decade, the basic learning methods, such as the IC algorithm, the SGS algorithm, and the PC algorithm, have taken a coarsely granular view on the learning problem. They directly learn the whole network without taking any optimization strategy so that they are always disabled with a large dimension of network structure. Later, the advanced learning methods, such as the SC algorithm, the GS algorithm, and the MMPC algorithm

(Brown *et al.* 2004), have adopted a finely granular view on the learning problem. They emphasize a local structure like a Markov blanket of variables, and learn the whole network stage by stage. They successfully avoid the computation problem. However, the property of local structure inhibits the general application of these advanced learning approaches. Hence, these two kinds of learning approaches, namely basic and advanced learning methods, stand in the two ends of learning granularity.

Clearly, the block learning algorithm falls in the middle concerning the learning granularity. It does not painstakingly seek the learning size to decrease the learning complexity; however, it utilizes the variable dependency to decide a suitable learning size naturally. A "block" could be defined as one "whole network" in terms of basic learning methods while it could be identified as one "Markov blanket" in terms of advanced learning methods.

Secondly, the block learning algorithm is equipped with an adaptive learning engine. A reservoir of structure learning algorithms exists in the available literature. The new learning approaches appear continuously while the old ones are never obsolete. This phenomenon arises since the existing learning techniques have their own ability or advantages for dealing with a certain type of learning problems. Moreover, some hybrid learning algorithms (Dash & Druzdzel 1999) are designed in order to absorb the advantages of the available learning techniques. Consequently, the accumulated learning algorithms with various paradigms are always confused.

As the block learning algorithm has a capability of encompassing other learning methods, it is easy to be configured with preferred learning techniques in the implementation. The parameters ALG1 and ALG2 allow desired learning techniques to be embedded in the block learning algorithm. For instance, the block learning

algorithm became "the PC algorithm" in the second experiment on the ALARM network and "the TPDA algorithm" in the first experiment on the Hailfinder network. Hence, the block learning algorithm could be customized with any learning technique. It provides a uniform view on various learning approaches.

Finally, the block learning algorithm aids a distributed design. For a large amount of computation involved in the learning process, a good algorithm expects to exploit the power of computing technologies for the purpose of constructing a network in reasonable time. The parallelized or distributed learning algorithms (Chu & Xiang 1997; Lam & Segre 2002) have saved a lot of computation time to learn a proper network. Adopting the divide-and-conquer strategy, the block learning algorithm supports a distributed design. For instance, the procedures LO and LB in which a majority of computation time is involved could be implemented and run in separate computing nodes in a distributed computing environment. Hence, the block learning algorithm is also a distributed framework. Furthermore, the block learning algorithm is prone to be run in a grid.

In conclusion, the block learning algorithm is not just a type of learning algorithm but a kind of learning strategy. It provides a unifying framework for learning Bayesian network structure. The unifying framework is built in Figure 7.10.

In Figure 7.10, the unifying framework is developed on the basis of the block learning algorithm. Step 2 is to build a rough dependency graph for identifying blocks. It is the procedure GMST. In fact, many approaches on detecting dependency can be used to generate this kind of graph. In addition, Step 2 can be avoided if the original network becomes the block. Steps 3-5 provide the power to unify all of the current learning algorithms or learning strategies. Step 3 decides the learning dimensionality, like the

procedure IBMB. Step 4 configures the learning engine with preferred learning techniques. Hence, the unifying framework can be customized into a hybrid learning framework. Step 5 is concerned with the implementation of the learning framework. It expects to be equipped with advanced computing technology, such as distributed computing and grid computing (Foster *et al.* 2002). Steps 6-7 are similar to the procedures LO, LB and CB. Hence, the unifying learning framework evolves from the block learning algorithm and may become a general architecture for learning Bayesian network structures.

A Unifying Learning Framework

Input: Data set D
Output: Bayesian Network Structure B

1. Load the data set D
2. Generate an MST
3. Determine the block (block, network, Markov blanket···)
4. Configure learning engine (all available learning techniques)
5. Design learning environment (serial, distributed···)
6. Learn the overlaps and blocks
7. Combine the learned blocks
8. Recover the whole network B

Figure 7.10: A Unifying Learning Framework

7.6 Summary

This chapter describes the block learning algorithm. Adopting the divide-and-conquer strategy, the block learning algorithm is proposed to address the challenging work -

learning Bayesian network structures from small data sets. After generating a rough dependency graph called MST, the block learning algorithm divides the whole network into several blocks that may have some overlaps. Then, the block learning algorithm learns blocks separately with the constraints from the learned overlaps. Finally, the whole network structure is recovered from the combination of the learned blocks.

A series of experimental results on the Hailfinder network and the ALARM network demonstrate the learning ability of the block learning algorithm on a small data set. They also show the capability of the block learning algorithm for encompassing other learning techniques. Subsequently, a theoretical analysis further verifies the advantages of the block learning algorithm.

With outstanding features, the block learning algorithm may provide a uniform view on the current learning algorithms. Its natural learning granularity, flexible configurations on learning engines and distributed implementation design facilitate the building of a unifying learning framework. Evolving from the block learning algorithm, the unifying framework generalizes the existing learning techniques. It also provides a concise and powerful template for a novice to categorize and study various learning approaches.

[This page intentionally left blank]

8 Conclusion and Future Work

In this Chapter, a summary of the merits and the limitations of the work conducted is offered and areas for future research are suggested to conclude this dissertation.

8.1 Conclusion

In this study, I developed approaches for solving multiagent decision problems in an uncertain environment, as well as techniques for learning large Bayesian network structures from small data sets.

Firstly, I developed a novel framework to represent multiagent decision problems. This framework comprises Multiply Sectioned Influence Diagrams (MSID) and Hyper Relevance Graph (HRG). An MSID is a probabilistic graphical decision model encoding agency features. It consists of a set of local influence diagrams that represent beliefs, capabilities and preferences of individual agents. The common portion of d-sepset between adjacent local influence diagrams in an MSID indicates the public information shared by the corresponding agents. Except for the public information in the d-sepset, the agents' privacy is protected in the local influence diagrams in an MSID. With the known information in the d-sepset, agents are able to make decisions using only their local information. Hence, an MSID not only describes the states of unpredictable environments, but also characterizes the properties of multiple agents.

On the other hand, an HRG describes organizational relationships in a multiagent system. It categorizes the agents' organizational relationships into two types: control and communication. Within this categorization, an HRG quantifies the information support for decision making of adjacent agents. In an HRG, the required information for the agents' decisions in the control relationship is distinguished from the supporting information in the communication relationship. Elements in an HRG can be obtained from a relevant MSID through the *Decision Bayes-ball* procedure. Due to its emphasis on the domain knowledge, an HRG is able to ensure a compact and accurate MSID.

It is clear that an MSID, together with HRG, is scalable to represent larger decision problems involving multiple agents. Moreover, with a distributed design, they are flexible and reusable to describe decision problems in the changing world. Consequently, this new framework could be utilized to address a general decision problem in the real world.

Secondly, I proposed a symbolic method to verify a valid model representation of MSID and HRG. A valid MSID must obey three constraints: DAG structure, d-sepset and irreducible d-sepset. The first two constraints are more relevant to a valid graphical structure while the third one concerns a compact and accurate knowledge representation. Concerning the verification of the first two constraints, I proposed a symbolic method that exploits an algebraic description of Bayesian networks. The combination of a factorization of joint probability distribution and a form of conditional probability results in a meaningful algebraic description that encodes a Bayesian network structure. Utilizing this algebraic description, the symbolic method is able to test the two constraints of DAG structure and d-sepset through some simple procedures that do not involve graphical operations. It seems that the symbolic method

does not require a strong foundation of graph theory which always complicates the traditional verification of graphical structures. At the same time, the symbolic method initiates an algebraic view on the research of Bayesian network and probabilistic decision models.

For the third constraint of irreducible d-sepset, I proposed a pairwise verification method that utilizes the joint information connecting an MSID with a corresponding HRG. An HRG explicitly represents the organizational relationships with the information support for decision making. Its elements could be obtained either from an MSID or from the abstraction of domain knowledge, or a combination of both of them. A pairwise verification method compared a d-sepset in an MSID with elements obtained in the *Decision Bayes-ball* procedure. These elements could be traced or verified in a relevant HRG. In this way, the pairwise verification method ensured a consistent and accurate representation of domain knowledge in a compact decision model of MSID and HRG.

Besides the verification methods, I also addressed the issue of verification failure. Gathering useful information in the verification process, I provided instructional strategies to correct the parts that cause the verification failure. However, a corrected model requires new verification from scratch, which may cause a correction cycle. Without doubt, future work on a local verification will benefit the solving of verification failure and facilitate the verification process.

Thirdly, I proposed three evaluation algorithms for solving the decision model of MSID. The three evaluation algorithms are categorized into two groups: one is a direct approach that includes a cooperative reduction algorithm and a distributed *evalID* algorithm; the other is an indirect approach that is based on junction tree algorithms for

influence diagrams. A cooperative reduction algorithm extended reduction algorithms in influence diagrams. The procedures *GSL* and *GER* were developed to produce a local and global elimination order for the involved nodes in order to design a valid and effective reduction algorithm in a distributed fashion. Furthermore, the P-Evaluation was recommended to preserve a valid decision model in the evaluation process. A distributed *evalID* algorithm was built based on the *evalID* algorithm in decision networks. A framework of multiple evaluation networks was designed to facilitate the distributed *evalID* algorithm. It ensured consistent information in the evaluation process and allowed an efficient computation process. On the other hand, an indirect approach solved an MSID in two steps: transformation and evaluation. It transformed an MSID into a multiple rooted cluster tree and solved the tree directly. In this approach, a multiple rooted cluster tree was developed to generate helpful working strategies. Clearly, all evaluation algorithms adopted effective strategies to ensure consistent information in an evaluation process. Moreover, they were designed in a distributed fashion so that an efficient computation process was executed.

I also compared these three evaluation algorithms. The comparison results show the outstanding features of these evaluation algorithms and indicate appropriate evaluation techniques for a relevant MSID. The cooperative reduction algorithm is more suitable to solve a typical MSID that includes regular influence diagrams while the other two algorithms have the capability to solve a large and complex MSID. For an MSID including local Bayesian networks, the indirect approach based on junction tree algorithms is recommended. The comparison also indicates that the three evaluation algorithms could be generalized to solve probabilistic decision models. Based on this work on evaluation algorithms in an MSID, it is recommended that an adaptive evaluation framework be built with the integration of more advanced computing

technologies. The adaptive evaluation framework could select appropriate algorithms for solving various MSIDs that cold be developed.

Fourthly, I illustrated a case study to demonstrate the practical applications of my proposed methodologies. The case concerned a decision problem of disease control in the medical domain. It involved a collaboration of multiple nations with the aim to control the spread of the SARS so as to achieve the best benefit for both individual nations and the whole world. I represented this decision problem with MSID and HRG as well as investigated the model validity, scalability and adaptability. To determine the optimal strategies for decisions, I adopted a hybrid evaluation algorithm that was composed of the three evaluation algorithms I proposed in this thesis. A detailed analysis on this case study showed my proposed methodologies could be utilized in practice. For future research, a practical tool should be developed to build the model of MSID and HRG effortlessly, to verify a built model automatically and to solve an MSID efficiently.

Finally, I proposed a block learning algorithm to learn a large Bayesian network structure from a small data set. The block learning algorithms adopted the divide-and-conquer strategy. Instead of learning a whole network structure, it learned individual blocks that have a small size. These learned blocks were combined to recover a whole network. Experimental results on the ALARM network and the Hailfinder network demonstrated that the block learning algorithm outperformed the TPDA algorithm and the PC algorithm. The results also indicated that the block learning algorithm is able to learn a large network structure from a small data set and can be easily configured by other learning techniques. In a theoretical way, I investigated the block learning algorithm and validated its advantages. Moreover, a unifying learning framework was

built on the basis of the block learning algorithm. It may provide a uniform view for a novice to apprehend various learning techniques.

It seems that the block learning algorithm could be a good structure learning algorithms in current literature. However, much effort should be put into the refining of some technical parts in the block learning algorithms such as designing various approaches to identify blocks and to find V-structures. More experiments are required to test the performance of the block learning algorithm and the unifying learning framework. Currently, the block learning algorithm is only tested on the structure learning problem. For the parameter learning problem, the block learning algorithm may have the same advantages as those shown in the structure learning problem. An avenue for future work would be the development of the block "parameter" learning algorithm and its theoretical discussions.

In conclusion, my proposed model representation of MSID and HRG is scalable to describe a large and complex decision problem since the model is designed in a distributed manner. It could be extended to solve a general decision problem in practice. However, it is only partly adaptive to model decision problems in the changing world since the MSID and HRG are a kind of advanced modeling language that disconnects the communication between model engineers and decision analysts. Future work on an evolutionary decision model may facilitate the modeling process. Also, my proposed model verification methods are the first graph verification methods that do not depend on a known graphical structure. The symbolic method provides an algebraic view on the research on probabilistic decision graphs. Future work could concentrate on the issue of model correction when the verification fails. My designed evaluation algorithms have shown a strong ability to solve an MSID and are able to solve general decision models effectively and efficiently. They can serve as a basis to

build an adaptive evaluation framework for solving decision models. However, their practical applications are not so extensive in the real world since there is a linear assumption on a global optimal solution. Further effort on more applications could improve these evaluation algorithms. Finally, my proposed block learning algorithm provides a unifying framework for learning Bayesian network structures. It seems that the same advantages in the block "structure" learning algorithm could be retained in the block "parameter" learning algorithm. Future work on the block learning algorithm for Bayesian network parameters should enrich the learning techniques for Bayesian networks. Thus it is worth developing a learning tool based on the block learning algorithm.

8.2 Future Work

This work has provided insights into the solving of multiagent decision problems as well as the block learning Bayesian network structures. The future agenda of this research work will be carried out in the following two aspects.

Firstly, on the theoretical aspect, there exist some improvements on my proposed methodologies. As mentioned above, the graphical decision models of MSID and HRG still lack enough ability to be adaptive in the changing world. Future work on the formal language design of decision models may facilitate the research work on adaptive decision making. Moreover, it is very significant work to explore the topic of evaluation algorithm selection for solving decision models. It will not only make full use of the existing algorithm reservoir, but will also provide an intelligent inference engine to solve decision models. As for the block learning algorithm, future effort should be invested in the parameter selection in the IB procedure. The theoretical

foundation which will help the development of the unifying learning framework deserves further investigation.

Secondly, on the practical and implementation aspect, there also exists much worthwhile work in the future. The goal of this research work is to pursue some new ideas or methodologies. The ability and capability of MSID and HRG have been demonstrated through a case study in the medical domain; however, some unexpected problems may arise in some agency domains that are more complex. Thus future work on extensive applications of my proposed methodologies on MSID and HRG is desirable and necessary. Concerning the block learning algorithm, future work should emphasize on the tool development. In addition, more effort should put into more experiments and comparisons with other learning algorithms.

Reference

Acid, S. and De Campos, L. M. (2000), Learning Right Sized Belief Networks by Means of a Hybrid Methodology, *Lecture Notes in Artificial Intelligence* **1910**, pp. 309-315.

Acid, S. and De Campos, L. M.. (2001), A Hybrid Methodology for Learning Belief Networks: Benedict, *International Journal of Approximate Reasoning* **27**, pp. 235-262.

Aliferis, C. F. & Cooper, G. F. (1995), A New Formalism for Temporal Modeling in Medical Decision-support Systems, In *Proceedings of the Nineteenth Annual Symposium on Computer Applications in Medical Care*, pp. 213-217.

Aliferis, C. F., Cooper, G. F., Pollack, M. E., Buchanan, B. G. and Wagner, M. M. (1997), Representing and Developing Temporally Abstracted Knowledge as a Means Towards Facilitating Time Modeling in Medical Decision Support Systems, *Computers in Biology and Medicine* **27(5)**, pp. 411-434.

Beinlich, I. A., Suermondt, H. J., Chavez , R. M. and Cooper, G. F. (1989), The ALARM Monitoring System: A Case Study with Two Probabilistic Inference Techniques for Belief Networks, In *Proceedings of the Second European Conference on Artificial Intelligence in Medicine*, pp. 247-256.

Blum, B., C. R. Shelton and Koller, D. (2003), A Continuation Method for Nash Equilibria in Structured Games, In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence*, pp. 757-765.

Bouckaert, R. R. (1993), Belief Networks Construction Using the Minimum Description Length Principle, *Lecture Notes in Computer Science* **747**, pp. 41-48.

Bouckaert, R. R. (1995), *Bayesian Belief Networks: From Construction to Inference*, Ph.D. Thesis, University of Utrecht.

Brown, L.E., Tsamardinos, I., Aliferis, C.F. (2004), A Novel Algorithm for Scalable and Accurate Bayesian Network Learning, In *Proceedings of the Eleventh World Congress on Medical Informatics (MEDINFO)*, pp. 711-716.

Buntine, W. (1991), Theory Refinement of Bayesian Networks, In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 52-60.

Buntine, W. (1994), Operations for Learning with Graphical Models, *Journal of Artificial Intelligence Research* **2**, pp. 159-225.

Buntine, W. (1996), A Guide to the Literature on Learning Probabilistic Networks from Data, *IEEE Transactions on Knowledge and Data Engineering* **8**, pp. 195-210.

Castelo, R. and Kocka, T. (2003), On Inclusion-Driven Learning of Bayesian Networks, *Journal of Machine Learning Research* **4**, pp. 527-574.

Castelo, R. and Siebes, P. (1999), Scaling Bayesian Network Discovery through Incremental Recovery, *Technical Report INS-R9901*, CWI, Amsterdam.

Cheng, J. and Druzdzel, M. (2000), AIS-BN: An Adaptive Importance Sampling Algorithm for Evidential Reasoning in Large Bayesian Networks, *Journal of Artificial Intelligence Research* **13**, pp. 155-188.

Cheng, J., Greiner, R., Kelly, J., Bell, D. A. and Liu, W. (2002), Learning Bayesian Networks from Data: an Information-Theory Based Approach, *The Artificial Intelligence Journal* **137**, pp. 43-90.

Chickering, D. M., Geiger, D. and Heckerman, D. (1995), Learning Bayesian Networks: Search Methods and Experimental Results, In *Preliminary Papers of the Fifth International Workshop on Artificial Intelligence and Statistics*, pp. 112-128.

Chickering, D. M. (1996), Learning Equivalence Classes of Bayesian Network Structures, In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 150-157.

Chickering, D. M. (2002a), Optimal Structure Identification with Greedy Search, *Journal of Machine Learning Research* **3**, pp. 507-554.

Chickering, D. M. (2002b), Learning Equivalence Classes of Bayesian-Network, *Journal of Machine Learning Research* **2**, pp. 445-498.

Chow, C. K. and Liu, C. N. (1968), Approximating Discrete Probability Distributions with Dependence Trees, *IEEE Transactions on Information Theory* **12**, pp. 462-467.

Chu, T. and Xiang, Y. (1997), Exploring Parallelism in learning Belief Networks, In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pp. 90-98.

Cooper, G. F. (1987), *Probabilistic Inference Using Belief Networks is NP-hard*, Knowledge Systems Laboratory, Stanford University, Memo KSL-87-27.

Cooper, G. F. (1988), A Method for Using Belief Networks as Influence Diagrams, In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 211-219.

Cooper, G. F. (1990), The Computational Complexity of Probabilistic Inference Using Bayesian Belief Networks, *Artificial Intelligence* **42**, pp. 393-405.

Cooper, G. F. and Herskovits, E. (1992), A Bayesian Method for the Induction of Probabilistic Networks from Data, *Machine Learning* **9**, pp. 309-347.

Cousot, R. (2005), Verification, Model Checking, and Abstract Interpretation, *Lecture Notes in Computer Science* **3385**, Springer.

Dagum, P. and Luby, M. (1993), Approximating Probabilistic Inference in Belief Networks is NP-hard, *Artificial Intelligence* **60(1)**, pp. 41-48.

Dash, H. D. and Druzdzel M. J. (1999), A Hybrid Anytime Algorithm for the Construction of Causal Models from Sparse Data, In *Proceedings of the Fifteenth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 142-149.

De Campos, L. M. (1998), Independency Relationships and Learning Algorithms for Singly Connected Networks, *Journal of Experimental and Theoretical Artificial Intelligence* **10**, pp. 511-549.

De Campos, L. M., Fernández-Luna, J. M., Gámez, J. A. and Puerta, J. M. (2002), Ant Colony Optimization for Learning Bayesian Networks, *International Journal of Approximate Reasoning* **31**, pp. 291-311.

De Campos, L. M., Fernández-Luna, J. M. and Puerta, J. M. (2003), An Iterated Local Search Algorithm for Learning Bayesian Networks with Restarts based on Conditional Independence Tests, *International Journal of Intelligent Systems* **18**, pp. 221-235.

Dean, T. and Kanazawa, K. (1989), A Model for Reasoning about Persistence and Causation, *Computational Intelligence* **5**, pp. 142-150.

Dean, T. and Wellman, M. P. (1991), *Planning and Control*, Morgan Kaufmann Publishers, San Mateo, CA.

Dechter, R. (1996), Bucket Elimination: A Unifying Framework for Probabilistic Inference, In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 211-219.

Dechter, R. (2003), *Constraint Processing*, Morgan Kaufmann.

Downing, T. E., Moss, S. and Pahl, W. C. (2001), Understanding Climate Policy Using Participatory Agent-based Social Simulation, In *Multi-Agent-Based Simulation*, pp. 198-213.

Draper, D. (1995), *Localized Partial Evaluation of Belief Networks*, PhD Thesis, Department of Computer Science, University of Washington.

Druzdzel, M. J. and Suermondt, H. J. (1994), Relevance in Probabilistic Models: "Backyards" in a "Small World", In *Working Notes of the AAAIFall Symposium Series: Relevance*, pp. 60-63.

Durfee, E. H. (1988), *Coordination of Distributed Problem Solvers*, Kluwer Academic, Boston, MA.

Durfee, E. H. (1996), Planning in Distributed Artificial Intelligence, In *Foundations of Distributed Artificial Intelligence* (Eds.: G. M. P. O'Hare and N. R. Jennings), pp. 231-245.

Durfee, E. H., Lesser, V. R. and Corkill, D. D. (1989a), Cooperative Distributed Problem Solving, In *Handbook of Artificial Intelligence* **4** (Eds.: E. A. Feigenbaum, A. Barr and P. R. Cohen), pp. 83-147.

Durfee, E. H., Lesser, V. R. and Corkill, D. D. (1989B), Trends in Cooperative Distributed Problem Solving, *IEEE Transactions on Knowledge and Data Engineering* **1(1)**, pp. 63-83.

Edwards, W. (1998), Hailfinder: Tools for and Experiences with Bayesian Normative Modeling, *American Psychologist* **53**, pp. 416-428.

Foster, I., Kesselman, C., Nick, J. and Tuecke, S. (2002), The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration, *Open Grid Service Infrastructure WG*, Global Grid Forum.

Friedman, N. and Goldszmidt, M. (1996), Learning Bayesian Networks with Local Structure, In *Proceedings of the Twelfth Conference on Uncertainty in Artificial Intelligence*, pp. 252-262.

Friedman, N., Nachman, I. and Pe'er, D. (1999), Learning Bayesian Networks Structure from Massive Dataset: The "Sparse Candidate" Algorithm, In *Proceedings of the Fifteen Conference on Uncertainty Artificial Intelligence*, pp. 206-215

Friedman, N. and Koller, D. (2000), Being Bayesian about Network Structure, In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 201-210.

Friedman, N., Ninio, M., Pe'er, I. and Pupko, T. (2002), A structural EM Algorithm for Phylogenetic Inference, *Journal of Computational Biology* **9**, pp. 169-191.

Fudenberg, D. and Tirole, J. (1991), *Game Theory*, The MIT Press.

Fung, R. and Chang, K. C. (1989), Weighting and Integrating Evidence for Stochastic Simulation in Bayesian Networks, In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 475-482.

Fung, R. and Favero, B. (1994), Backward Simulation in Bayesian Networks, In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 227-234.

Galan, S. F., Aguado, F., Diez, F. J. and Mira, J. (2002), NasoNet: Modeling the Spread of Nasopharyngeal Cancer with Networks of Probabilistic Events in Discrete Time, *Artificial Intelligence in Medicine* **25(4)**, pp. 247-264.

Garcia, S. D. and Druzdzel, M. J. (2004), An Efficient Sampling Algorithm for Influence Diagrams, In *Proceedings of the Second European Workshop on Probabilistic Graphical Models*, pp. 97-104,.

Geiger, D. and Heckerman, D. (1995), A Characterization of the Dirichlet Distribution with Application to Learning Bayesian Networks, In *Proceedings of the Eleventh Conference on Uncertainty in Artificial Intelligence*, pp. 196-207.

Geiger, D., Verma, T. and Pearl, J. (1989), D-separation: From Theorems to Algorithms, In *Proceedings of the Fifth Workshop on Uncertainty in Artificial Intelligence*, pp. 118-125.

Glymour, C. and Cooper, G. F. (1999), *Computation, Causation, and Discovery*, Cambridge, MA, USA, MIT press.

Heckerman, D. (1990), *Probabilistic Similarity Networks*, PhD Thesis, The MIT Press.

Heckerman, D., Geiger, D. and Chickering, D. M. (1995), Learning Bayesian Networks: The Combination of Knowledge and Statistical Data, *Machine Learning* **20**, pp. 197-243.

Heckerman, D. (1995), A Tutorial on Learning Bayesian Networks, *Technical Report* MSR-TR-95-06, Microsoft Research.

Heckerman, D. (1996), Bayesian Networks for Knowledge Discovery, In *Advances in Knowledge Discovery and Data Minin*g (Eds.: Fayyad, U. M., Piatetsky-Shapiro, G., Smyth, P. and R. Uthurusamy), Cambridge: MIT Press, pp. 273-305.

Heckerman, D., Meek, C. and Koller, D. (2004), Probabilistic Models for Relational Data, *Technical Report*, Microsoft Research.

Henrion, M. (1988), Propagating Uncertainty in Bayesian Networks by Probabilistic Logic Sampling, In *Uncertainty in Artificial Intelligence* **2** (Eds.: Lemmer, J. F. and L. N. Kanal), pp. 149-163.

Henrion, M. (1991), Search Based Methods to Bund Diagnostic Probabilities in Very Large belief Nets, In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 142-150.

Herskovits, E. (1991), *Computer-based Probabilistic Network Construction*, Doctoral Dissertation, Medical Information Sciences, Stanford University, Stanford, CA.

Herskovits, E. and Cooper, G. F. (1990), Kutató: An Entropy-driven System for the Construction of Probabilistic Expert Systems from Databases, In *Proceedings of the Sixth Conference on Uncertainty in Artificial Intelligence*, pp. 54-62.

Howard, R. A. and Matheson, J. E. (1984), Influence Diagrams. In *Readings on the Principles and Applications of Decision Analysis* **2** (Eds.: Howard, R. A. and J. E. Matheson), pp. 719-726.

Jennings, N. R. (2000), On Agent-base Software Engineering, *Artificial Intelligence* **117**, pp. 227-296.

Jensen, F. V., Lauritzen, K. G. and Olesen, K. G. (1990), Bayesian Updating in Causal Probabilistic Networks by Local Computations, *Computational Statistical Quarter* **4**, pp. 269-282.

Jensen, F., Jensen, F. V, and Dittmer, S. L (1994)， From Influence Diagrams to Junction Trees, In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 367-374.

Jensen, F. V. (1996), *An Introduction to Bayesian Networks*, Springer, New York.

Jensen, F. V. (2001), *Bayesian Networks and Decision Graphs*, Springer, New York.

Jensen, F. V. and Marta Vomlelova (2002), Unconstrained Influence Diagrams, In *Proceedings of the Eighteenth Conference of Uncertainty in Artificial Intelligence*, pp. 234-241.

Jensen, F. V., Nielsen, T. D. and P. P. Shenoy (2004), Sequential Influence Diagrams: A Unified Asymmetric Framework, In *Proceeding of the Second Workshop on Probabilistic Graphical Models*, pp. 121-128.

Jiang, C.A., Poh, K.L. and Leong, T.Y. (2005), Integration of Probabilistic Graphic Models for Decision Support, In *Proceedings of the 2005 AAAI Spring Symposium on Challenges to Decision Support in a Changing World*, pp. 40-47.

Joseph, L., Parmigiani, G. and Hasselblad, V. (1998), Combination Expert Judgment by Hierarchical Modeling: An Application to Physician Staffing, *Management Science* **44**, pp.149-161.

Kearns, M., M. L. Littman and Singh, S. (2001a), Graphical Models for Game Theory, In *Proceedings of the Seventeenth Conference on Uncertainty in Artificial Intelligence*, pp. 253-260.

Kearns, M., M. L. Littman and Singh, S. (2001b), An Efficient Exact Algorithm for Singly Connected Graphical Games, In *Proceedings of the Fourteenth Conference on Neural Information Processing Systems*, pp. 817-823

Kjærulff, U. (1994), Reduction of Computation Complexity in Bayesian Networks Through Removal of Weak Dependencies, In *Proceedings of Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 374-382.

Kjærulff, U. (1997), A Computational Scheme for Reasoning in Dynamic Probabilistic Networks, In *Proceedings of Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 121-129.

Koller, D. and Pfeffer, A. (1997), Object-Oriented Bayesian Networks, In *Proceedings of the Thirteenth Conference of Uncertainty in Artificial Intelligence*, pp. 302-313.

Koller, D. and Pfeffer, A. (1998), Probabilistic Frame-based Systems, In *Proceedings of the Fourteenth Conference of Uncertainty in Artificial Intelligence*, pp. 580-587.

Koller, D. (1999), Probabilistic Relational Models, In *Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP-99)*, pp. 3-13.

Koller, D. and B. Milch (2001), Multi-Agent Influence Diagrams for Representing and Solving Games, In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, Washington, pp. 1027-1034.

La Mura, P. (2000), Game Networks, In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 335-342.

La Mura, P. and Shoham, Y. (1999), Expected Utility Networks, In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 366-373.

Lam, W and Bacchus, F. (1994), Learning Bayesian Belief Networks: An Approach Based on the MDL Principle, *Computational Intelligence* **10(4)**, pp. 269-293.

Lam, W. and Segre, A. M. (2002), A distributed Learning Algorithm for Bayesian Inference Networks, *IEEE Transactions on Knowledge and Data Engineering* **12(1)**, pp. 93-105.

Larranaga P., Poza M., Yurramendi Y., Murga R. H. and Kuijpers C. M. H. (1996), Structure Learning of Bayesian Networks by Genetic Algorithms: A Performance Analysis of Control Parameters**,** *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18(9)**, pp. 912-926.

Lauritzen, S. L. and D. J. Spiegelhalter (1988), Local Computations with Probabilities on graphical Structures and Their Applications to Expert Systems (with discussion), *Journal of the Royal Statistical Society Series B* **50**, pp. 157-224.

Lauritzen, S. L. and D. Nilsson (2001), Representing and Solving Decision Problems with Limited Information, *Management Science* **47**, pp. 1238-1251.

Leong, T. Y. (1994), *An Integrated Approach to Dynamic Decision Making under Uncertainty*, TR-631, MIT Laboratory for Computer Science.

Lesser, V. R. and Erman, L. D. (1980), Distributed Interpretation: A Model and Experiment, *IEEE Transactions on Computers* **29(12)**, pp. 1144-1163.

Getoor, L. (2001), *Learning Statistical Models from Relational Data Networks*, PhD Thesis, Stanford University.

Li, Z. and Ambrosio, B. D. (1994), Efficient Inference in Bayes' Nets as a Combinatorial Optimization Problem, *International Journal of Approximate Reasoning* **4**, pp. 55-81.

Madigan, D. and Raftery, A. (1994), Model Selection and Accounting for Model Uncertainty in Graphical Models Using Occam's Window, *Journal of the American Statistics Association* **89**, pp. 1535-1546.

Madsen, A. L. and Jensen F. V. (1998), Lazy Propagation in Junction Trees, In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pp. 211-219.

Margatitis, D. (2003), *Learning Bayesian Network Model Structure from Data*, PhD Thesis, School of Computer Science, Carnegie Mellon University.

Matzkevich, I. and Abramson, B. (1992), The Topological Fusion of Bayes Nets, In *Proceedings of the Eighth Annual Conference on Uncertainty in Artificial Intelligence*, pp. 191-198.

Mckelvey, R. and McLennan, A. (1996), Computation of Equilibria in Finite Games. *Handbook of Computational Economics*, pp. 87-142.

Muntenau, P. and Cau, D. (2000), Efficient Score-based Learning of Equivalence Classes of Bayesian Networks, *Lecture Notes in Artificial Intelligence* **1910**, pp. 96-105.

Myers, J. W., Laskey, K. B. and Levitt, T. (1999), Learning Bayesian Networks from Incomplete Data with Stochastic Search Algorithms, In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 476-485.

Nash, J. (1950), Equilibrium Points in N-Person Games, *PNAS* **36**, pp. 48-49.

Neapolitan, R. E. (1990), *Probabilistic Reasoning in Expert Systems*, Wiley and Sons, New York.

Neapolitan, R. E. (2004), *Learning Bayesian Networks*, Prentice Hall.

Ndilikikesha, P. (1994), Potential Influence Diagrams, *International Journal of Approximate Reasoning* **11**, pp. 251-285.

Nicholson, A. E. (1992), *Monitoring Discrete Environments Using Dynamic Belief Networks*, PhD Thesis, Department of Engineering Sciences, Oxford.

Nicholson, A. E. and Brady, J. M. (1992), The Data Association Problem When Monitoring Robot Vehicles Using Dynamic Belief Networks, In *Proceedings of the Tenth European Conference on Artificial Intelligence*, pp. 689-693.

Nicholson, A. E. and Brady, J. M. (1992), Sensor Validation Using Dynamic Belief Networks, In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 207-214.

Nielsen, T. D. (2001), *Graphical Models for Partially Sequential Decision Problems*, PhD Thesis, Department of Computer Science, Aalborg University.

Nielsen, T. and Jensen, F. V. (1999), Well-defined Decision Scenarios, In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp 502-511.

Olmsted, S. M. (1983), On Representing and Solving Decision Problems, PhD Thesis, Department of Engineering-Economic Systems, Stanford University.

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

Pearl, J. and Verma, T. S. (1991), A Theory of Inferred Causation, In *Principles of Knowledge Representation and Reasoning* (Eds.: Allen, J. F., Fikes, R. and E. Sandewall), pp. 441-452.

Poole, D. (1993), Probabilistic Horn Abduction and Bayesian Networks, *Artificial Intelligence* **64(1)**, pp. 81-129.

Ramoni, M. and Sebastiani, P. (1997), Learning Bayesian Networks from Incomplete Databases, In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pp. 401-408.

Rao, A. S. and Georgeff, M. P. (1995), BDI Agents: From Theory to Practice, *Technical Report **56***, Australian Artificial Intelligence Institute, Melbourne, Australia.

Rebane, G. and Pearl, J. (1987), The Recovery of Causal Poly-trees from Statistical Data, In *Uncertainty in Artificial Intelligence* **3** (Eds.: Kanal, L.N., Levitt, T.S. and J. F. Lemmer), Amsterdam: North-Holland, pp. 222-228.

Rege, A. and Agogino, A. M. (1988), Topological Framework for Representing and Solving Probabilistic Inference Problems in Expert Systems, *IEEE Transactions on Systems, Man and Cybernetics* **18 (3)**, pp. 402-414.

Robert, T. C. and Terry, R. (2001), *Making Hard Decisions with Decision Tools*, Duxbury/Thomson Learning.

Russell, S. and Norvig, P. (2003), *Artificial Intelligence: A Modern Approach*, Prentice Hall, Englewood Cliffs.

Ryan, P., Eugene, N. and Shoham, Y. (2004), Simple Search Methods for Finding a Nash Equilibrium, In *Proceedings of American Association for Artificial Intelligence (AAAI)*, pp. 664-669.

Sanguk, N. and Gmytrasiewicz, P. J. (1998), Rational Communicative Behavior in Anti-air Defense, In *Proceedings of the Third International Conference on Multi-Agent Systems*, pp. 214-221.

Segal, E. Pe'er, D., Regev, A., Koller, D. and Friedman, N. (2003), Learning Module Networks, In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pp. 7-15.

Shachter, R. D. (1986), Evaluating Influence Diagrams, *Operations Research* **34** (6), pp. 871-882.

Shachter, R. D. (1988), Probabilistic Inference and Influence Diagrams, *Operations Research* **36**, pp. 589-605.

Shachter, R. D. (1999), Efficient Value of Information Computation, *In Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pp. 594-601.

Shacthter, R.D., B. D'Ambrosio, B. and B. A. Del Favero (1990), Symbolic Probabilistic Inference in Belief Networks, In *Proceedings of the Eighth National Conference on Artificial Intelligence I*, pp. 126-131.

Shachter, R. and M. A. Peot. (1989), Simulation Approaches to General Probabilistic Inference on Belief Networks, In *Proceedings of the Fifth Conference on Uncertainty in Artificial Intelligence*, pp. 311-318.

Shachter, R. and M. A. Peot. (1992), Decision Making Using Probabilistic Inference Methods, In *Proceedings of the Eighth Conference on Uncertainty in Artificial Intelligence*, pp. 276-283.

Shafer, G. (1996), *Probabilistic Expert Systems, Society for Industrial and Applied Mathematics*, Philadelphia.

Shenoy, P. (1992), Valuation-Based Systems for Bayesian Decision Analysis, *Operations Research* **40 (3)**, pp. 463-484.

Singh, M. and Valtorta, M. (1993), An Algorithm for the Construction of Bayesian Network Structures from Data, In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 259-265.

Singh, M. and Valtorta, M. (1995), Construction of Bayesian Network Structures from Data: A brief Survey and an Efficient Algorithm, *International Journal of Approximate Reasoning* **12**, pp. 111-131.

Smith, R. G. (1977), The CONTRACT NET: A Formalism for the Control of Distributed Problem Solving, In *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, pp. 472.

Smith, R. G. (1980a), The Contract Net Protocol, *IEEE Transactions on Computers* **29(12)**, pp. 1104-1113.

Smith, R. G. (1980b), *A Framework for Distributed Problem Solving*, UMI Research Press.

Smith, R. G. and Davis, R. (1980), Frameworks for Cooperative in Distributed Problem Solving, *IEEE Transactions on Systems, Man and Cybernetics* **11(1)**, pp. 24-33.

Spirtes, P., Glymour. G. and Scheines, R. (1990), Causality from Probability, In *Proceedings of Advanced Computing for the Social Sciences*, Williamsburgh, VA.

Spirtes, P., Glymour, G. and Scheines, R. (1993), *Causation, Prediction and Search*, New York, Springer-Verlag.

Spirtes, P. and Meek, C. (1995), Learning Bayesian Networks with Discrete Variables from Data, In *Proceedings of the First International Conference on Knowledge Discovery and Data Mining*, pp. 294-299.

Spirtes, P., Glymour. G. and Scheines, R. (2000), *Causation, Prediction and Search*, Cambridge, Mass, MIT Press.

Srinivas, S. (1994), A Probabilistic Approach to Hierarchical Model-Based Diagnosis, In *Proceedings of the Tenth Conference of Uncertainty in Artificial Intelligence*, pp. 538-545.

Steck, H. (2000), On the Use of Skeletons When Learning in Bayesian Networks, In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 558-565.

Suermondt, J., and Cooper, G. (1991), Initialization for the Method of Conditioning in Bayesian Belief Networks, *Artificial Intelligence* **50**, pp. 83-94.

Suzuki, J. (1993), A Construction of Bayesian Networks from Databases Based on the MDL Principle, In *Proceedings of the Ninth Conference on Uncertainty in Artificial Intelligence*, pp. 266-273.

Suzuki, J. (1996), Learning Bayesian Belief Networks Based on MDL Principle: An Efficient Algorithm Using the Branch and Bound Technique, In *Proceedings of the International Conference on Machine Learning*.

Tatman, J. A. and Shachter, R.D. (1990), Dynamic Programming and Influence Diagrams, *IEEE Transactions on Systems, Man and Cybernetics* **20 (2)**, pp. 365-379.

Tian, J. (2000), A branch-and-bound Algorithm for MDL Learning Bayesian Networks, In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pp. 580-587.

Tsamardinos, I., Aliferis, C. F., Statnikov, A. and Brown, L. E.(2003), Scaling-Up Bayesian Network Learning to Thousands of Variables Using Local Learning Technique, *DSL TR-03-02*, March 12, 2003, Vanderbilt University, Nashville, TN, USA.

Tsamardinos, I., Aliferis, C. F., and Statnikov, A. (2003), Algorithms for Large Scale Markov Blanket Discovery, In *Proceedings of the Sixteenth International FLAIRS Conference*.

Vickrey, D. and Koller, D. (2002), Multi-Agent Algorithms for Solving Graphical Games, In *Proceeding of American Association for Artificial Intelligence (AAAI)*, pp. 345-351.

Von Stengel, B. (2002), Computing Equilibria for Two-Person Games, *Handbook of Game Theory*, pp. 1723-1759.

Von Neumann, J. and Morgenstern, O. (1947), *The Theory of Games and Economic Behavior*, Princeton: Princeton University Press, 2nd Edition.

Wallace, C., Korb, K. B. and Dai, H. (1996), Causal Discovery via MML, In *Proceedings of the Thirteenth International Conference on Machine Learning*, pp. 516-524.

Weiss, G. (1999), *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, The MIT Press.

Wellman, M. P. and Liu, C. L. (1994), State-space Abstraction for Anytime Evaluation of Probabilistic Networks, In *Proceedings of the Tenth Conference on Uncertainty in Artificial Intelligence*, pp. 567-574.

Wellman, M. P., Birmingham, W. P. and Durfee, E. H. (1996), The Digital Library As a Community of Information Agents, *IEEE Expert* **11(3)**, pp. 10-11.

Wermuth, N. and Lauritzen, S. (1983), Graphical and Recursive Models for Contingency Tables, *Biometrika* **72**, pp. 537-552.

Wong, M. L., Lam, W. and Leung, K. S. (1999), Using Evolutionary Computation and Minimum Description Length Principle for Data Mining of Probabilistic Knowledge, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **21**, pp. 174-178.

Wong, S. K. M. and Wu, D. (2002), An Algebraic Characterization of Equivalent Bayesian Networks, In *Proceeding of the Seventeenth World Computer Congress - TC12 Stream on Intelligent Information*, pp. 177-187.

Wooldridge, M. and N. R. Jennings (1995), Intelligent Agents: Theory and Practice, In *Knowledge Engineering Review* **10 (2)**, pp. 115-152.

Wooldridge, M. (2002), *An Introduction to Multiagent Systems*. John Wiley and Sons Ltd.

Wu, X. and Poh, K. L. (1998), Decision Model Construction with Multilevel Influence Diagrams, In *Proceedings of AAAI 1998 Spring Symposium on Interactive and Mixed-Initiative Decision Theoretic Systems*, pp. 142-147.

Xiang, Y. (1996), A Probabilistic Framework for Cooperative Multi-agent Distributed Interpretation and Optimization of Communication, *Artificial Intelligence* **87 (1-2)**, pp. 295-342.

Xiang, Y. (1998), Verification of DAG Structures in Cooperative Belief Network Based Multiagent Systems, *Networks* **31**, pp. 183-191.

Xiang, Y. (2002), *Probabilistic Reasoning in Multiagent Systems: A Graphical Models Approach*, Cambridge University Press.

Xiang, Y. and Chen, Y. (2002), Cooperative Verification of Agent Interface, In *Proceedings of the First European Workshop on Probabilistic Graphical Models*, pp. 194-203.

Xiang, Y., Poole, D. and Beddoes, M. P. (1993) Multiply Sectioned Bayesian Networks and Junction Forests for Large Knowledge Based Systems, *Computational Intelligence* **9(2)**, pp.171-220.

Xiang, Y., Pant, B., Eisen, A., Beddoes, M. P. and Poole, D. (1993), Multiply Sectioned Bayesian Networks for Neuromuscular Diagnosis, *Artificial Intelligence in Medicine* **5**, pp. 293-314.

Xiang, Y. (1994), Distributed Multi-agent Probabilistic Reasoning with Bayesian Networks, *Methodologies for Intelligent Systems* (Eds.: Z. W. Ras and M. Zemankova), pp. 285-294.

Xiang, Y. (2003), Comparision of Multiagent Inference Methods in Multiply Sectioned Bayesian Networks, *International Journal of Approximate Reasoning* **33(3)**, pp. 235-254.

Xiang, Y. P. and Poh, K. L. (1999), Time-Critical Dynamic Decision Making, In *Proceedings of Fifteen Conference on Uncertainty in Artificial Intelligence*, pp. 688-695.

Yuan, C. and Druzdzel, M. J. (2003), An Importance Sampling Algorithm Based on Evidence Pre-propagation, In *Proceedings of the Nineteenth Conference on Uncertainty in Artificial Intelligence*, pp. 624-631.

Zambonelli, F., Jennings N. R. and Wooldridge, M. (2001), Organizational Abstractions for the Analysis and Design of Multiagent Systems, In *Agent-Oriented Software Engineering* (Eds.: Ciancarini, P. and M. Wooldridge), Springer-Verlag Lecture Notes in Artificial Intelligence.

Zhang, N. L. (1994), *A Computational Theory of Decision Networks*, Ph.D. Dissertation, Department of Computer Science, University of British Columbia.

Zhang, N. L., R. Qi and D. Poole (1994), A Computational Ttheory of Decision Networks, *International Journal of Approximate Reasoning* **11(2)**, pp. 83-158.

Zhang, N. L. (1998), Probabilistic Inference in Influence Diagrams, *Journal of Computational Intelligence* **4**, pp. 476-497.