COMMUNICATION MANAGEMENT IN DISTRIBUTED SENSOR INTERPRETATION

A Dissertation Presented

by

JIAYING SHEN

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2007

Computer Science

COMMUNICATION MANAGEMENT IN DISTRIBUTED SENSOR INTERPRETATION

A Dissertation Presented

by

JIAYING SHEN

Approved as to style and content by:

Victor Lesser, Chair

Norman Carver, Member

Abhijit Deshmukh, Member

Sridhar Mahadevan, Member

Shlomo Zilberstein, Member

W. Bruce Croft, Department Chair
Computer Science

ABSTRACT

COMMUNICATION MANAGEMENT IN DISTRIBUTED SENSOR INTERPRETATION

MAY 2007

JIAYING SHEN

B.E., SHANGHAI JIAO TONG UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Victor Lesser

Distributed Sensor Interpretation (DSI) problems have been the subject of considerable research within the cooperative MAS community. In a DSI system, data is collected from different sensors and must be integrated to produce the best interpretation. Distributed approaches to DSI emphasize not only distributed collecting of data, but also distributed processing of data. However, in virtually all real-world DSI systems the agents must exchange data, local results, and/or other information to develop a solution with acceptable quality. Unless communication among the agents is appropriately limited, the cost of communication may negate much of the benefit of distributed processing. Unfortunately, the state-of-the-art in MAS is such that there are not yet formal design methods that allow one to evaluate a potential DSI domain and determine the optimal coordination strategy. I believe that this is a serious issue that will hinder the deployment of many important applications of sensor networks.

My work is one of the first attempts to address this issue. I formalized the communication problem in DSI with a Distributed Bayesian Network and solved the question of what to communicate with a decentralized Markov Decision Process (DEC-MDP). With this model, one is able to generate a communication strategy for a given DSI problem such that only minimum communication cost is needed to achieve a required confidence level in the interpretation task.

Though general communication can be naturally modeled with a DEC-MDP, techniques need to be developed to address the complexity issue before the system can be scaled up. I approach this problem from two perspectives. First, I proposed an algorithm to automatically generate a set of abstract communication actions such that when the abstract information is transferred between the agents, the goal of the system is more likely to be reached. By allowing only abstract communication actions in certain states, both the expected communication cost required and the time needed to solve the DEC-MDP are reduced. Second, I established that the interaction present among the agents is the cause of the high complexity of a DEC-MDP. This understanding is crucial to identifying new and more tractable models as well as developing appropriate approximations to otherwise intractable problems. I proved that deciding a distributed MDP whose interaction history contains information of a size polynomial in the number of states is NP-complete, and that deciding a non-polynomially encodable distributed MDP is harder than NP. This is the first time that a well defined condition has been identified that can distinguish between multi-agent problems in NP and those that are strictly harder than NP. It is an important step in mapping out the complexity hierarchy of multi-agent systems. The significance of this theoretical result also has a more practical side. Most multi-agent systems are provably harder than NP and solving them optimally is not possible. This work provides theoretical guidance in understanding how the approximations in a model limit the search space and reduce the complexity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
INTRODUCTION

Decision theoretic models have received considerable attention in recent years by the artificial intelligence community. Markov Decision Processes (MDPs) [68, 7] and Partially Observable Markov Decision Processes (POMDPs) [55, 48, 15] have been adopted as a general framework for planning under uncertainty [21] and reinforcement learning [1]. An MDP/POMDP describes a problem as a finite set of states and solves the problem by finding the optimal action that maximizes the agent's expected cumulative reward. From a formal perspective, they are attractive because they offer an extremely general framework and because they have been studied extensively within operations research and control theory for many years. Building on earlier work, AI researchers have developed attractive new techniques that can solve large MDP/POMDPs using domain background knowledge and structure [9, 24, 30].

However, extending the decision theoretic models to the multi-agent systems is not straightforward. The interactions among the agents and their limited knowledge of the entire system increases the complexity of the system significantly. Therefore, most traditional research in multi-agent systems is focusing on heuristic approaches. Examples include the belief-desire-intention (BDI) architecture [18, 71], the Generalized Partial Global Planning (GPGP) approach [42], and the STEAM model [77]. Though all of these approaches have been used in multi-agent coordination problems successfully, they are largely heuristic and experimental. In order to understand the behaviors and performance of a multi-agent system, we need to look at the problems from a more formal perspective.

Recent work has extended the standard MDP/POMDP models to address the problem of coordination and control of collaborative multi-agent systems [8, 66, 5]. Decentralized Markov decision process (DEC-MDP), a generalization of MDP, is proposed in [5]. In a DEC-MDP, the decision process is controlled by multiple distributed agents, each with possibly different information about the state. At each time step the agents' observations together uniquely determine the global state. Though solving DEC-MDP is an extremely hard problem, new techniques and algorithms have been developed to solve it exactly or approximately [13, 4, 28, 75, 57, 32, 6]. This makes it a suitable formal tool to model various multi-agent coordination problems.

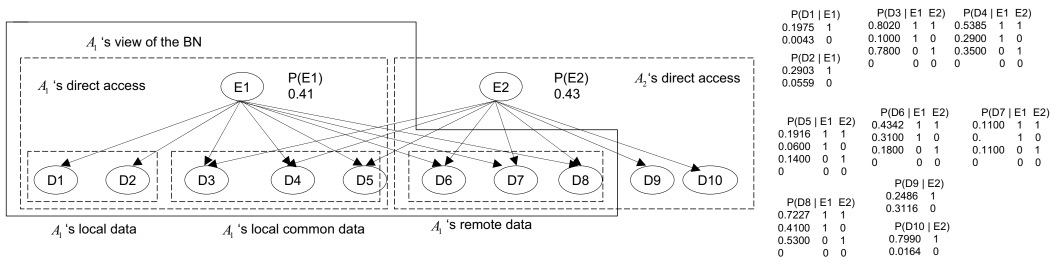
Extensions to the DEC-MDP that explicitly model communication between agents have been proposed and studied [69, 28]. However, most research is focused on the type of communication that synchronizes the partial views of the global state [81, 28, 2, 3]. Whenever communication occurs in the system, all the agents inform the other agents of their observations since the last time communication happened, such that all the agents have the same view of the system. This type of communication is useful when the communication cost for different messages are the same, but not suitable for situations where different communication actions have different costs. Therefore, a study of more general form of communication is needed.

In this work, I will look at a class of problems where the only actions in the DEC-MDP are connected with deciding what and when to communicate and try to understand from a formal perspective the characteristics of these problems that make them hard to solve. I will study the problem of communication management in a distributed sensor network. It is a realistic example problem that is indicative of some of the issues that need to be solved in order to construct effective (though not necessarily) optimal DEC-MDPs for the solution of this example problem.

Distributed Sensor Interpretation (DSI) has been the subject of considerable research within the *multi-agent systems* (MAS) community because advances in sensor technology are leading to the deployment of large networks of sophisticated sensors [43]. Interest in getting robots and other computer systems to interact more autonomously with their environments is also driving the introduction of sensors into a wider array of environments. A key characteristic of large-scale networks of sensors is that they are inherently *distributed*: the sensors will be spatially distributed, possibly over large geographic areas.

An interpretation of a sensor data set is typically expressed as a set of events that could have occurred in the environment and would explain the sensor data. Events may be things like aircraft or other vehicle moving through the sensed region, the detection of walls or other obstacles as a robot moves, or so forth. In most cases, there are multiple possible interpretations for any reasonably sized set of data. The algorithm used by an sensor interpretation system must not only identify possible interpretations, it must also select the best of the possibilities to report as the solution.

In a DSI system, data is collected from different sensors and need to be integrated to produce the best interpretation. This can be done either in a centralized fashion or distributedly. In centralized approaches to such problems, the data from all of the sensors is transmitted to a single agent that does all of the computation to interpret the data [74]. Distributed approaches to DSI emphasize not only distributed processing of data, but also decentralized/distributed control via a system of intelligent agents. When a sensor network is distributed in nature, the distributed approach can have many potential advantages over the centralized approach. The use of multiple processors has the potential to speed up processing and/or reduce the cost of the computing hardware. System costs can also be reduced if each agent/processor is located in close physical proximity to its sensor(s), because local processing of data will lower the bandwidth requirements of the communication elements of the network. Local processing has the added benefit of improving real-time control over control-



P(D1 | E1)	
0.1975	1
0.0043	0

P(D2 | E1)	
0.2903	1
0.0559	0

P(D3 | E1	E2)	
0.8020	1	1
0.1000	1	0
0.7800	0	1
0	0	0

P(D4 | E1	E2)	
0.5385	1	1
0.2900	1	0
0.3500	0	1
0	0	0

P(D5 | E1	E2)	
0.1916	1	1
0.0600	1	0
0.1400	0	1
0	0	0

P(D6 | E1	E2)	
0.4342	1	1
0.3100	1	0
0.1800	0	1
0	0	0

P(D7 | E1	E2)	
0.1100	1	1
0.	1	0
0.1100	0	1
0	0	0

P(D8 | E1	E2)	
0.7227	1	1
0.4100	1	0
0.5300	0	1
0	0	0

P(D9 | E2)	
0.2486	1
0.3116	0

P(D10 | E2)	
0.7990	1
0.0164	0

Figure 1.1. There are two events E_1 and E_2. Data $D_1, D_2, ...D_{10}$ are distributed to two agents. A_1 has access to $D_1, ...D_5$ and is responsible for solving E_1, while A_2 can only see $D_6, ...D_{10}$ and is responsible for solving E_2. The objective is for A_1 and A_2 to figure out what E_1 and E_2 are with required confidence and with minimum expected communication cost

lable sensors. Finally, they can lead to more reliable systems, where performance degrades gracefully if processors and/or communication links fail.

The distributed approach to DSI can often be represented in a Distributed Bayesian Network (DBN). As a graphical representation, A Bayesian Network (BN) [65] captures both the quantitative and qualitative aspects of the causal relationships between related events. A DBN is a BN whose nodes are distributed to different agents and is a powerful probabilistic reasoning tool to represent the causal relationship between the data collected by sensors and the events that may have occurred in the environment in the DSI problem. An example DSI problem represented in a DBN is shown in Figure 1.1. The top level nodes are the events that are the possible causes of the observed data, while the leaves are the observed data gathered by various agents. There are two agents, each of whom has direct access to only a part of the observable data. The interpretation task is distributed to the two agents as well. Each agent is only responsible for its part of the overall problem and only has the knowledge of the part of the network that is relevant to its task.

I use SensEye [41] as an example of a distributed sensor network structure and show how an interpretation problem is represented in a Bayesian Network. SensEye is a multi-tier network of heterogeneous wireless nodes and camera sensors. Resource-

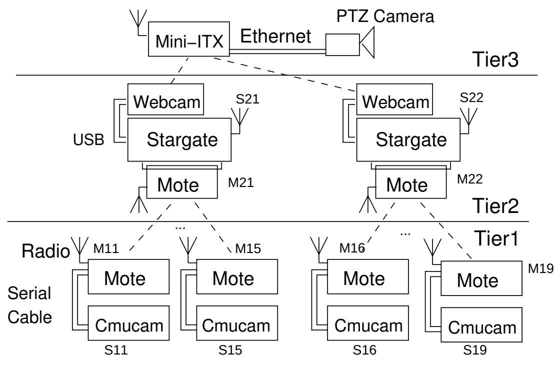


Figure 1.2. SensEye hardware architecture.

constrained, low power elements are employed to perform simpler application tasks, while more capable, high-power elements take on more complex tasks. Doing so results in more judicious use of precious energy resources. Only when necessary are the higher level sensors woken up to perform the more complex tasks that the lower level sensors cannot complete. Figure 1.2 shows an example of the SensEye hardware architecture. For each camera sensor there is a mote associated with it that has networking and processing abilities. The higher tier motes have considerably better networking and processing capacities than the lower ones.

A distributed sensor network such as SensEye is often deployed in object tracking applications. Since object detection is a relatively simple task to perform, it is carried out by the tier 1 sensors in SensEye. The higher tier sensors are normally in sleeping mode until woken up. Once an object is detected at Tier 1 by multiple sensors, a localization algorithm is performed to decide the location of the object and decide which Tier 2 sensor(s) to wake up in order to better track the target. If no Tier 2 sensor is in the appropriate location and point to the correct direction, a retargetable Tier 3 sensor is woken up to properly track the target. When the target is moving,

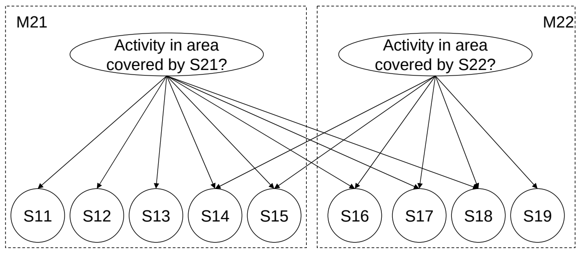

M21
M22
Activity in area covered by S21?
Activity in area covered by S22?
S11
S12
S13
S14
S15
S16
S17
S18
S19

Figure 1.3. The interpretation problem in SensEye modeled with a Bayesian Network.

the new data collected by the sensors can be used to predict the movement of the target and decide which sensors to wake up the next.

In this setup, when an object is detected by tier 1 sensors, the sensor interpretation problem is for the higher level motes to decide whether there is activity detected in the areas covered by their associated camera sensors. Based on this interpretation and other information such as availability and processing loads of the sensors, the agents can then decide which sensors to wake up. Figure 1.3 shows the interpretation problem for $M21$ and $M22$ in the SensEye setup illustrated in Figure 1.2. In this setup, $M21$ has direct communication channels with $M11$ through $M15$. When any of $S11$ through $S15$ detects something, the associated motes send the information to $M21$. Additionally, $M21$ and $M22$ can also communicate with each other when necessary, and are responsible for deciding whether there is activity in the area covered by $S21$ and $S22$, respectively.

Though distributed approaches to DSI problems can have many advantages, the existence of subproblem interactions means that the agents must communicate during the problem-solving process to obtain information that is needed to solve their subproblems optimally. A coordination strategy needs to be generated that specifies how agents should interact. In the SensEye example, even though $M21$ does not have direct access to $S16$ through $S18$, they do cover areas overlapping with $S21$. Therefore, in order to decide whether there is activity in the area covered by $S21$,

$M21$ may need the data provided by $S16$ through $S18$, and communication between $M21$ and $M22$ may become necessary.

As the number of sensors grows, the amount of communication among agents required to guarantee global optimality may be significant. Unfortunately, in many DSI applications communication cost is high. Communication cost can take on many different forms. It can simply be bandwidth allocation or power consumption needed to set up the channel and transfer the message. In a wireless network, one or both of these resources can be limited. In some application such as distributed scheduling, privacy arises as a key issue. Agents may be reluctant to share private information with others and therefore communication incurs a privacy violation cost. Sensor networks are also often deployed in an adversarial environment, where communication is undesirable in fear of possible eavesdroppers. In such applications, communication has a security cost.

"Satisficing" approaches have been developed that trade off optimality for reduced communication [12]. One approach is for agents to exchange local data related to each other's subproblem until a sufficient level of credibility has been achieved among the agents. An important characterization of such distributed protocols is how much communication is required and the likelihood that the solution will be the same as what would be generated by an optimal centralized algorithm which uses all available information. Most approaches to managing communication trade off solution quality for reduced communication, but only from a statistical view. The behavior of the algorithms are often analyzed over a collection of problems to say that p percent of the time they will get the required solution quality q with an average amount of communication c [12].

I would like to take this satisficing approach to the next step by exploring whether a parameterized communication management algorithm can be designed so that one can determine the expected amount of communication the agents need to guarantee

actions have different costs depending on the bandwidth and power consumption for the message, the possible violation of the privacy and the danger of broaching the security. In contrast, other research studying communication in DEC-MDP focuses on synchronizing communication where every communication action exchanges all the state information among agents and leads to the same view of the system. While the study of synchronizing communication tries to answer the question of **when** to communicate, my work is trying to address both the questions of **what** and **when** to communicate.

The size of the state space of the DEC-MDP I will build for the DSI problem is exponential to the amount of data in the DBN, and so is the action space. It is computationally infeasible to solve such a large DEC-MDP when the quantity of observed data grows. In addition to developing algorithms to solve DEC-MDP, other techniques are needed to generate a more compact DEC-MDP for such problems. One approach I will look at is to transfer abstract information. Instead of transmitting observed data directly, the agents can abstract the useful information contained in the observed data and communicate them. Not only do we need algorithms to generate appropriate abstract data to be transferred between the agents, we also need to develop techniques to incorporate such abstract data in order to simplify the DEC-MDP. Another benefit of transferring abstract information instead of the local observations themselves is that it can potentially reduce the expected minimum communication cost since the abstract data carries information more efficiently. However, how to appropriately incorporate the transmission of abstract data is an important question. By adding the abstract data, the solution quality of the DSI problem should not be compromised, nor should the average necessary communication cost be increased.

One of the major reasons why decision theoretic models are not widely used in multi-agent systems is the high complexity class. It has been proved that deciding the DEC-MDP is NEXP-complete [5]. One class of DEC-MDP has been studied in detail

With this model, one will be able to generate a communication strategy for a given DSI problem represented in a DBN such that only minimum communication cost is needed to achieve the required confidence level in the interpretation task. This is different from the previous work in this area. Previous work either focuses on finding the globally optimal solution without taking into consideration the potentially significant communication cost [79], or studies the tradeoff between solution quality and communication cost only from a statistical view [12]. In contrast, this work produces a parameterized algorithm to determine the expected amount of communication the agents need to guarantee a desired confidence level for any given DSI problem.

This work is not limited to the DSI domain, but can be extended to the Cooperative Distributed Problem Solving (CDPS) [44] in general if its global state does not change in the communication process. In a CDSP system, a problem is decomposed into sub-problems and they are distributed to a group of agents. Each agent often needs information from other agents to solve its local problem and communication is necessary to guarantee solution quality. Though the advances in networks have made it possible to transfer reasonable amount of data between agents without significant cost, there are applications where communication is extremely costly, such as planetary rovers [78], and battlefield teamwork [77]. In the first example, resource constraints such as battery power limit communication and in a battlefield situation communication can be dangerous. In these domains, appropriate communication strategies are inevitably needed to minimize the communication cost while still maintaining the required solution quality. If one can identify the information (observations) accessible to each agent, the goal that the system tries to achieve and how the information acquired is related to the progress towards the goal, then a DEC-MDP similar to the one I build for the DSI problem can be generated for the problem as well and be used to provide the required communication strategy. In the DSI example, the information accessible to each agent is the local sensor data collected and the goal of the system is the desired confidence level of the subproblems, which is used to decide whether a state of the DEC-MDP is a final state. The relationship between the two can be derived through the Bayesian Net inference.

1.1.2 Developed some of the first algorithms to find approximate solutions for general DEC-MDPs.

The interaction between the agents makes it hard to construct algorithms that are guaranteed to find the globally optimal solution for a DEC-MDP. I have designed two algorithms to approximate the globally optimal solution [75]. One is an iterative algorithm that is guaranteed to converge to a local optimal solution, but the quality of the policy it generates largely depends on the starting policy of the iterative process. The other approach is based on a lookup algorithm which is much less computationally expensive and can be easily extended to more than 2 agents. Though there is no guarantee that can be made about the solution either generates, experimental work indicates that in the problems studied both approaches lead to policies that are of good quality. This is some of the first work providing algorithms to approximate optimal solution of communication problems in a complex problem solving setting that are formulated in a decision-theoretic model.

1.1.3 Formalized the use of abstraction in communication minimization problems.

Though general communication can be naturally modeled with a DEC-MDP, the state space and action space grow exponentially with the data that needs to be transferred. Therefore techniques need to be developed to address this problem before the system can be scaled up. I propose to generate a set of abstract communication actions such that when such abstract information is transferred between the agents, the goal of the system is more likely reached. By allowing only abstract communication actions in certain states, the expected communication cost required is shown by experiments to be improved, and the time needed to solve the DEC-MDP is reduced on average [73].

In the original Distributed Problem Solving work [23], abstraction was used as a mechanism for controlling the information needing to be communicated. However, the use of abstraction was not formalized. Nor was there a clear understanding of when the lower level data needed to be transmitted. Carver and Lesser [11] studied the use of multiple levels of abstraction to reduce the necessary communication. Like [23], the abstraction layers were predefined and the use of abstraction was not formalized. In contrast, the algorithm I introduce allows the system to generate appropriate abstraction data automatically without predefinition. My study of the addition of transferring abstraction data as actions in the DEC-MDP provides us with a formal view of the use of abstraction in the management of communication cost in a distributed problem solving system.

1.1.4 Established the relationship between the interactions among agents and the complexity of a DEC-MDP.

While communication management remains a key research problem in the MAS community, previous research has demonstrated that interactions among a group of agents may significantly increase the complexity of a DEC-MDP. A DEC-MDP with independent transitions and observations [4] has minimum interactions between the agents and is NP-complete, while a general DEC-MDP with general communication [5, 69] is NEXP-complete. However, adding synchronizing communication to a DEC-MDP with independent transitions and observations does not increase its complexity class and it remains NP-complete [29]. The relationship between the complexity class of a DEC-MDP and the amount of interaction between agents remains an open question.

This work attempts to quantify the characteristics of a class of multi-agent coordination problems that determines its complexity. The key result is that the complexity of the problem depends on the amount of important information each agent has about the other agents, and whether this information can be represented in a succinct way. Information is important if knowing it could allow the agents to achieve a higher expected reward, and by succinct I mean that the set of all possible important information the agents could acquire is polynomial in the number of states. I prove that this criteria is both sufficient and necessary for the class of problems to be NP-complete [72].

One implication of this result is that it may start to change the way people view interactions between agents in the context of distributed POMDPs. Multi-agent researchers have long intuitively understood that the interaction between the agents is the cause of their high complexity. The theoretical results of this paper are significant in that they both formally justify this intuition as well as explain how the interaction affects the complexity. This new understanding of interaction and its relationship to complexity will help us to identify new classes of multi-agent systems with a lower complexity.

The significance of this theoretical result also has a more practical side. Most multi-agent systems are provably harder than NP and solving them optimally is very difficult. Much work has been put into developing good algorithms for approximating these problems. This work provides theoretical guidance in understanding how the approximations in a model limit the search space and reduce the complexity. I demonstrate this on two problems that do not meet the condition by providing two approximations that impose additional restrictions on the interactions among the agents and reduce the complexity to no harder than NP.

The rest of the thesis is organized as follows. In Chapter 2, I first define the communication management problem in Distributed Sensor Interpretation applications and model it with a DEC-MDP. I present two algorithms to approximate its optimal solution. Chapter 3 describes how abstraction data can be generated and transferred in the DEC-MDP as communication actions to further reduce the communication cost as well to improve the solution time of the DEC-MDP. Chapter 4 studies agent interaction in distributed POMDPs and its implications on the complexity class. It situates the type of DEC-MDP represented by the one built for the DSI problem in the context of previous decision theoretic work studying communication in co-operative distributed systems. The related work is discussed throughout the three chapters where applicable. Chapter 5 concludes the thesis with a summary of the main contributions and a discussion of possible future directions that this research can be taken.

CHAPTER 2

MINIMIZING COMMUNICATION COST USING A DECENTRALIZED MDP

Advances in sensor technology are leading to the deployment of large networks of sophisticated sensors [43]. Interest in getting robots and other computer systems to interact more autonomously with their environments is also driving the introduction of sensors into a wider array of environments (involving distributed characteristics). Typically, such large-scale sensor systems are inherently distributed because the sensors are geographically distributed. As a result, Distributed Sensor Interpretation (DSI) problem is of particular interest and related research becomes increasingly important. Sensor interpretation involves the fusion of data from one or more sensors, over time, to determine the situation in the environment. An interpretation of a sensor data set is typically expressed as a set of events that could have occurred in the environment and would explain the sensor data. Events may be things like aircraft or other vehicle moving through the sensed region, the detection of walls or other obstacles as a robot moves, or so forth. Typically, there will be multiple possible interpretations for any reasonably sized set of data. The algorithm used by an sensor interpretation system must not only identify possible interpretations, it must also select the best of the possibilities to report as the solution.

In a DSI system, data is collected from different sensors and need to be integrated to produce the best interpretation. This can be done either in a centralized fashion or distributedly. In centralized approaches to such problems, the data from all of the sensors is transmitted to a single agent that does all of the computation to interpret the data [74]. In distributed approaches, the interpretation task itself is decomposed into subproblems and distributed to different agents. Each agent is responsible for collecting the related data from local sensors and or other agents in order to solve its local subproblem. Though distributed approaches to DSI problems can have many advantages, the existence of subproblem interactions means that the agents must communicate during the problem-solving process at least to insure that their local solutions are globally consistent and perhaps also to obtain data or other information that is needed to solve their subproblems optimally. A coordination strategy needs to be generated that specifies how agents will interact: when and with whom they will communicate, and what information they will send or request.

Unfortunately, as the number of sensors grows, the amount of communication among agents required to guarantee global optimality or global consistency may be very significant while in a lot of DSI applications communication cost is high or sufficient communication bandwidth and reliable channel is not available. Thus, "satisficing" approaches have been developed that trade off optimality for reduced communication [12]. One approach is for agents to generate local solutions based on their own data and then transmit these high level solutions to other agents. Based on consistency and credibility of these local solutions, new local solutions may be generated or more detailed data sent until a sufficient level of consistency and credibility has been achieved among the agents. An important characterization of such distributed protocols is how much communication is required and the likelihood that the solution will be the same as what would be generated by an optimal centralized algorithm which uses all available information.

Most approaches to managing communication trade off solution quality for reduced communication, but only from a statistical view. The behavior of the algorithms are often analyzed over a collection of problems to say that p percent of the time they will get the required solution quality q with an average amount of communication c [12].

changes the other agent's view. The local MDPs of the two agents are largely dependent on each other. This makes it hard to construct algorithms that are guaranteed to find the globally optimal solution. I have designed two algorithms to approximate the globally optimal solution for the DEC-MDP. One is an iterative algorithm that is guaranteed to converge to a local optimal solution, but the quality of the policy it generates largely depends on the starting policy of the iterative process. The other approach is based on a lookup algorithm which is much less computationally expensive and can be easily extended to more than 2 agents. Though there is no guarantee that can be made about the solution either generates, experimental work described in Section 2.3.4 indicates that in the problems studied both approaches lead to policies that are of good quality. To our knowledge, this is some of the first work providing algorithms to approximate optimal solution of communication problems in a complex problem solving setting that are formulated in a decision-theoretic model.

2.1 Problem Definition

Probabilistic reasoning with graphical models, also known as Bayesian Networks (BNs) has become an active field of research and practice in artificial intelligence, operations research, and statistics in the past two decades. A BN is a directed acyclic graph whose arrows represent causal influences or class-property relationships [65]. Because it is a compact graphic representation of independence relationship between nodes, it is widely used as knowledge representation scheme in inference systems. A lot of work has been done to study its properties and different algorithms have been developed to propagate evidence and answer queries efficiently [65, 37]. The success of these techniques in modeling intelligence decision support systems in centralized and single-agent applications has been striking. The natural causal relationship between the events and sensor data makes it appropriate to represent the DSI problem structures in a Bayesian Network [12].

Recent work on Multiply Sectioned Bayesian Networks (MSBN) extends the traditional BN to the distributed and multi-agent case [80]. This framework allows distributed representation of uncertain knowledge on a large and complex environment in multi-agent systems and effective, exact and distributed probabilistic inference. Xiang [79] introduced an algorithm that can produce the same final solution for a distributed interpretation problem modeled by a MSBN as is generated by a centralized problem solving system. However, this approach can potentially require significant communication.

In this work, the interpretation problem is represented in a two-layer Bayesian Network (Figure 2.1). The top level nodes are the events that are the possible causes of the observed data, while the leaves are the data gathered by various agents. It is worth noting that the data nodes in this work are not the raw sensor output, but rather the results of some initial processing to produce categorical/discrete values such that they can be processed by the probabilistic model. There are two agents, each of whom has direct access to only a part of the observable data. The interpretation task is distributed to the two agents as well. Each agent is only responsible for its part of the overall problem and only has the knowledge of the part of the network that is relevant to its task. The agents can either request or send data. The objective is to figure out the most likely interpretation of the causal events with a certain level of confidence using as little communication as possible. For example, in Figure 2.1, there are two agents A_1 and A_2. A_1 is responsible for interpreting the event E_1 and therefore knows about the part of the causal network that is relevant to E_1. Out of the necessary data $D_1, ...D_8$, it can directly observe only D_1 through D_5.

Definition 1. *Based on the nature of different parts of the relevant data to an interpretation task, I divide them into three categories.* **Local data** *are the data that can be directly observed by the agent and are relevant only to its local task. They do not need to be transmitted to the remote agent at any time.* **Local common data** *are the*

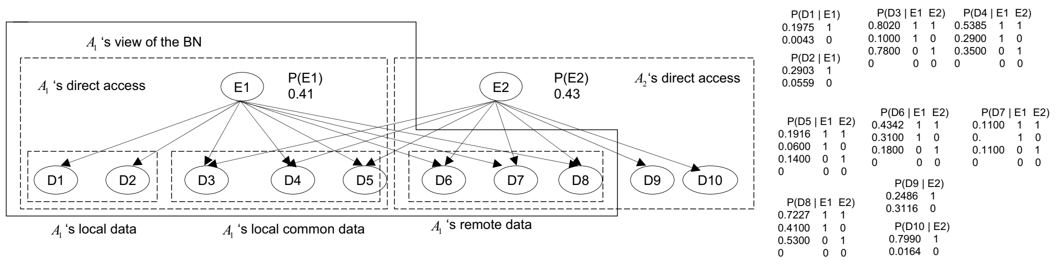


Figure 2.1. There are two events E_1 and E_2. Data $D_1, D_2, ...D_{10}$ are distributed to two agents. A_1 has access to $D_1, ...D_5$ and is responsible for solving E_1, while A_2 can only see $D_6, ...D_{10}$ and is responsible for solving E_2. The objective is for A_1 and A_2 to figure out what E_1 and E_2 are with required confidence and with minimum expected communication cost

rest of the data that are observable by the local agent. They not only are important for the local task but also would help remote agents in their interpretation tasks. They are the candidates to be sent in the local agent's decision process. **Remote data** *are the ones that cannot be directly observed by the local agent, but knowing them would be helpful in increasing the confidence of its local solution. When an agent is considering requesting data, remote data are the natural candidates. In Figure 2.1, A_1's local data is D_1 and D_2. Its local common data is $D_3, \ldots, D_5$, and its remote data is $D_6, \ldots, D_8$.*

Definition 2. *The* **evidence** ϵ_{A_i} *of an agent A_i are the values of the data that the agent has collected so far. They can be the values that are observed directly by that agent or acquired from the remote agent. The* **complete evidence** $\epsilon^*_{A_i}$ *of A_i are the values of all the relevant data of the agent. ϵ^* are the values of all the low level data in the system. At any time, $\epsilon_{A_i} \subseteq \epsilon^*_{A_i} \subseteq \epsilon^*$. In Figure 2.1, A_1 initially can only observe the data values of $D_1, \ldots, D_5$, and ϵ_{A_1} can be one of the 32 possible configurations. As communication goes on, an agent will gather more evidence from the remote agents, and the set ϵ_{A_i} will grow. The relevant data of A_1's interpretation task is $D_1, \ldots, D_8$. Therefore, $\epsilon^*_{A_1}$ is one of the 64 possible configurations.*

Definition 3. Likelihood L_{A_i}. *Based on the evidence observed, an agent can calculate the conditional probabilities of every possible interpretation of its local events based on the current evidence, i.e.,* $L_{A_i}(E_{A_i}) = P(E_{A_i}|\epsilon_{A_i})$. *e.g. in Figure 2.1, the local event of* A_1 *is* E_1. *Therefore,* $L_{A_1}(E_1) = P(E_1|\epsilon_{A_1})$. *Specifically, before any communication occurs,* $L_{A_1}(E_1) = P(E_1|D_1, \ldots, D_5)$.

Definition 4. MAPI (Maximum A Posteriori Interpretation). *Based on the current evidence set available to an agent, there is a most likely interpretation of the events.* $MAPI(\epsilon_{A_i}) = argmax_h P(E_{A_i} = h|\epsilon_{A_i})$.

The task of an agent A_i in the system is to find the most likely interpretation of the local events given its current evidence set, i.e., $MAPI(\epsilon_{A_i})$. Ideally the decentralized system should generate the interpretation that a centralized system will generate given the complete data values, i.e., $MAPI(\epsilon^*_{A_i})$. Unfortunately, with only partial knowledge of the relevant data values, an agent cannot always guarantee that the current local MAPI is the global MAPI. On the other hand, with the conditional probability table given by the BN, an agent can predict the probability of different relevant data configurations given its current evidence, i.e., $P(\epsilon^*_{A_i}|\epsilon_{A_i})$. Hence we have the following definition.

Definition 5. Confidence $C(MAPI(\epsilon_A))$ *is the likelihood of the local MAPI being the global MAPI of* E_A *given the current known evidence of* A.

$$\begin{aligned} C(MAPI(\epsilon_A)) &= P(MAPI(\epsilon_A) = MAPI(\epsilon^*_A)) \\ &= \sum_{e \in \{\epsilon^*_A | MAPI(\epsilon_A) = MAPI(\epsilon^*_A)\}} P(e|\epsilon_A) \end{aligned}$$

Confidence is not a novel idea, but the way I define it is different from previous work. [12] defined it simply as the probability of an event being the local MAPI, i.e.,

$P(E_A = MAPI(\epsilon_A)|\epsilon_A)$. What we really care about is not the likelihood of an event being the local MAPI, but whether the local MAPI agrees with the global MAPI, i.e., whether the decentralized system will generate the same result as a centralized system would. That is exactly what our definition of confidence achieves. Using confidence as a measurement of the solution quality of a decentralized system as compared to that of a centralized one, we can make a tradeoff between the quality of the solution and the communication cost. Another interesting and useful property of our definition of confidence is that it is guaranteed to reach 100% when the values of all the relevant data are known. As a result, a given confidence level can always be satisfied. One major drawback of the traditional definition of confidence is its possible introduction of selection bias [34]. The selection bias is introduced when an agent only acquires the part of evidence that raises the probability of an event, which is not the global MAPI, being the local MAPI sufficiently high. The new confidence measure addresses this problem by measuring the likelihood that the MAPI of the known evidence set is correct, i.e. - it matches the MAPI of the actual complete evidence set. The new confidence can be viewed as a measurement of the probability of not inducing this selection bias given the known evidence set. A higher confidence implies a lower likelihood of inducing the selection bias. Though this definition does not fully exclude the introduction of the selection bias, it does provide a measure of it. If the introduction of selection bias is highly undesirable in an interpretation problem, then the confidence threshold needs to be set sufficiently high to prevent it.

Given a problem structure in a two level BN such as the one in Figure 2.1 and a specified confidence threshold, we need to find a communication strategy. Such a communication strategy should specify what communication action each agent should take based on their current knowledge at each stage. In this work, I am only considering the case of synchronous communication. In other words, the two agents take

2.2 Decentralized MDP Model

The system I described in the last section is both decentralized and cooperative at the same time. The communication actions of each agent are entirely dependent on its current local evidence and the communication history. Based on these characteristics, I model this problem as a decentralized MDP.

Unlike centralized MDPs, in a decentralized MDP, the process is controlled by multiple agents, each with possibly different information about the global state. At each time step the agents' observations together will uniquely determine the global state, though possibly none of them have the complete information of the global state. A local policy for an agent is a mapping from its local histories of observations to an action. The task of a decentralized MDP is normally to find an optimal joint policy of all the agents that maximizes the expected total return.

First let me give a formal definition of decentralized MDP, adapted from [5]. I only define the two agent case in this section, which can be easily extended to n agent case.

Definition 7. *A* **decentralized MDP** *is a tuple* $\langle S, Ac_1, Ac_2, P, R, \Omega_1, \Omega_2, O\rangle$, *where*

- S *is a finite set of global states, with distinguished initial state* s_0.
- $Ac = Ac_1 \times Ac_2$ *is a finite set of joint actions.* Ac_1 *and* Ac_2 *are the local actions for the two agents.*
- $P : S \times A \times S \rightarrow \Re$ *is the transition function.* $P(s'|s, a_1, a_2)$ *is the probability of the outcome state* s' *when the joint action* (a_1, a_2) *is taken in state* s.
- $R : S \times A \times S \rightarrow \Re$ *is the reward function.* $R(s, a_1, a_2, s')$ *is the reward obtained from taking joint action* (a_1, a_2) *in state* s *and transitioning to state* s'.
- $\Omega = \Omega_1 \times \Omega_2$ *is a finite set of joint observations.* Ω_i *is the set of observations for agent* i.

- *$O : S \times Ac \times S \times \Omega \rightarrow \Re$ is the observation function. $O(s, a_1, a_2, s', o_1, o_2)$ is the probability of agents 1 and 2 seeing observations o_1 and o_2 after the sequence s, (a_1, a_2), s' occurs.*

- *Joint full observability: the pair of observations made by the agents together fully determine the current state. If $O(s, a_1, a_2, s', o_1, o_2) > 0$ then $P(s'|o_1, o_2) = 1$.*

Definition 8. *A* **local policy** *$\pi_i : \overline{\Omega_i} \rightarrow Ac_i$ for agent A_i is a mapping from local histories of observations $\overline{o}_i$ to actions in Ac_i. A* **joint policy** *$\pi = \langle \pi_1, \pi_2 \rangle$ is a pair of local policies, one for each agent.*

Definition 9. *The value $V_\pi(s)$ of a state s following policy $\pi = \langle \pi_1, \pi_2 \rangle$ is:*

$$V_\pi(s) = \sum_{\langle \overline{o}_1, \overline{o}_2 \rangle} \sum_{q \in S} \sum_{s' \in S} P_\pi(\overline{o}_1, \overline{o}_2, q|s) \cdot P(s'|q, \pi_1(\overline{o}_1), \pi_2(\overline{o}_2)) \cdot R(q, \pi_1(\overline{o}_1), \pi_2(\overline{o}_2), s'),$$

where $P_\pi(\overline{o}_1, \overline{o}_2, q|s)$ is the probability of observing $\overline{o}_1, \overline{o}_2$ and reach state q from state s following policy π. The value of the initial state $V_\pi(s_0)$ is the expected total reward following the policy π from state s_0. To solve a decentralized MDP is to find a joint policy $\langle \pi_1, \pi_2 \rangle$ to maximize $V_\pi(s_0)$.

Definition 10. *An* **underlying centralized MDP** *of a decentralized MDP $\langle S, Ac_1, Ac_2, P, R, \Omega_1, \Omega_2, O \rangle$ is a tuple $\langle S, Ac_1, Ac_2, P, R \rangle$, where S, Ac_i, P and R are the same as its corresponding part in the DEC-MDP. A* **centralized policy** *$\pi^c = \langle \pi_1^c, \pi_2^c \rangle$ for this MDP is a mapping from the global state $s \in S$ to a pair of the local actions $\langle a_1, a_2 \rangle \in Ac_1 \times Ac_2$. The value of the state is denoted as $V^c(s)$. Solving an underlying centralized MDP is to find a centralized policy that maximizes the value of the start state. We denote the value of a state for the optimal policy as $V^{c*}(s)$.*

The key difference between a decentralized MDP and its underlying centralized MDP are their policies. The joint policy of a DEC-MDP is a pair of local policies each of which is dependent on the local observation sequence of the corresponding agent, while the centralized policy for the centralized MDP is itself dependent only on the global state, and is not directly related to the local policies. When the two agents have access to the global state at any time, a DEC-MDP is reduced to its underlying centralized MDP. In this sense, an underlying centralized MDP is equivalent to a MMDP.

My work is based on the DEC-MDP because it enables us to explicitly model the partial observability of a multi-agent system and does not impose any specific requirement on communication among the agents. The natural way to model the DSI problem described in the last section is as follows:

- Every global state is in the form of $s = \langle \epsilon^*, i \rangle$, where ϵ^* is all of the low level data values observed by the system, and i is an external feature that indicates it is A_i's turn to communicate. s_0 is a dummy start state $\langle \emptyset, 0 \rangle$. When all the sensors collect their data, the start state transitions to one of the possible real global states $\langle \epsilon^*, 1 \rangle$ before any communication takes place.

- Ac_1 and Ac_2 are action sets of the two agents. $Ac_i \in$ { NULL, SEND x, REQUEST y}, where x is a subset of the local common data of A_i and y is a subset of the remote data. When it is not A_i's turn to communicate, its action is the NULL action. A joint action $\langle Ac_1 \neq NULL, NULL \rangle$ transitions a state $\langle \epsilon^*, 1 \rangle$ to $\langle \epsilon^*, 2 \rangle$, and a joint action $\langle NULL, Ac_2 \neq NULL \rangle$ transitions a state $\langle \epsilon^*, 2 \rangle$ to $\langle \epsilon^*, 1 \rangle$. If at state s, both agents have reached confidence level t for its local interpretation task, i.e., when $C(MAPI(\epsilon_{A_i})) \geq t$, $i = 1, 2$, then s is a final state.

- P is a transition probability table. $P(s'|s, a_1, a_2) = P(\langle \epsilon^*, i \rangle | \langle \epsilon^*, j \rangle, a_1, a_2) = 1$ iff $i \neq j$. Otherwise, $P(s'|s, a_1, a_2) = 0$. As a special case, the transition

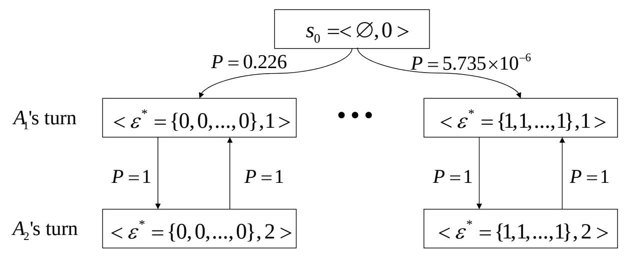

$s_0 = <\varnothing, 0>$
$P = 0.226$
$P = 5.735\times10^{-6}$
A_1's turn
$<\varepsilon^* = \{0,0,...,0\},1>$
$<\varepsilon^* = \{1,1,...,1\},1>$
$P=1$
$P=1$
$P=1$
$P=1$
A_2's turn
$<\varepsilon^* = \{0,0,...,0\},2>$
$<\varepsilon^* = \{1,1,...,1\},2>$

Figure 2.2. The decentralized MDP generated for the problem in Figure 2.1.

probability from s_0 to $s = \langle \epsilon^*, 1\rangle$ after both agents execute NULL action is $P(s|s_0, NULL, NULL) = P(\epsilon^*)$. It can be calculated given the BN structure.

- R is a reward function. $R(s, a_1, a_2, s') = R(a_1, a_2) = -c(a_1) - c(a_2)$, where $c(a_i)$ is the cost of the communication action a_i, $i = 1, 2$.

- Ω_1 and Ω_2 are the sets of observations for the two agents. An observation $o_i \in \Omega_i$ is the data value just sent or received by A_i. As a special case, after s_0, each agent observes the values of its local data. An observation sequence $\overline{o}_i = \langle \epsilon^0_{A_i}, H\rangle$, where $\epsilon^0_{A_i}$ is the initial observation of A_i's local data set before any communication occurred in the system, and H is the communication history of the system.

- $O(s, a_1, a_2, s', o_1, o_2) = 1$ iff o_1 and o_2 are the new data sent or received by the agents after them taking actions a_1 and a_2 at state s. Otherwise it equals to 0.

For example, take the BN structure in Figure 2.1. A possible global state s might be $\langle\{D_1 = 1, D_2 = 0, D_3 = 0, D_4 = 1, D_5 = 1, D_6 = 0, D_7 = 1, D_8 = 1, D_9 = 0, D_{10} = 0\}, 1\rangle$. The possible actions for A_1 are NULL, SEND x, and REQUEST y, where $x \subseteq \{D_3 = 0, D_4 = 1, D_5 = 1\}$ and $y \subseteq \{D_6, D_7, D_8\}$. Since it is not A_2's turn to communicate, its only action is NULL. If A_1 chooses REQUEST $\{D_7, D_8\}$, then the next state is $\langle\{D_1 = 1, D_2 = 0, D_3 = 0, D_4 = 1, D_5 = 1, D_6 = 0, D_7 = 1, D_8 =$

$1, D_9 = 0, D_{10} = 0\}, 2\rangle$ and both agents observes $\{D_7 = 1, D_8 = 1\}$. Remembering the entire communication history H for both agents have two purposes. First, it includes the newly acquired evidence received from the other agent. Second, even for the agent who just SEND part of its own data without receiving any new evidence, it is important to remember this so that in the future it does not send the same information unnecessarily.

Figure 2.2 is the DEC-MDP built for the problem in Figure 2.1. It is interesting to note that the structure of the DEC-MDP is stochastic only in the first transition, from s_0. Once the sensors have collected their data, the remaining transitions of the DEC-MDP are all deterministic. While here I use the DSI problem as an example to build the DEC-MDP with only communication actions, other such DEC-MDPs have the same structure. The only difference is the actual details of the state information, observations and the probability tables. We will explore this point in detail later in the chapter.

2.3 Solving the DEC-MDP

In this section I present two approximate algorithms to solve the DEC-MDP built in the last section. Both of them can be used to solve a general DEC-MDP without modification.

Though the DEC-MDP is a very hard problem, recently there has been various work developing different algorithms to solve it. Hansen et al. [32] introduced an exact dynamic programming algorithm, which uses pruning to eliminate dominated partial policies to improve efficiency. However, it suffers from extensive memory use and its usage is limited to small problems. Becker et al. [4] proposed an exact algorithm called Coverage Set Algorithm to solve the DEC-MDP with independent transitions and observations. Since the DEC-MDP has a complexity of NEXP-complete, more work is focused on developing approximate algorithms. Peshkin et al. [66] studied gradient descent based algorithms that converge on local optimal solutions. Nair et al. [57] presented JESP, a class of algorithms that iteratively fix one agent's policy and maximize the value for the other agent. The iterative algorithm I introduce in this work is very similar to JESP, but they were developed around the same time independently. Ghavamzadeh and Mahadevan [27] applied reinforcement learning techniques to solve the problem in their Hierarchical Reinforcement Learning algorithm. Bernstein et al. [6] generalized the Bounded Policy Iteration algorithm for POMDPs [67] to the multi-agent case. Policies are represented as joint stochastic finite-state controllers, which consist of a local controller for each agent and a correlation device that allows the agents to correlate their behavior without exchanging information during execution. The algorithm uses a fixed amount of memory, and each iteration is guaranteed to produce a controller at least as good as the previous one for all possible initial state distributions. For DEC-MDP with synchronizing communication, algorithms based on myopic assumptions are developed [29, 3].

2.3.1 Local MDP

In a DEC-MDP, a local policy for an agent is entirely dependent on its observation sequence so far, since the agent does not know exactly which global state it is in. In this sense, when looked from a local agent's perspective, a DEC-MDP is similar to a POMDP. Unfortunately, in most cases, an agent's observation function is dependent on the other agent's policy and therefore a DEC-MDP cannot be easily decoupled into two local POMDPs to be solved separately. Any problem with communication actions, whether they are implicit or explicit, falls into this category, because communication actions have direct impact on the observation sequences of both agents. Without assuming the other agent's policy, it is impossible to build the transitional probability table for the local MDP. However, if the remote agent's policy is fixed, one is able to build a local POMDP and its belief MDP for each agent. By solving

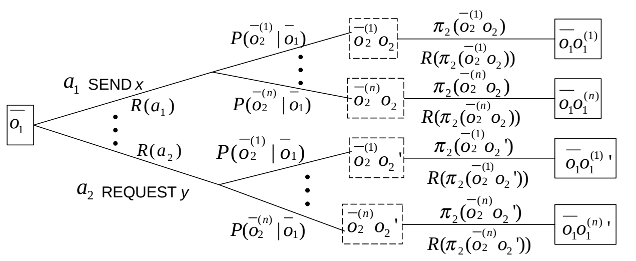

$\overline{o}_1$
a_1 SEND x
$R(a_1)$
$R(a_2)$
a_2 REQUEST y
$P(\overline{o}_2^{(1)} \mid \overline{o}_1)$
$P(\overline{o}_2^{(n)} \mid \overline{o}_1)$
$P(\overline{o}_2^{(1)} \mid \overline{o}_1)$
$P(\overline{o}_2^{(n)} \mid \overline{o}_1)$
$\overline{o}_2^{(1)} o_2$
$\overline{o}_2^{(n)} o_2$
$\overline{o}_2^{(1)} o_2'$
$\overline{o}_2^{(n)} o_2'$
$\pi_2(\overline{o}_2^{(1)} o_2)$
$R(\pi_2(\overline{o}_2^{(1)} o_2))$
$\pi_2(\overline{o}_2^{(n)} o_2)$
$R(\pi_2(\overline{o}_2^{(n)} o_2))$
$\pi_2(\overline{o}_2^{(1)} o_2')$
$R(\pi_2(\overline{o}_2^{(1)} o_2'))$
$\pi_2(\overline{o}_2^{(n)} o_2')$
$R(\pi_2(\overline{o}_2^{(n)} o_2'))$
$\overline{o}_1 o_1^{(1)}$
$\overline{o}_1 o_1^{(n)}$
$\overline{o}_1 o_1^{(1)}{}'$
$\overline{o}_1 o_1^{(n)}{}'$

Figure 2.3. A snapshot of the local belief MDP of A_1 with the internal states.

such a local (belief) MDP, we can find the optimal local policy given a specific remote local policy.

The interesting element of such a local MDP is its transition probability table. To make it more understandable I generate a group of internal states for each local state. Each internal state represents the possible observation sequences of the remote agent after the agent executes a certain action at the current local state. Since the remote policy is fixed for each local MDP, for each pair of current local state and internal state there is one and only one next local state. For example, Figure 2.3 shows a snapshot of A_1's local MDP. Here, before A_1 chooses its next action, its observation is $\overline{o}_1$ and A_2 may have observation sequence of either $\overline{o}_2$ or $\overline{o}_2'$ which are dependent on A_1's current local evidence ϵ_{A_1} and the past communication history of the system H, which is exactly $\overline{o}_1$. Hence, the probabilities of A_2 observing $\overline{o}_2$ and $\overline{o}_2'$ are $P(\overline{o}_2|\overline{o}_1)$ and $P(\overline{o}_2'|\overline{o}_1)$ respectively. If A_1 sends data to A_2, no matter what A_2's current observation sequence is, it will have the same new observation o_2, i.e., A_1's action and the new data from A_1. This updates A_2's observation history to $\overline{o}_2 o_2$ and $\overline{o}_2' o_2$, which are the two internal states for the local state $\overline{o}_1$ after action a_1. Since A_2 has a fixed policy π_2, A_2's action when observing $\overline{o}_2 o_2$ and $\overline{o}_2' o_2$ are determined, namely $\pi(\overline{o}_2 o_2)$ and $\pi(\overline{o}_2' o_2)$, which will change A_1's new observation sequence to be $\overline{o}_1 o_1^{(1)}$ and $\overline{o}_1 o_1^{(2)}$. Therefore, the next states of $\overline{o}_1$ after the action are $\overline{o}_1 o_1^{(1)}$ and $\overline{o}_1 o_1^{(2)}$ with the transitional probability of $P(\overline{o}_2|\overline{o}_1)$ and $P(\overline{o}_2'|\overline{o}_1)$ respectively. Similarly, we

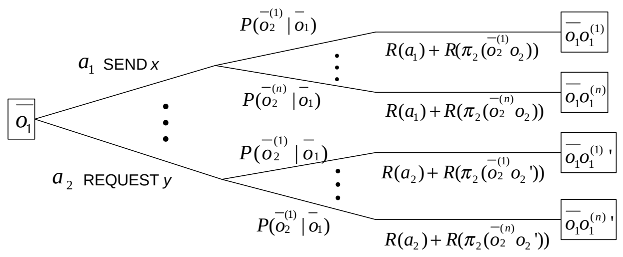

$\overline{o}_1$
a_1 SEND x
a_2 REQUEST y
$P(\overline{o}_2^{(1)} \mid \overline{o}_1)$
$P(\overline{o}_2^{(n)} \mid \overline{o}_1)$
$P(\overline{o}_2^{(1)} \mid \overline{o}_1)$
$P(\overline{o}_2^{(1)} \mid \overline{o}_1)$
$R(a_1) + R(\pi_2(\overline{o}_2^{(1)} o_2))$
$R(a_1) + R(\pi_2(\overline{o}_2^{(n)} o_2))$
$R(a_2) + R(\pi_2(\overline{o}_2^{(1)} o_2'))$
$R(a_2) + R(\pi_2(\overline{o}_2^{(n)} o_2'))$
$\overline{o}_1 o_1^{(1)}$
$\overline{o}_1 o_1^{(n)}$
$\overline{o}_1 o_1^{(1)}{}'$
$\overline{o}_1 o_1^{(n)}{}'$

Figure 2.4. A snapshot of the local MDP of A_1 without the internal states

get the next states from $\overline{o}_1$ by executing REQUEST actions. Figure 2.4 shows the corresponding snapshot of the actual local MDP of A_1 without the internal states.

Since the DSI system is a cooperative one, the goal of the agents is to maximize the global utility instead of the local one. This means that in a local MDP, when calculating the reward received from an action, we not only need to include the reward from the local action alone but also that gained by the remote action. For example, in Figures 2.3 and 2.4, the reward for executing action a_1 at state $\overline{o}_1$ and reaching state $\overline{o}_1 o_1^{(1)}$ is $R(\overline{o}_1, a_1, \overline{o}_1 o_1^{(1)}) = R(a_1) + R(\pi_2(\overline{o}_2 o_2))$.

The local MDP of an agent A_i when fixing the other agent's policy π_j can be summarized as follows:

- S is a finite local state set. Each state is the observation sequence of the local agent $\overline{o}_i = \langle \epsilon_{A_i}, H \rangle$. To resolve the uncertainty of the initial observation, a dummy start state o_i^0 is added. When the confidence level is reached at a state, it is a final state.

- Ac is the action set. For A_i, it is Ac_i in the DEC-MDP.

- P is the transition probability table, the calculation of which has been illustrated.

- $R(s, a, s')$ is the reward function. It is a sum of the reward gained directly by A_i executing action a and that gained by the remote agent's action as a result of a.

- $V_{\langle \pi_i, \pi_j \rangle}(\overline{o}_i)$ is the value of the state $\overline{o}_i$ for a local policy π_i of the local MDP. To solve the local MDP is to find the policy π_i that maximizes the state value of the start state o_i^0.

The CPT of the BN is used to calculate the transitional probability tables and decide whether a local state is a final state of the local MDP. By constructing such an MDP, we are utilizing the information provided by the BN to model the local agent's belief in the current global state based on its local state. Its belief in the observation sequence of the remote agent directly influence the transitional probability of the states. Therefore, by finding an optimal policy for the local MDP, the agent is implicitly using the knowledge obtained from both the current evidence set and the communication history of the system.

In this DEC-MDP, the two agents interact through actions, which directly affect each other's local observations. The local MDPs of the two agents are tightly coupled. The rewards, transitional probability table of the MDPs are both dependent on each other. This means that it is very hard to solve such a DEC-MDP without exhaustive search. In the next two subsections I will present two approximate algorithms which utilize the property of the local MDPs as expressed by Theorem 1.

Theorem 1. *Maximizing the utility of the local MDP of A_i maximizes the utility of the decentralized MDP given π_j is fixed.*

Proof. The value function (9) of a DEC-MDP defined in Definition 9 can be simplified due to the alternating action behavior of the decentralized MDP.

$$V_{\langle \pi_1, \pi_2 \rangle}(s_0)$$

$$
\begin{aligned}
&= \sum_{\langle \bar{o}_1,\bar{o}_2\rangle} \sum_{q\in S} \sum_{S'\in S} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1,\bar{o}_2,q|s) P(s'|q,\pi_1(\bar{o}_1),\pi_2(\bar{o}_2)) R(q,\pi_1(\bar{o}_1),\pi_2(\bar{o}_2),s') \\
&= \sum_{\langle \bar{o}_1,\bar{o}_2\rangle} \sum_{q\in S_1} \sum_{S'\in S_2} P_{\pi_1,\pi_2\rangle}(\bar{o}_1,\bar{o}_2,q) P(s'|q,\pi_1(\bar{o}_1)) R(\pi_1(\bar{o}_1)) \\
&\quad + \sum_{\langle \bar{o}_1,\bar{o}_2\rangle} \sum_{q\in S_2} \sum_{S'\in S_1} P_{\pi_1,\pi_2\rangle}(\bar{o}_1,\bar{o}_2,q) P(s'|q,\pi_2(\bar{o}_2)) R(\pi_2(\bar{o}_2)) \\
&= \sum_{\bar{o}_1} R(\pi_1(\bar{o}_1)) \sum_{\bar{o}_2} \sum_{q\in S_1} P_{\langle \pi_1,\pi_2\rangle}(q,\bar{o}_1,\bar{o}_2|s_0) \sum_{s'\in S_2} P(s'|q,\pi_1(\bar{o}_1)) \\
&\quad + \sum_{\bar{o}_2} R(\pi_1(\bar{o}_2)) \sum_{\bar{o}_1} \sum_{q\in S_2} P_{\langle \pi_1,\pi_2\rangle}(q,\bar{o}_1,\bar{o}_2|s_0) \sum_{s'\in S_1} P(s'|q,\pi_2(\bar{o}_2)) \\
&= \sum_{\bar{o}_1} R(\pi_1(\bar{o}_1)) P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) + \sum_{\bar{o}_2} R(\pi_2(\bar{o}_2)) P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_2)
\end{aligned}
$$

Next, let us see what the global utility that the local MDP is trying to maximize is. Here I am taking advantage of the independence relationships due to the MDP setup.

$$
\begin{aligned}
& V_{\langle \pi_1,\pi_2\rangle}(o_1^0) \\
&= \sum_{\bar{o}_1} \sum_{o_1} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1|o_1^0) P(o_1|\bar{o}_1,\pi_1(\bar{o}_1)) R(\bar{o}_1,\pi_1(\bar{o}_1),\bar{o}_1 o_1) \\
&= \sum_{\bar{o}_1} \sum_{o_1} \sum_{\bar{o}_2} \sum_{o_2} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) P(\bar{o}_2|\bar{o}_1) P(o_2|\bar{o}_1,\pi_1(\bar{o}_1)) \\
&\quad P(o_1|\bar{o}_2 o_2,\pi_1(\bar{o}_1),\pi_2(\bar{o}_2 o_2))(R(\pi(\bar{o}_1)) + R(\pi_2(\bar{o}_2 o_2))) \\
&= \sum_{\bar{o}_1} \sum_{\bar{o}_2} \sum_{o_2} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) P(\bar{o}_2|\bar{o}_1) P(o_2|\bar{o}_1,\pi_1(\bar{o}_1)) R(\pi(\bar{o}_1)) \\
&\quad + \sum_{\bar{o}_1} \sum_{\bar{o}_2} \sum_{o_2} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) P(\bar{o}_2|\bar{o}_1) P(o_2|\bar{o}_1,\pi_1(\bar{o}_1)) R(\pi_2(\bar{o}_2 o_2)) \\
&= \sum_{\bar{o}_1} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) R(\pi_1(\bar{o}_1)) \\
&\quad + \sum_{\bar{o}_1} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) \sum_{\bar{o}_2} \sum_{o_2} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_2|\bar{o}_1,\pi_1(\bar{o}_1)) P(o_2|\bar{o}_1,\pi_1(\bar{o}_1)) R(\pi_2(\bar{o}_2 o_2)) \\
&= \sum_{\bar{o}_1} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) R(\pi_1(\bar{o}_1)) \\
&\quad + \sum_{\bar{o}_1} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) \sum_{\bar{o}_2} \sum_{o_2} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_2 o_2|\bar{o}_1,\pi_1(\bar{o}_1)) R(\pi_2(\bar{o}_2 o_2)) \\
&= \sum_{\bar{o}_1} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) R(\pi_1(\bar{o}_1)) + \sum_{\bar{o}_2} R(\pi_2(\bar{o}_2)) \sum_{\bar{o}_1} P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_2|\bar{o}_1) \\
&= \sum_{\bar{o}_1} R(\pi_1(\bar{o}_1)) P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_1) + \sum_{\bar{o}_2} R(\pi_2(\bar{o}_2)) P_{\langle \pi_1,\pi_2\rangle}(\bar{o}_2)
\end{aligned}
$$

Hence, we get

$$V_{\langle \pi_1,\pi_2\rangle}(o_1^0) = V_{\langle \pi_1,\pi_2\rangle}(s_0) \tag{2.1}$$

The same equation holds for o_2^0. □

2.3.2 Iterative Algorithm

The first approximate algorithm I present is called Iterative Algorithm (Algorithm 1). The basic idea behind it is to fix one agent's local policy at each iteration and maximize the value of the local MDP of the other agent. The process continues until the value cannot be improved anymore.

```
1.1 Start from $A_1$, choose a random policy $\pi_2$ for $A_2$ (the simplest being that
    $A_2$ will not do any action at any state), generate the local MDP for $A_1$.
    Solve the MDP and get the local policy $\pi_1$ for $A_1$.
1.2 repeat
        Fix $\pi_1$ for $A_1$, generate the local MDP for $A_2$. Generate the local
        optimal policy $\pi'_2$ for $A_2$. If $V_{\langle \pi_1, \pi_2 \rangle}(o_2^0) == V_{\langle \pi_1, \pi'_2 \rangle}(o_2^0)$, go to step 3;
        else, update $\pi_2$ to $\pi'_2$.
        Fix $\pi_2$ for $A_2$, generate the local MDP for $A_1$. Generate the local
        optimal policy $\pi'_1$ for $A_1$. If $V_{\langle \pi_1, \pi_2 \rangle}(o_1^0) == V_{\langle \pi'_1, \pi_2}(o_1^0)$, go to step 3; else,
        update $\pi_1$ to $\pi'_1$
    until false
1.3 Return $\langle \pi_1, \pi_2 \rangle$.
```
Algorithm 1: Iterative Algorithm

The following theorem states that the Iterative Algorithm converges to a local optima.

Theorem 2. *The Iterative Algorithm converges.*

Proof. Let me first prove that the global utility of the local MDP's is monotonically increasing at each iteration, i.e., $V_{\langle \pi_1, \pi'_2 \rangle}(o_2^0) \geq V_{\langle \pi_1, \pi_2 \rangle}(o_1^0)$ and
$V_{\langle \pi'_1, \pi_2 \rangle}(o_1^0) \geq V_{\langle \pi_1, \pi_2 \rangle}(o_2^0)$.

Since π'_2 is the optimal policy of A_2 when fixing A_1's policy as π_1, we have $V_{\langle \pi_1, \pi'_2 \rangle}(o_2^0) \geq V_{\langle \pi_1, \pi_2 \rangle}(o_2^0)$. According to Theorem 1, $V_{\langle \pi_1, \pi_2 \rangle}(o_2^0) = V_{\langle \pi_1, \pi_2 \rangle}(s_0) = V_{\langle \pi_1, \pi_2 \rangle}(o_1^0)$. Therefore, we have $V_{\langle \pi_1, \pi'_2 \rangle}(o_2^0) \geq V_{\langle \pi_1, \pi_2 \rangle}(o_1^0)$. Similarly we can prove $V_{\langle \pi'_1, \pi_2 \rangle}(o_1^0) \geq V_{\langle \pi_1, \pi_2 \rangle}(o_2^0)$.

On the other hand, since there is a finite joint policy space, there exists a globally optimal solution. The iteration is bound to stop when the utility hits the upper bound, i.e., that of the optimal solution if it does not stop before that. Therefore, the algorithm converges. □

A natural corollary of this theorem is that the global optimal solution is at one of the local convergence points generated by the Iterative Algorithm. From the proof of Theorem 2, we can see that for an iterative algorithm like ours, the sufficient condition for it to converge to a local optimal solution is that the local MDP is maximizing the same global utility as the decentralized MDP. Therefore, for any decentralized MDP, if we can construct a local MDP for the local agents which maximizes the same global utility as that of the DEC-MDP, an iterative algorithm will converge to a local optimal solution. There is other work using the same general idea, such as described in [13].

The Iterative Algorithm is a naturally anytime algorithm. The algorithm can stop at any time and generate a policy and after each iteration the policy generated has a better value than the last one. Although the Iterative Algorithm is guaranteed to converge to a local optimal solution, it needs to dynamically regenerate the local MDPs at each iteration. As a result, it is suitable to be run off-line. One variation that can potentially reduce the time to converge is not to generate the local optimal policy at each iteration. Instead, run only one iteration of the value iteration algorithm to improve the local policy before switching between the agents. The other main disadvantage is that it depends on the starting policy chosen; it may be stuck at some fairly low quality local optimal solution without being able to reach the globally optimal one. One solution to the later problem is to randomly restart with a new initial policy and pick the best joint policy generated after a few time steps.

The Iterative Algorithm can be extended to more than two agents fairly easily. The most natural extension is a Flat Iterative Algorithm (Algorithm 2). In this algorithm,

put back to the end of the queue. The second method is a random selection method, where a random agent (except the one who has just finished the process) is selected for the next iteration. However, there is a potential issue for both methods that is related to the halting criteria. The criteria for the two agent version of the algorithm is for the agent at the current iteration to simply compare the value of its new policy to that of the old one. The algorithm halts if they are the same. This criteria does not work with either the round robin method or the random selection method without modification. In order for the algorithm to work properly, i.e., it halts when and only when the process has converged, we need to take extra steps before deciding whether the convergence has been reached. Once an agent discovers that the new policy after its iteration has the same value as the old policy, it initiates a process during which each agent takes turn to run another iteration. If and only if none of the agents changes the value of the joint policy as a result of its iteration will the algorithm halt.

```
3.1 Input: $n$ agents $\{A_k\}, 1 \leq k \leq n$;
    $m$ threads/computing resources $\{R_i\}, 1 \leq i \leq m$, each with computing
    capacity of $c_i$, where $\sum_{1 \leq i \leq m} c_i = n$;
    $t$: fixed length for each episode.
3.2 Output: Policies $\{\pi_k\}$ for all the agents.
3.3 Initiate policies for all the agents to $\{\pi_k\}$. Denote the value of the start
    state for these policies as $V$.
3.4 For each computing resource $R_i (1 \leq i \leq m)$, select $c_i$ agents,
    $\mathcal{A}_i = \{A_{i1}, A_{i2}, \ldots, A_{ic_i}\} \subseteq A$.
3.5 For each computing resource $R_i$, run Iterative algorithm for the agents $\mathcal{A}_i$
    for a maximum time of $t$, fixing the policies of the agents $\{A_k\} - \mathcal{A}_i$ to
    $\{\pi_k\} - \{\pi_{i1}, \pi_{i2}, \ldots, \pi_{ic_i}\}$. At the end of the episode, the algorithm
    generates a new set of policies for agents $\mathcal{A}_i$: $\{\pi'_{i1}, \pi'_{i2}, \ldots, \pi'_{ic_i}\}$ with a value
    of $V_i$.
3.6 If $\forall i$, $V_i = V$, then terminate the algorithm. Return $\{\pi_k\}$.
3.7 Otherwise, for $j = argmax_i V_i$, replace all the policies for agents $\mathcal{A}_i$, i.e.,
    $\pi_{j1} \Leftarrow \pi'_{j1}, \ldots, \pi_{jc_j} \Leftarrow \pi'_{jc_j}$. Replace $V$ with $V_j$. Goto 3.4.
```
Algorithm 3: Parallel Iterative Algorithm

There is an alterative to the Flat Iterative Algorithm, the Parallel Iterative Algorithm (Algorithm 3). The basic idea is to divide the agents to subgroups and run

value of the states in the decentralized MDP given a joint policy. The Lookup Algorithm (Algorithm 4) is an attempt to take advantage of Theorem 3 to approximate the values of the local MDP without knowing the remote policy.

```
4.1 Solve the underlying centralized MDP of the DEC-MDP, build a lookup
    table of the optimal state value of each state $V^{c*}(s)$ (see Definition 10).
4.2 When reaching a new observation sequence $\overline{o}_i$, choose the action according
    to the following equation, where $s'$ is the next state of $s$ in the decentralized
    MDP after taking action $a$.
        $\pi_i(\overline{o}_i) = argmax_a(R(a) + \sum_{s \in S} P(s|\overline{o}_i)V^{c*}(s'))$    (2.2)
```

Algorithm 4: Lookup Algorithm

This algorithm makes use of the simple structure of the decentralized MDP. The mostly deterministic nature of the DEC-MDPs makes the underlying centralized MDP very easy to solve. The Lookup Algorithm utilizes the optimal state values of the underlying centralized MDP as a guidance to approximate the globally optimal solution. According to Theorem 3, (2.2) is trying to maximize a very optimistic approximation of $V_{\langle \pi_1, \pi_2 \rangle^*}(\overline{o}_i)$, since $V^{c*}(s')$ is an optimistic estimate of $V_{\langle \pi_1, \pi_2 \rangle^*}(s')$. Nevertheless, without knowing what the optimal solution is, it is still a fairly good estimate. The main merit of the algorithm is the simplicity of the heuristic. If along with the state value lookup table, we also store the prior probability of each value combination of the data set, then even $P(s|\overline{o}_i)$ can be easily calculated without resorting back to the original BN. This algorithm can be executed online efficiently after the lookup table is constructed. A simple algorithm described in [47] uses the same idea to solve POMDP.

2.3.4 Experimental Results

I implemented both both the Iterative Algorithm and the Lookup Algorithm described above. I have run experiments on 100 problem structures with 2 agents, 2 high level events and 10 low level data (5 local to each agent) for different confidence

It is interesting to observe how the minimum expected communication cost changes when the required confidence level increases. At first, increasing the confidence does not require much increase in the communication cost, while later the same amount of improvement in the confidence needs a lot more communication. When the confidence reaches a certain level, no further communication is required for improvement. This is understandable because the critical data are chosen to be transferred first, which has the most impact on increasing the confidence. Later, more non-critical data needs to be communicated to gain the same amount of improvement. Finally, the comparatively irrelevant data will not contribute a lot to improving the solution quality.

The framework I have presented can also be used to analyze the suitability of a problem to be solved by a distributed system. Figure 2.6 shows the minimum expected communication cost curve of two different problems. Problem structure 1 is much more suitable for a distributed solution since it needs comparatively little communication to achieve high confidence level. Further experiments and theoretic work need to be done to help design suitable systems for different problems with different communication properties.

2.4 Complexity Results

The general finite state DEC-MDP assumes that its horizon $T = O(|S|)$. However, the horizon of our problem is much shorter. Since each agent can at most request all the other agent's data and send all its own data, the horizon of the DEC-MDP is therefore $T = O(|D|) = O(log|S|)$, where $|D|$ is the number of sensor data in the network. As a result, we have the following complexity result.

Theorem 4. *Deciding a communication management problem in Distributed Sensor Interpretation is in NEXP in the number of sensor data.*

Proof. In order to prove the theorem, we need to prove that there is a verifier that is exponential in the number of sensor data.

A joint policy in a DEC-MDP can be evaluated by representing it as a belief state graph. Each node in the graph is composed of the state, the sequence of observations for agent i and for agent j. Each node has a single joint action, which is defined by the joint policy. The transitions between the nodes depends only on the transition and observation functions, and each transition has a reward defined by the reward function. The belief state graph can be evaluated using the standard MDP recursive value function and policy evaluation, which runs in time polynomial in the size of the graph. For a DEC-MDP, this size is $|\Omega_i|^T \times |\Omega_j|^T$. Since $T = O(|D|)$, the policy evaluation takes $O(|\Omega_i|^{|D|} \times |\Omega_j^{|D|}$, which is exponential in the number of the sensor data $|D|$. Therefore, the communication management problem $\in NEXP_{|D|}$. $\square$

Since the size of the state space is exponential to the number of sensor data in the network, i.e., $|D| = log|S|$, the communication management problem is NEXP in $log|S|$, which is of lower complexity than a general DEC-MDP. However, we have the following theorem.

Theorem 5. *Deciding a communication management problem in Distributed Sensor Interpretation is harder than NP in the size of the state space.*

Proof. I prove this theorem by contradiction. Assume that the problem is in NP. This means that there exists a polynomial time verifier for any policy in the DEC-MDP built for it. If a policy can be verified in polynomial time, then it must have a polynomial sized representation. On the other hand, the policy of our DEC-MDP is in the form of $\overline{\Omega}_i \times \overline{\Omega}_j \rightarrow A_i \times A_j$. We have $|\overline{\Omega}_i \times \overline{\Omega}_j| = |\Omega_i|^T \times |\Omega_j|^T = |\Omega_i \times \Omega_j|^{|D|} = |\Omega_i \times \Omega_j|^{log|S|}$, which is not polynomial in the size of the state space. Contradiction. Therefore, the communication management problem is harder than NP in the size of the state space. $\square$

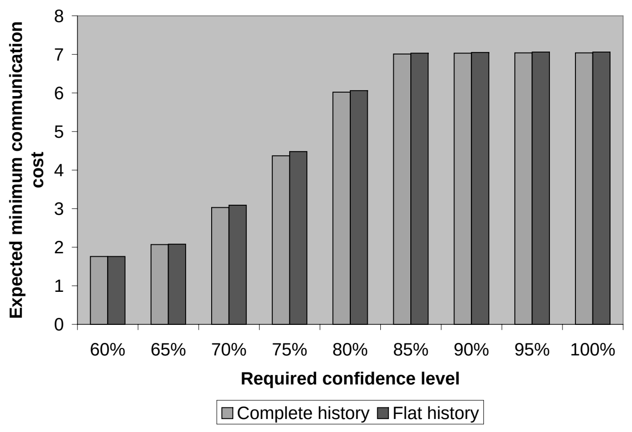


Figure 2.7. The comparison of the expected communication cost of the joint policies generated by the exhaustive search algorithm using complete history of communication and flat history.

The theorem tells us that while the communication management is easier than the general DEC-MDP, it is still hard to solve. If we can build an approximate model of the problem that has a lower complexity and yet generate a near optimal policy, then the problem is easier to scale.

One way to approximate the optimal solution to this problem is that instead of remembering the entire observation history $\overline{\Omega}$, each agent only remembers the values of the data exchanged so far without remembering the order in which they were transferred. This is a reasonable approximation since to calculate the confidence level of the local interpretations, only the sensor data values are needed. In this approximation, the policy is of the form $E_i \times E_j \rightarrow A_i \times A_j$, where E_i and E_j are the flat observation history of agent i and agent j. If we assume that each sensor data has at most m possible values, then $|E_i| = O((m+1)^{|D|}) = O((m+1)^{log|S|}) = O(|S|^{log_m(m+1)})$. Since m is independent of $|S|$, E_i is in the size polynomial in the state space. Therefore, there exists a polynomial verifier for this approximation. As a result, this approximation is no harder than NP in the size of the state space.

Figure 2.7 shows the comparison of the average expected communication cost of the joint policies generated by the exhaustive search algorithm using complete history of communication and flat history for 100 networks of 2 agents with 1 higher level hypothesis and 5 lower level data each. While the flat history approximation did not lose considerable performance, it did show at least 70% time savings to solve the DEC-MDP.

2.5 An example Application: SensEye

This section is to illustrate how we can model the sensor wake up on demand problem in a multi-tier sensor network [41] with a distributed Bayesian Network. The DEC-MDP based approach introduced in previous sections can then be taken to manage the communication among agents in order to decide which high level sensors to wake up based on the available sensor data and each sensor's capabilities.

2.5.1 SensEye: Applications and System Model

Technological growth has led to a variety of sensors and networked sensor platforms with different cost, form-factor, resolution, and functionality. For example, camera sensor products range from expensive pan-tilt-zoom cameras to inexpensive webcams and cell-phoneclass cameras to even cheaper, tiny cameras such as Cyclops [70]. Similarly, a set of options become available for sensor platforms, with choices ranging from embedded PCs and PDA-class Stargates [76], to low-power Motes [20, 35] and even lower power systems-on-a-chip [39].

Due to these advances, the design and deployment of camera sensor networks - wireless networks of sensor nodes equipped with cameras - is now feasible in a variety of application scenarios, especially in target tracking and monitoring applications. Examples include environmental monitoring, where a camera sensor network is used to monitor wild-life habitats, rare species in remote locations being disturbed by humans, and adhoc surveillance, where camera sensors are used in disaster management scenarios like fire and floods.

A well designed sensor network should have the desired quality of low cost, high coverage, high functionality, and high reliability. However, a flat, single tier network of homogeneous sensor nodes invariably leads to the achievement of some of these properties at the sacrifice of the others. For example, low cost can be achieved by using a single tier of inexpensive sensors but at the expense of functionality. High functionality can be achieved by employing high fidelity sensors but at the expense of sacrificing coverage due to the high cost. Kulkarni et. al [41] demonstrated that by employing camera sensors with different power consumption and capabilities to perform tasks with different requirements, multi-tier networks provide a better balance of cost, coverage, functionality, and reliability. SensEye [41], a multi tier network of heterogeneous wireless nodes and cameras, can potentially achieve an order of magnitude reduction in energy usage while providing comparable system performance as the more traditional single tier network.

In this approach, resource-constrained, low power elements are employed to perform simpler application tasks, while more capable, high-power elements take on more complex tasks. Doing so results in more judicious use of precious energy resources. Only when necessary are the higher level sensors woken up to perform the more complex tasks that the lower level sensors cannot complete.

SensEye assumes a three-tier architecture (see Figure 2.8). The lowest tier in SensEye comprises Mote nodes equipped with 900MHz radios and low-fidelity Cyclops or CMUcam camera sensors. The second SensEye tier comprises Stargate nodes equipped with web-cams. Each Stargate is equipped with an embedded 400MHz XScale processor that runs Linux and a web-cam that can capture higher fidelity images than Tier 1 cameras. Each Tier 2 node also consists of two radiosa 802.11 radio that is used by Stargate nodes to communicate with each other, and a 900MHz

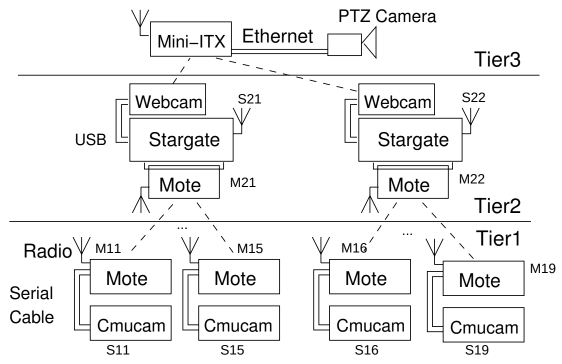


Figure 2.8. SensEye hardware architecture.

radio that is used to communicate with Motes in Tier 1. The third tier of SensEye contains a sparse deployment of high-resolution pan-tilt-zoom cameras connected to embedded PCs. The camera sensors at this tier are retargetable and can be utilized to fill small gaps in coverage provided by Tier 2 and to provide additional redundancy for tasks such as localization. Nodes in each tier and across tiers are assumed to communicate using their wireless radios in ad-hoc mode and no base-stations are assumed in this environment.

I use a standard object tracking environment as an example. Since object detection is a relatively simple task to perform, it is carried out by the tier 1 sensors in SensEye, and the higher tier sensors are normally in sleeping mode until woken up. Once an object is detected at Tier 1 by multiple sensors, localization algorithm is performed to decide the location of the object and decide which Tier 2 sensor to wake up in order to better track the target. If no Tier 2 sensor is at an appropriate location and pointed in the correct direction, a retargetable Tier 3 sensor is woken up to properly track the target. When the target is moving, the new data collected by the sensors can be used to predict the movement of the target and decide which sensors to wake up the next.

Low energy consumption and low latency are two important performance measurements of a sensor network such as SensEye. Wakeup On Demand task is a crucial problem from both perspectives. From the energy perspective, there should be no wasteful wakeups of the higher tier nodes. The localization algorithm plays a key part in this issue. It requires information from more than one sensor nodes, and therefore communication is necessary. From the latency perspective, the separation of detection and tracking tasks across two different tiers introduces latency between the execution of the two tasks. The delay includes the time needed for communication that is involved in localization of the target and the receiving of any wakeup packets that are transferred if the command is issued by agents other than the processor controlling the sensor that is being woken up. Additionally, communication requires considerable power consumption and should be limited in a sensor network in general. Therefore, minimizing communication becomes an important issue in order to advance the system performance in SensEye.

2.5.2 The Wakeup On Demand Problem

The wakeup on demand problem can be broken down to two stages. At the first stage, the higher level mote needs to decide whether there is activity in the area covered by the sensor associated with it. At the second stage, armed with the localization knowledge from the last stage and other information such as the high level sensors' availability and work load, the high level motes can then in turn decide which sensor is the best to assign the task to and wake up. Stage two is essentially a task allocation problem, and can employ well studied techniques such as distributed constraint optimization algorithms [54, 51]. I will focus on discussing the first stage of the problem.

In the current prototype, upon detecting a new target, the tier 1 node broadcasts a wakeup trigger for the higher tier including the location parameters, and each wakeup

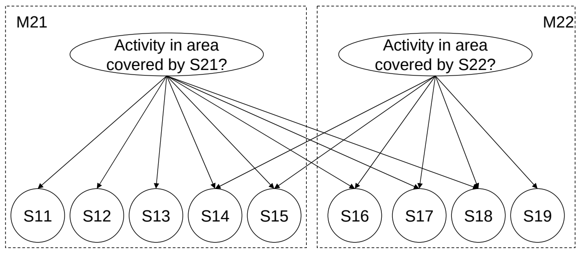

M21
M22
Activity in area covered by S21?
Activity in area covered by S22?
S11
S12
S13
S14
S15
S16
S17
S18
S19

Figure 2.9. The interpretation problem in SensEye modeled with a Bayesian Network.

mote at the higher tier uses the localization algorithm to decide whether to wake up the sensor associated with it. This broadcasting method can be disadvantageous in certain environments. For example, if there are multiple objects detected around the same time, the communication traffic can be too crowded and it may lead to lost or garbled packets. Additionally, in certain applications, broadcasting should be used at caution if not totally avoided because of privacy issues. In such environment, I propose the following approach.

Instead of broadcasting the wakeup trigger upon object detection, each Tier 1 mote is assigned one Tier 2 wakeup mote to communicate with at the initialization stage. Once a new object is detected, the Teir 1 mote sends the necessary information to the Tier 2 mote assigned. The Tier 2 mote then can choose to communicate with other Tier 2 motes to acquire additional localization data if necessary, in order to decide whether there is activity in the area covered by the sensor associated with it. The assignment of Tier 2 motes to Tier 1 motes should take into consideration whether the two motes have overlapping coverage to lessen unnecessary communication. In addition, load balancing should be considered as well so that no Tier 2 mote is overburdened with packets from a large number of Tier 1 motes.

We can model this approach with a standard Distributed Sensor Interpretation problem using a Bayesian Network. Figure 2.9 shows the interpretation problem for $M21$ and $M22$ in the SensEye setup illustrated in Figure 2.8. In this setup,

$M21$ has direct communication channels with $M11$ through $M15$. When any of $S11$ through $S15$ detects something, the associated motes send the information to $M21$. Additionally, $M21$ and $M22$ can also communicate with each other when necessary, and are responsible for deciding whether there is activity in the area covered by $S21$ and $S22$, respectively.

Though distributed approaches to DSI problems can have many advantages, the existence of subproblem interactions means that the agents must communicate during the problem-solving process to obtain information that is needed to solve their subproblems optimally. A coordination strategy needs to be generated that specifies how agents should interact. In the SensEye example, even though $M21$ does not have direct access to $S16$ through $S18$, they do cover areas overlapping with $S21$. Therefore, in order to decide whether there is activity in the area covered by $S21$, $M21$ may need the data provided by $S16$ through $S18$, and communication between $M21$ and $M22$ may become necessary.

After formulating the first stage of the wakeup on demand problem in SensEye, we can further formulate its communication management problem with a DEC-MDP as discussed in previous sections. With this approach, we can generate a communication policy such that when followed, each Tier 2 mote needs to only transfer the least amount of information to and from the other Tier 2 motes in order to decide whether to wake up its associated camera sensor with the required confidence. Since in SensEye, we assume that each mote knows the location and direction of the other sensors, the BN can be pre-generated and is fixed until the sensor network topology is changed. As a result, the communication policy can be generated off-line beforehand and stored at each mote.

There are several questions that need to be answered before this approach can be employed effectively. First, the data that is collected by Tier 1 sensors are the location data of the object, and are continuous numbers. In order for the BN to be practical, they need to be discretized. This is reasonable since most times we do not need the accurate location to decide whether to wake up a Tier 2 sensor. As long as the new target is within the effective coverage of the sensor, it can perform the task satisfactorily.

The second problem to be addressed is to build a good cost model for the communication actions. There are several factors to be considered: the packet cost, which is related to both the initialization cost and the packet length, and the possible stages that may be required depending on the data on hand and the previous data received. Intuitively, large number of cycles should be avoided since it increases the latency caused by communication.

The last problem is how to decide the proper setting of the required confidence level. Ideally, the confidence level should be set as high as possible such that it is still achievable within the communication channel limit. This can be achieved by solving DEC-MDPs with different levels of required confidence and generating a graph that shows the relationship between the confidence level and the minimum expected communication cost needed. Such a graph can then be used to decide the proper confidence setting given the current communication channel limit.

There is a potentially interesting issue of the wakeup on demand problem if both stages are to be implemented and connected. During the second stage when the higher level motes are trying to decide which sensor is the best to wake up, they may discover that for various reasons the initial required confidence level set for the stage one interpretation is too low for the cost of waking up / reassigning any of the high level sensors. In other words, with the initial confidence level in the location of the new activity, it is not worth the cost for waking up / reassigning a sensor for monitoring it. Under such circumstances the motes may decide to set a higher confidence requirement and revisit the interpretation problem.

There are two different ways in which this can be done. Suppose that the initial required confidence level is $X\%$, and the new required confidence level is $Y\%$, where $X < Y$. In the first approach, initially the motes follow the communication policy generated for the DEC-MDP with the $X\%$ confidence. When a higher confidence level $Y\%$ needs to be reached at a later time, it will then start from the local observation sequence at which it halted during the first process, but now following the policy generated for the DEC-MDP with the new $Y\%$ confidence instead. While the complete policy to the DEC-MDP for $Y\%$ includes policies for all the possible local observation sequences, including the terminal sequences of the $X\%$ DEC-MDP, we would ordinarily only store the policies for the reachable sequences in the motes' memory to save space. As a result, in order to be able to start a communication process from where was left off at the first stage, the motes need to remember the policies for the terminal observation sequences of the $X\%$ DEC-MDP as well.

The second approach differs from the first one in that at the first stage, the policies followed by the motes are those generated by the $Y\%$ DEC-MDPs instead of those by the $X\%$ DEC-MDPs. The communication continues until the $X\%$ confidence level is reached. When later it is discovered that the confidence level needs to be increased to $Y\%$, the motes can simply pick up where the communication was left off. We notice that the policy the motes follow to reach $Y\%$ confidence is not optimal for $X\%$ DEC-MDP. As a result, the first approach is more suitable for the situation where it is unlikely that at a later stage the confidence needs to be raised to $Y\%$, while the second approach works better when it is more likely that the confidence has to be increased.

2.6 Other work studying Distributed Sensor Networks

Large scale sensor networks are a useful technology for a wide range of application. Therefore there has been an increasing interest in the networks and artificial intelligence community to study various aspects of employing distributed scale sensor networks, including the placement of sensors, representation of the interpretation problem and communication problem among the sensors.

Graphical models are widely used in the study of sensor networks. Huang and Russell [38] are the first to represent the interpretation problem in a sensor network as a Bayesian Network. They formulate the raw sensory data as evidence and the object identification, i.e., how an object can be expected to appear at subsequent observations given its current appearance as appearance probability. With this formulation, they are able to achieve high levels of performance in the task of recognizing cars observed by cameras at widely separated sites in a freeway network. Krause et al [40] use a non-parametric probabilistic models called Gaussian Processes to study the best sensor placement in a network both for the spatial phenomena of interest and for the spatial variability of link qualities. We choose Bayesian Networks in our work to represent the interpretation problem because it captures the causal relationship both qualitatively and quantitatively between the observable data and the events that cause them.

The study of communication management in Distributed Sensor Networks can be largely divided into two categories: all the relevant data are collected by a central agent who processes them and find the best interpretation; or the data are interchanged among the agents such that the processing is distributed and there is no central node to collect all the data. In the first category, much work has focused on how to route the data through the network from the data observing agents to the central node that is collecting and processing all the data. Deshpande et al [22] and Meliou et al [53] both study this query routing problem where the communication cost is based on the communication distance between the sink node and the data observing agent. Cristescu et al [19] study a similar problem but the network is composed of a sink node and a tree communication structure. Li et al [46] propose a routing protocol that minimizes the communication time and therefore minimizes delay involved in transmitting the data. In the second category where the data processing and communication happen distributedly, the research has been focused on a higher level than the network routing problem. Krause et al [40] study the problem of sensor placement such that the communication is minimized while the sensing quality of the sensors is maximized. In their work, the communication is assumed to be potentially unreliable and the communication cost is associated with the reliability of the communication channel between nodes. However, the communication cost of a network's organization is averaged among all the possible routes existing in the network and is not associated with the quality of the interpretation yielded by the data communication and processing. Paskin et al [63, 64] study the communication problem in a very similar setting to that of this thesis work where the goal is to minimize communication cost in a distributed sensor network represented by a Bayesian Network. However, in their work the communication cost is only minimized for the accurate interpretation of the network data, while in this work, it is minimized for any given confidence level in the interpretation required by the problem.

2.7 Conclusions

There is beginning to be research applying the decision-theoretic framework to multi-agent systems. The Multi-agent Markov decision process (MMDP) defined in [8] is a centralized multi-agent extension of an MDP. In that work, each agent observes the global state directly and takes an joint action. Xuan and Lesser [81] proposed a decentralized extension, in which an agent has its local state in addition to the global state. When an agent has ambiguity over which action it should take based on its local state, it needs to communicate with the other agent to synchronize the global view. Similarly, researchers studying communication in multi-agent systems focus their attention on the type of communication that forces the agents to exchange their observations whenever they communicate and therefore synchronizes each agent's partial view of the system. My work is unique in the way that the agents cannot simply observe the global state directly or even synchronize to it occasionally. In fact, this inability to observe the global state at any time is the key factor that leads to the difficulty of the problem. Instead of trying to answer the question of "when" to communicate as most other researchers do, this work is addressing the issue of "what" to communicate instead. Pynadath and Tambe [69] proposes a general framework (COM-MTDP) which extends the DEC-MDP to incorporate communication actions as well as domain actions. This framework provides an excellent foundation for theoretical analysis of different coordination strategies in the teamwork context, but does not provide a practical algorithm for approximate or optimal solutions. In contrast, my framework is designed for the distributed systems whose connection between its communication cost and the progress of the collective problem solving progress can be identified. Additionally, I am modeling only the communication actions of the system. This enables us to find the optimal or near-optimal solutions for a given problem.

Modeling the problem structure as a BN is also unique. By utilizing the information provided by the BN structure, I am able to build a complete model of the DEC-MDP and calculate the local agent's belief in the global state and the other agent's local state. In this respect, my model of the local MDPs resembles a POMDP. In fact, the techniques of solving a DEC-MDP presented in this paper can be used for general DEC-MDPs as long as they have a complete model. That is exactly what the BN representation achieves. The DEC-MDP framework I presented for the DSI problem can be easily extended to minimize communication cost in the more general Cooperative Distributed Problem Solving context, such as distributed scheduling problems. The only required condition is that one can identify the information (observations) accessible to each agent, the goal that the system tries to achieve, the cost for sharing information among the agents, and how the content of the observations affect the achievement of the goal. In the DSI example, the information accessible to each agent is the local sensor data collected and the goal of the system is the desired confidence level of the subproblems, which is used to decide whether a state of the DEC-MDP is a final state. The Distributed BN structure is used to update the new observations and calculate whether the final state is reached. Similarly, for the distributed scheduling problem, the information accessible to each agent is the scheduling constraints of the agent itself, and any constraints the other agents are willing to share with it. The goal of the problem is whether the task(s) can be successfully scheduled and how satisfied the agents concerned are about the schedule. The cost of sharing local constraints with other agents may involve both the communication cost and how much the agents want to keep the information secret from the others, i.e., the privacy cost. If we can establish the relationship between the amount and content of local constraints shared among the agents and the probability of successfully scheduling the task(s), then the techniques introduced in this chapter can be applied to the distributed scheduling problem to decide what local constraints to share with other agents.

What makes my problem more difficult than some of the others [4] is its tightly coupled nature. An agent's communication action directly changes the other agent's view. The local MDPs of the two agents are largely dependent on each other. This makes it hard to construct algorithms that are guaranteed to find the globally optimal solution. My complexity result shows that even though our problem is of a lower complexity class than the general DEC-MDP, it is still harder than NP. I have designed two algorithms to approximate the globally optimal solution for our DEC-MDP. One is an iterative algorithm that is guaranteed to converge to a local optimal solution, but the quality of the policy it generates largely depends on the starting policy of the iterative process. The other approach is based on a lookup algorithm which is much less computationally expensive and can be easily extended to more than two agents. Though there is no guarantee that can be made about the solution either generates, experimental work indicates that in the problems studied both approaches lead to policies that are of very good quality. To my knowledge, this is some of the first work providing algorithms that approximate the optimal solution for communication problems in a complex problem solving setting that are formulated in a decision-theoretic model.

There are different ways that my framework can be used. First, it can be used as described in this chapter to find the optimal communication policy for a given goal. In the DSI problems, this goal is a required confidence level. Conversely, in certain applications, there is a known fixed communication constraint. For such applications, it is often useful to estimate the expected best confidence level that can be achieved under the given constraint. Our framework can be used for this purpose. For a problem structure, we first solve the communication management problem for different confidence levels. A confidence–communication cost graph such as the one in Figure 2.6 can be generated. It is then trivial to look up the maximum expected confidence level for a given communication constraint. The third way to use this framework is to view the expected minimum communication cost generated as a measurement of decomposability of a distributed problem or as a measurement of privacy. For example, Figure 2.6 shows that problem structure 1 requires more communication cost to achieve the same confidence level than problem structure 2, and therefore problem structure 2 is more decomposable. This is particularly useful when we need to distribute sensor data to different agents. We can generate confidence–communication cost curves for different network configurations, and choose the one that requires the least communication cost.

The main problem with using a decision theoretic model such as DEC-MDP in multi-agent applications is its high complexity class. It makes the problem hard to

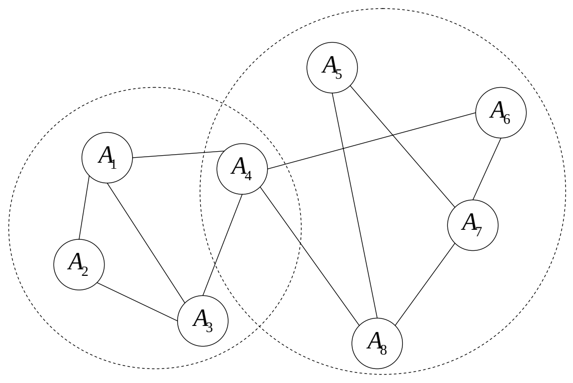


Figure 2.10. There are two subnets in the system. Agent A_4, as the overlapping agent of the two subnets, serves as the gateway of the two subnets.

scale up. For example, the current BN structure I work with for the DSI problem is comparatively small and simple. To effectively model real world applications I will need to scale it along several dimensions: the number of agents, the number of events, the number of sensor data elements, and the layers of intermediate results between the events and the data.

One of the easiest ways to extend the approach to handle multiple agents is for every agent to treat the external network as a virtual agent and model the problem the same way I discussed in this chapter. As the number of agents in the system increases, there is a need to divide the agents into nearly decomposable groups such that there is little overlap between different groups (Figure 2.10). Since each agent only has a partial view of the network that is relevant to its problem, the entire system is composed of multiple nearly disjoint sub-networks. The agents that are in the overlapping areas will serve as gateways between sections (e.g., in Figure 2.10, A_4 serves as the gateway between the two subnets). The other agents do not know of the existence of the subnets that they are not members of. This enables us to use our current scheme and scale up the network to include a large number of agents.

The effectiveness of this scheme for scaling up the system depends to a degree on being able to partition the events and data so that fairly limited communication will be required between agent groups. While this may not be possible in all domains, [14] have done experiments with a simulation framework built for the DBN structure I study in this work and the results are encouraging. For the the vast majority of domain models they have studied, events can be pursued independently of one another (i.e., determining their correct values, such as true/false). Where significant data from another group is required for determining the correct value of an event with the desired confidence, they have been quite successful at limiting communication by identifying "markers" in the data/results that indicate the need for additional evidence.

Increasing the number of sensor data elements will inevitably increase the state and action space of the DEC-MDP under the current mapping, and therefore makes it almost impossible to solve the problem exactly. There are several ways to address the scalability issue. One is to design approximate algorithms that tradeoff solution quality for lower complexity. Another way to handle scaling is to change the way we map the original communication problem to DEC-MDP. For example, in the current mapping, the state space and action space of the DEC-MDP are exponential to the amount of data in the BN. It is computationally infeasible to generate a large DEC-MDP when the quantity of data grows. One approach is to extend the BN to multiple layers, where the intermediate nodes represent the intermediate causes. Instead of transmitting the observed data directly, the agents can abstract the information contained in the data into the intermediate nodes and communicate them. The next chapter discusses this approach that utilizes abstraction communication actions.

CHAPTER 3
THE USE OF ABSTRACTION

If the DEC-MDP allows general communication and the communication actions have different cost, the number of actions allowed is exponential to the number of local observations. For example, the size of the state space of the DEC-MDP I built for the DSI problem in the last chapter is exponential to the amount of data in the BN, and so is the action space. This means that such a DEC-MDP is very hard to solve when the BN grows in the number of sensor data. If we have techniques that can efficiently reduce the size of the DEC-MDP, then it is possible to find an approximate solution in reasonable amount of time.

The technique that I present in this chapter uses the idea of transferring abstract information between the agents instead of the part of the observations themselves. Abstract layers are introduced into the Bayesian Network as a way of carrying more useful information in one piece of transmitted data to further reduce the number of messages that need to be sent. Initially the abstract information is generated locally by each agent such that its observations can be encoded more efficiently. However, this efficiency is not without loss, since part of the necessary information may be missing from the abstract data transferred between the agents. Therefore, when the agents discover that transferring the abstract information alone cannot enable them to reach the final states, they fall back to transmitting the less abstract information. This process continues until a final state is reached.

In order to effectively use the abstraction technique, there are two questions that need to be answered: How to generate the abstract information to be transferred, and

When to transfer the abstract information. To answer the first question, I present an algorithm that automatically generates appropriate abstraction data, which reduces the expected communication cost necessary to achieve the required confidence level. This is a greedy approach since the abstraction data enables the agent to immediately reach the desired confidence level. I chose this type of abstraction data because it is intuitive and straightforward to generate, yet it enables us to study the concept of abstraction and its use in reducing communication cost from a formal perspective. There are other forms of abstraction that can be potentially useful. For example, some abstraction data may contain useful information but is not sufficient to reach the confidence by itself. Other abstraction may be an intermediate interpretation result that can be incorporated into the local result. I plan to look into these different types of abstraction in future work. Another simplification I use in the algorithm and examples is that all the data are binary valued as multi-valued. There are two motivations behind this. First, a multi-valued data can be broken down into multiple binary valued data and the same techniques discussed can be applied, though this may drastically increase the number of data present in the network. The second reason is that as discussed before, the data nodes in this work are not the raw sensor output, but rather the results of some initial processing to produce categorical/discrete values. In many applications, these data nodes are binary feature nodes that denote whether certain characteristics are present [49, 52].

To answer the second question, I examine three different approaches to incorporate the new abstract communication actions to the existing DEC-MDP. The *all data action selection* approach allows abstract communication actions as well as the normal communication actions in all the states. The *abstraction data action selection* approach allows only the abstract communication actions in all the states. In the *hierarchical action selection* approach, before an agent has acquired all of the abstraction data, the agents are not allowed to transfer any of the observed data. Experimental work demonstrates that the hierarchial approach, simplifies the DEC-MDP while still reducing the expected communication cost. Furthermore, experiments on larger networks with multiple abstraction layers show that the hierarchical approach is able to solve more and larger networks than the other approaches and therefore is more suitable for scaling up the approach.

Abstraction has often been used in hierarchical planning community to represent abstract plans that can be further refined into specific actions. In particular, Clement and Durfee [17] used summary conditions, a form of abstraction, to coordinate the concurrent interactions of plans at different levels of abstraction between agents. Abstraction has also been used in distributed diagnosis research to identify the sources of problem. Chung and Barrett [16] demonstrated how Boolean expressions can facilitate finding minimal cost diagnoses in linear time. However, neither work studies the tradeoff between abstraction and communication. In the original Distributed Problem Solving work [23], abstraction was used as a mechanism for controlling the information needing to be communicated. However, the use of abstraction was not formalized. Nor was there a clear understanding of when the lower level data needed to be transmitted. Carver and Lesser [11] studied the use of multiple levels of abstraction to reduce the necessary communication. Like [23], the abstraction layers were predefined and the use of abstraction was not formalized. In contrast, the algorithm I introduce allows the system to generate appropriate abstraction data automatically without predefinition. My study of the addition of transferring abstraction data as actions in the DEC-MDP provides us with a formal view of the use of abstraction in the management of communication cost in a distributed problem solving system. As the sensor network gets larger and sensor technologies advance, "high fan-in" architectures are being developed. Such an architecture consists of a network of widely distributed sensors that are receptors. Their readings then will be aggregated locally with data from other nearby devices, and then further aggregated within a larger area, and so on. Franklin et al [25] discuss the key characteristics and data management challenges presented by high fan-in systems, and argue for a uniform, query-based approach towards addressing them. The techniques developed in this chapter can be seen as an implementation of the query-based approach proposed in their work. The abstraction layer generated by my algorithm is composed of the queries that guide the aggregation of the lower level data and these queries are not formulated randomly but motivated by the desire to minimize the communication cost in the system.

The previous chapter formally defined the problem I study and summarized the DEC-MDP model I developed to generate the communication strategy. In this chapter I first introduce the algorithm that automatically generates the abstraction data that when transmitted can greatly facilitate the achievement of the confidence level. The following section discusses how the abstraction data can be incorporated to improve the system performance. Finally I extend the different approaches discussed so far to larger networks with multiple abstraction layers and more than two agents.

3.1 Generating the Abstraction Layer

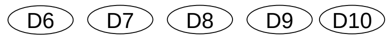
In the DSI example presented in the last chapter, I discussed two layer networks, which means that there are no intermediate abstraction data. Many domains, however, have structures that can be exploited to improve the performance of both local processing and inter-agent merging of evidence. Domain structure can result from a number of factors such as independencies among subsets of events or between events and certain data subsets and cases where only a fraction of the combinations of some data are informative. Many of these situations are best captured by using BNs that include intermediate nodes that lie between the event and data levels.

Research on learning Bayesian Networks from data [10, 33] focuses on finding the BN structure that best matches the collected data. These techniques are important in constructing the BN structure that represents the problem I am trying to solve.

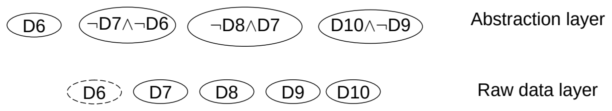
```
5.1 Input: agent $A$'s view of BN, the values of the local data, the
    required confidence
5.2 Output: a set of logic expressions of the remote data that, if true,
    put the local confidence above the required confidence
5.3 Set DataList = all the remote data of $A$, potential = null
5.4 Initialize LogicTree
5.5 while DataList.next is not null do
        currentLeaf = DataList.next
        add currentLeaf and $\neg$currentLeaf as new children to
        LogicTree.root
        if confidence given currentLeaf $\geq$ required confidence then
            mark currentLeaf
            continue
        if confidence given $\neg$currentLeaf $\geq$ required confidence then
            mark $\neg$currentLeaf
            continue
        if potential == null then
            potential.add(currentLeaf)
            potential.add($\neg$currentLeaf)
            continue
        for each $p$ in potential do
            potential.add(addChildren (currentLeaf, $p$), addChildren
            ($\neg$currentLeaf, $p$))
5.6 convert the marked part of LogicTree to a set of logic expressions
    and return it
5.7 addChildren (currentLeaf, $p$)
    begin
        Add $p$ to LogicTree as a child of currentLeaf in a depth first
        fashion, check the confidence at every step. If confidence $\geq$
        required confidence, stop going deeper into this branch and
        mark it.
        Output: the unmarked subtree of LogicTree starting from
        currentLeaf
    end
```
Algorithm 5: The algorithm for generating the abstraction layer

However, we need other methods to find an appropriate abstraction layer from the existing BN that, when transmitted from the remote agent, more efficiently conveys the necessary information to facilitate the local problem solving. In other words, this abstraction layer, when acquired, should be able to reduce the expected communication necessary to achieve the required confidence level.

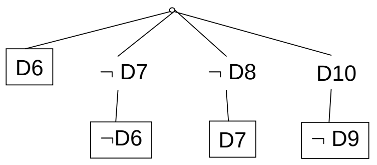
I achieve this goal by developing an algorithm (Algorithm 5) that automatically generates an abstraction layer given a value combination of an agent's local data and the desired confidence level. The basic idea behind the algorithm is to find a set of logic expressions consisting of the remote data such that if at least one of the expressions is true the required confidence is reached. For example, an agent has 5 pieces of remote data $\{D6, \cdots, D10\}$ and needs to achieve a confidence level of 75%. In the original system, the only possible actions are to transfer some or all of these observed data. I will call this approach *raw data action selection.* However, if one or more of $D6$, $\neg D7 \wedge \neg D6$, $\neg D8 \wedge \neg D7$ and $D10 \wedge \neg D9$ are true, then the agent immediately has a local solution with a confidence level of 75%. These data carry more abstract information than the observed data themselves and therefore are more efficient, though the information is not as refined. If these data values can be acquired as well as the observed data, it can potentially yield huge savings on the communication cost. Additionally, the values of the abstraction data also give valuable information about what the values the observed data may have. For example, if the value of $\neg D7 \wedge \neg D6$ is transferred and is true, then the process can immediately terminate because the confidence level of 75% has been reached. If the value is false, even though the desired confidence level is not immediately achieved the agent does now have the knowledge that $D6$ and $D7$ cannot be false at the same time. This information can be retained by updating the BN. Figure 3.1(a) shows the action options of the *raw data action selection* approach for this example while Figure 3.1(b) shows the action options if the abstraction data can also be transferred.



(a) The action options for the *raw data action selection* approach



(b) The action options for the *all action selection* approach. The data that can be transferred are divided into two categories.



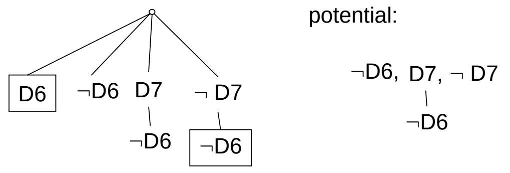
(c) The corresponding **LogicTree** to the abstraction data.

Figure 3.1. An example of the abstraction layer. Remote data includes $\{D6, \cdots, D10\}$. The required confidence level is 75%

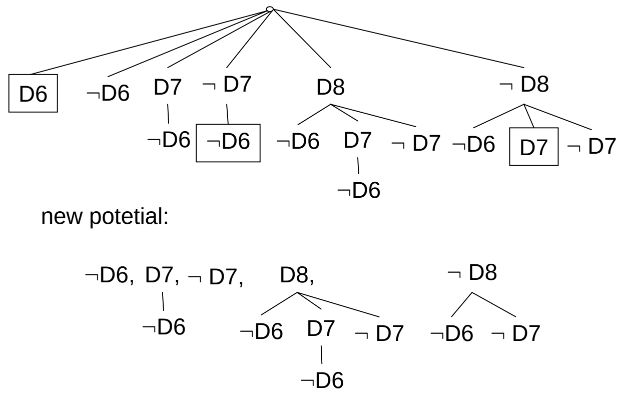
The desired set of logic expressions can be generated by simply enumerating all of the possible combinations of the remote data values and selecting those that will enable the agent to reach its desired confidence level. Our algorithm improves efficiency by a logarithmic factor over the exponential brute force enumeration by reusing the subgraphs of logic expressions.

Figure 3.2 gives an illustration of an episode showing how the algorithm is run for the example in Figure 3.1. I use a tree, which I call a **LogicTree**, to keep track of all the data values examined so far. The nodes along the same path are connected by the $\wedge$ operator, and the different branches represent different value assignments. The marked branches are the ones that pass the confidence test, i.e., if true, the desired confidence level is reached. The set **potential** is used to keep track of the subtrees examined so far that did not pass the confidence test. Only the members of **potential** should be examined as a subgraph of longer expressions. Figure 3.1(c) shows the final **LogicTree** for this example.

If the values of all the remote data are acquired, the local confidence is guaranteed to reach 100%. Therefore, I limit the depth of the **LogicTree** to be less than the



(a) **LogicTree** and **potential** before adding $D8, \neg D8$ to **LogicTree**.



(b) **LogicTree** and **potential** after adding $D8, \neg D8$ to **LogicTree**.

Figure 3.2. An example of the key step of Algorithm 5.

number of remote data. When the maximum depth is the number of remote data minus 1, the algorithm generates all of the expressions that have the desired property. Reducing the depth of the **LogicTree** results in a smaller number of abstraction nodes generated and therefore a smaller DEC-MDP. This tradeoff will be further discussed in the next section. In general, the number of abstraction data generated should be no more than the number of observed data. For each path in the **LogicTree**, we record the likelihood of the abstraction data represented by the branch being true, and we choose the n abstraction data with the highest likelihood of being true to incorporate into the DEC-MDP, where $n <$ the number of observed data.

When given a BN and a desired confidence level, the agent generates an abstraction layer for each observed data value combination and adds them to its action options for the states that have the corresponding observed data values. The expanded DEC-MDP can be solved to generate a communication strategy. I call this approach the *all data action selection* approach. Since the observed data communication actions are competing with the abstraction actions, an agent chooses to transfer an abstraction data if and only if it will result in a lower expected communication cost than transferring any of the observed data. Therefore, the new system performance should be no worse than the original system in terms of the expected communication cost.

I compared the performance of the *all action selection* approach and the *raw data action selection* approach. The cost of sending a piece of abstraction data was equal to the cost of sending observed data. I used the Iterative Algorithm introduced in [75] to solve the DEC-MDP. I ran experiments on 100 problem structures with 2 high level events and 10 observed data (5 local to each agent) for different confidence levels. All of the networks were fully connected, which means that for both agents to have the complete evidence, 10 pieces of data needed to be transmitted. The corresponding curves in Figure 3.3 shows a comparison of the minimum expected communication cost generated by both systems. Column (a) in Table 3.1 shows the percent improvement in the expected communication cost when transmitting abstraction data in addition to the observed data. As shown, the *all data action selection* has a noticeable improvement over the *raw data action selection* approach. This illustrates that the addition of the abstraction data does help reduce the communication cost. The improvement is most significant when the required confidence is between 70% and 80%. When the confidence level is low, the expected communication cost is already low so that it is difficult to achieve a significant improvement. On the other hand, when the required confidence level is high, there are much fewer abstraction data generated resulting in a smaller improvement.

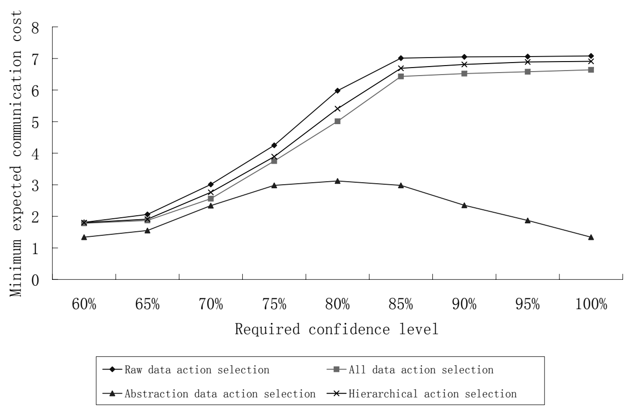


Figure 3.3. A comparison of the minimum expected communication cost given different action selections

3.2 Hierarchical Action Selection

Introducing the communication actions that transmit the values of the new abstraction data leads to both a larger state space and a larger action space for the generated DEC-MDP. As a result, the time needed to solve the new DEC-MDP can be significantly greater than the original DEC-MDP, which only has the observed data transmission as its actions. Column (c) in Table 3.1 shows the average time the *all data action selection* approach took to solve the DEC-MDP, where the average time needed for the *raw data action selection* approach equals 1.00. In this section, I introduce techniques that address this problem.

First I examine the case where the agents only transfer the abstraction data between them. I call this approach the *abstraction data action selection* approach. While the size of the DEC-MDP generated is often much smaller than that of the original DEC-MDP, one major drawback of this approach is that we can no longer guarantee that the required confidence level can be reached. Only when at least one of the abstraction data is true will the desired confidence level be reached. The corresponding curve in Figure 3.3 shows the expected communication cost achieved for the cases where the desired confidence level can be reached. When the required confidence level

required confidence	(a)	(b)	(c)	(d)
60%	1.66%	0.55%	1.25	0.89
65%	9.22%	7.28%	1.53	0.77
70%	14.95%	8.31%	1.49	0.69
75%	11.76%	8.47%	1.52	0.63
80%	16.22%	9.53%	1.61	0.59
85%	8.27%	4.56%	1.41	0.65
90%	7.52%	3.40%	1.21	0.77
95%	6.80%	2.41%	1.10	0.82
100%	6.21%	2.40%	1.09	0.87

Table 3.1. Performance compared to raw data action selection approach for single abstraction layer in a 2-10 network. (a) Expected communication cost improvement of *all action selection*. (b) Expected communication cost improvement of *hierarchical action selection*. (c) Time needed to solve the DEC-MDP for *all action selection* normalized by that for raw data action selection. (d) Time needed to solve the DEC-MDP for *hierarchical action selection* normalized by that for raw data action selection.

is high, there are fewer abstraction data generated, and therefore the expected cost is much lower than those of the two approaches I discussed in the previous section. Notice, however, that this is a false savings as the desired confidence level cannot always be reached, which is not reflected in the figure.

I seek to combine the advantages of the *all data action selection* approach and the *raw data action selection* approach so that we can save time on solving the DEC-MDP as well as guarantee the required confidence level. I achieve this by restricting legal actions for different states. Before an agent has acquired all of the abstraction data, the agents are not allowed to transfer any of the observed data. I call this approach the *hierarchical action selection* approach. An agent prefers to transfer abstraction data because it abstracts from multiple pieces of observed data and thus is a more efficient information carrier. Only when the acquisition of all of the abstraction data cannot achieve the desired confidence level will an agent start to acquire the observed data in order to get the necessary information.

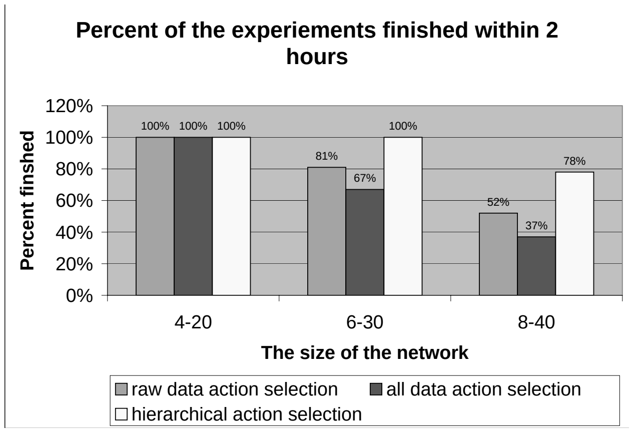


Figure 3.4. The percentage of the problems tested that each approach was able to finish solving within 2 hours.

will explore this tradeoff between computation time and communication cost as the depth is varied.

3.3 Multiple Levels of Abstraction

The idea of varying the depth of the **LogicTree** leads to an interesting way of generating a hierarchy of abstraction data. We can modify Algorithm 5 to generate a set of abstraction data of different lengths. The more literals in the abstraction data, the more abstract it is, the more efficiently it carries information, and the less refined the information is.

Just like in the single abstraction layer case, there are different ways of incorporating multiple layers of abstraction data into the action space of the DEC-MDP. I did experiments on the all data action selection approach and the hierarchical action selection approach, and compared their performance to that of the raw data action selection approach. In the all data action selection approach, all the abstraction data of various length are added to the action space to compete with the observed data. In the hierarchical action selection approach, the agents transfer the most abstract data

required confidence	(a)	(b)	(c)	(d)
60%	3.87%	2.14%	1.33	0.91
65%	10.42%	8.19%	1.49	0.79
70%	16.01%	12.34%	1.52	0.62
75%	19.84%	15.62%	1.59	0.56
80%	18.26%	13.43%	1.71	0.49
85%	12.22%	9.53%	1.59	0.58
90%	8.49%	6.72%	1.39	0.68
95%	5.71%	4.43%	1.19	0.76
100%	4.23%	3.12%	1.10	0.78

Table 3.2. Performance compared to the raw data action selection approach for multiple abstraction layers in a 6-30 network. (a) Expected communication cost improvement of *all action selection*. (b) Expected communication cost improvement of *hierarchical action selection*. (c) Time needed to solve the DEC-MDP for *all action selection* normalized by that for raw data action selection. (d) Time needed to solve the DEC-MDP for *hierarchical action selection* normalized by that for raw data action selection.

first. If the confidence level is not achieved, then the agents start transferring the next level of abstract data. This process goes on until the confidence level is reached.

I tested these approaches on networks of different sizes: 4 high level events and 20 low level data (4-20 networks); 6 high level events and 30 low level data (6-30 networks); 8 high level events and 40 low level data (8-40 networks). I generated two abstraction layers for each of the networks. Figure 3.4 shows the percentage of the networks I tested on that each of the three approaches was able to finish within two hours. It shows that hierarchical action selection approach was able to finish the policy search better than either of the two other approaches. The larger the network, the bigger an advantage the hierarchical approach has over the other two. Table 3.2 shows the performance comparison of the all data action selection approach and hierarchical action selection approach over the raw data action selection approach. The data is shown only for the 6-30 networks for which all of the three approaches finished searching for the optimal policy. The first abstraction layer is composed of the abstraction data with no more than 5 literals, and the second layer is composed of the abstraction data with no more than 10 literals and greater than 5 literals. The results show a similar pattern as those of the single abstraction layer experiments. The all data approach is not practical for large networks or more than a few abstraction layers, since it substantially expands both of the state space and action space of the DEC-MDP. However, it does give a lower expected communication cost than the raw data action selection approach when it can solve the problem. On the other hand, the hierarchical approach proves to be more effective than both the raw data action selection approach and the all data action selection approach. It is able to solve more and larger networks, and on average it takes much less time. Furthermore, for the networks that both approaches were able to finish the policy search, the solution generated by the hierarchical action selection approach has a lower expected communication cost on average than that generated by the raw data action selection approach.

3.4 More than Two Agents

This section extends the techniques of transferring abstraction data discussed so far to more than two agents. In order to extend the abstraction generation algorithm, we need to answer the following question first: Do we restrict each abstraction to data from one single agent or do we allow it to span across data from several agents? If we only allow abstraction to have data from one single agent, then we run the same abstraction generation algorithm for every neighbor of the current agent and the abstraction generated can be directly put to the action space for the DEC-MDP as a possible request from that neighbor. However, this approach can be limiting in the abstraction that is allowed to be generated, especially if we do not allow less greedy abstraction (i.e., the abstractions that only increase a certain amount of confidence instead of enable an immediate realization of the required confidence).

If we do allow a piece of abstraction to span across several agents, then when running the abstraction generation algorithm for a specific agent, all the other agents are seen as a virtual agent. Every remote common data that does not belong to the current agent is a part of the DataList. The algorithm remains unchanged, though there are two points worth noting. First, the observed data in DataList needs to be grouped by the agents they belong to. The reason is that in the abstraction data generated, if the data from each agent is grouped together, then the portion of the abstraction can be isolated from the rest and request from the agent as one entity regardless of the rest of the abstraction. For example, suppose we have four agents, $A_1, \ldots, A_4$. Each of them have five pieces of data. A valid abstraction generated for agent A_4 would be $(D_1 \wedge D_2) \vee (\not{D}_7 \wedge D_9) \wedge (D_{13} \vee D_{15})$, while $D_1 \wedge D_2 \vee D_7 \wedge D_5 \vee D_{15} \wedge D_9$ is not valid. For the same reason, I bring up the second point. When calculating the confidence given the currentLeaf, the data along the path are grouped together by parentheses according to the agents they belong, like the example I have just shown.

There are two ways for an agent to request a piece of abstraction that consists of data from multiple agents. In the first approach, requesting the abstraction is seen as a single action. The agent breaks the abstraction down to parts belonging to different agents and sends a request message to all the neighbors whose data is a part of the abstraction. The neighbors then send their part of the abstraction back as response to the request. The cost of this REQUEST action includes the cost for all the request messages and that of every response message. I call this approach *single request approach.*

In the second approach, requesting the abstraction that involves multiple agents is seen as multiple actions. I call it *multiple request approach.* In the previous example, agent A_4 is requesting the abstraction $(D_1 \wedge D_2) \vee (\not{D}_7 \wedge D_9) \wedge (D_{13} \vee D_{15})$ from the other agents. In the single request approach, A_4 will request $(D_1 \wedge D_2)$ from A_1, $(\not{D}_7 \wedge D_9)$ from A_2, and $(D_{13} \vee D_{15})$ from A_3 simultaneously, and it is seen as one

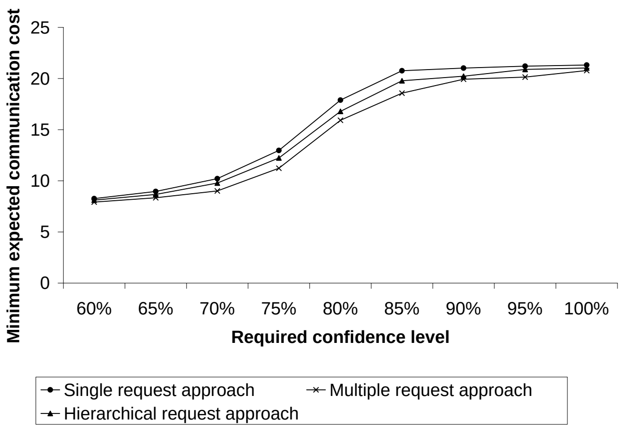


Figure 3.5. A comparison of the minimum expected communication cost given different abstraction action request approaches

single action in the DEC-MDP. In contrast, the multiple approach will see them as three separate actions in the planning space. The difference is that in the multiple request approach, A_4 can choose to only request part of the abstraction at one time and request the other parts at a later time or even never again. The reason why the multiple request approach may generate plans with lower expected communication cost is that sometimes it may be beneficial to only acquire part of the abstraction instead of the entire abstraction all at once. There are two situations that this is the case. First, acquiring part of the abstraction may be enough for the agent to deduce the value of the entire abstraction and therefore there is no need to acquire the rest. In the second situation, even though the acquired part does not tell the value of the entire abstraction, it does give enough information such that acquiring or sending information other than the rest of the abstraction has a higher expected value than immediately acquiring the rest of the abstraction.

I ran experiments on 50 problem structures with 3 agents. Each agent has 1 local high level event and 5 observed data. All of the networks were fully connected. The corresponding curves in Figure 3.5 show a comparison of the minimum expected communication cost generated by the single request approach and the multiple request

required confidence	(a)	(b)	(c)	(d)
60%	4.12%	1.82%	1.21	1.01
65%	6.92%	3.24%	1.61	1.02
70%	11.75%	4.21%	1.78	1.01
75%	13.42%	5.71%	1.84	1.01
80%	11.01%	6.15%	1.89	1.02
85%	10.55%	4.72%	1.81	1.02
90%	5.19%	3.76%	1.75	1.01
95%	5.04%	1.51%	1.59	1.01
100%	2.53%	1.36%	1.23	1.01

Table 3.3. Performance compared to single request approach for transferring abstraction data that may span multiple agents in a 3-15 network. (a) Expected communication cost improvement of *multiple request approach.* (b) Expected communication cost improvement of *hierarchical request approach.* (c) Time needed to solve the DEC-MDP for *multiple request approach* normalized by that for single request approach. (d) Time needed to solve the DEC-MDP for *hierarchical request approach* normalized by that for single request approach.

approach. Column (a) in Table 3.3 shows the percent improvement in the expected communication cost when using multiple request approach over the single request approach. Column (c) shows the average time the multiple request approach needed to solve the DEC-MDP normalized by the time needed for the single request approach. As shown, the multiple request approach has a noticeable improvement over the single request approach. However, since the action space action space of the multiple request approach is considerably larger than that of the single request approach, it takes significantly longer time to solve.

The next approach I introduce, i.e., the *hierarchical request approach*, finds a middle ground between the previous two approaches. Like in the single request approach, when first solving the DEC-MDP, the abstraction request action is seen as one single action. However, when an agent needs to execute the abstraction request action, it breaks it down to multiple actions, just like in the multiple request approach. A small MDP with the new request actions as its action space is then solved to decide whether and in what order to request the individual parts of the abstraction from their sources. Since the hierarchical request approach does not allow transferring other data between transferring different parts of the same abstraction, it does not account for the second situation where multiple request approach is more advantageous than the single request approach. However, it does take the first situation into account and combines the strengths of both approaches together. Figure 3.5 and Table 3.3 (b) show that while it does not perform as well as the multiple request approach, it outperforms the single request approach in terms of the expected minimum communication cost achieved. On the other hand, Table 3.3 (d) shows that the hierarchical request approach takes only slightly longer than the single request approach to solve the DEC-MDP and that it is much faster than the multiple request approach.

3.5 Conclusions

In this chapter I investigated the techniques of transferring abstraction data in addition to observed data in Distributed Bayesian Networks to reduce the required communication cost. I introduced an algorithm that automatically generates appropriate abstraction data that facilitates the achievement of the required confidence level and reduces the necessary communication cost. I also discussed approaches to incorporate the new abstraction data into the DEC-MDP framework effectively. Both the improvement in the minimum expected communication cost and the time savings in solving the DEC-MDP make the *hierarchical action selection* an attractive approach, especially for the systems which require a mid-ranged confidence level. I further extended the different action selection approaches to larger networks and multiple abstraction layers. The hierarchical action selection approach was shown to be able to solve problems with larger networks than the other approaches. When extending the problems to more than two agents, different ways to transfer abstractions that span multiple agents were examined along with their advantages and disadvantages. The hierarchical request approach proves to be a reasonable tradeoff between the solution quality and time needed to solve the problem.

This work allows us to look at the use of abstraction to reduce communication cost from a formal perspective. The *hierarchical action selection* approach defines when and how the abstraction data and the observed data should be transferred in the communication process. The addition of transferring abstraction data to the communication actions can be extended to reduce the communication cost of other Cooperative Distributed Problem Solving (CDPS) applications as well. Though the algorithm that I used to generate the abstraction data is specific to the DBN used to represent the DI problems, given a CDPS problem with specific semantics, there often is abstraction data that can be generated and used to transfer information more efficiently. The hierarchical action selection approach then can be used to incorporate this abstraction data to further minimize the communication cost.

I predict that the savings of the *hierarchical action selection* approach shown in this paper will be more significant for larger networks. The influence of different depths of **LogicTree** in Algorithm 5 on the system performance will also be investigated. In this work, I looked at one particular type of abstraction. There are other forms of abstraction that can be potentially useful. For example, some abstraction data may contain useful information but is not sufficient to reach the confidence by itself. Other abstraction may be an intermediate interpretation result that can be incorporated into the local data. I plan to look into these different types of abstraction in future work. I also intend to investigate other techniques in generating multiple layers of abstraction data, such as different levels of intermediate interpretation results.

CHAPTER 4

AGENT INTERACTION IN DISTRIBUTED POMDPS AND ITS IMPLICATIONS ON COMPLEXITY

The objective of this thesis work is to formally study the communication management in Cooperative Distributed Problem Solving applications. While communication management remains a key research problem in the Multi-Agent Systems (MAS) community, the type of communication problem I study is different from previous research. Though several general decision theoretic models, i.e., distributed POMDP models, have been proposed [69, 29], most research in this field focuses on the type of communication that synchronizes the agents' partial views of the global state and has a constant message cost, [82, 81, 28, 3]. Very little research has been done studying transferring only partial observation history between agents. However, some researchers have studied the case where the agents send their policies or actions to each other [45, 27]. In a sense, exchanging actions or policies between the agents accomplishes the same goal as transferring partial observation history. Both of them change each agent's belief of the global state and/or the other agent's local view of the global state without completely synchronizing the global views. The type of DEC-MDPs I study in this work deals with general communication where the agents choose to transfer part of its own observation and the messages have different costs depending on their sizes and contents. One objective of this chapter is to understand how my work relates to the rest of the field.

Another difference between the way I model communication in DEC-MDP and previous study of communication using DEC-MDP framework is that previously communication actions are mostly modeled as special actions that are different from other actions and therefore frameworks were proposed to extend DEC-MDP to explicitly represent communication actions. In contrast, this work models communication actions directly with DEC-MDP framework. Though there are several advantages to explicitly distinguish between communication actions and other types of actions, this distinction is artificial. Consider the type of actions of an agent that affects the other agents' view of the world and their decision making without explicit communication. Such actions incur interactions among the agents and transfer information without "communication actions" in the common sense. If we treat both types of interactions among the agents similarly, then the DEC-MDP framework itself is general enough to model them both.

Distributed POMDP frameworks assume that the agents have partial or noisy observations about the current global state, and that communication between the agents may have a cost. This partial observability of the global state leads to an increased complexity of NEXP-complete [5, 69]. However, two subclasses of distributed POMDPs have been identified whose complexity is much more tractable than the general case. First is the Transition Independent Decentralized MDP (TI-DEC-MDP), which is a class of distributed POMDPs where the agents interact with each other only through a global reward function while their local transitions and observations remain independent [4]. The other class is the TI-DEC-MDP extended with Synchronizing Communication, in which the agents are allowed to communicate but when they do they are required to transmit sufficient information such that all agents know the global state [28, 3]. Both subclasses have been proven to be NP-complete [29, 4]. The relationship between the complexity class of a DEC-MDP and the amount of interaction (whether explicit or implicit) between agents remains an open question. If we can identify the type of interaction that allows the DEC-MDP to remain at a lower complexity class, then it may enable us to design appropriate communication actions for a more tractable solution.

The focus of this chapter is to quantify the characteristics of a class of multi-agent coordination problems that determines its complexity. The key result is that the complexity of the problem depends on the amount of important information each agent has about the other agents, and whether this information can be represented in a succinct way. Information is important if knowing it could allow the agents to achieve a higher expected reward, and by succinct I mean that the set of all possible important information the agents could acquire is polynomial in the number of states. I prove that this criteria is both sufficient and necessary for the class of problems to be NP-complete. I illustrate this idea with two examples from the literature and prove that both have this property.

My goal in this chapter is not to introduce new models or algorithms, but to change the way people view interactions between agents in the context of distributed POMDPs. Multi-agent researchers have long intuitively understood that the interaction between the agents is the cause of their high complexity. The theoretical results of this paper are significant in that they both formally justify this intuition as well as explain how the interaction affects the complexity. This new understanding of interaction and its relationship to complexity will help us to identify new classes of multi-agent systems with a lower complexity.

The significance of this theoretical result also has a more practical side. Most multi-agent systems are provably harder than NP and solving them optimally is very difficult. Much work has been put into developing good algorithms for approximating these problems. This work provides theoretical guidance in understanding how the approximations in a model limit the search space and reduce the complexity. I demonstrate this on two problems that do not meet our condition by providing two approximations that impose additional restrictions on the interactions among the agents and reduce the complexity to no harder than NP.

This chapter first gives an overview of various distributed POMDP models including those with explicit communication. I prove the equivalence of these different models and present the related complexity results. Then I discuss different types of interactions between agents and present the condition that determines the complexity of a distributed MDP. Two examples of NP-complete interaction protocols are discussed as well as two examples of how harder problems can be approximated.

4.1 A Review of the Decision Theoretic Models

In the context of single agent control and learning, Markov Decision Processes (MDPs) have been widely used to model sequential decision making under uncertainty [68, 7]. The basic idea is to describe the problem as a finite set of distinct states. Each state is fully observable, i.e., it includes all the necessary information for the agent to make the best possible decision. The optimal action is the action that maximizes the agent's expected cumulative reward. This is a P-Complete problem for both the finite horizon and the discounted infinite horizon cases [62].

While the MDP can handle a large variety of single agent problems, the Partially Observable MDP (POMDP) is suitable for situations where the states cannot be fully observed [55, 48, 15]. The basic idea is that at each step, the agent makes an observation that changes the belief the agent has about its current state. The agent's policy is a mapping from a sequence of observations to an action. This introduces an exponential increase in the size of the policy, and therefore is a much harder problem to solve. The finite horizon POMDP is PSPACE-complete [62, 56] and the infinite horizon case is undecidable [50].

Recently, multi-agent coordination problems have started to receive more attention from the decision theory community. Traditionally, multi-agent decision problems are modeled directly using POMDPs by focusing on individual agent's decision making and viewing the rest of the agents as a part of the partially observable environment. This approach ignores the fact that all the agents have their own states, observations and action decisions. Furthermore, the actions of the agents interact with each other and have impacts on the observations and rewards received by the other agents. This is very different from the passiveness of an environment. Therefore, models specifically designed for multi-agent decision problems are needed.

The first decision theoretic model that explicitly represent multiple agents is the Multi-agent Markov Decision Process (MMDP) [8]. It is a centralized multi-agent extension of an MDP. The only change is to factor the action space into actions for each agent. Each agent observes the global state directly and takes an joint action. The MMDP is equivalent to the MDP, and therefore it has the same polynomial complexity. However the action space of the MMDP is exponential in the number of agents. Other researchers have also used this view of multi-agent systems. Guestrin et al. focused on the interactions between agents through the reward focus and their observation model resembles the full observability of the MMDP [30, 31]. Ghavamzadeh and Mahadevan [26] developed a reinforcement learning algorithm for the fully observable multi-agent setting.

The major disadvantage of the MMDP model is that it assumes that the agents can fully observe the global state. This assumption is often not the case for multi-agent systems. Instead, each agent often has a different partial view of the environment. Only when each agent can freely communicate its observations with every other agent at every time step can most multi-agent systems be mapped into an MMDP. Xuan and Lesser [81] proposes a decentralized extension to the MDP, in which an agent has its local state in addition to the global state. When an agent has ambiguity over which action it should take based on its local state, it needs to communicate with the other agent to synchronize the global view. As a variation, Ooi and Wornell [59] studied the case where all the agents synchronize their views of the global state every k time steps. The dynamic programming algorithm developed for such a system is doubly exponential in terms of k. Hsu and Marcus [36] introduced a more tractable algorithm but assumed that the agents share state information every time step. The common problem with these models is that they all require the agents to share state information periodically without considering the communication cost associated with such state synchronization.

Several more general multi-agent decision theoretic models have been independently developed to overcome this problem, among which are the Partially Observable Identical Payoff Stochastic Game (POIPSG) [66] and Decentralized Partially Observable Markov Decision Processes (DEC-POMDP) [5]. These models explicitly represent the partial observability of the world state for each agent. Take DEC-POMDP for example. Like the MMDP, the action space is factored into actions for each agent. Moreover, the observations are factored into separate observations for each agent as well, and each agent observes only a potentially different part of the global state. This is a key difference between POMDP and DEC-POMDP and leads to the increased complexity class of the DEC-POMDP. In the DEC-POMDP, each agent maintains the belief about the current global state just as in the POMDP case. Additionally, it also maintains the belief about the beliefs of the other agents in the system. This belief of beliefs gives the problem a complexity of NEXP-complete [5]. Similar to the fact that a POMDP can be reduced to an equivalent MDP with an exponential increase in the state space, a DEC-POMDP can be reduced to an MDP with a doubly exponential increase in the state space.

Bernstein et al. also formalized a variation of the DEC-POMDP known as the Decentralized Markov Decision Process (DEC-MDP) [5]. The difference between the two models is whether the system has joint full observability (as in DEC-MDP) or joint partial observability (as in DEC-POMDP), i.e., whether combining the observations of all the agents can fully determine the current global state. This distinction turns out to be fairly minor. One can convert a DEC-MDP to an equivalent DEC-POMDP by adding a random variable to every state that no agent observes, but that does not have any effect on the problem. This makes the problem now only joint partially observable. Similarly, one can convert an arbitrary DEC-POMDP into an equivalent DEC-MDP by adding an all-seeing agent that observes everything and does nothing. This means that these two models are equivalent to each other, and for the rest of the proposal, I will only discuss in terms of DEC-MDP.

The formal definition of DEC-MDP was given in Chapter 2. It is repeated here for ease of reference.

Definition 11 (adapted from [5]). *A* **decentralized MDP** *is a tuple* $\langle S, A_i, A_j, P, R, \Omega_1, \Omega_2, O \rangle$*, where*

- *S is a finite set of global states, with distinguished initial state s_0.*
- *$A = A_i \times A_j$ is a finite set of joint actions. A_i are the local actions for agent i.*
- *$P : S \times A \times S \to \Re$ is the transition function. $P(s'|s, a_i, a_j)$ is the probability of the outcome state s' when the joint action (a_i, a_j) is taken in state s.*
- *$R : S \times A \times S \to \Re$ is the reward function. $R(s, a_i, a_j, s')$ is the reward obtained from taking joint action (a_i, a_j) in state s and transitioning to state s'.*
- *$\Omega = \Omega_i \times \Omega_j$ is a finite set of joint observations. Ω_i is the set of observations for agent i.*
- *$O : S \times A \times S \times \Omega \to \Re$ is the observation function. $O(s, a_i, a_j, s', o_i, o_j)$ is the probability of agents i and j seeing observations o_i and o_j after the sequence s, (a_i, a_j), s' occurs.*
- *Joint full observability: the pair of observations made by the agents together fully determine the current state. If $O(s, a_i, a_j, s', o_i, o_j) > 0$ then $P(s'|o_i, o_j) = 1$.*

Definition 12. *A* **local policy** $\pi_i : \overline{\Omega_i} \rightarrow A_i$ *for agent* A_i *is a mapping from local histories of observations* $\overline{o}_i$ *to actions in* A_i*. A* **joint policy** $\pi = \langle \pi_1, \pi_2 \rangle$ *is a pair of local policies, one for each agent. The goal for a DEC-MDP is to find a joint policy* $\pi = \langle \pi_i, \pi_j \rangle$ *that maximizes the expected value.*

The basic idea of the definition is that the state, now known as the global state, encompasses all of the necessary information about the state of each agent (local state) and the environment. Each agent has its own set of actions it takes and observations it makes. The transition function is from world states to world states given the actions taken by each agent, the joint action. Bernstein et al. proved the following complexity result:

Theorem 6 ([5])**.** *Deciding a DEC-MDP is NEXP-complete.*

The reason that the DEC-MDP is harder than the POMDP is that the agents are all receiving different observations, and only together could they identify the current global state. To predict what action agent i will take requires agent j to maintain not only a belief about the global state (similar to a POMDP) but also a belief about the belief agent i has. Interactions between the agents are built into the transition and observation functions. The likelihood that agent i will see a particular observation depends directly on the actions of both agents as well as the new global state, which in turn depends on the actions of both agents.

The DEC-MDP stops there in its definition. However, two formal models have been proposed to explicitly represent another form of interaction, explicit communication, between agents: COM-MTDP [69] and DEC-MDP-Com [28]. They make explicit distinction between a domain level action and an explicit communication action. Here I give a formal definition of DEC-MDP-Com because it uses the same notation system as DEC-MDP introduced before and COM-MTDP is equivalent to DEC-MDP-Com.

Definition 13 (adapted from [29]). *A DEC-MDP-Com is a tuple:* $\langle S, A, \Sigma, C_\Sigma, P, R, \Omega, O\rangle$, *where:*

- S *is a finite set of global states, with a distinguished initial state* s_0.
- $A = A_i \times A_j$ *is a finite set of joint actions.* A_i *are the local actions for agent* i.
- $\Sigma = \Sigma_i \times \Sigma_j$ *is the alphabet of messages.* $\sigma_i \in \Sigma_i$ *denotes an atomic message sent by agent* i. $\overline{\sigma}_i$ *is a sequence of atomic messages sent by agent* i. $\epsilon_\sigma \in \Sigma_i$ *denotes not sending a message to the other agent.*
- $C_\Sigma \to \Re$ *is the cost of transmitting an atomic message.*
- $P : S \times A \times S \to \Re$ *is the transition function.*

  $P(s'|s, a_i, a_j)$ *is the probability of the outcome state* s' *when the joint action* (a_i, a_j) *is taken in state* s.
- $R : S \times A \times S \to \Re$ *is the reward function.* $R(s, a_i, a_j, s')$ *is the reward obtained from taking joint action* (a_i, a_j) *in state* s *and transitioning to state* s'.
- $\Omega = \Omega_i \times \Omega_j$ *is a finite set of joint observations.* Ω_i *is the set of observations for agent* i.
- $O : S \times A \times S \times \Omega \to \Re$ *is the observation function.* $O(s, a_i, a_j, s', o_i, o_j)$ *is the probability of agents* i *and* j *seeing observations* o_i *and* o_j *after the sequence* s, (a_i, a_j), s' *occurs.*
- *Joint full observability: the pair of observations made by the agents together fully determine the current state. If* $O(s, a_i, a_j, s', o_i, o_j) > 0$ *then* $P(s'|o_i, o_j) = 1$.

The policy for an agent is a mapping from all of the available information to a domain action and a communication action. Even though the global state is Markov, the agents base their decisions on the history of observations just like in a POMDP.

However, if the agents ever communicate such that they both know the global state, i.e., they exchange their most recent observation, then they do not need to remember the prior history due to the Markov property. In effect, this synchronization resets the problem but with a new starting state.

Definition 14. *The local policy for agent i, π_i, is a mapping from the history of observations $\overline{\Omega}_i$, the history of messages sent $\overline{\Sigma}_i$, and the history of messages received $\overline{\Sigma}_j$ since the last synchronized world state S to a domain action and a communication action.*

$$\pi_i : S \times \overline{\Omega}_i \times \overline{\Sigma}_i \times \overline{\Sigma}_j \longrightarrow A_i \times \Sigma_i.$$

The goal for a DEC-MDP-Com is to find a joint policy $\pi = \langle \pi_i, \pi_j \rangle$ that maximizes the expected value.

Definition 14 is different from the original definitions of the local policies of a DEC-MDP-Com in [28, 29]. The policies defined here are a mapping from the complete message exchange history, which includes the messages sent as well as those received, instead of from only the messages received.

An important question is whether or not the communication explicitly represented in the DEC-MDP-Com increases the expressiveness of the model. It turns out that it does not – the communication actions are just special types of domain actions and the message received are just special types of observations.

Theorem 7. *DEC-MDP-Com is equivalent to DEC-MDP.*

Proof.

- **DEC-MDP $\leq_p$ DEC-MDP-Com.**

This reduction is trivial. To any DEC-MDP, we add $\Sigma = C_\Sigma = \emptyset$ and we get an equivalent DEC-MDP-Com.

- **DEC-MDP-Com $\leq_p$ DEC-MDP.**

We reduce a DEC-MDP-Com $\langle S, A, \Sigma, C_\Sigma, P, R, \Omega, O\rangle$ to an equivalent DEC-MDP $\langle \hat{S}, \hat{A}, \hat{P}, \hat{R}, \hat{\Omega}, \hat{O}\rangle$.

The basic idea is to introduce the two step process of the DEC-MDP-Com into the DEC-MDP by doubling the state space: $\hat{S} = S \times \{0, 1\}$. The states $\hat{s} = [s, 0]$ are for taking domain actions A_i and receiving observations Ω_i. The states $\hat{s} = [s, 1]$ are for taking communication actions Σ_i and receiving communications Σ_j. The total space of actions is therefore $\hat{A}_i = A_i \cup \Sigma_i$. The observations that agent i receives include both the messages sent by agent i and the messages received from agent j, i.e., $\hat{\Omega}_i = \Omega_i \times \Sigma_i \times \Sigma_j$. When taking domain actions nothing changes in the functions $\hat{P}$, $\hat{R}$ and $\hat{O}$:

$$\hat{P}([s,0], a_1, a_2, [s',1]) = P(s, a_1, a_2, s').$$
$$\hat{R}([s,0], a_1, a_2, [s',1]) = R(s, a_1, a_2, s').$$
$$\hat{O}([s,0], a_1, a_2, [s',1], o_1, o_2) = O(s, a_1, a_2, s', o_1, o_2).$$

When taking the communication actions, they do change:

$$\hat{P}([s,1], \sigma_1, \sigma_2, [s,0]) = 1.$$
$$\hat{R}([s,1], \sigma_1, \sigma_2, [s,0]) = C_\Sigma(\sigma_1) + C_\Sigma(\sigma_2).$$
$$\hat{O}([s,1], \sigma_1, \sigma_2, [s,0], \sigma_1\sigma_2, \sigma_2\sigma_1) = 1.$$

Therefore, the DEC-MDP-Com is equivalent to the DEC-MDP. □

Theorem 7 states that DEC-MDP-Com and DEC-MDP have the same expressiveness. However, the distinction between the communication actions and the domain actions can be very useful, as I will show in the next section.

4.2 Polynomially Encodable Interactions

The DEC-MDP-Com and related models allow for a very general form of interaction between the agents. The complexity for those problems has been proven to be NEXP-complete [5, 29]. At the other end of the spectrum would be to disallow all interactions between the agents. In effect, each agent would be independently solving a local MDP that represents its part of the system. MDPs are P-complete, so there is something about the interactions between the agents which is the cause of the complexity. As additional evidence, Becker et al [4] defined a class of multi-agent problems in which the agents were almost completely independent. Each agent had a local MDP that described its part of the system. The agents could not communicate in any way nor could they take an action that would influence another agent. However, the system received a reward that depended on the local states and actions of all of the agents, and the goal was to find a joint policy that maximized the sum of the expected local and global rewards. This class of problems in which the agents can only interact through the reward function proved to be NP-complete. Goldman and Zilberstein [29] also showed that by following certain communication protocols the agents from the previous example could communicate with each other and the problem remained at NP-complete.

This section will examine different types of interactions between agents and provide theoretical results to explain how and why the interactions affect the complexity of finding optimal solutions. The next section will elaborate on the two NP-complete examples introduced above and prove that they meet this condition.

I classify the actions agents can take into two groups: non-interacting (or independent) actions and interacting (or dependent) actions. Independent actions are those that do not affect the other agent and neither agent receives any information about the other. Dependent actions are those that affect the other agent in some way. For example, robot i could pick up robot j and move it, which would affect the local

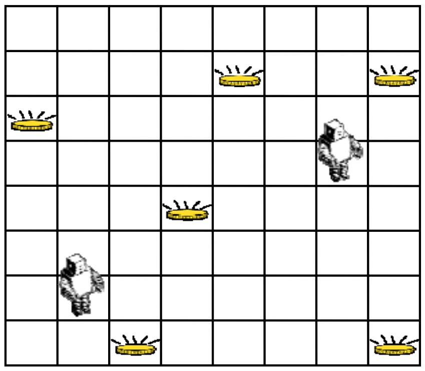


Figure 4.1. The token collecting example

state of robot j. Communication is another example: agent i could send a message to agent j, which would change the knowledge agent j has. We can further subdivide dependent actions into explicit and implicit communication. Normally when one thinks about communication, i.e., sending a message, one is talking about explicit communication. This is the communication part of the DEC-MDP-Com. Implicit communication is the information an agent receives by a domain action, like the example of a robot picking up and moving another robot. The robot being picked up gains information about the local state and belief of the other robot through interaction with it, namely the location of the other robot and the fact that the other robot felt this was a useful action to take.

I will illustrate these interactions with a token collecting example (Figure 4.1). There is a $n \times n$ grid world, which is populated by two agents and a number of tokens. The agents can observe their own locations and the locations of the tokens. When an agent picks up a token, the system gets a positive reward. The goal of the system is to maximize the total reward within time T. This problem can be modeled as a DEC-MDP. The world state includes the locations of both agents, the locations of the tokens and the time left. The agents' observations at each time step include the agents' own location, the location of the tokens, and the time left. At every time step, each agent can either move to an adjacent square or pick up a token at its current location. If an agent moves, its action does not affect the other agent's observations, and therefore the movement actions are independent actions. However, if agent i picks up a token, agent j can observe the fact that one of the tokens just disappeared. By comparing its current observation and the observation at the last time step, agent j can infer the exact location of agent i and therefore has the complete knowledge of the current world state. As a result, the token collecting action is a dependent action even though there is no explicit communication in the system.

I defined a dependent action as an action that affects the observations of the other agent and therefore changes its belief about the global state. If a dependent action is explicitly modeled in a transition and observation independent DEC-MDP-Com, its effect is recorded by the communication action σ_i itself. On the other hand, if it is not explicitly modeled in a DEC-MDP-Com, its effect is recorded by the observations of the agents. The observation history $\overline{\Omega}_i$ records the interaction history of agent i in the DEC-MDP, including all the actions taken, i.e., $\overline{A}$. Consequently, in a DEC-MDP-Com where there are communication actions explicitly modeled, the interaction history of agent i is $\overline{\Omega}_i \times \overline{\Sigma}_i \times \overline{\Sigma}_j$.

Definition 15. *I call E_i an* **encoding** *of the interaction history of agent i, if a joint policy $\tilde{\pi} = \langle \tilde{\pi}_1, \tilde{\pi}_2 \rangle$ is sufficient to maximize the global value, where $\tilde{\pi}_i$ is of the form $S \times E_i \rightarrow A_i \times \Sigma_i$ for a DEC-MDP-Com, or of the form $E_i \rightarrow A_i$ for a DEC-MDP.*

The encoding represents removing all elements from the interaction history that are unnecessary to generate an optimal policy. Please note that the encoding at any time in the history needs to be able to incorporate the information contained in the encoding of the previous moment plus the new observation at the current time. The important characteristic is the size of the smallest encoding. The interaction history is normally considered to be exponential in $|S|$ because the length of the observation sequence is $O(|S|)$. In some problems, however, the smallest encoding is only polynomial in $|S|$.

Definition 16. *The interaction history of a subclass of DEC-MDP/ DEC-MDP-Com is* **polynomially encodable** *if there exists an encoding E_i for each interaction history $\overline{\Omega}_i$ and a constant c_i, such that $|E_i| = O(|S|^{c_i})$.*

The criteria that determines the complexity of a multi-agent system is whether the interaction history can be polynomially encoded. If it can, then the problem is in NP. If it cannot be polynomially encoded, then it is provably harder than NP. The following two theorems prove this relationship between the encoding and the complexity.

Theorem 8. *Deciding a polynomially encodable DEC-MDP/DEC-MDP-Com is NP-complete.*

Proof. Here I prove the DEC-MDP case, the DEC-MDP-Com is essentially the same.

To prove NP-completeness, I (1) provide a polynomial time verifier and (2) show a reduction from an NP-Complete problem to this one.

(1) A joint policy in a DEC-MDP can be evaluated by representing it as a belief state graph. Each node in the graph is composed of the state, the sequence of observations for agent i and for agent j. Each node has a single joint action, which is defined by the joint policy. The transitions between the nodes depends only on the transition and observation functions, and each transition has a reward defined by the reward function. The belief state graph can be evaluated using the standard MDP recursive value function and policy evaluation, which runs in time polynomial in the size of the graph. For a DEC-MDP, this size is $|S| \times |\Omega_i|^T \times |\Omega_j|^T$, where $T = O(|S|)$. However, since there exists a polynomial encoding E_i for each observation sequence $\overline{\Omega}_i$, the size of the graph is only $|S| \times |E_i| \times |E_j|$ and the policy evaluation takes

$O((|S|^{c_i+c_j+1})^c)$, which is polynomial in the size of the state space for constants c_i, c_j and c.

(2) To prove the lower bound I will reduce the NP-complete problem DTEAM [60, 61] to this problem. DTEAM is a single-step discrete team decision problem with two agents. Agent i observes a random integer k_i, $1 \leq k_i \leq N$, and takes an action $\gamma_i(k_i) \in \{1, \dots, M\}$. Their actions incur cost $c(k_i, k_j, \gamma_i(k_i), \gamma_j(k_j))$. The problem is to find policies γ_i and γ_j that minimize the expected cost:

$$\sum_{k_i=1}^{N} \sum_{k_j=1}^{N} c(k_i, k_j, \gamma_i(k_i), \gamma_j(k_j)).$$

The reduction is quite straightforward. In the initial state, the agents take a null action and transition to one of N^2 intermediate states that correspond to the random integers k_i and k_j. Agent i observes k_i and takes its action to reach the final state. The reward is negative the cost, $R(\cdot) = -c(\cdot)$. The size of the observation sequence $|\overline{\Omega}_i| = N = O(|S|)$ is polynomial in the size of the state space.

Given the polynomial time verifier and the reduction, a DEC-MDP with a polynomially encodable interaction history is NP-complete. $\square$

Theorem 9. *Deciding a non-polynomially encodable DEC-MDP/DEC-MDP-Com is harder than NP.*

Proof. I prove this by contradiction. Assume that an arbitrary DEC-MDP without a polynomial encoding is in NP. This means that there exists a polynomial time verifier for any policy in the DEC-MDP. If a policy can be verified in polynomial time then it must have a polynomial sized representation. Since a policy is a mapping from $E_i \rightarrow A_i$, this polynomial sized representation is the encoding of the interaction history. Contradiction. Therefore, the DEC-MDP without a polynomial encoding is not in NP.

The proof for DEC-MDP-Com is similar. $\square$

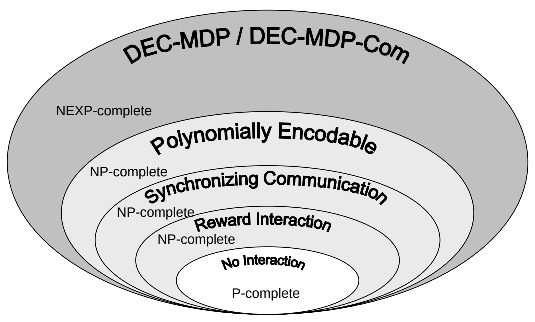


Figure 4.2. The relationships and complexity between various distributed MDP models.

Figure 4.2 illustrates the relationship and complexity between the models discussed in this paper. Polynomially Encodable interaction histories are an NP-complete subset of DEC-MDPs. Other models, like the synchronizing communication are NP-complete subsets of polynomially encodable problems. No interaction between the agents is a P-complete class of problems.

4.3 Examples of Protocols

In this section, I present two interaction protocols known to be NP-complete, and demonstrate how to prove the existence of a polynomial encodings.

4.3.1 Reward Dependence

The first example is a DEC-MDP in which the agents are mostly independent of each other. All of their actions are independent actions. The dependence between the agents comes from the reward function which depends on the world state and joint action. Becker et al. [4] formally defined this class of problems as a Transition Independent DEC-MDP (TI-DEC-MDP). It is a DEC-MDP with a factored state space, which means that there is a local state for each agent and the global state is the product of all of the local states, $S = S_i \times S_j$. The transition from one local state to the next depends only on the actions of that agent. Similarly, the observations of agent i depends only on i's local states and actions. We call these properties transition and observation independent.

Definition 17 ([4]). *A factored DEC-MDP is said to be* **transition independent** *if the new local state of each agent depends only on its previous local state and the action taken by that agent:*

$$P(s_i'|(s_i, s_j), (a_i, a_j), s_j') = P_i(s_i'|s_i, a_i).$$

Definition 18 ([4]). *A factored DEC-MDP is said to be* **observation independent** *if the observation an agent sees depends only on that agent's current and next local state and current action:* $\forall o_i \in \Omega_i$

$$P(o_i|(s_i, s_j), (a_i, a_j), (s_i', s_j'), o_j) = P(o_i|s_i, a_i, s_i').$$

At each step each agent fully observes its local state, and observes no information about the local state of the other agent.

Definition 19 ([4]). *A factored DEC-MDP is said to be* **locally fully observable** *if each agent fully observes its own local state at each step., i.e.,* $\forall o_i \; \exists s_i \; : P(s_i|o_i) = 1$.

The interaction history of the DEC-MDP is the sequence of local observations, which in this problem translates into the sequence of local states. However, due to the limited interaction, the most recent local state and the time step is sufficient to maximize the global value. Proving that this encoding is sufficient is the primary component in proving that this class of problems is polynomially encodable.

Theorem 10. *The interaction history of a DEC-MDP with independent transitions and independent observations is polynomially encodable.*

Proof. The interaction history of a transition independent and observation independent DEC-MDP is a sequence of local states $\overline{s}_i$ with an upper bound on the length being $T = O(|S|)$. I will prove that there exists an encoding of the interaction history composed of the last state in the sequence and the length of the sequence. This encoding $E_i = S_i \times T$ is polynomial: $|S_i \times T| = O(|S|^2)$.

The Q value of the belief state graph (see Theorem 8) is the Q value of taking the given action in the current state sequence and then following the given optimal policy $\overline{\pi}^* = \langle \overline{\pi}_1^*, \overline{\pi}_2^* \rangle$, where $\overline{\pi}_i^* : \overline{S}_i \rightarrow A_i$.

$$Q_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) = \sum_{s_1' s_2'} P(s_1' s_2' | \overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) \times$$
$$[R(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2, s_1' s_2') + \gamma V_{\overline{\pi}^*}(\overline{s}_1 s_1 s_1', \overline{s}_2 s_2 s_2')].$$

If for all possible histories $\overline{s}_1', \overline{s}_2'$ of length T that led to the current state, the Q value is the same, then the history of states is irrelevant and can be replaced in the policy by just the length of the sequence T.

To prove $Q_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) = Q_{\overline{\pi}^*}(\overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2)$, I have to show three things:

1. $P(s_1' s_2' | \overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) = P(s_1' s_2' | \overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2)$.

From the definition of Markov,

$$P(s_1' s_2' | \overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) = P(s_1' s_2' | s_1 s_2, a_1 a_2). \tag{4.1}$$

2. $R(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2, s_1' s_2') = R(\overline{s}_1' s_1, \overline{s}_2' s_2, a_1 a_2, s_1' s_2')$.

From the definition of reward,

$$R(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2, s_1' s_2') = R(s_1 s_2, a_1 a_2, s_1' s_2'). \tag{4.2}$$

3. $V_{\overline{\pi}^*}(\overline{s}_1 s_1 s_1', \overline{s}_2 s_2 s_2') = V_{\overline{\pi}^*}(\overline{s}_1' s_1 s_1', \overline{s}_2' s_2 s_2')$

I show $V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) = V_{\overline{\pi}^*}(\overline{s}'_1 s_1, \overline{s}'_2 s_2)$ by induction.

Base case: s_1 and s_2 are final states and their values are always zero. $V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) = 0$, $V_{\overline{\pi}^*}(\overline{s}'_1 s_1, \overline{s}'_2 s_2) = 0$.

Inductive case: We assume it is true for

$$V_{\overline{\pi}^*}(\overline{s}_1 s_1 s'_1, \overline{s}_2 s_2 s'_2) = V_{\overline{\pi}^*}(\overline{s}'_1 s_1 s'_1, \overline{s}'_2 s_2 s'_2), \tag{4.3}$$

I need to show that it is true for

$$V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) = V_{\overline{\pi}^*}(\overline{s}'_1 s_1, \overline{s}'_2 s_2).$$

The value function is very similar to the Q function, except the current action is chosen from the policy.

$$\begin{aligned}
& V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) \\
= \quad & \sum_{s'_1 s'_2} P(s'_1 s'_2 | \overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2) \times \\
& [R(\overline{s}_1 s_1, \overline{s}_2 s_2, a_1 a_2, s'_1 s'_2) + \gamma V_{\overline{\pi}^*}(\overline{s}_1 s_1 s'_1, \overline{s}_2 s_2 s'_2)] \\
\overset{(4.1),(4.2)}{=} \quad & \sum_{s'_1 s'_2} P(s'_1 s'_2 | s_1 s_2, a_1 a_2) \times \\
& [R(s_1 s_2, a_1 a_2, s'_1 s'_2) + \gamma V_{\overline{\pi}^*}(\overline{s}_1 s_1 s'_1, \overline{s}_2 s_2 s'_2)] \\
\overset{(4.1),(4.2),(4.3)}{=} \quad & \sum_{s'_1 s'_2} P(s'_1 s'_2 | \overline{s}'_1 s_1, \overline{s}'_2 s_2, a_1 a_2) \times \\
& [R(\overline{s}'_1 s_1, \overline{s}'_2 s_2, a_1 a_2, s'_1 s'_2) + \gamma V_{\overline{\pi}^*}(\overline{s}'_1 s_1 s'_1, \overline{s}'_2 s_2 s'_2)]
\end{aligned}$$

Since a'_1 and a'_2 are optimal actions, we have:

$$\begin{aligned}
& \sum_{s'_1 s'_2} P(s'_1 s'_2 | \overline{s}'_1 s_1, \overline{s}'_2 s_2, a_1 a_2) [R(\overline{s}'_1 s_1, \overline{s}'_2 s_2, a_1 a_2, s'_1 s'_2) \\
& + \gamma V_{\overline{\pi}^*}(\overline{s}'_1 s_1 s'_1, \overline{s}'_2 s_2 s'_2)]
\end{aligned}$$

$$\begin{aligned}&\leq \sum_{s'_1 s'_2} P(s'_1 s'_2 | \overline{s}'_1 s_1, \overline{s}'_2 s_2, a'_1 a'_2)[R(\overline{s}'_1 s_1, \overline{s}'_2 s_2, a'_1 a'_2, s'_1 s'_2) \\ &\quad + \gamma V_{\overline{\pi}^*}(\overline{s}'_1 s_1 s'_1, \overline{s}'_2 s_2 s'_2)] \\ &= V_{\overline{\pi}^*}(\overline{s}'_1 s_1, \overline{s}'_2 s_2)\end{aligned}$$

As a result, we have $V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) \leq V_{\overline{\pi}^*}(\overline{s}'_1 s_1, \overline{s}'_2 s_2)$. Due to symmetry, we can also show that $V_{\overline{\pi}^*}(\overline{s}'_1 s_1, \overline{s}'_2 s_2) \leq$
$V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2)$. Therefore $V_{\overline{\pi}^*}(\overline{s}_1 s_1, \overline{s}_2 s_2) = V_{\overline{\pi}^*}(\overline{s}'_1 s_1, \overline{s}'_2 s_2)$.

Since the value of taking a joint action while following the optimal policy does not depend on the history, the same joint action is optimal for all histories and the policy need not include it. The interaction history can be summarized by the current state and time. □

Theorem 10 implies that a DEC-MDP with independent transitions and observations can be decomposed into two independent MDPs, with local states only affected by the local actions. The policies are standard policies for MDPs with the addition of time, and the goal is to maximize the expected reward received from a global value function.

4.3.2 Synchronizing Communication

In the DEC-MDP with independent transitions and observations presented above, the agents interact only through reward function. This highly restricted form of interaction does not reveal any information about other agent's local state or observations. In this section, I look at a less restricted form of interaction, which I call **synchronizing communication**. I will define it in the context of DEC-MDP-Com, since the explicit modeling of communication allows it to remain distinct from the domain actions.

Definition 20. *A communication protocol is said to be a* **synchronizing communication protocol** *if whenever any agent communicates, all agents send sufficient* *information to all other agents to unify their world views. Such an exchange is viewed as a single communication action with a single cost, even though there are potentially many messages sent.*

In a DEC-MDP-Com with a synchronizing communication protocol, whenever there is communication, each agent has the same view of the world. Since the world state is jointly fully observable, each agent has a complete view of the world state, and knows that the other agent has a complete view as well. The DEC-MDP-Com is essentially reset to an identical problem with a different start state, and the agents are safe to forget their past observation histories and communication histories. The communication actions essentially divide the DEC-MDP-Com into individual episodes, each of which is a DEC-MDP with no communication actions. The length of each episode varies depending on when it is optimal to communicate.

There are many applications in which synchronizing communication is an appropriate protocol. In certain problems, the communication setup cost is so high that it does not matter how much actual information is transferred. In other systems, the minimum packet size sent over the network may be larger than the messages the agents send, giving them a constant cost per message. For applications such as these, the amount of information contained in each message does not change its cost. A communication protocol is said to have **constant cost** if all the communication actions have the same cost. Specifically, a synchronizing communication action has the same cost as any other communication actions, no matter how many messages are actually exchanged to synchronize their partial views of the world state. Goldman and Zilberstein [29] proved that given a DEC-MDP-Com with constant communication cost, there is an optimal communication policy such that whenever there is communication, the agents exchange their last observations. Since a DEC-MDP-Com is jointly fully observable, when the agents exchange their last observations, they synchronize their views of the global state. As a result, if a DEC-MDP-Com has constant communication cost, there is an optimal communication policy such that whenever there is communication between the agents, it is synchronizing communication.

The second example of a polynomially encodable interaction protocol is the synchronizing communication in a DEC-MDP-Com with independent transitions and observations and a constant communication cost. This protocol has been studied in other work [29, 3].

Theorem 11. *The interaction history of a DEC-MDP-Com with independent transitions and observations and constant communication cost is polynomially encodable.*

Proof. From Definition 14, a local policy for a DEC-MDP-Com is of the form $\pi_i : S \times \overline{\Omega}_i \times \overline{\Sigma}_i \times \overline{\Sigma}_j \to A_i \times \Sigma_i$, where S is the last synchronized state, and $\overline{\Omega}_i$, $\overline{\Sigma}_i$ and $\overline{\Sigma}_j$ are the observation history and communication history since S. Since the DEC-MDP-Com has a constant communication cost, a synchronizing communication protocol is optimal. As a result, whenever there is communication the last synchronized global state S is updated to the newly synchronized state, and $\overline{\Omega}_i$ is set to $\emptyset$. When there is no communication, the observation is appended to $\overline{\Omega}_i$. Neither $\overline{\Sigma}_i$ nor $\overline{\Sigma}_j$ are used. Therefore, between communications the DEC-MDP-Com is equivalent to a DEC-MDP with independent transitions and observations, which is polynomially encodable (from Theorem 10). Since the interaction history between communications is polynomially encodable and communication resets the interaction history to $\emptyset$, the problem is polynomially encodable. $\square$

Even though a DEC-MDP-Com with independent transitions and observations and constant communication cost has considerably more communication than a DEC-MDP with only reward dependence, Theorem 11 shows that its interaction protocol is still polynomially encodable, and therefore it remains in NP.

4.4 Examples of Approximations

For applications where communication cost is constant, one can find an optimal policy that periodically synchronizes the world views of the agents. However, there are many other applications where the cost of message depends on its contents, and it may be beneficial to send less information than would synchronize the agents' world views. Unfortunately, most such interactions do not seem to be polynomially encodable because each piece of information changes an agent's belief about the local state of the other agent. A good guideline is that an agent needs to keep track of both the other agent's local state as well as the other agent's knowledge. However, we may still be able to design approximate encodings that are polynomial for such problems. The purpose of these encodings is to put extra restrictions on the interactions so that the complexity of approximating the optimal policy is reduced to at most NP. In this section, I show two examples of such approximations.

4.4.1 Constant Horizon

Consider the token collecting example. While there is no explicit communication between the agents, both of the agents observe the location of all available tokens. When agent i picks up a token at time t, agent j observes the fact that a token disappeared and can infer the location of agent i. With this observation, agent j knows the global state but agent i does not, so the interaction is not synchronizing but what I call asymmetric synchronizing. Asymmetric synchronization is where an agent either gains no information about the other agent's local state or complete information, giving it a belief of 1.0 about the current global state. The difference between synchronization and asymmetric synchronization is in the asymmetric case the agents do not necessarily both synchronize at the same time.

This difference precludes a polynomial encoding. In the token collecting example when agent i picks up a token both agents i and j know that i picked it up as well as both know the other knows, and so on. Agent i, then, must keep track of the information j has collected about i to best predict what j will do at least until j picks up a token itself and i learns where j is located. Remembering this sequence of information is exponential in the size of the state space in the worst case.

While the problem itself is harder than NP, it can be approximated by making assumptions that allow the interaction history to be polynomially encoded. For example, instead of keeping track of the entire history of interaction, one could assume that the last c interactions, for some constant c, was sufficient. The interaction history is now of size $|S|^c$, which is polynomial in the size of the state space. In the token collecting example this could correspond to agent i remembering the last 5 tokens it collected.

4.4.2 Flat Representation

Now let us look at the communication optimization problem in a Distributed Sensor Interpretation (DSI) system as studied in this thesis. I modeled this problem with a DEC-MDP where every action is explicit communication. Instead of explicitly modeling it as Σ in a DEC-MDP-Com, these actions are implicitly modeled in the DEC-MDP framework itself. The observation history $\overline{\Omega}$ records the interaction history of the system. Since every observation at every time step is needed to calculate the confidence level achieved, it is necessary for the agents to remember all the data values transferred between the agents in the past. Furthermore, remembering the order in which the data are exchanged is valuable because it carries useful information. An agent can infer why the other agent chose to transfer this piece of data before the other piece. Therefore, remembering the entire $\overline{\Omega}$ is essential to generating the optimal policy. We write the number of local sensor data as n, and assume that each sensor data has at most m possible values. As a result, $|S| = m^n$, and

$|\overline{\Omega}| = O(n! \cdot m^n) = O(n^n) = O(|S|^{\log_2 n})$. Since n is not independent of $|S|$, $\overline{\Omega}$ is not polynomially encodable and therefore the DEC-MDP is harder than NP.

As discussed in Chapter 2, one way to approximate the optimal solution to this problem is that, instead of remembering the entire $\overline{\Omega}$, each agent only remembers the values of the data exchanged so far without remembering the order in which they were transferred. This is a reasonable approximation since to calculate the confidence level of the local interpretations, only the sensor data values are needed. In this approximation, the approximate encoding E_i of $\overline{\Omega}$ is of the size $O((m+1)^n) = O(|S|^{\log_m(m+1)})$. Since m is independent of $|S|$, E_i is a polynomial encoding, and therefore this approximation is no harder than NP.

4.5 Conclusions

Distributed POMDPs have been developed and employed to model various multi-agent coordination problems. Understanding the source of their high complexity is crucial to identifying new and more tractable models as well as developing appropriate approximations to otherwise intractable problems. This paper establishes that the interactions present among the agents is the cause of the high complexity of distributed POMDPs. I proved that deciding a distributed POMDP whose interaction history contains information of a size polynomial in the number of states is NP-complete, and that deciding a non-polynomially encodable distributed POMDP is harder than NP. I demonstrated how two subclasses of distributed POMDPs known to be NP-complete can be polynomially encoded. This is the first time that a well-defined condition has been identified that can distinguish between multi-agent problems in NP and those that are strictly harder than NP. It is an important step in mapping out the complexity hierarchy of multi-agent systems.

My goal in this chapter was not to introduce new models or algorithms, but to change the way people view interactions between agents in the context of distributed

POMDPs. Multi-agent researchers have long intuitively understood that the interaction between the agents is the cause of their high complexity. The theoretical results of this paper are significant in that they both formally justify this intuition as well as explain how the interaction affects the complexity. This new understanding of interaction and its relationship to complexity will help us to identify new classes of multi-agent systems with a lower complexity.

The significance of this theoretical result also has a more practical side. Most multi-agent systems are provably harder than NP and solving them optimally is not possible. Much work has been put into developing good algorithms for approximating these problems. This work provides theoretical guidance in understanding how the approximations in a model limit the search space and reduce the complexity. I demonstrated this on two non-polynomially encodable problems by providing two assumptions that reduce the complexity to no harder than NP. One of them is to put bound on the length of the history that an agent can remember, and the other puts constraint on the history information that an agent retains. There are other types of approximation techniques that can potentially used to reduce the complexity of the problem. For example, we can restrict the agents to remember only a number of key features in the communication history, similar to the techniques used by Xuan and Lesser [81]. Abstraction techniques that allows agents to retain an abstract communication history that is of a constant size may also reduce the problem's complexity.

CHAPTER 5

CONCLUSIONS

Distributed Sensor Interpretation (DSI) problems have been the subject of considerable research within the cooperative MAS community. In a DSI system, data is collected from different sensors and must be integrated to produce the best interpretation. Distributed approaches to DSI emphasize not only distributed collecting of data, but also distributed processing of data. However, in virtually all real-world DSI systems the agents must exchange data, local results, and/or other information to develop a solution with acceptable quality. Unless communication among the agents is appropriately limited, the cost of communication may negate much of the benefit of distributed processing. Unfortunately, the state-of-the-art in MAS is such that there are not yet formal design methods that allow one to evaluate a potential DSI domain and determine the optimal coordination strategy. I believe that this is a serious issue that will hinder the deployment of many important applications of sensor networks. My work is one of the first attempts to address this issue.

5.1 Contributions

1. Formally understand the tradeoff between communication cost and solution cost in DSI problems.

   I formalized the communication problem in DSI with a Distributed Bayesian Network and solved the question of what to communicate with a decentralized Markov Decision Process (DEC-MDP). With this model, one is able to generate a communication strategy for a given DSI problem such that only minimum communication cost is needed to achieve a required confidence level in the interpretation task. This is a new way to study the satisficing approach to the communication management in DSI problems. The DEC-MDP model and the communication strategy derived from it enables us to study the tradeoff between the communication cost and the solution quality formally.

Such a DEC-MDP is different from the DEC-MDPs with communication that most researchers have studied. Once the sensor data are collected by the agents, the global state is not changed. The only actions that an agent may choose to execute are to request part of the other agent's observations or to send part of its own observations in order to get a better view of the global state. The communication actions have different costs depending on the bandwidth and power consumption for the message, the possible violation of the privacy and the danger of broaching the security. In contrast, other research studying communication in DEC-MDP focuses on synchronizing communication where every communication action exchanges all the state information among agents and leads to the same view of the system. While the study of synchronizing communication tries to answer the question of **when** to communicate, my work is trying to address the question of **what** to communicate.

2. Developed some of the first algorithms to find approximate solutions for general DEC-MDPs.

   In the problem I study, an agent's communication action directly changes the other agent's view. The local MDPs of the two agents are largely dependent on each other. This makes it hard to construct algorithms that are guaranteed to find the globally optimal solution. My complexity result shows that even though my problem is of a lower complexity class than the general DEC-MDP, it is still harder than NP. I have designed two algorithms to approximate the globally optimal solution for our DEC-MDP. One is an iterative algorithm that is guaranteed to converge to a local optimal solution, but the quality of the policy it generates largely depends on the starting policy of the iterative process. The other approach is based on a lookup algorithm which is much less computationally expensive and can be easily extended to more than two agents. Though there is no guarantee that can be made about the solution either generates, experimental work indicates that in the problems studied both approaches lead to policies that are of very good quality. To my knowledge, this is some of the first work providing algorithms that approximate the optimal solution for communication problems in a complex problem solving setting that are formulated in a decision-theoretic model.

3. Understand the use of abstraction in communication management.

   Though general communication can be naturally modeled with a DEC-MDP, techniques need to be developed to address the complexity issue before the system can be scaled up. I approached this problem from two perspectives. First, I investigated the techniques of transferring abstraction data in addition to observed data in Distributed Bayesian Networks to reduce the required communication cost. I introduced an algorithm that automatically generates appropriate abstraction data that facilitates the achievement of the required confidence level and reduces the necessary communication cost. I also discussed approaches to incorporate the new abstraction data into the DEC-MDP framework effectively. Both the improvement in the minimum expected communication cost and the time savings in solving the DEC-MDP make the *hierarchical action selection* an attractive approach, especially for the systems which require a mid-ranged confidence level. I further extended the different action selection approaches to larger networks and multiple abstraction layers. The hierarchical action selection approach was shown to be able to solve problems with larger networks

is not possible. Much work has been put into developing good algorithms for approximating these problems. This work provides theoretical guidance in understanding how the approximations in a model limit the search space and reduce the complexity. I demonstrated this on two non-polynomially encodable problems by providing two assumptions that reduce the complexity to no harder than NP.

5.2 Future Directions

There are several direction in which I would like to extend this work. First, I would like to look at other forms of abstraction that can be potentially useful. For example, some abstraction data may contain useful information of the rest of the network but is not sufficient to reach the confidence by itself. Other abstraction may be an intermediate interpretation result that can be incorporated into the local data. Control information can also be used as a type of abstraction. I also intend to investigate other techniques in generating multiple layers of abstraction data, such as different levels of intermediate interpretation results. Even though I have proposed techniques and algorithms to extend this work to more than two agents, it is still difficult to scale up the problems we can solve without developing more sophisticated methods. We already started investigating the possibility of partitioning a large network into smaller subnets that are connected through small number of gateway agents. Abstraction then can be used to carry information between the subnets in order to divide the entire problem into smaller sub-problems with more manageable sizes. Finally, I am searching for new examples of polynomially encodable interaction protocols, as well as common protocols that we can prove are not polynomially encodable. I am also trying to find new NP approximations appropriate to the protocols that we can prove to be harder than NP.

BIBLIOGRAPHY

[1] Barto, A.G., Bradtke, S.J., and Singh, S.P. Learning to act using real-time dynamic programming. *Artificial Intelligence 72* (1995), 81–138.

[2] Becker, Raphen, Lesser, Victor, and Zilberstein, Shlomo. Decentralized Markov decision processes with event-driven interactions. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems* (2004), vol. 1, IEEE Computer Society, pp. 302–309.

[3] Becker, Raphen, Lesser, Victor, and Zilberstein, Shlomo. Analyzing myopic approaches for multi-agent communication. In *Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology* (Compiegne, France, September 2005), IEEE Computer Society, pp. 550–557.

[4] Becker, Raphen, Zilberstein, Shlomo, Lesser, Victor, and Goldman, Claudia V. Solving transition independent decentralized MDPs. *Journal of Artificial Intelligence Research 22* (2004), 423–455.

[5] Bernstein, Daniel S., Givan, Robert, Immerman, Neil, and Zilberstein, Shlomo. The complexity of decentralized control of Markov decision processes. *Mathematics of Operations Research 27*, 4 (November 2002), 819–840.

[6] Bernstein, Daniel S., Hansen, Eric A., and Zilberstein, Shlomo. Bounded policy iteration for decentralized POMDPs. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)* (Edinburgh, Scotland, July 2005).

[7] Bertsekas, Dimitri P. *Dynamic Programming and Optimal Control, Vols. I and II.* Athena Scientific, Belmont, MA, 1995.

[8] Boutilier, Craig. Sequential optimality and coordination in multiagent systems. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence* (1999), pp. 478–485.

[9] Boutilier, Craig, Dearden, Richard, and Goldszmidt, Moise's. Exploiting structure in policy construction. In *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence* (San Francisco, 1995), Chris Mellish, Ed., Morgan Kaufmann, pp. 1104–1111.

[10] Buntine, Wray. A guide to the literature on learning probabilistic networks from data. *IEEE Transactions on Knowledge and Data Engineering 8*, 2 (1996), 195–210.

[11] Carver, Norman, and Lesser, Victor. The DRESUN testbed for research in FA/C distributed situation assessment: Extensions to the model of external evidence. In *Proceedings of the First International Conference on Multi-Agent Systems* (January 1995), AAAI Press, pp. 33–40.

[12] Carver, Norman, and Lesser, Victor. Domain monotonicity and the performance of local solutions strategies for CDPS-based distributed sensor interpretation and distributed diagnosis. *International Journal of Autonomous Agents and Multi-Agent Systems 6* (2003), 35–76.

[13] Chades, Iadine, Scherrer, Bruno, and Charpillet, Francois. A heuristic approach for solving decentralized-POMDP: Assessment on the pursuit problem. In *Proceedings of the 17th ACM Symposium on Applied Computing (SAC 2002)* (Madrid, 2002), pp. 57–62.

[14] Chakravarthi, Muralidar. Decomposability in distributed sensor interpretation. Master's thesis, Computer Science Department, Southern Illinois University, 2003.

[15] Chelsea C. White, III. A survey of solution techniques for the partially observed Markov decision processes. *Annals of Operations Research 32* (1991), 215–230.

[16] Chung, Seung H., and Barrett, Anthony. Distributed real-time model-based diagnosis. In *Proceedings of the 2003 IEEE Aerospace Conference* (2003).

[17] Clement, Bradley J., and Durfee, Edmund H. Theory for coordinating concurrent hierarchical planning agents using summary information. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (1999), pp. 495–502.

[18] Cohen, Philip R., and Levesque, Hector J. Intention is choice with commitment. *Artificial Intelligence* (1990), 213–261.

[19] Cristescu, Razvan, Beferull-Lozano, Baltasar, Vetterli, Martin, Ganesan, Deepak, and Acimovic, Jugoslava. On the interaction of data representation and routing in sensor networks. In *Proceedings of the 30th IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2005)* (Philadelphia, PA, 2005).

[20] Crossbow Wireless Sensor Platform. http://www.xbow.com/Products/Wireless Sensor Networks.htm.

[21] Dean, T., Kaelbling, L.P., Kirman, J., and Nicholson, A. Planning under time constraints in stochastic domains. *Artificial Intelligence 76* (1995), 35–74.

[22] Deshpande, Amol, Guestrin, Carlos, Madden, Sam, Hellerstein, Joseph, and Hong, Wei. Model-based approximate querying in sensor networks. *International Journal on Very Large Data Bases* (2005).

[23] Erman, Lee D., Hayes-Roth, Frederick, Lesser, Victor R., and Reddy, D. Raj. The HEARSAY-II speech understanding system: Integrating knowledge to resolve uncertainty. *Computing Surveys 12*, 2 (June 1980), 213–253.

[24] Feng, Zhengzhu, and Hansen, Eric. Approximate planning for factored POMDPs. In *Proceedings of the Sixth European Conference on Planning* (2001).

[25] Franklin, Michael J., Jeffery, Shawn R., Krishnamurthy, Sailesh, Reiss, Frederick, Rizvi, Shariq, Wu, Eugene, Cooper, Owen, Edakkunni, Anil, and Hong, Wei.

[26] Ghavamzadeh, Mohammad, and Mahadevan, Sridhar. A multiagent reinforcement learning algorithm by dynamically merging Markov decision processes. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi Agent Systems* (Bologna, Italy, 2002), ACM Press.

[27] Ghavamzadeh, Mohammad, and Mahadevan, Sridhar. Learning to communicate and act using hierarchical reinforcement learning. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS-2004)* (New York City, New York, July 2004), pp. 1114–1121.

[28] Goldman, Claudia V., and Zilberstein, Shlomo. Optimizing information exchange in cooperative multi-agent systems. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems* (Melbourne, Australia, July 2003), ACM Press, pp. 137–144.

[29] Goldman, Claudia V., and Zilberstein, Shlomo. Decentralized control of cooperative systems: Categorization and complexity analysis. *Journal of Artificial Intelligence Research 22* (2004), 143–174.

[30] Guestrin, Carlos, Koller, Daphne, and Parr, Ronald. Multiagent planning with factored MDPs. In *Advances in Neural Information Processing Systems 14* (Cambridge, MA, 2002), T. G. Dietterich, S. Becker, and Z. Ghahramani, Eds., MIT Press, pp. 1523–1530.

[31] Guestrin, Carlos, Venkataraman, Shobha, and Koller, Daphne. Context specific multiagent coordination and planning with factored MDPs. In *Proceedings of the Eighteenth National Conference on Artificial Intelligence* (2002), pp. 253–259.

[32] Hansen, Eric A., Bernstein, Daniel S., and Zilberstein, Shlomo. Dynamic programming for partially observable stochastic games. In *Proceedings of the 19th National Conference on Artificial Intelligence (AAAI)* (San Jose, California, July 2004), pp. 709–715.

[33] Heckerman, David. A tutorial on learning with bayesian networks. In *Learning in Graphical Models*, Michale Jordan, Ed. MIT Press, Cambridge, MA, 1999.

[34] Heckman, James J. Sample selection bias as a specification error. *Econometrica 47*, 1 (January 1979), 153–162.

[35] Hill, J., Szewczyk, R., Woo, A., Hollar, S., Culler, D., and Pister, Kr. System architecture directions for networked sensors. In *Proceedings of ASPLOS* (2000), pp. 93–104.

[36] Hsu, Kai, and Marcus, Steven I. Decentralized control of finite state Markov processes. *IEEE Transactions on Automatic Control 27*, 2 (1982), 426–431.

[37] Huang, Cecil, and Darwiche, Adnan. Inference in belief networks: A procedural guide. *International Journal of Approximate Reasoning 11* (1994), 1–158.

[38] Huang, Timothy, and Russell, Stuart J. Object identification in a bayesian context. In *IJCAI* (1997), pp. 1276–1283.

[39] Kahn, J. M., Katz, R. H., and Pister, K. S. J. Mobile networking for Smart Dust. In *Proceedings of ACM MOBICOM* (1999), pp. 271–278.

[40] Krause, Andreas, Guestrin, Carlos, Gupta, Anupam, and Kleinberg, Jon. Near-optimal sensor placements: Maximizing information while minimizing communication cost. In *Proceedings of Fifth International Conference on Information Processing in Sensor Networks (IPSN'06)* (April 2006).

[41] Kulkarni, Purushottam, Ganesan, Deepak, Shenoy, Prashant, and Lu, Qifeng. SensEye: A multi-tier camera sensor network. In *Proceedings of ACM Multimedia* (2005).

[42] Lesser, Victor, Decker, Keith, Wagner, Thomas, Carver, Norman, Garvey, Alan, Horling, Bryan, Neiman, Daniel, Podorozhny, Rodion, Prasad, M. Nagendra, Raja, Anita, Vincent, Regis, Xuan, Ping, and Zhang, Xiaoqin. Evolution of the GPGP/TAEMS domain-independent coordination framework. *Autonomous Agents and Multi-Agent Systems 9*, 1 (July 2004), 87–143.

[43] Lesser, Victor, Ortiz, Charles, and Tambe, Milind, Eds. *Distributed Sensor Networks: A multiagent perspective.* Kluwer Academic Publishers, 2003.

[44] Lesser, V.R., and Erman, L.D. Distributed interpretation: A model and an experiment. *IEEE Transactions on Computers, Special Issue on Distributed Processing C-29*, 12 (December 1980), 1144–1163.

[45] Li, Haksun, Durfee, Edmund H., and Shin, Kang G. Multiagent planning for agents with internal execution resource constraints. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multiagent Systems* (July 2003), pp. 560–567.

[46] Li, Huan, Shenoy, Prashant, and Ramamritham, Krithi. Scheduling communication in real-time sensor applications. In *Proceedings of the Tenth IEEE Real-Time/Embedded Technology and Applications Symposium (RTAS04)* (Toronto, Canada, May 2004).

[47] Littman, Michael, Cassandra, Anthony, and Kaelbling, Leslie. Learning policies for partially observable environments: Scaling up. In *Proceedings of the Twelfth International Conference on Machine Learning* (San Francisco, CA, 1995), Armand Prieditis and Stuart Russell, Eds., Morgan Kaufmann, pp. 362–370.

[48] Lovejoy, W. S. A survey of algorithmic methods for partially observed markov decision processes. *Annals of Operations Research 28* (1991), 47–66.

[49] Lowe, David G. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision 60*, 2 (November 2004), 91 – 110.

[50] Madani, Omid, Hanks, Steve, and Condon, Anne. On the undecidability of probabilistic planning and infinite-horizon partially observable Markov decision problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence* (1999), pp. 541–548.

[51] Mailler, Roger, and Lesser, Victor. Solving distributed constraint optimization problems using cooperative mediation. *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi Agent Systems* (2004), 438–445.

[52] Mattar, M., Hanson, A., and Learned-Miller, E. Sign classification using local and meta-features. In *Proceedings of IEEE Workshop on Computer Vision Applications for the Visually Impaired (in conjunction with CVPR)* (San Diego, California, 2005).

[53] Meliou, Alexandra, Chu, David, Guestrin, Carlos, Hellerstein, Joseph, and Hong, Wei. Data gathering tours in sensor networks. In *Proceedings of the Fifth International Conference on Information Processing in Sensor Networks (IPSN'06)* (April 2006).

[54] Modi, Pragnesh Jay, Shen, Wei-Min, Tambe, Milind, and Yokoo, Makoto. An asynchronous complete method for distributed constraint optimization. In *Proceedings of the Second International Joint Conference on Autonomous Agents and Multi Agent Systems* (2003), pp. 161–168.

[55] Monahan, George E. A survey of partially observable Markov decision processes: Theory, models, and algoritms. *Management Science 28* (1982), 1–16.

[56] Mundhen, Martin, Goldsmith, Judy, Lusena, Christopher, and Allender, Eric. Complexity of finite-horizon Markov decision process problems. *Journal of the ACM 47*, 4 (2000), 681–720.

[57] Nair, R., Tambe, M., Yokoo, M., Pynadath, D., and Marsella, S. Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings. In *Proceedings of the International Joint conference on Artificial Intelligence (IJCAI)* (2003).

[58] Nair, Ranjit, Varakantham, Pradeep, Tambe, Milind, and Yokoo, Makoto. Networked distributed pomdps: A synthesis of distributed constraint optimization and pomdps. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)* (2005).

[59] Ooi, James M., and Wornell, Gregory W. Decentralized control of a multiple access broadcast channel: Performance bounds. In *Proceedings of the 35th Conference on Decision and Control* (1996), pp. 293–298.

[60] Papadimitriou, Christos H., and Tsitsiklis, John. On the complexity of designing distributed protocols. *Information and Control 53* (1982), 211–218.

[61] Papadimitriou, Christos H., and Tsitsiklis, John. Intractable problems in control theory. *SIAM Journal on Control and Optimization 24*, 4 (1986), 639–654.

[62] Papadimitriou, Christos H., and Tsitsiklis, John. The complexity of Markov decision processes. *Mathematics of Operations Research 12*, 3 (1987), 441–450.

[63] Paskin, Mark, and Guestrin, Carlos. Robust probabilistic inference in distributed systems. In *Proceedings of Twentieth Conference on Uncertainty in Artificial Intelligence (UAI 2004)* (Banff, Canada, July 2004).

[64] Paskin, Mark, Guestrin, Carlos, and McFadden, Jim. A robust architecture for distributed inference in sensor networks. In *Proceedings of Fourth International Conference on Information Processing in Sensor Networks (IPSN'05)* (April 2005).

[65] Pearl, Judea. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers, Inc., 1988.

[66] Peshkin, Leonid, Kim, Kee-Eung, Meuleau, Nicolas, and Kaelbling, Leslie P. Learning to cooperate via policy search. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence* (San Francisco, CA, 2000), Morgan Kaufmann, pp. 489–496.

[67] Poupart, Pascal, and Boutilier, Craig. Bounded finite state controllers. In *Advances in Neural Information Processing Systems 16*, Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, Eds. MIT Press, Cambridge, MA, 2004.

[68] Puterman, Martin L. *Markov Decision Problems*. Wiley, New York, 1994.

[69] Pynadath, David V., and Tambe, Milind. The communicative multiagent team decision problem: Analyzing teamwork theories and models. *Journal of Artificial Intelligence Research 16* (2002), 389–423.

[70] Rahimi, M., Baer, R., Warrior, J., Estrin, D., and Srivastava, M. Cyclops: In situ image sensing and interpretation in wireless sensor networks. In *Proceedings of ACM SENSYS* (2005).

[71] Rao, Anand S., and Georgeff, Michael P. Modeling rational agents within a BDI-architecture. In *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning* (1991).

[72] Shen, Jiaying, Becker, Raphen, and Lesser, Victor. Agent Interaction in Distributed MDPs and its Implications on Complexity. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Hakodate, Japan, May 2006).

[73] Shen, Jiaying, and Lesser, Victor. Communication Management Using Abstraction in Distributed Bayesian Networks. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Hakodate, Japan, May 2006).

[74] Shen, Jiaying, Lesser, Victor, and Carver, Norman. Controlling information exchange in distributed bayesian networks. Tech. Rep. 02-22, University of Massachusetts, 2002.

[75] Shen, Jiaying, Lesser, Victor, and Carver, Norman. Minimizing Communication Cost in a Distributed Bayesian Network using a Decentralized MDP. In *Proceedings of Second International Joint Conference on Autonomous Agents and MultiAgent Systems (AAMAS 2003)* (Melbourne, AUS, July 2003), vol. AAMAS03, ACM Press, pp. 678–685.

[76] Stargate Platform. http://www.xbow.com/Products/XScale.htm.

[77] Tambe, Milind. Towards flexible teamwork. *Journal of Artificial Intelligence Research 7* (1997), 83–124.

[78] Washington, Rich, Golden, K., Bresina, J., Smith, D., Anderson, C., and Smith, T. Autonomous rovers for mars exploration. In *Proceedings of the IEEE Aerospace Conference* (1999).

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

[80] Xiang, Yang. *Probabilistic Reasoning in Multiagent Systems, A graphical models approach.* Cambridge University Press, 2002.

[81] Xuan, Ping, and Lesser, Victor. Multi-agent policies: from centralized ones to decentralized ones. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-Agent Systems (AAMAS-02)* (Bologna, Italy, 2002).

[82] Xuan, Ping, Lesser, Victor, and Zilberstein, Shlomo. Formal modeling of communication decisions in cooperative multi-agent systems. In *Proceedings of the Second Workshop on Game Theoretic and Decision Theoretic Agents (GTDT 2000)* (2000).