TIME CONSTRAINT AGENTS' COORDINATION AND LEARNING IN COOPERATIVE MULTI-AGENT SYSTEM

WU XUE

(B.Eng. (Hons.), UNIVERSITY OF SCIENCE AND TECHNOLOGY OF CHINA)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY
DEPARTMENT OF INDUSTRIAL AND SYSTEMS ENGINEERING
NATIONAL UNIVERSITY OF SINGAPORE

2011

Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirely. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university in any university previously.

Wu xue

Wu Xue

20 Jan 2013

Acknowledgements

I would like to thank all people who have helped and inspired me during my doctoral study.

I would like to show my gratitude to my advisor, Prof. Poh Kim Leng, for his guidance during my study and research years at National University of Singapore. His perpetual energy and enthusiasm in research had motivated all his advisees, including me. And his encouragement, guidance and support have enabled me to overcome all the obstacles in the completion of the Ph. D. research work. As a result, research life became smooth and rewarding for me.

I was delighted to interact with Prof. Leong Tze Yun by attending the Biomedical Decision Engineering (BiDE) group seminars and having her as my research qualification examiner. Her insights to artificial intelligence and machine learning positively influenced my research work. And she has given many valuable advices for my dissertation.

My seniors, including Zeng Yifeng, Xiang Yanping, Wang Yang, Cao Yi, have helped me in finding my research topic and commenting my research work. I would like to especially thank Zeng Yifeng. I got his help and advice even when he graduated and became an assistant professor in Denmark.

My colleagues at the BiDE group, including Li Guoliang, Jiang Changan, Chen Qiongyu, Rohit, Yin Hongli, Ong Chenghui, Zhou Peng, Zhu Ailing, Nguyen Thanh Trung and Guo WenYuan , have asked interesting and challenging questions in my presentation and offered helpful comments on my research. I enjoyed the four years BiDE group seminar discussion with them.

All my lab buddies at the systems modeling and analysis laboratory of NUS made it a convivial place to work. They are Fan Liwei, Wang Xiaoying, Guo Lei, Han Yongbin, Liu Na, Luo Yi, Long Yin, Wang Guanli, Cui Wenjuan, Hu Junfei, Jiang Yixin and Didi. We have all got along very well. With their company, I more enjoy my stay in Singapore. I also would like to thank Tan Swee Lan, the lab technician, who has provided a convenient working environment for us.

My deepest gratitude goes to my parents for their unflagging love and support throughout my life. This dissertation is simply impossible without them.

Lastly, I offer my regards and blessings to all of those who supported me in any respect during the completion of the thesis.

Wu Xue

Table of Contents

Summary

Multi-agent System (MAS) is a system composed of multiple interacting autonomous intelligent agents. Since it is impossible for the designers to determine the behavioral repertoire and concrete activities of a MAS at the time of its design and prior to its use, agents should to be adaptive to meet the changing environment. The thesis proposes that in order to be adaptive, agents in the MAS need to learn from and coordinate with other agents. The learning ability enables agents to evolve with the changing environment. Furthermore, the coordination of the agents let MAS solve complex and dynamic problems adaptively.

To efficiently learn from and coordinate with other agents, the first and fundamental problem of MAS is to have a suitable agent model and a supporting MAS framework. A Bayesian-Networked – Believe, Desire, and Intention Model (BN-BDI) agent model is proposed to build the adaptive MAS. It is a hybrid architecture that has the merits of both the deliberative and reactive architecture. The BDI part maintains an explicit representation of the agents' world. The BN part measures the uncertainty that an agent faces and the dependent relationship it has with other agents. The BN-BDI agent can learn other agent's model, their preferences, their beliefs and their capacities. A hierarchical MAS architecture consisting of the BN-BDI agents is formed. Agents with the similar characteristics or capacities constitute a group, and some agents act as coordinators.

For the coordination of the MAS, a time constraint task-based model is proposed. This model borrows the task sharing idea from the distributed problem solving domain and adds in a time-critical component. The communication and coordination complexity of the model is $O(n \log n)$ and it significantly reduces the amount of information exchanged and scales well with the number of agents. The BN-BDI agent model makes the MAS framework ready for cooperation and coordination.

To verify the proposals, simulations are carried out using a foraging problem scenario. Two heuristic algorithms have been tested and the simulation results support these hypothesis.

Key words: Multi-agent systems, Coordination, Foraging, Contract Net, Multi-agent Learning, BDI

List of Figures

List of Abbreviations

ACE: Action Estimation
AGE: Action Group Estimation
AOS: Agent Oriented Software
BDI: Believe, Desire, Intention
BN: Bayesian Network
BN-BDI: Bayesian-Networked Believe, Desire, Intention Model
CBN: Casual Bayesian Network
C-component: Confounded Component
CDPS: Cooperative Distributed Problem Solving
CNP: Contract Net Protocol
CPT: Conditional Probability Table
DAG: Directed Acyclic Graph
DAI: Distributed Artificial Intelligence
DMAID: Dynamic Multi-agent Influence Diagram
DMC: Distance Modulated Communication
EBL: Explanation-based Learning
FIPA: Intelligent Physical Agent
HBM: Hierarchical Bayesian Model
ID: Influence Diagram
ILP: Inductive Logic Programming
JADE: Java Agent Development Framework
MAASP: Multi-agent Action Selection Problem
MACM: Multi-agent Causal Model
MAID: Multi-agent Influence Diagram
MAIDF: Multi-agent Influence Diagram Fragments
MAL: Multi-agent Learning
MAS: Multi-agent System
ML: Maximum Likelihood
MSBN: Multiply-sectioned Bayesian Network
LRTA*: Learning Real-time A* Algorithm
PRS: Procedural Reasoning System
PN: Praxeic Network
RTA*: Real-time A* Algorithm
TCTM: Time Constraint Task-based Model

Chapter 1 Introduction

1.1 The Adaptive Multi-agent System

With the advances in computer science – multi-tasking, communicating process, distributed computing, modern interpreted languages, real-time systems, communication networks, and networked environment, Intelligent Agents has become the most vibrant and fastest growing research area in both Artificial Intelligence and Computer Science,. New agent-based products, application and services arise on an almost daily basis, as it is a promising new paradigm for conceptualizing, designing and implementing software system. From funs games such as robot soccer to mission critical applications such as targeting and monitoring enemy movements, and to smart home systems, the number of applications for MAS is endless.

Although agents have been under study for almost 20 years, many research fields are still unsolved and are on the emerging stage, especially in the agent theories, models and architectures part. Let's look at the following problems:

<u>Personal Information Assistants</u>

The traditional approach to assist the personal information is through direct manipulation, which requires the user to tediously initiate and execute each action even in cases where sequences of actions are better automated, e.g., locating, retrieving, and extracting relevant information from distributed data collections. The direct approach is practical and possible for tens of items and it becomes unwieldy and impractical for thousands of them. This approach does not offer support for initiating actions on behalf of user in response to situations that might arise and does not facilitate easy composition of basic actions and objects in complex action structures. As a result, users have to grapple with the complexity and heterogeneity of distributed, heterogeneous information sources and computing devices. However, agent-based approach can solve these difficulties easily through indirect manipulation. Agents can automate routine tasks of locating, retrieving, and processing information from heterogeneous distributed information sources and can hide the heterogeneity and complexity of the underlying information sources. Moreover, adaptive agents allow customization of generic software to the needs and interests of specific users. Agents can learn by observing users and interacting with them, and thereby improve their behaviour. Agents can potentially work around unforeseen problems and exploit unforeseen opportunities as they go about doing their tasks.

Collect Collective Satellites Information

To better respond to transient Earth phenomenon that can cause loss of life or damage to economic assets (tornadoes, mudslides, flash floods, etc.) an increase in the amount and timeliness of information collected on phenomenon is needed. One method for collecting this information is by using groups of Earth observing satellites with the ability to perform autonomous orbital maneuvers and view phenomenon on demand. However, as satellites are very costly, creating a group of satellites large enough to perform this task is currently beyond the abilities of any one organization. One method of gathering a group of satellites that is large enough is by several organizations “pooling” their satellite resources together temporarily. In order to pool autonomous maneuverable satellites, several technical problems must be overcome. These include for example how to schedule large numbers of satellites to effectively collect critical information on phenomenon, even in the face of unexpected events, like satellite failures that can prohibit the collection of this information in (McConnell 2003).

Robot Search Team

One space shuttle lands on Mars. Several robots are sent out to collect as many useful material samples as possible. These robots are autonomous and self-interested. However, they are required to cooperate with each other to finish the discovery job as a team. How robots adjust their behavior to achieve a good system performance and at the same time keep their integrity and self-interest is a tough problem that researchers have to deal with.

These three examples mentioned above can be represented in MAS. MAS has many interesting yet unique abilities that make it outperform single agent system. No single agent has sufficient competence (e.g., in medical diagnosis, knowledge about heart disease, blood disorders and respiratory problems may need to be combined to diagnose a patient's illness), resource (power, memory, communication, that belong to different agents are to be harnessed), and information (in concurrent engineering systems, the same product may be viewed from a design, manufacturing and marketing perspective). For example, MAS can solve problems that are too large for a centralized agent; it can interconnect and interoperate with the multiple existing legacy systems; it can provide solutions to problems that can naturally be regarded as a society of autonomous interacting agents; it can provide solutions that efficiently use information sources that are spatially distributed; it can provide solutions in situations where expertise is distributed.

Although MAS provides many potential advantages, it also presents many challenges. The difficulties of the examples mentioned above represent the typical challenges in MAS. In the example of “Personal Information Assistants”, the challenges include how agents grapple with the complexity and heterogeneity of distributed, heterogeneous information sources and computing devices; how autonomous agents in the system which represent the users behave adaptively to meet the uncertainty and to achieve a good global performance of all the personal information and meanwhile keep their integrity of each information item and increase its local utility value.

Typically MAS is of considerable complexity with respect to both its structure and its functionality. For most application tasks, and even in environments that appear to be more or less simple, it is extremely difficult or even impossible to correctly determine the behavioral repertoire and concrete activities of a MAS system at the time of its design and prior to its use. This would require, for instance, that it is known a priori which environmental requirements will emerge in the future, which agents will be available at the time of emergence, and how the available agents will have to interact in response to these requirements. This problem is even more complex when the agent is situated in an environment that contains other agents with potentially different capabilities and goals. This kind of problems results from the complexity of MAS and can be avoided or at least reduced by endowing the agents with the ability to adapt to and to learn from the environment and their fellow agents.

1.2 Background Information

1.2.1 Agent Theories

The term agent is used by many people working in different areas, and it can be used to describe several very different kinds of systems. However, there is no precise definition about it. Generally, an agent is supposed to have four basic components, which are the sensor component, the motor component, the information base, and the reasoning engine. The sensor and motor components let an agent to interact with its environment. The information base contains the information an agent has about its environment. The reasoning engine enables an agent to perform processes like inferring, planning and learning. Although this basic conception is widely accepted, people have controversial views on the agent characteristic properties. In (Wooldridge and Jennings 1995) an agent is a computer system, situated in some environment that is capable of flexible autonomous action in order to meet its design objectives. In their mind, there exist a weak and a strong notion of agency. According to the weak notion, an agent must have autonomy, reactivity and pro-activeness. As for the strong notion, more specified properties are included in an agent. It enjoys the properties of belief, knowledge, intention, commitment, desire, goal, etc.

(Wooldridge and Jennings 1995) describes the characteristics of agents' modules when applied to software entities.

Autonomy: an agent operates without direct intervention of other agents or humans and has control over its actions and its internal state.

Responsiveness: an agent perceives its environment and responds in a timely fashion to changes that occur in it.

Pro-activeness and deliberation: an agent does not simply react to changes in the environment, but exhibits goal-directed behaviors and takes the initiative when it considers them to be appropriate.

Social-ability: an agent interacts with other agents and possibly humans via some kind of agent-communication language. If it is needed to complete its tasks and help others to achieve their goals, agents can cooperate with each other.

Mobility: the ability to change the physical position

Benevolence: the property of always doing what it is asked to do

The main point about agents is that they are autonomous: capable of acting independently, exhibiting control over their internal state. Thus an agent is a computer system capable of flexible autonomous action in some environment. The flexible action means reactive, pro-active, and social.

The unique characteristic properties of an agent distinguish it from other terminologies, like object or expert system.

Agent-based system' vs. 'object-oriented system'

Objects are defined as computational entities that encapsulate some state, are able to perform actions, or methods on this state, and communicate by message passing. While there are obvious similarities, there are also significant differences between agents and objects.

Firstly agents are autonomous. Agents embody stronger notion of autonomy than objects, and in particular, they decide for themselves whether or not to perform an action on request from another agent. Secondly agents are smart. They are able to have flexible behavior like being reactive, pro-active, and social, while the standard object model has nothing to say about such types of behavior. Thirdly agents are active. A MAS is inherently multi-threaded, in that each agent is assumed to have at least one thread of active control. Moreover, agents carry out the task to maximize its utility value.

An agent-based system is a system in which the key abstraction used is that of an agent. In principle, an agent-based system might be conceptualized in terms of agents, but implemented without any software structures corresponding to agents at all. Object-oriented software, on the other hand, allows the possibility to design a system in terms of objects, but to implement it without the use of an object-oriented software environment. However, this would be at best unusual, and at worst, counter-productive. A similar situation exists with agent technology. Therefore an agent-based system is expected to be both designed and implemented in terms of agents.

‘Agent-based system’ vs. ‘expert system’

Expert systems typically disembodied expertise about some domain of discourse. Take for example MYCIN in (Spiegelhalter, Dawid et al. 1993), which knows about blood diseases in humans. It has a wealth of knowledge about blood disease, in the form of rules. A doctor can obtain expert advice about blood diseases by giving MYCIN facts, answering questions, and posing queries.

The main differences between the agent and expert system are as follows. Agents are situated in an environment while expert system is not, like MYCIN is not aware of the world. Agents act while expert system does not, like MYCIN does not operate on patients. However, it has to be noticed that some real-time, typically process control, expert systems are agents.

1.2.2 Agent Software Development

The agent-based approach to software systems development views these autonomous software agents as components of a much larger business function in (Faltings 2000). The main benefit of viewing them from this perspective is that the software components can be integrated into a coherent and consistent software system in which they work together to better meet the needs of the entire application.

In the software development history, agent oriented programming is an innovation of the last decade, with specific techniques for specification, implementation and verification in (Jennings, Faratin et al. 2001), which help the programmer to analyze and design a complex software for distributed problem domains. Originally, as described in (Parunak 2000), the basic unit of software was the complete program, where, the algorithm, the programming code, and data were responsibilities of the programmer. At the next stage of software evolution programs could be designed in smaller packages, like loops and subroutines, and although the code was somehow encapsulated in subroutines, it had to be called externally to be executed. Within object-oriented programming, there are localized objectives with abilities like inheritance. Software agents take the next step of giving each object the ability to be autonomous, reactive, proactive and social. The software components are communicative and have their own thread of control and internal goals.

Agent theorists have developed different kinds of theories in order to conceptualize the properties that the autonomous software components should embody and the way they should reason based on their representations. These specifications can assist the software engineer to formally develop the software system in a structured and efficient way.

Agents as Intentional Systems: An agent can be described by an intentional instance. The attitudes that are required in order to represent agents can be grouped into information attitudes and pro-attitudes. The first involves the beliefs or knowledge that the agents hold and the second involves the desires, intention, obligations, commitments, choices etc. of the agents. Software components that embody information within these terms are called the Belief, Desire, and Intention agents (BDI) in (Rao and Georgeff 1995) and (Georgeff, Pell et al. 1999), and the combination of these attitudes has been an issue of debate over the last few years.

Possible world semantics: This specification uses possible worlds from the semantics of model logic, and linked together via accessibility relations (facts and/or rules that specify the connection between two different worlds) (Moreira, Vieira et al. 2004) (Steels and

Hanappe 2006); researchers have overcome many problems by seeing agents' belief characterized as a set of possible worlds.

Alternatives to possible worlds: In order to solve difficulties that arise from the previous theories, Levesque has attempted to develop an alternative theory (Levesque 1984), where there is a distinction between explicit and implicit belief. Also, Konolige, with his deduction model (Konolige 1986), tried to model the beliefs, by representing them in a database and having a logical inference mechanism.

1.2.3 Multi-agent System

MAS is a system composed of multiply interacting intelligent agents. Computational intelligence research originally focused on complicated, centralized intelligent systems with expertise in particular domains. However, as researchers address increasingly complex and distributed applications, single-agent system could not meet the requirements and the needs for Multi-agent System (MAS) are becoming apparent (Jennings, Sycara et al. 1999) (Ren and Williams 2003).

MAS can be used to solve problems that are difficult or impossible for an individual agent to solve. The MAS systems are based on the idea that a cooperative working environment comprising synergistic software components can cope with problems which are hard to solve using the traditional centralized approach to computation. Smaller software entities – software agents – with special capabilities (autonomous, reactive, pro-active and social mentioned above) are used instead to interact in a flexible and dynamic way to solve problems more efficiently (Weiss. 1999).

Research in MAS is concerned with study, behavior, and construction of collection of possibly preexisting autonomous agents that interact with each other and their environments. The target of MAS is to demonstrate how goal-directed, robust and optimal behavior can arise from interactions between individual autonomous intelligent software agents. Developing separate modules, where each one provides a solution, and allowing them to co-operate and exchange information in order to solve the larger problem makes the problems solving process easier to manage. This is something that would not be feasible by merely integrating the knowledge and inference mechanisms into a single software component.

The agents in MAS has several important characteristics which includes that agents are at least partially autonomous; each agent has incomplete knowledge or capabilities for solving the problem; there is no global control on problem-solving activities; data are distributed; computation is asynchronous.

MAS has been developed and used for various applications ((Abraham, Franke et al. 2003) (Abraham, Köppen et al. 2003) (Blum and Merkle 2008) (Garcia 2003) (Moreno and Nealon 2003)). MAS can be applied to proactive, reactive, and context sensitive information retrieval; decision support using distributed, heterogeneous data and knowledge sources (e.g. in health care, defense, collaborative scientific discovery, etc.); distributed design and manufacturing in virtual enterprises; electronic commerce; adaptive self-managing complex dynamic systems (e.g. large communication networks, power systems, transportation systems); adaptive user interfaces; mobile and ubiquitous computing; mail and message handling; collaborative work environments; system monitoring, intrusion detection, and countermeasures; knowledge discovery from heterogeneous distributed data and knowledge source (e.g., genome databases, protein databanks, laboratories).

Over the last decade, a number of successful applications have appeared in the field of business process, electricity management, control, transportation, logistics, networking, and mobile technologies.

Based on the description of intelligent systems given in the previous paragraph and moving to the next generation of MAS, researchers have followed the approach of extending existing intelligent systems methodologies by including the aspects of agents (Brazier, DuninKeplicz et al. 1997), (Iglesias, Garijo et al. 1999),(Weiss 1999). Although there is not a formal framework for MAS development, due to the dependence on application domains, it has been agreed that the construction of MAS requires a different approach from that of conventional software system development (Maes 1990). Certain issues like responsibilities and tasks assignment to agents need to be considered. These form the links between the problem domain and the development of individual software components and the MAS as a whole (Wooldridge, Jennings et al. 2000) and include:

Application domain functional analysis: identification of domain functions, task decomposition and distribution.

Software components design: definition of goals, role assignment (description, permissions, responsibilities, and commitments) and internal knowledge (explicit or implicit).

MAS design: co-operation, communication, control, planning, commitments, conflicts, and negotiation.

Following these three general stages and embodying abilities of communication and co-operation between the software components, the MAS will be able to process the functions depending on the goals to be accomplished within a certain problem domain.

An agent-based system may contain one or more agents. There are cases in which a single agent solution is appropriate. However, the multi-agent case, which means the system is designed and implemented as several interacting agents, is arguably more general and more interesting from a software engineering standpoint. The MAS systems are ideally suited to representing problems that have multiple problem solving methods, multiple perspectives and multiple problem solving entities. Such systems not only have the traditional advantages of distributed and concurrent problem solving, but also have the additional advantage of sophisticated patterns of interactions. It is the flexibility and high-level nature of these interactions which distinguishes MAS from other forms of software and which provides the underlying power of the paradigm. Apart from the usual benefits provided by distributed systems, MAS has the substantial benefit of containing the spread of uncertainty, with each agent locally dealing with the problems created by an uncertain and changing world.

To launch the agent technology successfully, researchers and business leaders need to reduce the costs and risks associated with adopting the technology. The costs and benefits that should be evaluated include (1) assessment of the technological and business issues involved in the development and operation of agent technologies, (2) understand the concerns and constraints of adopters, and (3) guide commercial development efforts by technology providers, agent standards, efforts by suppliers and users, and research efforts by agent computing researchers.

As the research in intelligent systems has progressed steadily over the past decade, it has become increasingly clear that there are classes of complex problems which cannot be solved by a single system in isolation and they require several systems to work together interactively in a cooperative framework. Furthermore, there are heterogeneous intelligent systems that were built in isolation, and their cooperation is necessary to achieve a new common goal. In situations where multiple agents are acting on different goals in the same environment, cooperation may be beneficial to all agents.

In cooperative MAS, some of the agents have the notion of global utility that they need to maximize, while some others have no global notion of utility and autonomous agents in them have to maximize their own utility functions. However, these two extreme cases may violate either the global utility or the local utility. Most of the time, it is expected to achieve a good system performance and at the same time protect the local agents' self-interest and integrity. As a result, how agents in MAS manage to do this is a challenging task. One promising way is through agent learning and coordination.

1.3 Problem Statement

For agents in a cooperative MAS to behave adaptively, they need to learn from and coordinate with other agents and their environment. To effectively and efficiently learn and coordinate, there needs to be a corresponding agent model and MAS architecture. Based on the hardware, advanced learning and coordination methods are proposed.

1.3.1 Scope of research

As the developments in MAS has been so rich and vast, encompassing the wide array of research topics and problems in MAS such as learning, reasoning, model, architecture, communication e.t.c is impossible. This thesis is focus on three axes on how to build an adaptive MAS, namely agent model, system architecture, coordination mechanism.

To effectively and efficiently build an adaptive MAS, the agents' model and the system's architecture are very important, for the framework of MAS setting the limits of the system performance potentials. A Bayesian-Networked Believe, Desire, Intention (BN-BDI) agent model and a corresponding hierarchical MAS architecture are presented in this thesis. Based on the agent model and the system architecture, the time constraint task-based model is proposed for coordination of the cooperative MAS. Through the learning and coordination process, agents can therefore be adaptive.

In short, in this thesis the learning and coordination of adaptive agents are investigated in a cooperative MAS system and proposed an advanced agent model as well as a supporting MAS framework.

1.3.2 Objectives

There are four main objectives of this thesis.

1. Propose an agent model for the cooperative MAS
2. To compare the existing MAS architecture and propose an MAS architecture
3. Propose an alternative coordination mechanism for the specific MAS architecture with the novel agent model
4. To verify the proposed framework. In the thesis, a foraging problem is used as the context of simulation

1.3.3 Assumptions

The assumptions that are made about the nature of the problem are summarized below.

Each agent has limited knowledge about its environment

Agent skills and knowledge differ.

Agent behavior is tightly coupled with the environment

Cooperation is required to solve tasks. Agents want to coordinate with each other so that they are conflict-free. That is, there should be no agent with contradictory goals that compete with other agents.

Agents have a common language for describing aspects of the world they sit in, so that they can know they are talking about the same thing. Otherwise, they would have no way of coordinating with each other.

Agents are able to communicate freely with each other so as to implement various synchronization decisions made at coordination time so that they can carry out their coordinated plans at run-time

Plan coordination is an offline process that takes place after agents produce their individual plans, but before agents execute their plans. This assumption is made to better frame the coordination problem as a straightforward optimization problem.

The problem of finding the globally optimal Multi-agent coordination plan for a given set of agents and goals is a fundamentally intractable problem, especially as the agents' number scales. Multi-agent coordination which is not guaranteed to produce a globally optimal utility but can be solved tractably under certain assumptions is design tradeoff between the quantity of the resultant Multi-agent coordination plan and the required computational overhead of producing such a plan.

1.3.4 Approach

Given this problem description, efficient agent models and system architecture are developed to enable agents to coordinate and learn with each other quickly. Firstly, a BN-BDI agent model is presented. It is a hybrid architecture that has the merits of both the deliberative and reactive architecture. The BDI part maintains an explicit representation of the agents' world. The BN part measures the uncertainty that an agent is facing and the dependent relationship it has with other agents. The BN-BDI agent can learn other agent's model, their preferences, their beliefs and their capacities. A hierarchical MAS architecture consisting of the BN-BDI agents is formed. Agents with the similar characteristics or capacities constitute a group in one layer and some agents act as coordinators in the upper layer. Agents use time-constraint contract net to communicate and coordinate with each other. Since there are a hierarchical MAS architecture and BN-BDI agent model, these architectures greatly reduced the amount of information exchanged among agents, and the information exchanged among agents grows not exponentially to the number of the agents. This greatly reduces the complexity of communication and coordination.

1.4 Organization of the Thesis

This thesis is organized as follows: In chapter 2 a literature survey of agent architecture and coordination mechanisms are presented. Chapter 3 focuses on the agent model and corresponding cooperative MAS architecture. The proposed BN-BDI agent model hierarchical MAS frameworks are presented. Chapter 4 investigates the coordination in the cooperative BN-BDI MAS and the proposed time constraint task-based coordination mechanisms. Chapter 5 is the simulation of a foraging problem. The proposed model and coordination mechanism are tested in the example. Results and analysis are given. It is shown that the novel agent model and the novel coordination mechanism are effective and efficiency. Final chapter is about the summary and conclusions.

Chapter 2 Literature Review

A distinguishing feature of MAS is the fact that the decision making of the agents can be distributed. This means that there is no central controlling agent that decides what each agent must do at each time step, and each agent is to a certain extent responsible for its own decisions. The main advantages of such a decentralized approach over a centralized one are efficiency, due to the asynchronous computation, and robustness in the sense that the functionality of the whole system does not rely on a single agent. In order for the agent to be able to take their actions in a distributed fashion, appropriate coordination mechanisms must be developed. In this chapter, the agent model and architectures and the cooperation and coordination mechanisms of MAS will be reviewed.

2.1 Agent Architectures

Intelligent agent has been a key concept in both AI and the main stream of computer science. The agent-based technology plays an important role in software engineering. The conception of multiple autonomous problem solvers interacting in various ways to achieve individual and system goals is a useful software engineering abstraction. In the past decades, the theory and application of agents have been well developed. Although the abstraction is useful in that it enables software engineers to do more or to do things more cheaply, to design and build the agent systems is difficult.

MAS has all the problems associated with building traditional distributed, concurrent systems, and has the additional difficulties which arise from having flexible and sophisticated interaction between autonomous problem solving components. For these reasons, most extant agent system applications are built by, or in consultation with, designers and developers who are themselves active in the agent research community. So this situation lays two major technical impediments to the widespread adoption of agent technology. One is the lack of a systematic methodology enabling designers to clearly specify and structure their applications as MAS. The other is that the lack of widely available industrial-strength MAS toolkits.

As for agent theory, many models and architectures for characterizing agents have been proposed. To date, much architecture has been implemented in a rather ad hoc manner. Since different environment types have various architectures, it is quite hard to evaluate one agent's architecture against another.

When classified in abstract agent architectures, there are purely reactive agents, Simple reflex agents, Perception-limited agents, Model-based reflex agents, Agents with internal state, Goal-based agents, Utility-based agents, and Learning agents, etc. When classified by concrete agent architectures, there are logic-based architectures, reactive architectures, BDI architectures, and layered architectures, etc.

Agent architectures represent the move from theoretical specification to the software agents' implementation:

Deliberative Architectures: the term 'deliberative agent' means a specific type of symbolic architectures. This is based on the physical symbol hypothesis of Newell and Simon, according to which intelligent activity in either humans or machines is achieved through the use of patterns of symbols, operations on these patterns and search (Newell and Simon 1976). This means that agents should maintain an explicit representation of their world, which can be modified by some form of symbolic reasoning. Research has explored the model of BDI agents. Although this approach is theoretically attractive, it is very hard to achieve in practice in real time application.

Reactive Architectures: this kind of architecture is based on Brook's approach 'Reasoning without representation'(Brooks 1991). This architecture aims to build autonomous mobile robots, which can adapt to changes in their environment and move in it, without any internal representation. The agents make their decisions at run time, usually based on very limited amount of information and simple situation –action rule. Decisions are based directly on sensory input.

Hybrid Architectures: Many researchers suggested that a combination of classical and alternative approaches would be more appropriate, as it would combine the advantages of both approaches and avoid the disadvantages. Some successful examples are the Procedural Reasoning System (PRS) (Georgeff 1989), TOURINGMACHINES (Ferguson 1992), COSY (Burmeister B. 1992) and INTERRAP (Muller J. P. 1995).

In logic-based architectures, decision making is realized through logical deduction. It is the traditional AI approach that intelligent behavior can be created in a system that manipulates symbols. In reactive architectures, decision making is implemented in some form of direct mapping from situation to action. Intelligent behavior emerges from the interaction of various simple behaviors, and intelligent behavior is not disembodied. It has to be a product of the interaction that the agent maintains with its environments. In BDI architectures, decision making depends upon the manipulation of data structures representing the beliefs, desires, and intentions of the agent. In layered architectures, decision making is realized via various software layers, each of which is more or less explicit reasoning about the environment at different levels of abstraction.

2.1.1 Belief-desire-intention Agent

Of the various agent architectures which have been proposed, BDI(Georgeff, Pell et al. 1999) is probably the most mature and has been adopted by a few industrial applications.

In this thesis, beliefs represent the information an agent has about the state of the environment being monitored, which is updated appropriately after each sensing action. The desires denote the objectives to be accomplished, including what priorities or payoffs are associated with the various objectives. The intentions represent the currently chosen courses of action and commitments are the conditions the agent is committed to maintain.

The BDI agent model is developed based on the theory proposed by (Bratman 1987). Unfortunately, the existing BDI systems have some shortcomings that prevent the mobility of agents (Rao and Georgeff 1995). The architecture lacks a paradigm for concurrency control among intentions performing conflicting operations, such as trying to manipulate the same set of beliefs at the same time. In theory, this problem is resolvable by writing context specific meta-level policies. However, in addition to being impractical, writing meta-plans which discover and handle race conditions in real-time is a very challenging task.

2.1.2 Bayesian Network

Bayesian network (BN) is a directed acyclic graph (DAG) of nodes representing variables, arcs representing probabilistic dependency relations among the variables and local probability distribution for each variable given values of its parent (Bacchus and Grove 1995).

A Bayesian network approach to self-organization and learning is introduced for use with intelligent agents in (Sahin 2000). Bayesian networks, with the help of influence diagrams, are employed to create a decision-theoretic intelligent agent. Influence diagrams combine both Bayesian network and utility theory. In (Sahin 2000), an intelligent agent is modeled by its belief, preference, and capabilities attributes. Each agent is assumed to have its own belief about its environment. The belief aspect of the intelligent agent is accomplished by a Bayesian network. The goal of an intelligent agent is said to be the preference of the agent and is represented with a utility function in the decision theoretic intelligent agent. Capabilities are represented with a set of possible actions of the decision-theoretic intelligent agent. Influence diagrams have utility nodes and decision nodes to handle the preference and capabilities of the decision-theoretic intelligent agent, respectively.

Learning is accomplished by Bayesian networks in the decision-theoretic intelligent agent. Because intelligent agents will explore and learn the environment, the learning algorithm should be implemented online. In (Heckerman 2008), an online Bayesian network learning method was proposed.

Self-organization of the intelligent agents is accomplished because each agent models interaction with other agents by observing their behavior. Agents have belief, not only about environment, but also about other agents. Therefore, an agent makes its decision according to the model of the environment and the model of the other agents. Even though each agent acts independently, they take the other agents' behaviors into account to make a decision(Zheng and Pavlou 2010). This permits the agents to organize themselves for a common task.

2.1.3 Bayesian Learning

A Bayesian network is made up of structures and parameters. It can be built either through domain knowledge or from data (Heckerman 2008). The first approach is called Bayesian network construction from domain knowledge and the second one is called Bayesian network learning from data. Constructing Bayesian networks from domain knowledge is far too subjective to apply, because the experts' judgment often leads to inconsistent networks. Moreover, it is difficult to elicit dependence relationship and variable probabilities from domain experts. Consequently, much effort has been made to devise the engines of Bayesian network learning from data.

In general, approaches for learning Bayesian network are categorized according to two ways: whether the structure is known and whether the data set is complete. When the structure is known, the problem becomes a parameter learning problem. Otherwise, the problem becomes a structure learning problem when the structure is unknown beforehand. Of course, it is possible to learn structures and parameters together from data. However, learning structure is much more difficult than learning parameters. An incomplete data set complicates the learning problem while a complete data set alleviates the learning challenge. The problem addressed leads toward the issue of learning Bayesian network structures from complete data set, which is also called as learning Bayesian network structures from data.

2.2 Cooperation and Coordination

A typical situation where coordination is needed is among cooperative agents that form a team, and through this team they make joint plans and pursue common goals (Klavins 2004). Cooperation is a key MAS concept. (Edmund H. Durfee 1989) have proposed four generic goals for agent cooperation: (1) increase the rate of task completion through parallelism; (2) increase the number of concurrent tasks by sharing resources (information, expertise, devices, etc.); (3) increase the chances for task completion by duplication and possibly using different modes of realization (4) decrease the interferences between tasks by avoiding the negative interactions.

Within MAS, cooperation is required to give robust global behavior, and is achieved through software components' communication. There are different methods for achieving coordination and cooperation (Schumacher 2001) (Scerri, Vincent et al. 2005) (Koulinitch and Sheremetov 1998) (Mataric 1998) (Ho and Kamel 1998) (Crowston 1991), which have formed three main approaches:

Cooperative interaction: This occurs when agents interact to assist each other in achieving their goals more efficiently. This coordination has to be built by the developer of the software, in terms of goals, roles and the relationship between them. The strategy can be complex, with rewards assigned to agents (David Carmel 1996), or with committed teams to achieve a goal by exchanging partial results (Norman Carver 1991), which might raise the need for sharing resources either centrally or in a distributed fashion (Zhongyan L. 1999).

Contract-based cooperation: This approach uses one of the common auction strategies, when there is some conflict between the agents. The common auction strategies include sealed-bid auction, English auction and Dutch auction. Sealed-bid auction: each agent submits a bid without knowing the bids of the other agents. The contract is awarded to the cheapest bidder. English auction: bids are accepted sequentially. Each new bid must be cheaper than the currently cheapest bid. The contract is awarded to the final bidder who offered the cheapest bid. Dutch auction: the initiator invites potential contractors to bid at a given price, which is systematically increased until a bid is received. The contract is awarded to the first bidder.

The approach that has most commonly been used within MAS is the contract-net protocol (Davis and Smith 2003) (He, Leung et al. 2003) which is based on a sealed –bid auction and has been proven to be more appropriate than blackboard architectures (Parunak, Ward et al. 1999). The agent cooperates by committing to a goal, which makes it able to predict the actions of the other agents contracted to it.

Negotiated cooperation: During cooperation conflicts might arise if resources are limited, in order for all the agents in MAS to carry out their actions (Sycara 1990). These conflicts can be solved with negotiation between the software components or the development of a software management mechanism. Using the latter means behavior rules have to be defined, while negotiation can contribute to the system's equilibrium in a dynamic fashion (Gilad Zlotkin 1993)

Weiss describes a learning system of how agents can learn to coordinate their actions (Weiss. 1999). Weiss makes a number of assumptions, including (1) each agent has limited knowledge about its environment (2) cooperation is required to solve tasks, (3) agent actions can conflict with each other (4) agent skills and knowledge differ, and (5) agent behavior is tightly coupled with the environment. Weiss' approach involves the use of two algorithms for collective learning, which are called ACE and AGE (these acronyms stand for "Action Estimation" and "Action Group Estimation"). In the ACE algorithm, the MAS learning consist of a repeated execution of three steps, which is called action determination, competition, and credit assignment. The application domain that Weiss addresses is the learning sequences of applicable actions that accomplish a given mission. Parker describes a similar learning system and his application domain is having agents select actions that allow them to robustly accomplish a set of independent subtasks in a minimal amount of time.

Cooperative MAS and cooperative mobile robotics share similar methods to coordinate the agents and robotics. The amount of research in the field of cooperative mobile robotics has grown substantially in recent years (J Deneubourg 1990). This work can be broadly categorized into two groups: swarm-type cooperation and "intentional" cooperation. A number of researchers have studied the issues of swarm robotics. (Blum and Merkle 2008) describes simulation results of a distributed sorting algorithm. (Theraulaz, Gervet et al. 1991) extracts the cooperative control strategies, such as foraging, from a study of Polistes wasp colonies. (Steels and Tokoro 1995) presents the simulation studies of the use of several dynamical systems to achieve emergent functionality as applied to the problem of collecting rock samples on a distant planet. (Drogoul and Ferber 1992) describe simulation studies of foraging and chain-making robots. In (Mataric 1998), describes the results of implementing group behaviors such as dispersion, aggregation, and flocking on a group of physical robots. (Beni and Wang 1989) describe methods of generating arbitrary patterns in cyclic cellular robotics. (Kube and Zhang 1996) present the results of implementing an emergent control strategy on a group of five physical robots performing the task of locating and pushing a brightly lit box. (Stilwell and Bay 1993) present a method for controlling a swarm of robots using local force sensors to solve the problem of the collective transport of a palletized load. (Arkin 1998) presents research concerned with sensing, communication, and social organization for tasks such as foraging. The CEBOT work of (Beni and Wang 1989), which stands for Cellular Robotic System, described in and many related papers, has many similar goals to other swarm-type multi-robotic systems; however, the CEBOT robots can be one of a number of robot classes, rather than purely homogeneous.

The above mentioned approaches are designed strictly for homogeneous robot teams, in which each robot has the same capabilities and control algorithm. Additionally, issues of efficiency are largely ignored. However, in heterogeneous robot teams, not all tasks can be performed by all team members, and even if more than one robot can perform a given task, they may perform that task quite differently, thus the proper mapping of subtasks to robots is dependent upon the capabilities and performance of each robot team members. This additional constraint brings many complications to a workable architecture for robot cooperation, and must be addressed explicitly to achieve the desirable level of cooperation.

The second primary area of research in cooperative control deals with achieving “intentional” cooperation among a limited number of typically heterogeneous robots performing several distinct tasks. In this type of cooperative system, the robots often have to deal with some sort of efficiency constraint that requires a more directed type of cooperation that is found in the swarm approach described above. Furthermore, this second type of mobile robotic mission usually requires that several distinct tasks be performed. These missions thus usually required a smaller number of possibly heterogeneous mobile robots involved in more purposeful cooperation. Although individual robots in this approach are typically able to perform some useful task on their own, groups of such robots are often able to accomplish missions that no individual robot can accomplish on its own. Key issues in these systems include robustly determining which robot should perform which task so as to maximize the efficiency of the team and ensuring the proper coordination among team members to allow them to successfully complete their mission.

In most of the existing work, heterogeneous physical robots use a traditional artificial intelligence approach, which breaks the robot controller into modules for sensing, world modeling, planning, and acting (hence, the sense-model-plan-act paradigm), rather than the functional decomposition of behavior-based approaches. (Kweon, Kuno et al. 1992) describes one such sense-model-plan-act control architecture which includes three layers of control: the planner level, which manages coordinated protocols, decomposition tasks into smaller subunits, and assigns the subtasks to a network of robots; the control level, which organizes and executes a robot's tasks, and the function level, which provides controlled activity. The paper reports on the implementation of this architecture on two physical mobile robots performing convoying and box pushing. In both of these examples, one of the robots acts as a leader, and the other acts as a follower.

(Parker 1998) describe another sense-model-plan-act architecture which includes a task planner, a task allocator, a motion planner, and an execution monitor. Each robot obtains goals to achieve either based on its own current situation or via a request by another team member. They use Petri Nets for interpretation of the plan decomposition and execution monitoring.

(Asama, Ishida et al. 1989) describe a decentralized robot system called ACTRESS, addressing the issues of communication, task assignment, and path planning among heterogeneous robotic agents. Their approach revolves primarily around a negotiation framework which allows robots to recruit help when needed. They have demonstrated their architecture on mobile robots performing a box pushing task.

Wang addresses a similar issue to that addressed in this thesis- namely, dynamic, distributed task allocation when more than one robot can perform a given task (Wang 1994). The paper proposes the use of several distributed mutual exclusion algorithms that use a "sign-board" for inter-robot communication. These algorithms are used to solve problems including distributed leader finding, the N-way intersection problem, and robot ordering. However, their research does not address issues of dynamic reallocation due to robot failure and efficiency issues due to robot heterogeneity.

(Parker 1998) proposes a hierarchical subdivision of authority to address the problem of cooperative fire-fighting. They describe their Phoenix system, which includes a generic simulation environment and a real-time, adaptive planner. The main controller in this architecture is called the Fireboss, which maintains a global view of the environment, forms global plans, and sends instructions to agents to activate their own local planning.

Ohko et al. describe a learning system, called LEMMING, which learns knowledge quite similar to that learned in L-ALLIANCE(Parker 1998). In their system, however, this knowledge is used by a case-based reasoner for reducing the communication flow between distributed agents. These distributed agents can often use point-to-point communication rather than broadcast communication to recruit help directly from those agents known to have the capabilities to perform a given task, thus reducing the overall communication traffic. They present results from a simulation application involving the movement of objects from one location to another by a team of distributed agents.

Bing Liu describes a heuristic, centralized method for coming up with a good job shop schedule (Liu 2003). The centralized scheduler has complete information of the tasks to be done and the capabilities of the “agents” (which in this case are machines). The system strives to deal with many constraints imposed by the organization, by precedence requirements, by resource requirements, or by preferences. It is assumed that the scheduler will always be operational and is not itself an agent, and that it fully knows the state of each agents and the state of the tasks to be performed. Sensory and effector uncertainty is not an issue. This type of system is not designed for real mobile robotic systems, since sensing and action are quite noisy, and since no single agent can be expected to know the full state of the environment (which includes the task states and the agent states) at all times(Parker 2000).

Learning Real-Time A* (LRTA*), introduced by Korf, is a generic heuristic search algorithm(Korf 1990), which is applicable to real-time path search for fixed goals. LRTA* builds and updates a table containing admissible heuristic estimates of the distance from each state in the problem space to the fixed goal state, which are learned in exploration time(Undeger and Polat 2010). In the early runs, the algorithm does not guarantee optimality, but when the heuristic table has converged, the solutions generated become optimal. Although LRTA* is convergent and optimal, the algorithm may find poor solutions in the first run. To solve the problem, Korf also proposed a variation of LRTA*, called real-time A* (RTA*), which gives better performance in the first run, but is lack of learning optimal table values.

2.2.1 Coordination Category

Several works have been proposed, covering different aspects of the problems of coordinating the plans of several agents operating in the same environment. These include scenarios where plan generation is distributed, where planning is centralized and plan execution is distributed, or where both planning and execution are distributed.

Coordinating the activities of multiple agents is widely regarded as the central problem of MAS research. The successful mechanisms contain three facets (Durfee, Lesser et al. 1989). (1) a structure within which agents can interact in predictable ways; (2) agents can flexibly operate in dynamic environments and can cope with their inherently partial and imprecise view of the community; (3) agents have appropriate knowledge and reasoning capabilities to intelligently use the structure and flexibility.

Since the performance of the system can be affected by the way of coordination, there is a need to evaluate proposed algorithms by a measure of coordination complexity. For the scalability of multi-agent algorithms, it is assumes that the less coordination a MAS requires, the better it should scale to large numbers of agents (Klavins 2002).

For agents in MAS to achieve a given task, the agents need to coordinate to share some information. Because more sharing requires more resources, like time and communication bandwidth, the amount of information that must be shared determines how coordinated a task is.

Agents need to cooperate with other agents to finish the problem that they could not do themselves. During the cooperation, coordination and negotiation are needed.

2.2.2 Coordinate through Negotiation

The word negotiation in general refers to communication processes that further coordination and cooperation. The process of negotiation may be of many different forms, such as auctions, protocols in the style of the contract net, and argumentation.

The research works on negotiation can be divided into three broad topics: negotiation protocols, negotiation objects and agents' reasoning models. Negotiation protocols are the set of rules that govern the interaction. This covers the permissible types of participants, the negotiation states, the events that cause state transitions and the valid actions of the participants in particular states. Negotiation objects are the range of issues over which agreement must be reached. These may be single issues, such as price, or multiple issues relating to price, quality, timing, etc. The agents' reasoning models provide the decision making apparatus by which participants attempt to achieve their objectives. The sophistication of the model is determined by the protocol used, the nature of the negotiation object, and the range of operations that can be performed on it.

Much theoretical work has been accomplished for intentional agent control by the Distributed Artificial Intelligence (DAI) community ((Weiss. 1999) contains many examples). In most of such work, the issue of task allocation has been the driving influence that dictates the design of the architecture for cooperation, since the selected approach to task allocation invariably restricts the potential solutions to other issues of cooperation, such as conflict resolution.

Typically, the DAI approaches use a distributed, negotiation-base mechanism to determine the allocation of tasks to agents, using a variety of proposed protocols. Under these negotiation schemes, no centralized agent has full control over which tasks individual team members should perform. Instead, many agents know which subtasks are required for various portions of the mission to be performed, along with the skills required to achieve those subtasks. These agents then broadcast a request for bids to perform these subtasks, which other agents may respond to if they are available and want to perform these tasks. The broadcasting agent then selects an agent from those that respond and awards the task to the winning agent, who then goes on to perform that task, recruiting yet other agents to help if required.

Although DAI work has demonstrated success in a number of domains (e.g. distributed vehicle monitoring and distributed air traffic control), the proposed solutions have rarely been demonstrated as directly applicable to situated agent teams (i.e. robotic), which have to live in and react to a dynamic and uncertain environment using noisy sensor and effectors, and a limited bandwidth, noisy communication mechanism. They typically rely on unrealistic “black boxes” to provide high-level, perfect sensing and action capabilities. Furthermore, in the approaches of the previous subsection, these DAI approaches typically ignore or only give brief treatment to the issues of agent performance of those tasks after the agents have been allocated. Such approaches usually assume the agents will eventually accomplish the task they have been assigned, or that some external monitor will provide information to the agents on dynamic changes in the environment or in agent performance. However, to realistically design cooperative approaches for agents, mechanisms within the software control of each agent that allow the team members to recover from dynamic changes in their environment or in the agent system should be included.

2.2.3 Coordination using Contract net

Contract net protocol (CNP) provides a flexible way for agent interaction (Smith 1980). Agents all have their own goals, are self-interested, and have limited reasoning resource. In contract net protocol for distributed problems solving, there are managers and contractors. Managers announce tasks, evaluate bids, make contracts, and monitor results. Contractors make bids, commit to contracts, and execute contracts. The same agent can be a manager as well as a contractor in different negotiations (Peng, Gao et al. 2008).

Although CNP was considered by Smith and Davis and many DAI researchers to be a negotiation principle, it is a coordination method for task allocation. CNP enables dynamic task allocation, allows agents to bid for multiple tasks at a time, and provides natural load balancing, since busy agents need not bid. Its limitations are that it does not detect or esolve conflicts: The manager does not inform nodes whose bids have been refused; agents cannot refuse bids; there is no pre-emption in task execution, so time-critical tasks may not be attended to; it is communication intensive.

Extensions to CNP have been made by Sandholm and Lesser(Sandholm 1993), where de-commitment penalties were introduced, and by Sycara (Sycara 1997), where the theory of financial option pricing has been used to achieve flexible contracting schemes in uncertain environment.

2.3 Summary

In the first half of the chapter, agent model and architecture have been reviewed. In the other half of the chapter, the literature of Multi-agent coordination has been reviewed. There are many methods available and can be classified according to different standards. For the coordination of agents in cooperative MAS, a hybrid approaches based on Bayesian network and BDI agent has been proposed in the thesis. The novel agent model combines the advantages of model-based and model-free approaches. Details of the BN-BDI agent model will be discussed in chapter 3.

Chapter 3 A BN-BDI Agent-based Cooperative Multi-agent System

Agents in MAS need to coordinate with their fellow agents to improve the system performance. The agent model and corresponding system architecture are very important for system coordination. Compared to the long-established areas of interaction protocol and agent communication language research, the development of agent architectures has received fairly little attention. In this chapter the agent models and the system architecture are the focus and a BN-BDI agent model for cooperative MAS is proposed.

This chapter is organized as following: the BN-BDI agent model will be introduced and the learning process for cooperation is illustrated. Then a hierarchical MAS architecture consisting of the BN-BDI agents is proposed. In the MAS architecture, agents with the similar characteristics or capacities will constitute a group and some agents will act as coordinators.

3.1 Proposed Agent Model

The term agent is used by many people working in different areas, and it can be used to describe several very different kinds of systems. Figure 3-1 illustrates the types of agents' systems from (Nwana 1996).

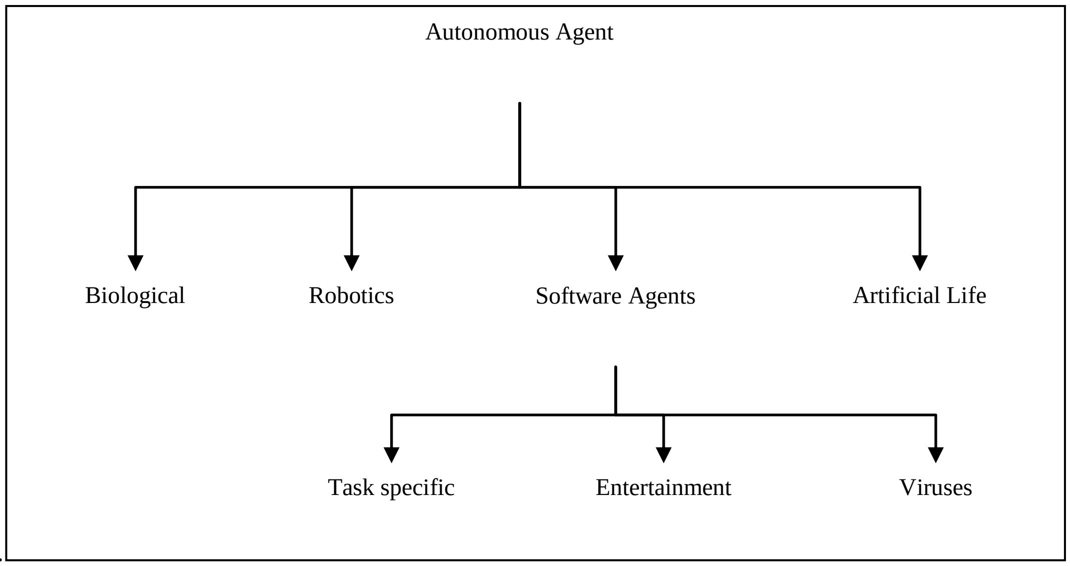


Figure 3-1Autonomous Agent

As the name indicates, BDI agents are characterized by a “mental state’ with three components (Rao and Georgeff 1991; Rao and Georgeff 1995). Intuitively, beliefs correspond to information that the agent has about its environment. Desires represent options available to the agent, which means different possible states of affairs that the agent may choose to commit to. Intentions represent states of affairs that the agent has chosen and has committed resources to. An agent’s practical reasoning involves repeatedly updating beliefs from information in the environment, deciding what options are available, filtering these options to determine new intentions, and acting on the basis of these intentions. A number of BDI agent systems have been implemented, the best-known of which is probably the Procedural Reasoning Systems (PRS).

Figure 3-2 illustrates the abstract architecture of BDI agents. The BDI architecture has been used in some products and a number of applications ranging from air traffic control to air combat simulations, and from telephone call centers to the handling of malfunctions on NASA’s Space shuttle.

```
BDI-interpreter
initialize-state();
repeat
    options :=option-generator(event-queue);
    selected-options :=deliberate(options);
    update-intentions(selected-options);
    execute();
    get-new-external-events();
    get-successful-attitudes();
    drop-impossible-attitudes();
end repeat
```

Figure 3-2 Abstract architecture of BDI agents

Compared with other models of agents, BDI agents provide the essential component necessary to cope with the real world, which is complex and dynamic, and is a place where chaos is the norm, not the exception. However conventional software systems are designed for static worlds with perfect knowledge. What would be of interest are environments that are dynamic and uncertain (or chaotic), and where the computational system only has a local view of the world (i.e., has limited access to information) and is resource bounded (i.e., has finite computational resources). In this situation, BDI agents have all the essential parts to model the MAS.

For BDI agents, Beliefs are essential because the world is dynamic (past events need therefore need to be remembered), and the system only has a local view of the world (events outside its sphere of perception need to be remembered). Moreover, as the system is resource bounded, it is desirable to cache important information rather than to re-compute it from base perceptual data. As Beliefs represent imperfect information about the world, the underlying semantics of the Belief component should conform to belief logics, even though the computational representation need not be symbolic or logical at all.

Desires or goals form another essential component of system state. In computational terms, a Goal may simply be the value of a variable, a record structure, or a symbolic expression in some logic. The important point is that a Goal represents some desired end state. Conventional computer software is “task oriented” rather than “goal oriented”; that is, each task (or subroutine) is executed without any memory of why it is being executed. This means that the system cannot automatically recover from failures, unless this is explicitly coded by the programmer, and cannot discover and make use of opportunities as they unexpectedly present themselves. For example, the reason that recovers from a missed train or unexpected flat tyre is that people know where they are (through their Beliefs) and people can remember to where they want to get (through their Goals). The underlying semantics for Goals, irrespective of how they are represented computationally, should reflect some logic of desire.

Only beliefs and goals are not enough for a system state. The system needs to commit to the plans and sub goals it adopts and must also be capable of reconsidering these at appropriate moments. These committed plans or procedures are called, in the AI literature, ‘Intention’, and represent the third necessary component of system state. Computationally, Intentions may simply be a set of executing threads in a process that can be appropriately interrupted upon receiving feedback from the possibly changing world.

Finally, for the same reasons the system needs to store its current Intentions, because it is resource-bounded, it should also cache generic, parameterized Plans for use in future situations rather than try to recreate every new plan from first principles. These plans, semantically, can be viewed as a special kind of Belief, but because of their computational importance, are sensibly separated out as another component of system state.

In summary, the basic components of a system designed for a dynamic, uncertain world should include some representation of Beliefs, Desires, Intentions and Plans, or what has come to be called a BDI agent.

After a brief literature review in the areas of BDI agent and Bayesian network in intelligent agent, the BDI agent and Bayesian network are found to complement each other and it is worthwhile to study the possibility of applying Bayesian network under the BDI agent framework to build a better learning system, which is named BN-BDI agent model.

The adaptive multi-agent framework proposed in this thesis aims to adaptively balance the global utility as well as the individual agent utility for MAS. The integration of BDI agent model and Bayesian networks can lead to synergistic effects including the capability of dealing with uncertain information in a BDI architecture and the improvement of the agent’s cognitive processes. Each node in a Bayesian network corresponds to exactly one random variable which has a finite set of mutually exclusive states. A BDI agent, whose beliefs are represented through Bayesian networks, believes that a state of a chance variable has a probability of occurrence given a set of conditions imposed by the parent variables. Desires can be assumed to states of affairs that an agent wishes to bring about. The BN-BDI model represents this mental state through states of chance variables that the agent desires to observe. Intensions of agent can also be represented in this way, since they are desires that an agent has committed to achieve.

Assuming that agent’s beliefs are represented through Bayesian networks, the belief updating process correspond to a probabilistic inference. Evidences perceived by the agent play an important role in that process, since up to date beliefs express the current state of world. Thus, agents can recognize circumstances where desires are considered feasible and intentions are considered successful. Up to date beliefs provide support to the deliberative process, responsible for deciding which states of affairs the agent will intend to achieve. In order to improve that process, it takes into account the quantitative and the qualitative aspects of the Bayesian networks to detect incompatible desires and to decide between competing ones.

It specifies a high-dimensional joint probability distribution compactly by exploiting their independence properties (Figure 3-6). Because of BN’s graphical model and relational probabilities, it can easily make the use of incoming data as well as take the advantage of the prior or expert domain knowledge (Heckerman 2008). These properties are very useful in multi-agent environment where the agent needs to learn its environment as well as other agents under the same environment.

If there is an arc from node A to another node B, then variable B depends directly on variable A, and A is called a parent of B. If for each variable $X_i, i = 1\ to\ n$, the set of parent variables is denoted by $parents(X_i)$, then the joint distribution of the variables is the product of the local distributions.

$$\Pr(X_1, \dots, X_n) = \prod_{i=1}^{n} \Pr(X_i | parents(X_i)) \qquad 3\text{-}1$$

If X_i has no parents, its local probability distribution is said to be unconditional, otherwise it is conditional. If the variable represented by a node is observed, then the node is said to be an evidence node.

Conditional independence is represented in the graph by the graphical property of d-separation: nodes X and Y are d-separated in the graph, given specific evidence nodes, if and only if variables X and Y are independent given the corresponding evidence variables. The set of all other nodes on which node X directly depends on is given by X’s Markov blanket.

BN has its unique properties that make it appropriate for MAL. Firstly, BN can deal with incomplete data sets without difficulty because it discovers dependencies among all variables. When one of the inputs is not observed, most models will end up with inaccurate prediction. That is because they do not calculate the correlation between the input variables. BN naturally encodes such dependencies. Secondly, the causal relationships by using BN are learnt. This is important because it will be helpful to understand the problem domain and make some predictions based on the knowledge of causal relationships. Thirdly, BN naturally combines the domain knowledge and data. Prior of domain knowledge is crucially important if one performs a real-world analysis when data is inadequate or expensive. The encoding of causal prior knowledge is straightforward because BN has causal semantics. Also BN encodes the causal relationships with probabilities. These are the reasons that prior knowledge and data can be put together in BN. Finally, BN can avoid over-fitting of data. Models can be "smoothed" in such a way that all available data can be used for training by using Bayesian approach.

Decision Trees and BDI

Informally, a decision tree consists of decision nodes, chance nodes, and terminal nodes, and includes a probability function that maps chance nodes to real-valued probabilities and a payoff function that maps terminal nodes to real numbers. The decision nodes are commonly represented by squares. The chance nodes are represented by circles and the terminal nodes are represented by triangles. A deliberation function, such as maximum or maximizing expected utility is the defined for choosing one or more best sequences actions to perform at a given node.

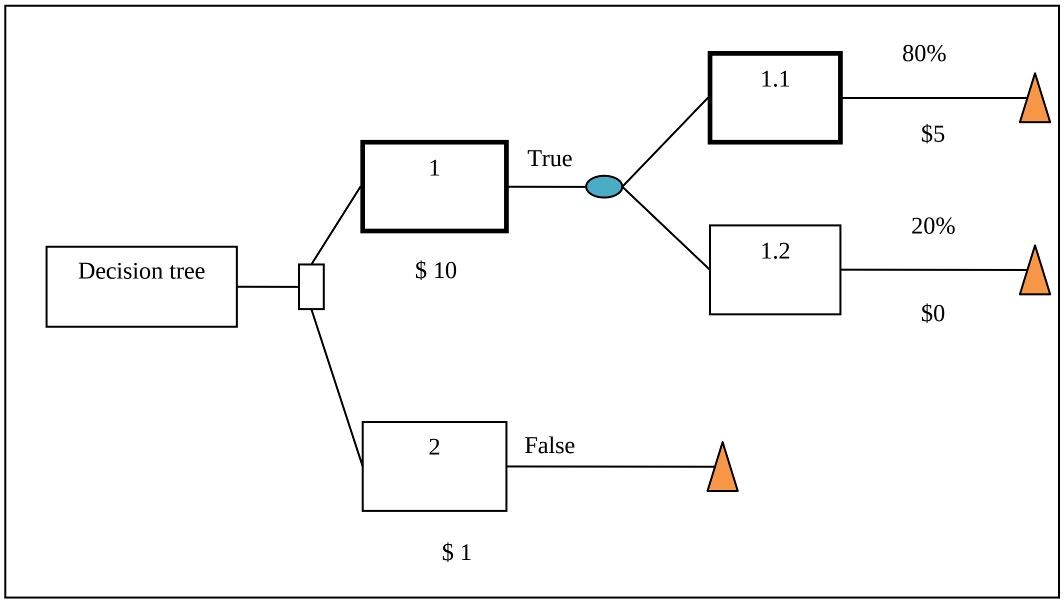


Figure 3-3 Decision Tree

Take Figure 3-3 for example, the decision tree has two choices: true choice will return $10, and false choice will return $1. To choose the true choice, there is 80% opportunity to get extra $5 and 20% opportunity to get nothing more.

Utility

Utility is a unifying, if sometimes implicit, concept in economics, game theory, and operations research. The idea is that each individual can somehow internally estimate the value (or the cost) of executing an action. It can also be called fitness, valuation, and cost. Since the exact formulation varies from system to system.

A decision tree with appropriate deliberation functions is transformed to an equivalent model that represents beliefs, desires, and intentions as separate accessibility relations over sets of possible worlds (Hernandez, El Fallah-Seghrouchni et al. 2004). This transformation provides an alternative basis for cases in which there is insufficient information on probabilities and payoffs and, perhaps more importantly, for handling the dynamic aspects of the problem domain.

Firstly, considering a full decision tree, in which every possible path is represented. Given such a decision tree, traverse from the root node to each arc. For each unique state labeled on an arc emanating from a chance node, a new decision tree is created that is identical to the original tree except that a chance node is removed and the arc incident on the chance node is connected to the successor of the chance node. This process is carried out recursively until there is no chance node left. This yields a set of decision trees, each consisting of only decision nodes and terminal nodes, and each corresponding to a different possible state of the environment. From a traditional possible-worlds perspective, each of these decision trees represents a different possible world with different probability of occurrence. Finally, the payoff function is assigned to paths in a straightforward way. Figure 3-4 and Figure 3-5 illustrate the transform from decision tree model to the possible-world model.

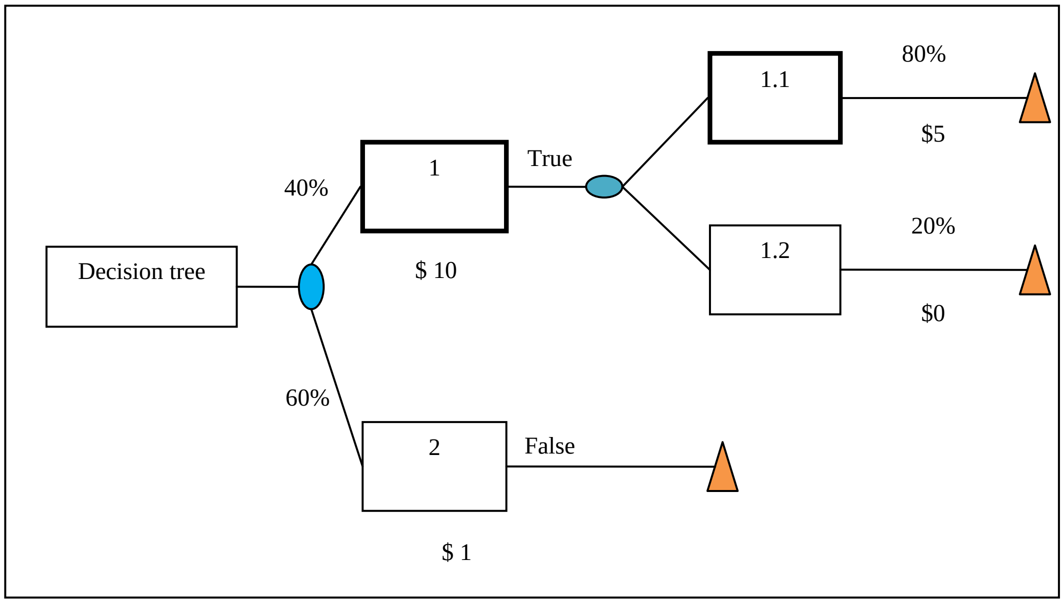


Figure 3-4 A full decision tree

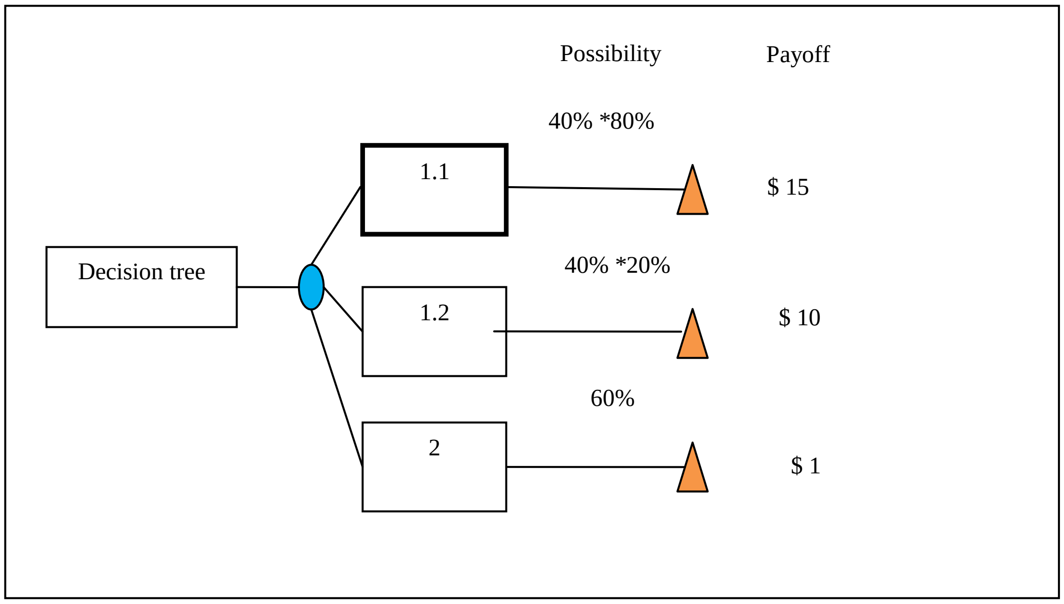


Figure 3-5 Resulting possible-worlds model

The resulting possible-worlds model contains two types of information, represented by the probabilities across worlds and the payoffs assigned to paths. The model is spited out into two accessibility relations, with the probabilities being represented in the belief-accessibility relation and the payoffs being represented in the desire-accessibility relation. At this point in the story, the sets of tree structures defined by these relations are identical, although without loss of generality, it can be deleted from the desire-accessible worlds all paths with zero payoffs.

Given a decision tree and the above transformation, an agent can now make use of the chosen deliberation function to decide the best course of action. It can be formally represented this selected path in the decision tree using a third accessibility relation on possible worlds, corresponding to the intention of the agent. In essence, for each desire-accessible world, there exists a corresponding intention-accessible world which contains only the best course of action as determined by the appropriate deliberation function.

Thus, the possible-worlds model consists of a set of possible worlds where each possible world is a tree structure. A particular index within a possible world is called a situation. With each situation, a set of belief-accessible worlds, desire-accessible worlds, and intention-accessible worlds is associated; intuitively, these are the worlds that the agent believes to be possible, desires to bring about, and intends to bring about, respectively.

Limitations of BDI Agent Model

3.1.1 Influence Diagram

An influence diagram is a graphical modeling language that represents the probabilistic inference and decision analysis model. It describes the dependencies in decision analysis and specifies the states of information for which independencies can be assumed to exist. Influence diagrams augment the Bayesian network framework with the notions of agents that make decisions strategically, to maximize their utility. Influence diagrams were introduced by (Ronald A. Howard 1989), and have been investigated almost entirely in a single-agent setting. Formally, an influence diagram is defined as follows:

Definition 3-1: An influence diagram is a pair $I = (G, P)$ whose elements are defined as follows:

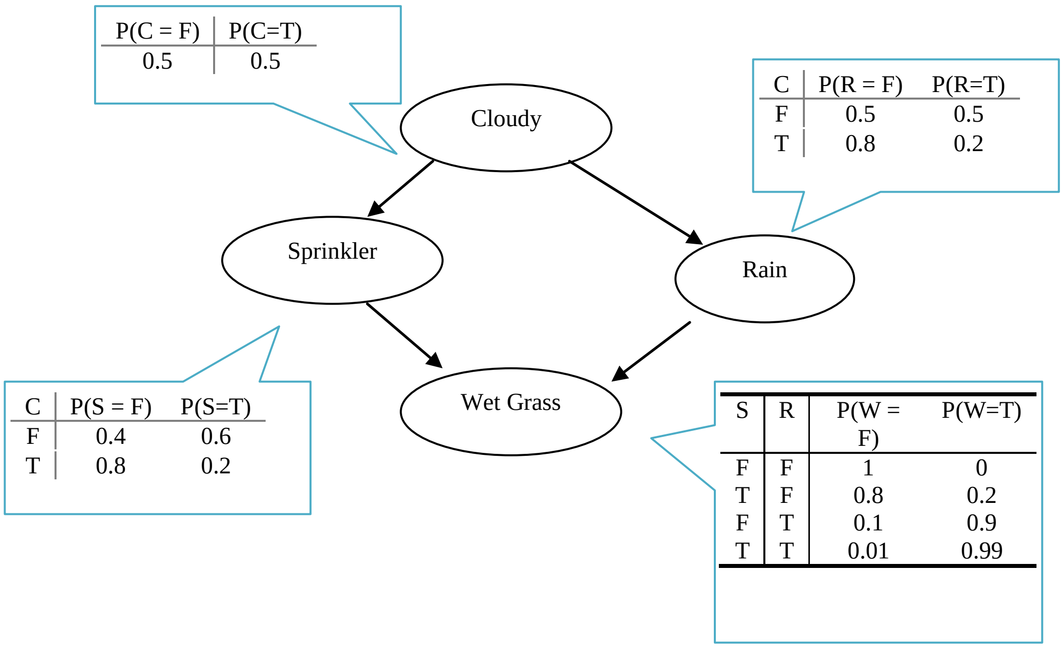
$G = (N, A)$ is a DAG such that $N \subseteq C \cup D \cup \{v\}$, where C and D are disjoint, and the following conditions are satisfied:

a) The value node v is a sink node that has no successors;
b) A directed path only consisting of all the decision nodes D exists in G ;
c) Each decision node and its parents are parents to all of its subsequent decision nodes;

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

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

Influence diagrams involving only one decision maker is a symmetric decision problem and it provides a standard representation to be extended to solve other types of decision problems. However, those influence diagrams-type of model representations only orient towards the single agent environment, which is not sufficient to deal with the Multi-agent decision problems.



P(C = F)	P(C=T)
0.5	0.5

C	P(R = F)	P(R=T)
F	0.5	0.5
T	0.8	0.2

C	P(S = F)	P(S=T)
F	0.4	0.6
T	0.8	0.2

S	R	P(W = F)	P(W=T)
F	F	1	0
T	F	0.8	0.2
F	T	0.1	0.9
T	T	0.01	0.99

Figure 3-6 Simple Bayesian Network

3.1.2 Learning Process

<u>Learning other agents' model.</u> This will help an agent to predict other agents' behavior correctly which in turn leads the learning agent to arrive at the best decision in terms of meeting the goal efficiently. The agent should already have a multi-agent reasoning system and maintain models of other agents in its knowledge base (d'Inverno, Luck et al.). Due to uncertainty, the agent assigns probabilities to the models, which are updated based on the history of the other agents' observed behavior. If a model is not accurate, its probability will gradually decrease. This signals the need for a better model, which can be obtained by learning.

In most situations, information about other agents only comes from the observation of their behavior. The history of an agent's behavior can be defined as a set of its observed actions during a particular time frame, in which the data of the world states are known. Given only a history of behavior, the initial model is modified by refining parameters in the BN-BDI model that are associated with capabilities, preferences and beliefs of the other agent. The refinement can be done in stages with the order as above, according to the increasing level of complexity. There can be a number of modified models that can be generated. The modified models can compete with each other and other models maintained by the learning agent. The probability of each model being correct can be arrived based on how well the model predicts the history of behavior.

Learn capabilities. An agent's capabilities in an BN-BDI model are represented as possible value assignments to the agent's decision node. These values need to be exhaustive. For example, if an agent has possible value of SR_M1 and SR_M2, then the value set should be {SR_M1, SR_M2, other}. Here "other" represents all the unknown or unlikely actions not explicitly represented. When the agent observes an action that is not explicitly present in the current model, the model is modified by specifying the action as explicit possible value of the decision node, and by updating the conditional probability table (CPT) of the nodes that are influenced by the decision node. Conversely, the other agent's capabilities may need to be collapsed and included as part of the "other" value

Learn preferences. The strategy in learning preferences of other agents is to modify the model by refining the utility function so that every action in the history of behavior always maximizes utilities of the resulting states.

$$U(S) = F(X_1, \ldots, X_N) \qquad 3\text{-}2$$

Utility function *U(S)* denotes the utility of state *S*, *X*={$X_1, \ldots, X_N$} is a set of features that directly influence the agent's preferences. The utility function *f*, is commonly postulated in multi-attribute utility theory to be a weighted sum of factors of values of feature X_k, k= *1,…, N.* For simplicity, assume that the weighted factors depend linearly on the features X_k.

$$U(S) = w_1 x_1 + \cdots + w_N x_N \qquad 3\text{-}3$$

The method of learning the agent's preferences is to modify the weights w_k. Let A^* denote an action that maximizes expected utility.

$$A^* = \arg\max_{a_i} \sum_{j=1}^{J} P(S_j | a_i, E) \times U(S_j) \qquad 3\text{-}4$$

A={$a_1, \ldots, a_N$} is a set of the agent's alternative actions, S_j are the possible outcome states given a, possibly non-deterministic, action a_i, with *j* ranging over *J* different outcome. *E* represents the known data of the world which are provided in the history of behavior. When combining the above two equations, the equation becomes

$$A^* = \arg\max_{a_i} \sum_{j=1}^{J} P(S_j | a_i, E) \sum_{k=1}^{N} w_k x_k^j \qquad 3\text{-}5$$

Define $X_k^i = \sum_{l=1}^{r_k} x_{k,l} \times P(X_k = x_{k,l} \mid a_i, E)$, and then the above equation can be written as

$$A^* = \arg\max_{a_i} \sum_{k=1}^{N} w_k X_k^j \qquad 3\text{-}6$$

From the history of other agent's behavior, a set of the agent's observed actions can be observed during a given time frame. Let $D=\{D(1),\ldots,D(T)\}$denote such a set, where T is the total number of actions, and $D(t)\in A,\ t=1,\ldots,T.$ The refinement of utility function is accomplished by having an initial utility function derived from the initial model, and subsequently adjusting the weights w_k based on the set D, so that $D(t)$ is equal to A^* for every situation t.

Weight adjustment is done by delta rule, which is to minimize a cost function based on the error signal so that each weight adjustment brings the actual output value closer to the desired one. The delta rule is similar to hill-climbing learning algorithm.

Learn beliefs. An agent's beliefs are represented in BN-BDI model as the nature nodes and the probabilistic relationship that exists among them. The learning method is similar as in the previous subsection, except that they set the weights fixed and learn the expected values of the features which influence the utility. The set Φ of feature nodes that are likely to be the source of inaccuracy in the model is to be found. Strategies to modify the model are based on different reasons. For example, incorrect local probability distributions of the features in Φ is modified by refining CPTs on the feature nodes in Φ; incorrect dependence relationship involving the features in Φ is modified by refining the structure of BN-BDI model, removing or adding links from certain nodes; if it is the parents of the feature in Φ that cause the problem, perform strategies on the parents nodes of Φ.

As intelligent systems are being applied to larger, open, and more complex problem domains, many applications are found to be more suitably addressed by MAS. More specifically, an application system in such a problem domain cannot be effectively constructed as a single intelligent agent due to the scale and complexity of the domain. Instead, a set of cooperating agents can be more effectively developed, each of which is embedded with a subset of knowledge and problem-solving abilities and pursues subtasks autonomously.

In this section a novel agent model has been proposed. The next section describes a hierarchical MAS architecture consisting of the BN-BDI agents.

3.2 Proposed Multi-agent System Architecture

3.2.1 Cooperative MAS

The MAS system can be grouped by the type of implemented cooperation which can range from total cooperation to the total antagonism. Figure 3-7 illustrates the change from cooperative agent to autonomous agent. Completely cooperative agents can change their goals to meet the needs of other agents. Antagonistic agents, on the other hand, will not cooperate and their respective goals may be blocked.

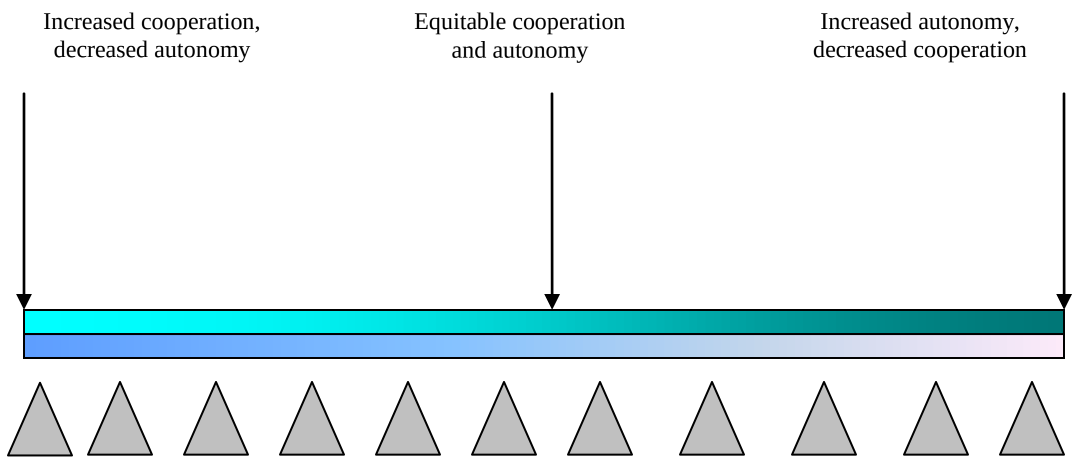

Increased cooperation, decreased autonomy
Equitable cooperation and autonomy
Increased autonomy, decreased cooperation

Figure 3-7 Agent Cooperation versus Autonomy

Researchers in (Bond 1990) describes the existence of two types of MAS architecture.

(1) Horizontal Architecture: this structure is useful in some contexts, for example, a situation where a group of agents having different (non-overlapping) capabilities and hence can work towards the goal without needing any conflict resolution. Here, all the agents are on the same level with equal importance without a Master/Slave relationship.

(2) Vertical Architecture: in a vertical architecture, the agents are structured in some hierarchical order. Agents at the same sub-level may share the characteristics of a horizontal structure. The 'horizontally structured' MAS model has several issues. One critical issue is that it quickly becomes too complex and unwieldy for practical applications, wherein agents in the MAS may share some common capabilities. Hence most current frameworks have adopted a hierarchical MAS model (vertical) by organizing the agents in some organizational structure.

As shown in Figure 3-8 it is a layered MSA architecture: And it involves the different decomposition within the agents to complete their tasks. The input is either processed from each layer individually (horizontal), or from each layer after the other so that the final one will form the output(vertical one pass) or from all the layers in a vertical two pass control so that the output is received from the same layer that the input was perceived.

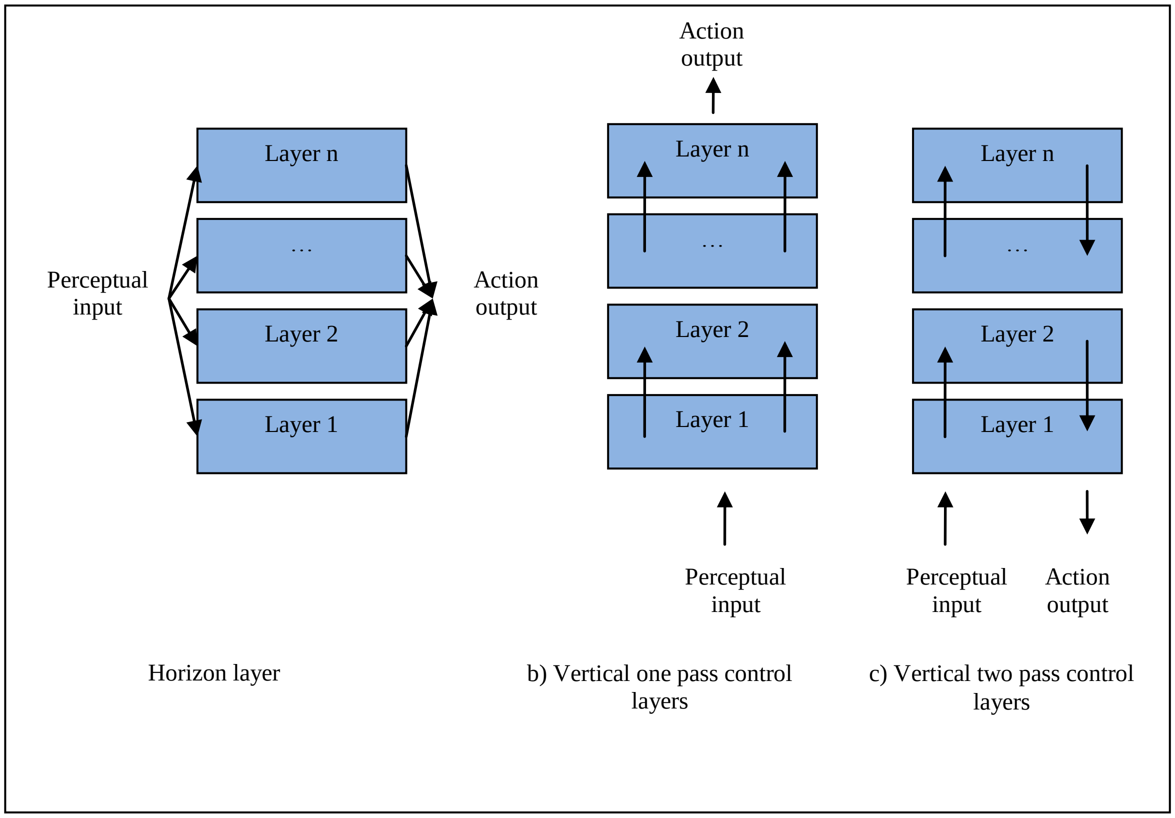


Figure 3-8 MAS Layered Architecture

In the previous section, a BN-BDI agent model is proposed. It is a hybrid architecture that has the merits of both the deliberative and reactive architecture. The BDI part maintains an explicit representation of the agents' world. The BN part measures the uncertainty that the agent is facing and the dependent relationship it has with other agents.

The BN-BDI agent can learn other agents' model, their preferences, their beliefs and their capacities.

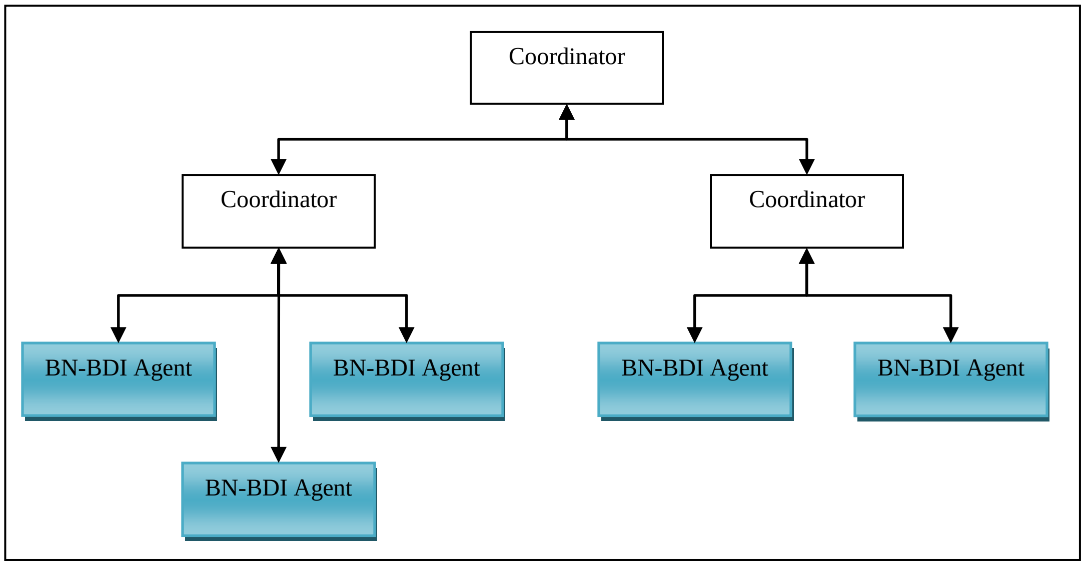


Figure 3-9 The dynamic hierarchical MAS architecture

To form an effective MAS consisting of BN-BDI agents, the architecture is very important. The horizontal architecture, where all the agents are on the same level with equal importance without a Master/Slave relationship, is useful when a group of agents having different (non-overlapping) capabilities can work towards the goal without needing any conflict resolution. However, as the number of agents increases in the system, the communication and computation complexity increase exponentially. The vertical architecture, where the agents are structured in some hierarchical order, can greatly reduce the complexity of the communication and computation. However, the fixed master/slave relationship is very rigid.

Considering the advantages and disadvantages of the horizontal and vertical architecture, a hierarchical MAS architecture consisting of the BN-BDI agents is proposed (Figure

3-9). Agents with the similar characteristics or capacities constitute a group and some other agents in MAS act as coordinators. The role of the coordinators is not fixed to any agents, which means in some situations, agent *A1* will be the coordinator; in other situations, when the agent *A1* is not available (e.g. lack of time, resource, or capacity), agent *A1* does not need to be a coordinator. This dynamic nature gives flexibility to MAS. So when each time agents are forming a system to perform some tasks, only capable agents will take the role of coordinator.

3.3 Summary

At the beginning of this chapter, BN-BDI agent model to build the adaptive cooperative MAS is proposed. It is a hybrid architecture that has the merits of both the deliberative and reactive architecture. The BDI part maintains an explicit representation of the agents' world. The BN part measures the uncertainty that the agent is facing and the dependent relationship it has with other agents. The BN-BDI agent can learn other agent's model, their preferences, their beliefs and their capacities.

It is important to carefully examine the problem settings and choose the best suitable modeling method. For the BN-BDI agent model, a hierarchical MAS architecture to form the Multi-agent system is proposed. Agents can choose to be or not to be a coordinator and this dynamic nature give the flexibility to agents. The role of coordinator helps to decrease the communication and coordination overhead. Such structure will facilitate the coordination of agents in cooperative MAS and give more adaptability to the MAS performance. The details of the coordination mechanism for the novel MAS framework will be presented in the next chapter.

Chapter 4 The Coordination Mechanism for Cooperative BN-BDI MAS

For the cooperative MAS, agent coordination is very crucial. Coordination can be regarded as the process by which the individual decisions of the agents result in good joint decision for the group. The reasons and difficulties lie in the following facets. 1) Inter-dependencies: this happens when individual agents take related goals. For example, decision of one agent impacts on decision of other society members and the possibility of harmful interactions distributes among agents. 2) Global constraints. These constraints exist when the solution being developed by a group of agents must satisfy certain conditions, if it is to be deemed successful. For example, a distributed monitoring system may have to react to critical events within 30 seconds; a distributed air traffic control system may have to control the planes with a fixed communication bandwidth.

In this chapter, how agent coordinates with each other based on the BN-BDI agent model and hierarchical MAS architecture is studied. This chapter is organized as follows. Firstly, the Time Constraint Task-based Model is proposed. Secondly, the corresponding coordinate protocol is discussed. After that, the analysis on the computational, communicational complexity and scalability are provided. Based on the theoretical analysis, the coordination mechanism for the novel cooperative BN-BDI MAS is effective.

4.1 Mechanisms to facilitate coordination

There are three common mechanisms to facilitate the coordination behavior in MAS. They are organizational structuring, exchanging meta-level information and multi-agent planning.

Our approach is based on meta-level information exchange. The procedure to solve a problem / finish a task in MAS: Problem suggested by one agent or a group of agents; Solve the problem or plan to get helpers; Decompose the problem; Team up a group of interest related and capable agents; Allocate sub-problems; Finish the sub-problem on time; Synthesize the result.

During the period of the problem or task execution, it is assumed that Self-interested agents form a team with one goal; no confliction between agents; all the heterogeneous agents follow or abide by the same norms and social laws. Under these assumptions, a MAS problem can be easily converted to a Cooperative Distributed Problem Solving Strategies (CDPS) problem. As a result, some ideas are deployed from CDPS to solve the MAS problem.

4.2 Time Constraint Task-based Model

In the literature, researchers used game theory, logic theory, and contract net to model the cooperation. In the thesis a contract net based model is provided and the novel improvement made to facilitate better cooperative MAS performance.

There are two Distributed Problem Solving Strategies: Task Sharing and Task Decomposition (Durfee, Lesser et al. 1989; Lesser 1999; Stone and Veloso 2000; Luck, Mark et al. 2001). The motivation for Task Sharing in Heterogeneous Systems includes: it is difficult to build agents that are competent in every possible task; if these agents which can carry out every possible task exist, they will result in agents that are "Jacks of all trades and masters of none"; Omni-capable agents would only use a small proportion of their capabilities at any one time. They will waste most of their capacity and it is not economically justifiable. Moreover, the master agent will be the bottleneck of the system performance. To avoid waste into building mighty agents, specialist agents that can work together to solve problems beyond their individual capabilities can be built.

Task sharing procedure	
Task Decomposition	Agents generate a set of tasks to be passed to others, it decompose large tasks into subtasks that can be tackled by different agents
Task Allocation	Agents assign subtasks to appropriate agents.
Task Accomplishment	Agents accomplish task, either individually or through further decomposition
Result Synthesis	Combine the results to generate the final results

Figure 4-1 Task sharing procedure

To find an agent that is able to solve a problem, a Na ïve Solution is to have a table that identifies the capabilities of the agents. Agents can look up the table to select appropriate candidates and give them the sub-problem. To select the best agent out if several candidate agents exist, the solution can be to announce a sub-problem to specific agents through contract net protocol. Request return bids describing acceptance/availability. Then Assign sub-problem to the highest bidder

There are other forms of task allocation in MAS, such as, Decentralized Task Allocation and Dynamic Task Allocation. In Decentralized Task Allocation, the problem is decomposed into subtasks and divided among the agents. The tasks could either be similar or different depending on the type of decomposition adopted. In some decentralized systems, the local goals for agents could be different but they all still have the same global goal. For Dynamic Task Allocation of MAS, it is highly useful in applications where the environment is constantly changing and agents have to adapt to the changing environment.

In this section the Time Constraint Task-based Model (TCTM) is discussed in detail.

A MAS has *n* agents.each agent has its own belief, desire and intention. It is assumed that each agent can model other agents' model and can improve its model from the actions other agents take. In TCTM, agents have five components to interact with other agents and environment, and to execute actions. They are problem solving module, task generating module, communication module, repository and goal module (Figure 4-2). Through the communication module, agents can communicate with each other and the environment. When a task is given to an agent, this information will go to the repository module first. If the repository module contains the solution, it will execute the plan. If no predefined solution is available, the task will be passed to problem solving module where agents try to solve it. This module will give the plan of actions. If the plan is too big or too complex to execute, the task generating module will decompose the task into sub-tasks and broadcast them to the fellow agents. A Task *T* has the following properties: Task name, Task requirements, Task rewards, Task time constraint. Tasks are loosely coupled. The task requirements and the task rewards are straight forwarding. The task time defines a deadline when the announced tasks need to be finished. This extra constraint protects the integrity of MAS but increases the complexity when learning and coordinating with other agents. The goal module stores the preference of the agent. The utility of agent's action will be calculated there. When agent has any action plan, it will execute the plan only if the execution of the plan will increase its utility level.

The TCTM's pre-conditions include: the group of agents which carry out the task follow the same norms and social laws; The group of agents share the same ultimate goal and agents are cooperative with each other; When working together, agents are reliable and trustful; When working together, there is no conflict of interest between agents; Agents are committed to carry out the task on time and satisfy the task requirements

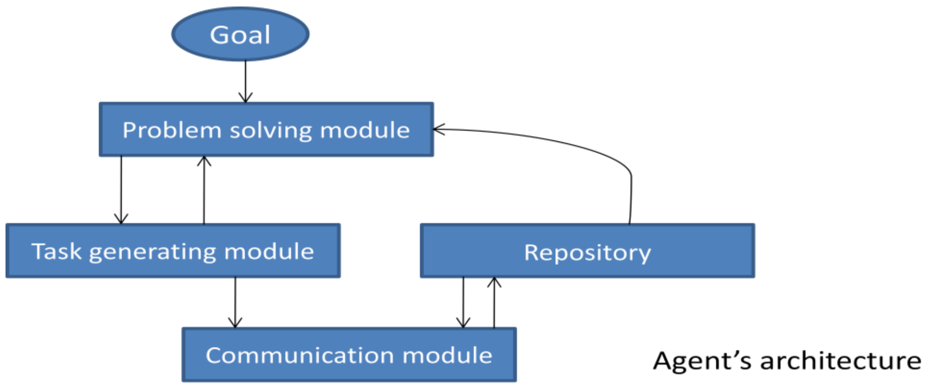


Figure 4-2 TCTM agent components

The Stages of Time Constraint Task-based Model are listed in the following (Figure 4-3):

Recognition stage: some agent in a Multi-agent community has a goal, and recognizes the potential for cooperative action with respect to that goal. In this stage, Agent cannot achieve the goal in isolation; Agent prefers not to work alone; Believe there is some group of agents that can achieve the goal.

Task decomposition stage: Generate a set of tasks to be passed to others; decompose large tasks into subtasks that can be tackled by different agents.

Team formation stage: Eligible candidates: can finish the task in the constraint time period. Methods like Search and broadcast can be used. Candidate agent with the highest bid gets the subtask. Agents accomplish task, either individually or through further decomposition. Team structure includes: Hierarchy structure, Task decomposition network, Result sharing in peer agents with the same roof agent.

Team action stage: Agents take over the task/subtask, Generate a plan, Execute the plan, Share results with peer agents, Agents learn from results sharing and improve the problem solving abilities, Father agent synthesizes the results, All actions need to be finished in constraint time

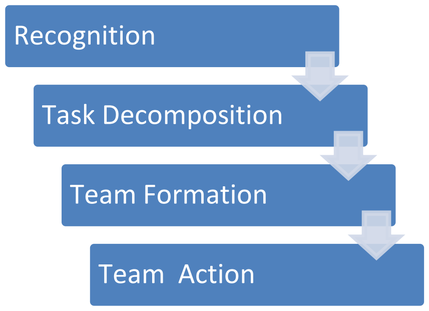


Figure 4-3 Stages of Time Constraint Task-based Model

4.2.1 Time Constraint Contract Net Protocol

Given a task to perform, an agent first determines whether it can break the tasks into sub-tasks that can be performed concurrently. It utilizes the protocol to announce the tasks that could be transferred and request bids from agents that could perform any of these tasks. An agent that receives a task-announcement message replies with a bid for the task, indicating how well it thinks it can perform the task. The contractor collects the bids and awards the task to the best bidder. The CNP allows nodes to broadcast bid requests to all others. These procedures are illustrated in Figure 4-4.

Agents dynamically create relationships in response to current processing requirements embodied in a contract. And a node with a task to be achieved forms a contract with other agents who proceed to accomplish the task. The contracts involve commitments to complete the agreed upon tasks among other agents. To make the award decision, it is typically based on marginal utility calculations. The contract net has been applied in many areas including: distributed delivery, information agents, distributed design and manufacturing, electric power systems, electronic commerce.

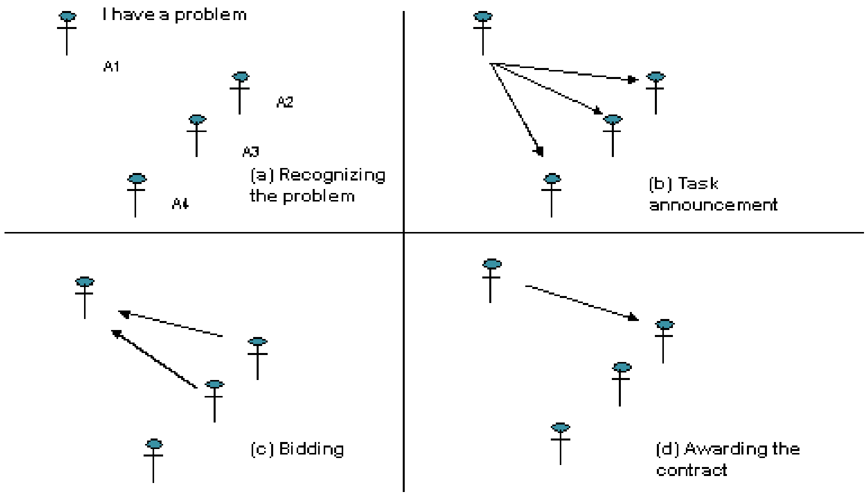


Figure 4-4 Contract net protocol

In the time constraint task-based model, the task *T* has a time constraint property (Figure 4-5). In the contract net bidding process, to make sure the deadline is met, a countdown time constraint should be included in the bid. Now the protocol becomes: 1) at time 0, agent A_1 has a problem. Agent A_1 decomposes the problem into tasks *T* with time constraints T_{cd} (deadline or countdown time). 2) At time t_1, agent A_1 announces the tasks *T* to its fellow agents. 3) At time t_2, those available agents that receive the announcement bid for the tasks *T*. 4) at time t_n, agent A_1 evaluates the bid and awards the contract with deadline T_{cd} to the suitable candidate agent. Candidate agent needs to finish the task and send the results back to agent A_1 before the deadline. Otherwise a heavy penalty will be placed on the candidate agents.

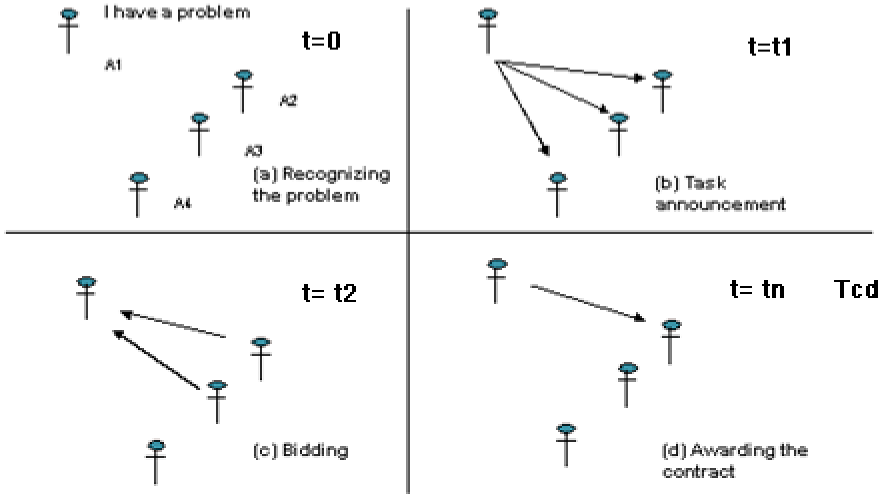


Figure 4-5 Time Constraint Contract Net Protocol

The TCTM model has many advantages over other models for MAS coordination. For example, TCTM simplifies MAS cooperation problem; TCTM includes time constraint variable, which means learning and coordination need to be done in a time-critical fashion. As a result, the MAS overall utilities increase in the same time period; in TCTM, the agent's role is not fixed. Agents can behave as a leader to sub-contract the task or as a bidder to bid for a sub-task in different course of team formation; in TCTM, Agents can learn from their team fellows and benefit from result sharing through communication and coordination.

4.3 Coordination Formation: Multi-agent action selection problem

In a multi-agent coordination scenario, a number of agents need to generate individual plans that achieve their respective goals and are not in conflict with each other. A multi-agent coordination scenario that is defined by two main characteristics is studied in this section. The first is that each agent is able to achieve his goals by himself. The second is that plan execution time is the criterion for evaluating the quality of both the individual and the joint plans, with preference given to the joint plan execution time. Therefore, agents seek to minimize the execution time of the joint plan, even in the case where this leads to non-optimal individual plans.

The coordination problem is defined formally as follows.

Generating teams of agents that are able to perform their tasks over long periods of time requires the agents to be responsive to continual changes in agent team member capabilities and to change in the state of the environment and the task mission.

In the previous chapter, the multi-agent architecture, which enables teams of agents to dynamically adapt their actions over time, is described. This architecture is a distributed, behavior-based architecture aimed for use in applications consisting of a collection of independent tasks. The key issue addressed here is the determination of which tasks agents should select to perform during their mission. In this approach, agents monitor the performance of their teammates forming common tasks, and evaluate their performance based upon the time of task completion. Agents then use this information throughout the lifetime of their mission to automatically update their belief parameters.

An important goal in the development of autonomous MAS is enabling agents to perform their tasks over a long period of time without human supervision. This system must be able to deal with dynamic changes that will occur over time, such as changes in the environment or incremental variations in their own performance capabilities because of their learning and evolving ability. The ability to adapt to these types of dynamic changes is especially important in multi-agent applications, since the effects of individual agent actions propagate across the entire team. In most real-world applications, a multi-agent team with static capabilities will not be able to continually achieve its goals over time as the system of agents and the environment drift further and further from the original state. Instead, a successful MAS is adaptive to changes in the environment, agent members capabilities, agent team compositions, mission requirements, and so on. One important consequence of dynamic changes in MAS is that the continuing drift in individual agent capabilities creates a team of heterogeneous agents, even if the original team is designed to be homogeneous. This heterogeneity in MAS presents a particular challenge to efficient autonomous control when overlapping in team member capabilities occurs. Overlap in team member capabilities means that more than one agent can perform a given task, but with different levels of efficiency. In these cases, the agent must continually determine which individual agent on the team is currently the best suited for a given task in the application. These types of decisions are usually not easy to make, especially when the multi-agent team control is distributed across all the team members. Even in the case of full global knowledge, however, the problem of optimally assigning tasks to agents that have different, but overlapping capabilities can be easily shown to be NP-hard.

Proof of NP-Hardness of the heterogeneous agent action selection problem

Definition 4-1: Let $A = \{a_1, a_2, ..., a_n\}$ represent the set of n agents on a cooperative team, and the set $T = \{task_1, task_2, ..., task_m\}$ represent the m independent tasks required in the current mission. Each agent a_i in A has a number of high-level task-achieving functions that it can perform, represented by the set $F_i = \{f_1, f_2, \ldots\}$. Since different agents may have different ways of performing the same task, the set of n functions is H where $H: F_i \rightarrow T, H = \{h_1(f_{1k}), h_2(f_{2k}), \ldots, h_n(f_{nk})\}$, and $h_i(f_{ik})$ returns $task_k$ that agent a_i is working on when it performs the high-level functions f_{ik}.

<u>Definition 4-2</u>: the metric evaluation function is denoted as $l(f_{ij})$, which returns the length of task execution time of agent i performing the action f_{ij} to finish task_j.

Since agent team members usually perform their actions in parallel during a mission, the total mission completion time is the time at which the last agent finishes its final task. However, agents that are unfamiliar with their own abilities or the abilities of their teammates could not have access to this $l(f_{ij})$ function. Thus, an additional aspect to the agent's learning problem is actually obtaining the performance quality information required to make an "intelligent" action selection choice.

<u>Definition 4-3</u>: Define the tasks an agent a_i selects to perform during a mission as the set $U_i = \{f_{ij} \mid \text{agent } a_i \text{ performs task } h_i(f_{ij}) \text{during the current mission}\}$.

In the most general form of this problem, distinct agents may have different collections of capabilities. Thus, it is not assumed that $\forall i.\forall j.(F_i = F_j)$. Furthermore, if different agents can perform the same task, they may perform that task with different qualities, i.e. with different lengths of task execution time. Thus, it is not assumed that if $h_i(f_{ix}) = h_j(f_{jy})$, then $l(f_{ix}) = l(f_{jy})$. For the simplified case in this section, it assumes that these agent performance measurements are known in advance.

The formal statement can be described as follows:

Multi-agent action selection problem (MAASP)

Given $A, T, F_i\ and\ H,$ determine the set of actions U_i for all a_i such that $\forall(a_i \in A), U_i \subseteq F_i$ and $\forall(task_j \in T), \exists i, \exists k, (task_j = h_i(f_{ik}))$ and $f_{ik} \in U_i$ and the performance metric is optimized, according to the desired performance metric; for example, in the case of the time metric, the goal is to minimize: $\max_i(\sum_{f_{ik} \in U_i} l(f_{ik}))$.

The first two constraints of the above problem definition ensure that each task in the mission is assigned to some agent that can actually accomplish that task. The goal is to minimize the maximum amount of time any agent will take to perform its set of actions.

For system robustness, fault tolerance, and flexibility, distributed decision making gives better results than a centralized decision maker most of the time. Thus, while this efficiency problem is formulated as a centralized decision problem, in real MAS applications, the problem should often be solved by the distributed multi-agent team using incomplete local information.

Nevertheless, even if complete global information is assumed, this efficiency problem can be easily shown to be NP-hard by restriction to the well-known NP-complete PARTITION problem.

Proof: MAASP problem is NP-hard

By restriction to PARTITION:

Allow only instances of MAASP where:

$$n = 2, (\text{i.e., } A = (a_1, a_2))$$

$$F_1 = F_2 = S$$

$$T = \bigcup_{a_i \in A,\, f_{ij} \in F_i} h_i(f_{ij}))$$

$$\forall (a_i \in A).\forall (f_{ij} \in F_i).(h_i(f_{ij}) = task_j)$$

$$\forall (f_{ij} \in F_i).(l(f_{1j}) = l(f_{2j}) = s(w_j)), \text{for } w_j \in W$$

Then since PARTITION is a special case of MAASP, MAASP must be NP-hard.

Figure 4-6 Proof of MAASP is NP-hard

The PARTITION problem is to decide whether a given multiset of integers can be partitioned into two "halves" that have the same sum. More precisely, given a finite set W and a "size" $s(w) \in \mathbf{Z}^+$ for each $w \in W$, determine whether there is subset $W' \in W$ such that $\sum_{w \in W'} s(w) = \sum_{w \in W - W'} s(w)$. Then it will be proved that the MAASP problem is NP-hard in the number of tasks required by the mission (Figure 4-6).

Since the PARTITION problem is stated in terms of finding two equally-sized subsets of tasks W and W', the proof of this theorem restricts MAASP to those instances involving two agents with identical capabilities and qualities of capabilities, the lengths of execution time. Furthermore, each agent has the same one-to-one mapping of behavior sets to tasks, meaning that all agents use the same behavior set to accomplish the same task, and all behavior sets are needed to accomplish the mission. These MAASP instances are the instances of PARTITION, so that, if MAASP can be solved, PARTITION can be solved too. In other words, even this simpler problem involving homogeneous agents with equal costs is NP-hard. The real-world application involving heterogeneous agents with unequal costs is certainly no easier, and thus this action selection problem is shown to be NP-hard.

Since this efficiency problem is NP-hard, the MAS will not be able to derive an optimal action selection policy in a reasonable length of time. Heuristic approximations to the problem that work well in practice will be investigated.

4.4 Coordination Scalability

Scalability is a problem for many learning techniques, but especially so for multi-agent learning. The dimensionality of the search space grows rapidly with the number and complexity of agent behaviors, the number of agents involved, and the size of the network of interactions between them. This search space grows so rapidly that it is believed no one agent can learn the entire joint behavior of a large, heterogeneous, strongly intercommunicating multi-agent system. Effective learning in an area of this complexity requires some degree of sacrifice: either by isolating the learned behaviors among individual agents, by reducing the heterogeneity of the agents, or by reducing the complexity of the agents' capabilities. Techniques such as learning hybrid teams, decomposing behaviors, or partially restricting the locality of reinforcement provide promising solutions in this direction, but it is not well understood under which constraints and for which problem domains these restricted methods will work best.

The scalability of multi-agent algorithms lie in that the less coordination a MAS requires, the better it should scale to large numbers of agents.

To evaluate the success of implementing MAS, there are two perspectives (Schumacher 2001). The first one is coherence. It describes how well the MAS behaves as a unit, along some dimension of evaluation. Coherence may be measured in terms of solution quality, efficiency of resource usage, conceptual clarity of operation, or system performance in the presence of uncertainty or failure. The other one is coordination, which means the degree to which agents can avoid extraneous activity such as synchronizing and aligning their activities. The presence of conflict between agents is an indicator of poor coordination, when agents destructively interact with one another which require time and effort.

The adverse effects of coordination are well known to researchers in parallel processing: often, increasing the number of processors results in a corresponding decrease in the amount of time needed to perform a given computation only up to a point, after which the time spent coordinating inter-process communication outweighs the benefits of having more processors. MAS has problems similar to those of the parallel algorithm designers: if the task charged to a MAS by necessity requires a great deal of coordination, then the performance of the system will necessarily degrade as more agents are employed.

Thus a measure of coordination complexity is needed to evaluate proposed algorithms. Ideally such a measure should be a function of the minimal information flow required to perform a given task and be independent of how that flow is mediated. It should, furthermore, account for how that information changes as a result of the dynamics of the environment. The model is that each agent knows its own state and can communicate directly with any other agent via some communication system. The cost of a communication event, in terms of bandwidth used, latency or time spent waiting, is summarized into a single, abstract cost.

The notion of communication complexity is based on the notions of communication complexity defined in the analysis of parallel algorithms with the difference being that in the coordinated systems some invariant maintains indefinitely as opposed to performs some finite computation.

In multi-agent settings, deliberative decision-theoretic methods allow an agent to compute its best action, i.e., one with the highest expected utility, given its available knowledge. The agent's knowledge may contain the information about the environment, about the agent itself, about the other agents present, about what other agents know, and so on. Therefore, when making coordinated decisions in complex scenarios with many agents present, the agent may have to consider large and complex knowledge structure, which may lead to it failing to decide on action in an acceptable time.

The communication schemes investigated range from full communication to no communication. Two of the schemes are from cooperative mobile robotics. The first, distance modulated communication (DMC), uses the idea that a robot might need a very accurate estimate of the position of nearby robots (to avoid colliding with them, for example) while needing only a coarse estimate of more distant robots. Thus, robots communicate more often with robots that are closer. It is shown that DMC has communication complexity $O(n \log n)$ or $O(n^{1.5})$ depending on assumptions about the dynamics of the robots (Klavins 2004). The second, the wandering communication scheme (WCS), defines a protocol by which only a small constant number of wandering robots are allowed to move and communicate, while others must remain immobile. A wanderer may transfer its right to move, and its information about the world, to an immobile robot in a short burst of communication. It is shown that WCS has communication complexity $O(n)$ or $O(1)$ depending on assumptions about how certain higher level decisions are made by the robots (Klavins 2004).

For the coordination problem in this thesis, an approach similar to distance modulated communication with a communication complexity of $O(n \log n)$ or $O(n^{1.5})$ is chosen. The simulation results in the next chapter which scales well with the growing number of agents support the proposal.

4.5 Summary

In this chapter, different facets of coordination mechanism of the cooperative BN-BDI Multiagent system have been discussed. The communication complexity and the coordination complexity give a benchmark to measure the efficiency of the coordination methods. These two metrics also measure the robustness and scalability of the coordination mechanism. The proposed TCTM model, which includes a time-critical component, coordinates the agents’ behavior based on contract net protocol. In the next chapter, a foraging simulation will be used to test the proposal.

Chapter 5 A Simulation Case Study of the Adaptive MAS

In the previous chapters, the BN-BDI agent model, the hierarchical MAS architecture and the time-constraint contract net coordination mechanism are investigated. In this chapter, a simulation which tries to solve a foraging problem to verify the proposals is carried out.

This chapter is organized as follows. Firstly the foraging problem will be introduced. Then the simulation test base's architecture will be presented. Next a basic model for agents in the foraging problem is proposed. After that the "sparse world" problem in the rock foraging is discussed. To solve this problem, an adaptive model which enables agents to adaptively change their behaviors when the world gets dynamic is extended based on the basic model. Finally, the simulation results that demonstrate the improved performance of the adaptive agents over the basic agents and conclusions are given.

5.1 The Foraging Problem

The foraging problem consists of a number of rocks that are scattered over a rectangular grid. Autonomous agents that live in this virtual world have to cooperate to collect these rocks.

The foraging problem solved in this chapter evolves from the swarm foraging. Artificial swarm foraging often utilizes various forms of "indirect communication", involving the implicit transfer of information from agent to agent through modification of the world environment. Examples of indirect communication include: leaving footsteps in snow, leaving a trail of bread crumbs in order to find one’s way back home, and providing hints through the placement of objects in the environment. Much of the indirect communication literature has drawn inspiration from social insects’ use of pheromones to mark trails or to recruit other agents for a task. Pheromones are chemical compounds whose presence and concentration can be sensed by fellow insects, and like many other media for indirect communication, pheromones can last a long time in the environment, though they may diffuse or evaporate. Early models of ants have demonstrated that pheromones make it possible for ants to optimize trails from the nest to a food source. It has been suggested that this happened because larger amounts of pheromones can more quickly accumulate on the shorter paths, rather than on the longer ones. Several pheromone-based learning algorithms have been proposed for foraging problem domains. A series of reinforcement learning algorithms have adopted a fixed pheromone depositing procedure, and use current pheromone amounts as additional sensor information while exploring the space or while updating the state-action utility estimates. Evolutionary computation techniques have also been applied to learn exploration/exploitation strategies using pheromones deposited by hardcoded mechanisms.

The task of the agents to collect all the rocks in the world is called a job. The idea is derived from swarm robotics, which is a subset of MAS. Figure 5-1 shows an example of a foraging problem in the grid world of size 6 * 9 with 4 agents.

When several robots are randomly sent to forage rocks in an open area, it is expected the robots are able to collect as many rocks as possible in the shortest time (Panait and Luke 2004). It is assumed that the robots are autonomous agents and only have local communication with each other and the environment, which means that no one agent knows all about the entire system. It is also assumed that the robots are homogeneous agents. This problem concerns how the agents communicate with each other, how they divide the job among themselves, and how they cooperate to finish the job.

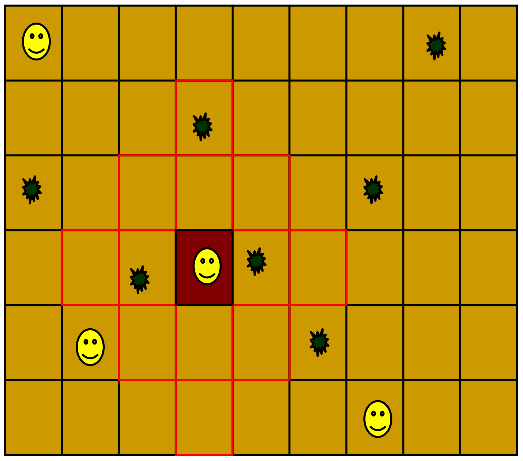


Figure 5-1 The virtual world with 4 agents and several rocks

The way of swarm robotics to solve the problem is efficient. The whole population of the robots is divided into sub-populations and different populations correspond to different areas. When the robot finds a rock in some place, it gets some information about the environment – the rocks distribution on that field. When more rocks are collected, more information is found which can give some guidance to the robot's following actions and to other robots in the same sub-population which later turns information to other sub-populations.

In the foraging problem, agents can interact with the environment in a number of ways. An agent can make a step to one of the four neighbor fields around him. If an agent meets any rocks in the field, the agent can pick it up. Some assumptions are needed. One assumption is that the capacity of the agent is unlimited, the energy for carrying so many rocks is the same and agents do not put down any of these collected rocks until the job is done for the whole team. Another assumption is that there is no collision detections in the foraging problems which means agents can be in the same position. It is also assumed that all the agents work cooperatively and honestly, so that they exchange true information with each other. Besides acting into the environment, agents can also send messages which contain its own belief about the virtual world to each other. A mailing system-like mechanism is used to assist the communication.

It is important to notice that each agent of the foraging problem has only a limited view of the world. The view-size of the world expresses how far, i.e. how many squares, an agent can observe around him. The red area in Figure 5-1 illustrates the limited view of one agent. Other than what agents actually observe about the world, each agent has its own belief of what the world should be. When each time an agent explores a new area, it will update its belief of the world; when each time agents communicate with each other, it will combine other’s belief of the world and use the part of what other agents really observe or really explore.

The performance of the foraging problem is measured by a counter, the efficiency of the agents in performing their job. This counter measures the energy invested by the agents. When an agent moves one step, it consumes one unit of energy. According to the problem assumptions, agents consume the same energy no matter whether they pick up rocks or not.

5.2 Cooperative BN-BDI Multiagent System for Foraging Problems

To solve the above mentioned foraging problems, the cooperative BN-BDI Multiagent system proposed in chapter 3 with the corresponding coordination mechanism proposed in chapter 4 are deployed.

To solve the foraging problem in an n*n grid world, one agent would take a long time to finish the task. The task here means find out all the possible rocks in the n*n grid world. Since the agent system has many other agents with same capabilities available, the task of the specific agent can be solved by coordinating with other agents in the system. The group of agents' model is BN-BDI agent model. The BN-BDI agents which can learn other agent's model, their preferences, their beliefs and their capacities. In the foraging problem context, the preference is the preference for task: which task to be preferred. The belief is the agent's belief about the rocks in the grid environment. The capacity is the agent's ability to solve a specific task. The BDI part maintains an explicit representation of the agents' world. The BN part measures the uncertainty that the agent is facing and the dependent relationship it has with other agents.

The time Constraint Task-based Model proposed in chapter 4 for coordination is used.

<u>Recognition stage</u>: agent A with task T at the beginning of the process. Agent A recognizes the task T of foraging the whole grid world takes too much time to finish itself and it plan to solve the task cooperatively with other agents in the MAS system.

Task decomposition stage: agent A generate a set of sub-tasks ti from the task T. sub-task t_i is defined as a 1*1 grid size foraging problem. As a result, the total number of the sub-task t_i is n*n.

Notation: Sub-task T_i: { Task name, Task requirements, Task rewards, Task time constraint }i=1 to n*n

- Task name: Task_i
- Task requirements: foraging grid area located in [x, y], i= x*y
- Task rewards: 1
- Task time constraint: T_{max}= n*n (one movement require one unit of time)

Team formation stage: in the cooperative BN-BDI MAS, there are many other agents with same capability available. Agent A coordinates with those agents through the Time Constraint Contract Net Protocol. In this coordination protocol, there are 5 stages to form the coordination.

- Recognizing the problem. Agent A has task T and divides task T into n*n sub-task t_i.
- Announcing tasks. Agent A announces all the sub-tasks out.
- Bidding. Available agent receives and evaluates the sub-task announcements. The eligible candidate agents bid for the sub-tasks. The agents evaluate the sub-task (i.e., the location of the foraging place) and compare it with their current locations. Since it is assumed that the reward for any sub-task is identical, agents will bid for the sub-task that near their current location in their belief world. The nearer location, the higher bid.

- Awarding the contract. Agent A receives the bids from candidate agents and awards each sub-task to the highest bidding agent. If more than one agent bids for the same sub-task, agent A will randomly award the sub-task to one of the bidding agents. The reason behind is that they are the homogenous agents in the simulation. In the foraging context, the number of available agents is far less than the sub-tasks that need to be done. After the first round of bidding, the other unassigned sub-tasks will enter into the second round of bidding until all the sub-tasks have been assigned and solved. Agents which are awarded with sub-tasks can choose to execute it their selves or through further tasks decomposition.

Team action stage: Agents who win the biding through the Time Constraint Contract Net Protocol own the sub-tasks in this stage. Agent A now acts as a coordinator agent and manages the agents with sub-tasks. The agents with sub-tasks can divide their sub-tasks and act as a sub-coordination in the team. All the agents in the team form a dynamic hierarchical MAS architecture as in Figure 3-9. Agents with sub-tasks will move towards the destination of the sub-task and pick up the rock to finish the sub-task. During the execution period, agent is not available to bid for other unassigned sub-tasks.

Result Synthesis stage: agents share results with peer agents under the same coordinator agent. Those BN-BDI agents learn from results sharing and improve the problem solving abilities. Roof coordinator agent A synthesizes all the results. All the sub-tasks need to be finished in required time Tmax.

The result sharing of the BN-BDI agents is through a mailing system in the foraging simulation.

5.3 Result Sharing of the BN-BDI agents in Foraging

A multi-agent simulation system for the foraging problem has been built. The framework of faster rock foraging is a system composed of several homogenous agents who works cooperatively to forage the rock. The entire framework was created using java multi-threading. The agent observation model has been simulated using a map window observable interface, using which an agent can observe only a part of the map. Figure 5-2 illustrates a mail system-like communication model that has been implemented using the java event objects, which raises an event on the mail box when a message is sent into it. Agents use an offline mailbox system to share information.

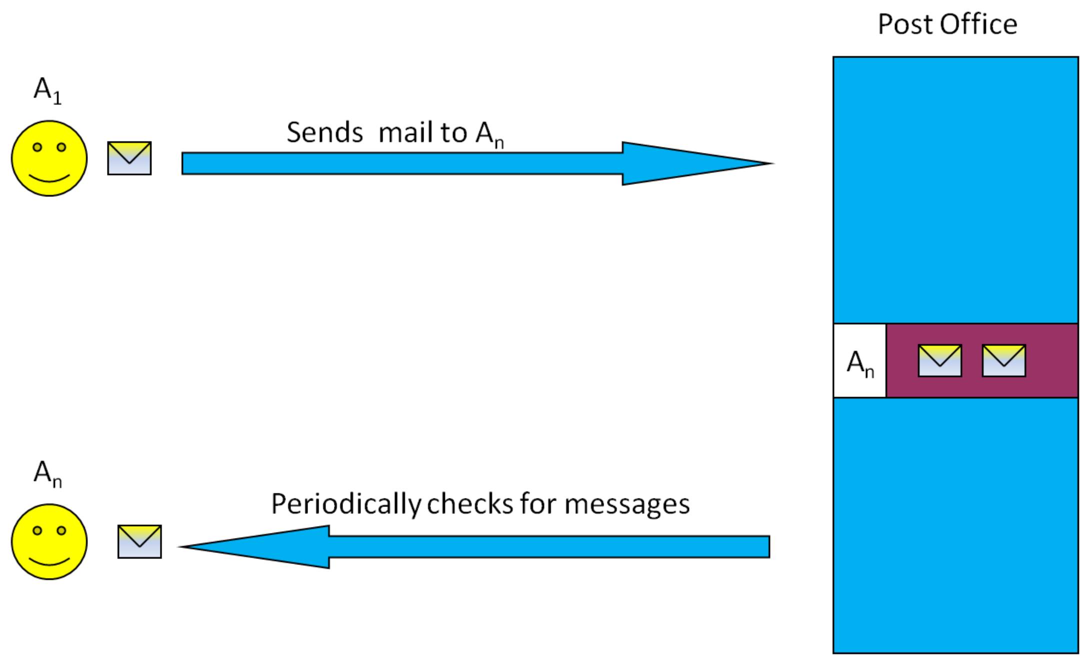


Figure 5-2 The mailing system in the Multi-agent Architecture

Although each agent can only observe a small part of the real virtual world, it has a map of the environment in its mind which is called agent's belief. The initial belief of each agent in the system is set to believe that there are rocks everywhere. The agent uses this belief to make decisions and plan the job scheduling process. In a cooperative Multi-agent environment, the agents can share their beliefs and help each other understand the unknown environment better. Hence in the architecture the agents share their beliefs with each other periodically. Before an agent can make a decision on the next job in hand, it augments its belief by observing the environment around it and also based on what the other agents believe about the environment. Figure 5-3 shows how an agent augments its belief using another agent's belief.

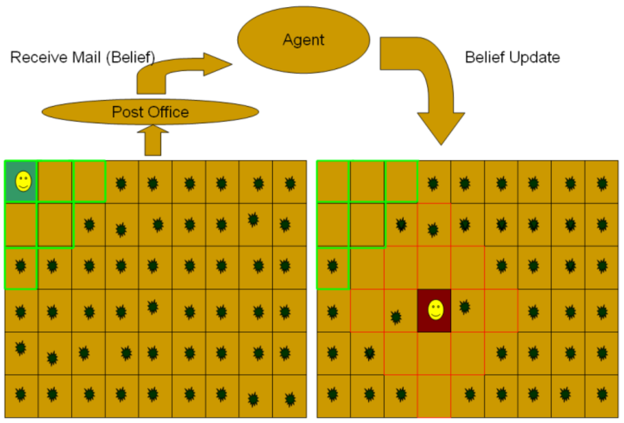


Figure 5-3 Belief augmentation process

5.4 Basic Model

This section introduces the agent behaviors in the basic model. The behavior of a basic agent is simple: it first finds out what it and other agents perceive about the virtual world, exchanges their information through the mailing system and updates its old belief of the world with the incoming new information. According to its belief – the map in its head, the agent moves to the nearest rock location in the belief map. If it perceives that there is any rock, the agent picks the rock up and updates its belief of that part of the world. If there is no rock, it does nothing and just updates its belief. After that, the agent exchanges information with other agents and continues to find another rock according to the map in its head ---its belief. The assumption is that the initial belief it has about the virtual world is that rocks are all over the field and rocks in the virtual world of the simulation are also everywhere. This process is repeated until the agent does not perceive any rock in its belief. Then it supposes the job is finished and it stops moving.

step	Process
1	Agent is created and placed on the global map
2	Observe the map
3	Send belief to all the other agents’ mail box
4	Augments belief based on the information received from the other agents
5	Greedily obtain the nearest job location based on its belief
6	Compute the next move along the path to the job location
7	Update the global map of its coordinates
8	Update its belief
9	If search complete is true, go to step 10; else go to step 2
10	Stop

Figure 5-4 Basic algorithm

The algorithm is fast and highly efficient as it has a built-in goal switching dynamic path planning strategy. Using its observation model and the belief augmentation, the agents can quickly adapt to the changing environment without incurring communication overheads. The rock foraging algorithm steps that every agent runs are as shown in Figure 5-4.

Several experiments about the basic model are conducted on the framework to test its performance. The tests were conducted for different grid sizes and for each case the number of agents was varied to test the scalability of the framework and the algorithm. Agents in the grid are assigned random initial positions at the beginning of the simulation.

5.5 Results and Analysis of Basic Model Performance

The simulation results in Figure 5-5, Figure 5-6 and Figure 5-7 tell that the maximum and average energy consumed for cooperative agents when they are in different virtual grid worlds: 5 by 5 grid, 10 by 10 grid and 50 by 50 grid respectively. As mentioned in the previous section, the energy consumed by agent is actually the number of steps it moves until the rock collection job is finished. The maximum steps taken (energy consumptions) can be seen as an indicator about the quality of the performance, which is how quickly the job can be finished by the MAS. The average steps mean the average steps agents move in the system. It is clear from the above three graphs that as the number of agents increases the time to have the job done decreases accordingly. From the three graphs, the maximum steps and the average steps taken by agents are close, which implies that all the agents are autonomous. All these support the basic model that it can coordinate autonomous multiple agents effectively and robustly to achieve a good system performance. It is also clear from these three figures that both the average time steps taken per agent and the maximum steps taken to complete the task scale well

Figure 5-5 Simulation results of basic model for 5 by 5 grid sizes

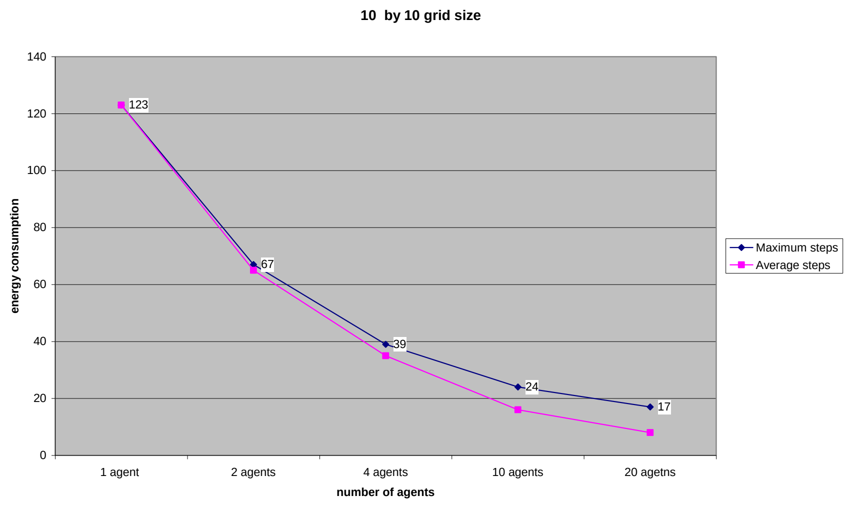


Figure 5-6 Simulation results of basic model for 10 by 10 grid size

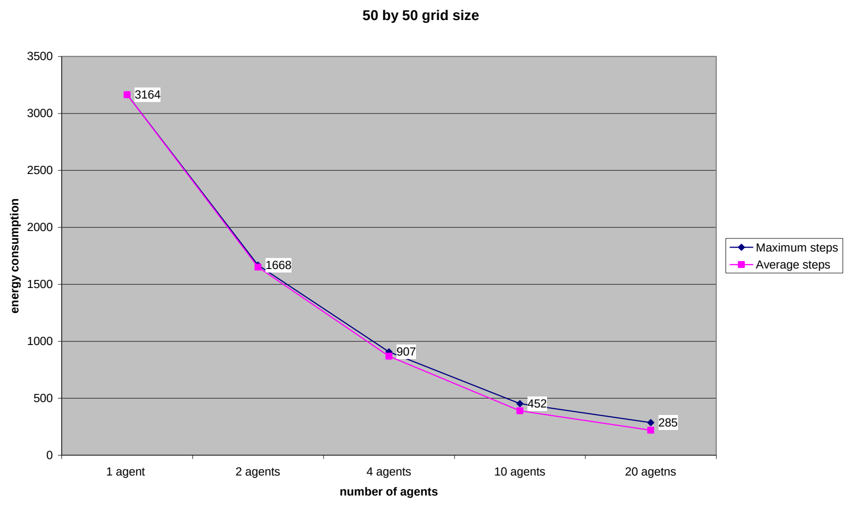


Figure 5-7 Simulation results of basic model for 50 by 50 grid size

as the numbers of agents increases. This justifies the need for running the rock foraging algorithm using cooperative MAS architecture. These figures also show that the decrease of energy consumption for the system is not linear. As more agents are employed in the system, the decreasing rate grows slower. The reason is that for cooperative systems, agents need to communicate with each other and divide the job among themselves.

As a result, some conflict and compromise arise among agents which affect the system's efficiency. Besides that, the rate of reduction in the time steps between the 10 agents case to the 20 agents case is the lowest in 5 by 5 grid size. This implies that the agents can over crowd on the grid if too many of them are used.

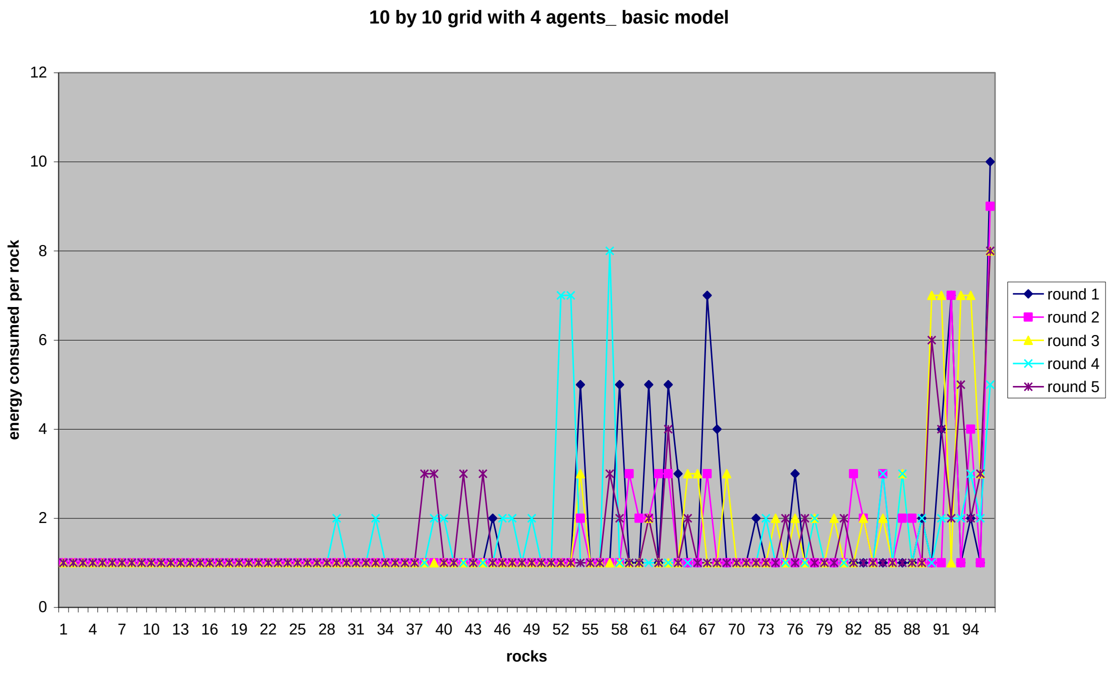


Figure 5-8 Energy consumed to pick up each rock in basic model

The behavior of the basic agents in the foraging problem is studied and a particular problem for the agents has been noticed. When the agents pick up more rocks, the remaining rocks in the environment decrease. From Figure 5-8 of the energy consumed by agent for picking up each rock, it can be observed that when the world becomes sparse and less than 60% of grid is occupied, the performance of the agents becomes inefficient. One round stands for one simulation process. At the beginning of the running, each rock only needs one-step energy to be picked up by agents. However, from the middle of the process until the end, a great part of the remaining rocks need much more step energies.

From the observations, one problem is identified for the agents when the world became sparse: when several agents detect one of the few rocks, all of them run at it while in the end only one agent is able to pick it up. This problem comes about under two conditions: Agents aggregate in a relatively small area and they are chasing for the same target. This is called repeat-chasing problem;

Agents are relatively far from each other and they are chasing for the same target. However one agent is further from the target than the others, but it still waste its energy to go after the rock that would definitely be picked up by other nearby agents. This is called far-chasing problem.

5.6 Adaptive Models

To solve the problems with the sparse world, an adaptive agent is proposed. The roles of the basic agent are reused in the adaptive agent and add two extra waiting roles: out-of-limit wait and over-crowned wait. These two improvements employ some ideas from Bayesian Theory. Agents have prior probability of what to do next. However, new evidence comes and their post probability changes. As a consequence, what they choose to do next changes. How agents can then achieve adaptive behavior is explained below. At the beginning is how the agent adapts its behavior to cope with the repeat-chasing problem. Then the far-chasing problem is studied.

<u>Repeat-chasing problem. </u>The process starts with the role of exchanging belief among agents. As long as the agent perceives any rock in its belief, it moves to the nearest target. However, if it finds out from those incoming information that the number of agents

around it in its perceiving range is more than the number of available rocks, it may switch to randomly stop for a while in case that they are chasing for the same target to avoid

Step	Process
1	Agent is created and placed on the global map
2	Observe the map
3	Observe the virtual world and check whether nearby agent number is more than nearby available rocks. If yes, agent freezes for a moment
4	Send belief to all the other agents' mail box
5	Augments belief based on the information received from the other agents
6	Greedily obtain the nearest job location based on its belief
7	Check whether search depth is out of a pre-defined limit. If yes, agent freezes for a moment
8	Compute the next move along the path to the job location
9	Update the global map of its coordinates
10	Update its belief
11	If search complete is true, go to step 12; else go to step 2
12	Stop

Figure 5-9 Adaptive algorithm

energy wasting. The stopping period depends on the number of the agents nearby. The more the nearby agents, the longer the random waiting time is. After the freezing period, the agent rechecks the target rock and continues moving.

Far-chasing problem. The process is the same as the basic model at the beginning. However, the agent here has a pre-defined belief limit range when it searches its belief map. If an agent finds out that the available rocks for it to collect is beyond the limit range according to its belief map in mind, the agent would suppose that it is not worthwhile for it to begin to go to collect that particular rock immediately. It is highly possible that there may be some other agents near the target rock. So the agent decides to let other agents collect the rock instead and it would stop moving for a while. But if the agent finds the target rock is still available, it would move to that direction. These two components are both added into the adaptive model and the process for the adaptive model changes to Figure 5-9.

5.7 Coordination Complexity

The scalability of multi-agent algorithms lie in that the less coordination a MAS requires, the better it should scale to large numbers of agents.

With this definition, it turns out that the worst case complexity of any reasonable algorithm is $O(n^2)$: all agents must communicate constantly with all other agents. With $O(n^2)$ complexity, the bandwidth of the communication system must increase with the square of the number of agents. For this reason, an $O(n^2)$ complexity algorithm or task is not considered scalable. In an $O(n)$ complexity algorithm, on the other hand, bandwidth needs only to scale linearly with the number of agents. In other words, for every 100 agents, if one new agent joins in the team, the MAS performance will not be suffered.

The communication schemes investigated range from full communication to no communication. In chapter 4 the coordination complexity section, two of the schemes are given. The first, distance modulated communication (DMC) has communication complexity $O(n \log n)$ or $O(n^{1.5})$ depending on assumptions about the dynamics of the robots. The second, the wandering communication scheme (WCS) has communication complexity $O(n)$ or $O(1)$ depending on assumptions about how certain higher level decisions are made by the robots (Klavins 2004).

In the adaptive models, the Repeat-chasing problem and Far-chasing problem use the similar communication schemes to improve the coordination performance which result in the scalability and robustness of the cooperative BN-BDI Multiagent system.

At every step of a multi-agent algorithm, the costs of the communication events that occurred in that step are summed up and then take the average over all the steps in the coordination algorithm. In other words, for each agent $j \in J$, the step taken by the agent is $i \in I_j$, $I = \cup_{j=1}^{J} I_j$. I_j is the dataset of steps taken by agent j, and I is the dataset of steps that the agents system takes to finish their job. The cost of step i taken by agent j is $cost_j(i)$. So agent j costs $\sum_{i=1}^{I_j} \mathrm{cost}_{\mathrm{j}}(\mathrm{i})$ and the system's performance is $\frac{1}{J}\sum_{j=1}^{J}\sum_{i=1}^{I_j} \mathrm{cost}_{\mathrm{j}}(\mathrm{i})$.

5.8 Results and Analysis of Adaptive Model Performance

The simulation results of the adaptive model tell that these two improvements really increase the system's efficiency. Figure 5-10 shows the comparison results between basic model and adaptive model on the energy spent on collecting each available rock. This figure is based on the average data values of 6 rounds of experiment results where agents are put randomly in the virtual world full of rocks at the beginning of the simulation. It can be seen from the figure that as the world becomes sparse, the inefficiency problem of the basic model has been largely avoided in the adaptive model.

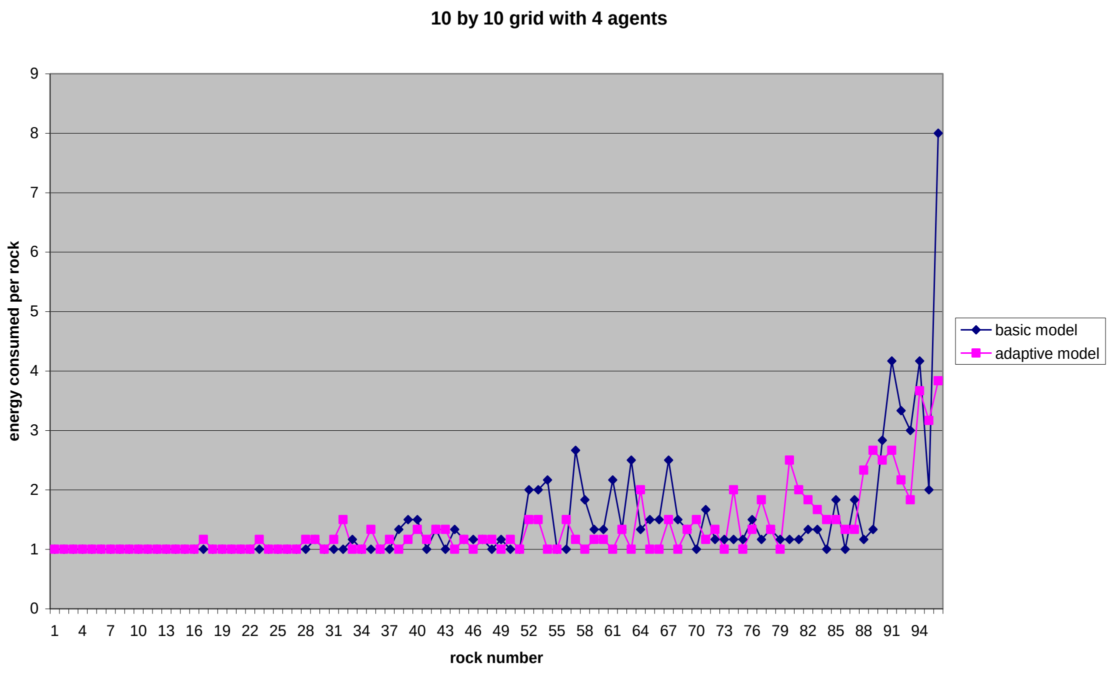


Figure 5-10 Energy needed to pick up each rock: comparison between basic and adaptive model

Besides the energy consumed to collect each rock, the total energy consumed in the system by all the agents is another important index to measure the performance of the MAS. This number shows how much the total cost of the system is, with a smaller number being better. Figure 5-11, Figure 5-12 and Figure 5-13 list this measurement in both basic model and adaptive model with different number of running agents under 3

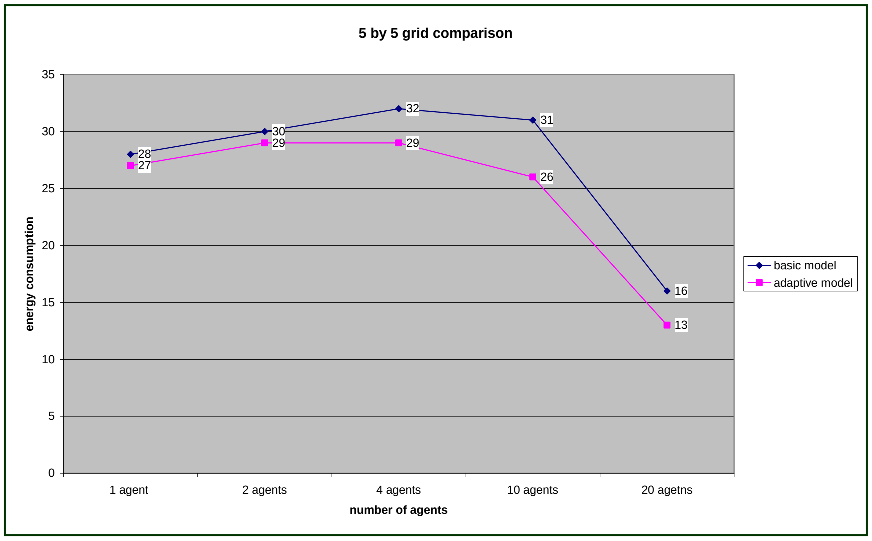


Figure 5-11 Total energy consumption for agents in 5 by 5 grid size

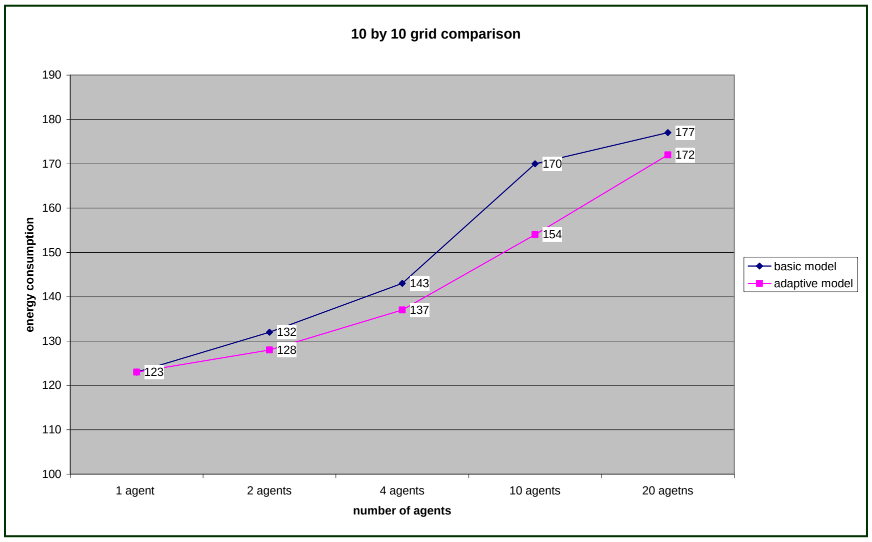


Figure 5-12 Total energy consumption for agents in 10 by 10 grid size

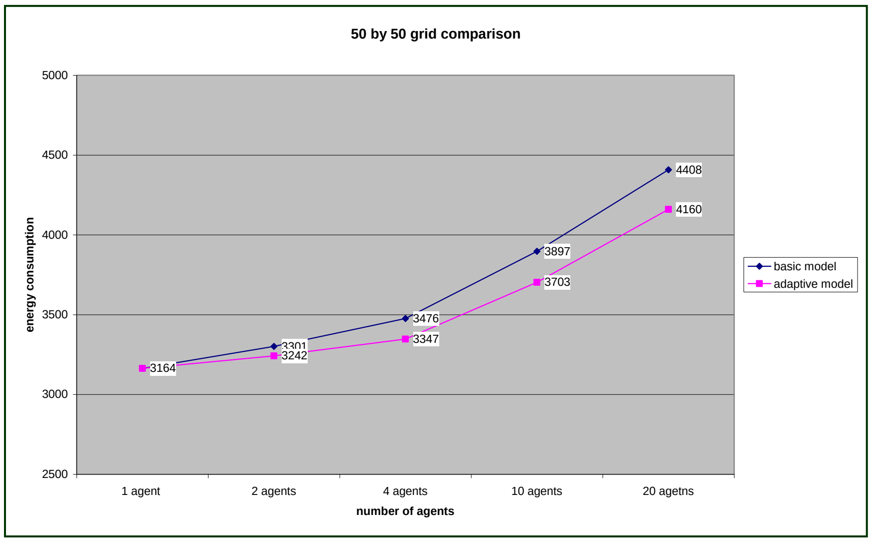


Figure 5-13 Total energy consumption for agents in 50 by 50 grid size

different grid sizes of environment. From these figures, it can be told that the adaptive model improves the system efficiency significantly compared with the basic model. For example, in the case of 10 by 10 grid size with 4 agents (Figure 5-12), MAS gets a 20% improvement. (Total energy consumed in basic model is 170; in adaptive model, the total energy consumed is 154. Number of rocks in the world is 90. The system performance improvement is 20 %=(170-154)/ (170-90))

Comparison tests are also done by pre assigning the initial positions of the agents. This helps to optimize the foraging of the environment. In the basic model, if the agents are located far apart, they bring in more information, reduce collisions among agents and will not bring about overlapping observations. All these can help in increasing the performance of the system and optimize the number of steps they take in completing the

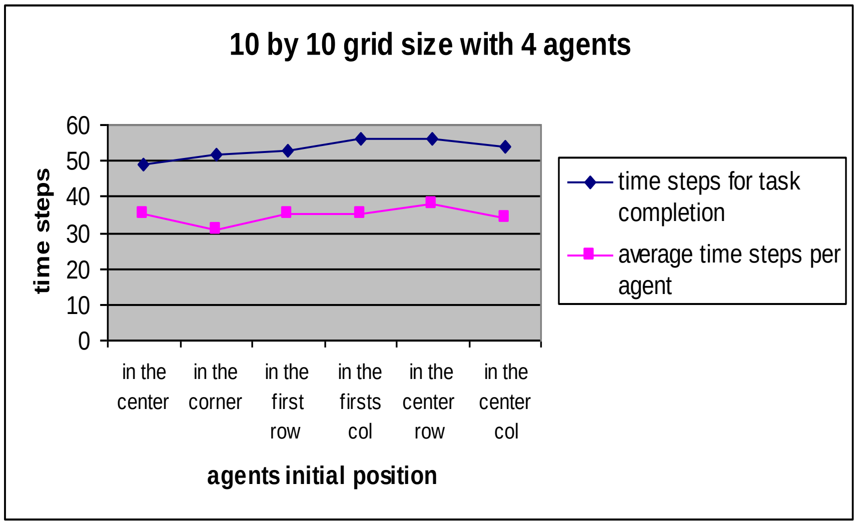


Figure 5-14 Basic model results with different agents' initial positions

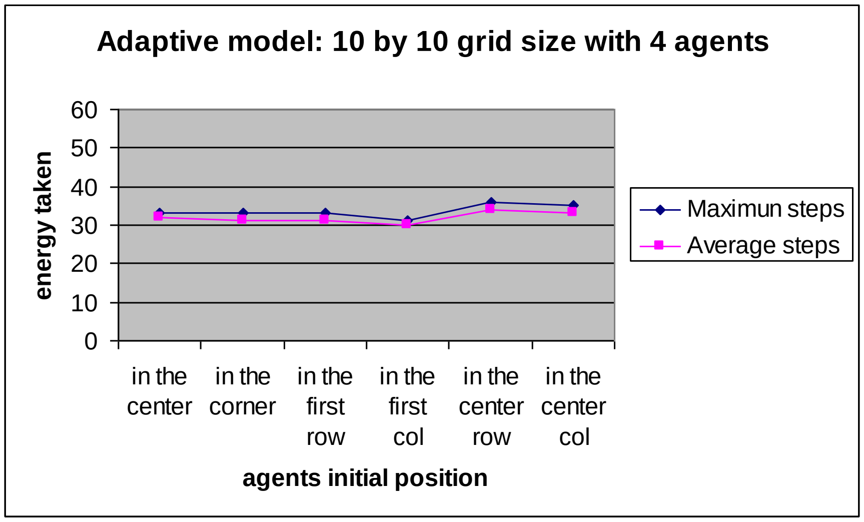


Figure 5-15 Adaptive model results with different agents' initial positions

job. From Figure 5-14, it is found that in the basic model, the agents did the best when placed in the four corners of the map. In the adaptive model (Figure 5-15), since there are two added opponents, agents placed together or near each other do not affect the results much compared with the basic model.

5.9 Summary

In this chapter, a cooperative BN-BDI multi-agent simulation environment is built for the foraging problems. Agents of the BN-BDI model forming a hierarchical MSA framework are applied in the foraging problem. The task of foraging the whole grid world is decomposed into n*n sub-task of foraging a specific location. The time constraint contract net protocol is used for the coordination of the agents awarded with sub-tasks. The results sharing and synthesis are communicated through the mailing system. Two models based on the theory are simulated. The basic model scales well as the number of foraging agents increasing. It gives a good system performance but it has some inefficiency when the rocks become less in the environment. The reasons are explored out and an improved model is proposed, and results in an adaptive model. The new model employs two added opponents and simulation results show it significantly improves the system's performance. In these two models, autonomous agents work cooperatively with each other to forage rocks in a virtual world. Information sharing among agents is allowed and agents not only can see the working environment but also can have a subjective perspective of it based on the belief component in their models. All these characteristics make the architecture scalable, reusable and robust.

Chapter 6 Conclusion and Future Work

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

6.1 Contribution

In this thesis, an adaptive Multi-agent system is built. Since it is impossible for designers to determine the behavioral repertoire and concrete activities of a MAS at the time of its design and prior to its use, agents need to be adaptive to the changes and new information from the environment, its fellow team agents and itself. To be adaptive, it is suggested that agents need to have the learning ability and can coordinate with one another to handle the big and complex problems.

Firstly, the BN-BDI agent model is proposed. It is a hybrid architecture that has the merits of both the deliberative and reactive architecture. The BDI part maintains an explicit representation of the agents' world. The BN part measures the uncertainty that the agent is facing and the dependent relationship it has with other agents. The BN-BDI agent can learn other agent's model, their preferences, their beliefs and their capacities.

Secondly, a hierarchical MAS architecture based on the BN-BDI agent is proposed. Agents can choose to be or not to be a coordinator and this dynamic nature give the flexibility to agents. The role of coordinator helps to decrease the communication and coordination overhead. Such structure will facilitate the coordination of agents in cooperative MAS and give more adaptability to the MAS performance.

Thirdly, the time constrained task-based model (TCTM) is proposed. TCTM simplifies MAS cooperation problem. In TCTM, the agent's role is not fixed. Agents can behave as a leader to sub-contract the task or as a bidder to bid for a sub-task in different course of team formation; in TCTM, Agents can learn from their team fellows and benefit from result sharing through communication through the proposed Time Constraint Contract Net Protocol. The communication complexity of the model is $O(n\log n)$ and it significantly reduces the amount of information being exchanged and scales well with the growing agent numbers.

Finally, in the foraging simulation, the proposed model, system architecture and coordination mechanism are tested using two heuristic algorithms with different sizes of the virtual world, with different number of agents, and with different agents' initial positions. The simulation results support the proposed hypothesis.

6.2 Future work

This thesis has provided insights into the adaptive Multi-agent system and coordination in MAS. New agent model, corresponding system architecture and coordination mechanism are proposed and deployed in experiments.

At present, the research in the thesis has only investigated the homogenous agents in the cooperative MAS without uncertainty. The possible extension could be in the following two aspects.

The first one is to extend the TCTM model and relax its constraints and assumptions, for example, to handle dynamic problems. The dynamic nature means that an agent may dysfunction when it accepts the bid; or a new agent might suddenly join the system and replace an old agent.

The second one is to improve the efficiency of the coordination method and at the same time to reduce the communication complexity. The relentless desire to pursue efficiency and effectiveness is applicable in adaptive MAS: How to deal with uncertainty, for example in the foraging problem, the rocks distribution in the grid world can be un-uniform and the status of rocks can be dynamic.

The third direction can be to extend the homogenous agents to heterogeneous agents. The heterogeneous agents can be applied in a wider practical area, but the learning and coordination mechanism are much more complex.

Bibliography

Abraham, A., K. Franke, et al. (2003). Intelligent systems design and applications. Berlin ; New York, Springer.

Abraham, A., M. Köppen, et al. (2003). Design and application of hybrid intelligent systems. Amsterdam ; Washington, DC.

Arkin, R. C. (1998). Social Behavior. Behavior-Based Robotics. Cambridge, MA, MIT Press.

Asama, H., Y. Ishida, et al. (1989). Design of an autonomous and distributed robot system: ACTRESS. IEEE/RSJ Inter- national Conference on Intelligent Robots and Systems, Tsukuba, Japan.

Bacchus, F. and A. Grove (1995). Graphical models for preference and utility, Montreal, Que., Canada, Morgan Kaufmann Publishers.

Beni, G. and J. Wang (1989). Swarm intelligence in cellular robotic systems. NATO Advanced Workshop on Robots and Biological Systems.

Blum, C. and D. Merkle (2008). Swarm intelligence : introduction and applications. Berlin, Springer.

Bond, A. H. (1990). Distributed decision making in organizations. IEEE International Conference on Systems, Man and Cybernetics, Los Angeles, CA , USA

Bratman, M. (1987). Intention, plans, and practical reason. Cambridge, Mass., Harvard University Press.

Brazier, F. M. T., B. M. DuninKeplicz, et al. (1997). "Desire: Modelling multi-agent systems in a compositional formal framework." International Journal of Cooperative Information Systems **6**(1): 67-94.

Brooks, R. (1991). "Intelligence without representation." Artificial Intelligence **47**(1-3): 20.

Burmeister B. , S. K. (1992). Cooperative problem solving guided by intentions and perception. the Third European Workshop on Modeling Autonomous Agents and Multi Agent Worlds, Amsterdam, The Netherlands, Elsevier Science Publishers.

Crowston, K. (1991). Towards a coordination cookbook : recipes for multi-agent action. Cambridge, Mass., Alfred P. Sloan School of Management, Massachusetts Institute of Technology.

d'Inverno, M., M. Luck, et al. "The dMARS architecture: a specification of the distributed multi-agent reasoning system." Autonomous Agents and Multi-Agent Systems **9**(1-2): 5-53.

David Carmel , S. M. (1996). Learning Models of Intelligent Agents. Proceedings of the Thirteenth National Conference on Artificial Intelligence, Portland, Oregon.

Davis, R. and R. G. Smith (2003). "Negotiation as a metaphor for distributed problem solving." Communication in Multiagent Systems **2650**: 51-97.

Drogoul, A. and J. Ferber (1992). From Tom Thumb to the Dockers: Some Experiments with Foraging Robots. International Conference on Simulation of Adaptive Behavior, MIT Press.

Durfee, E., V. Lesser, et al. (1989). "Trends in cooperative distributed problem solving." IEEE Transactions on Knowledge and Data Engineering **1**(1): 20.

Edmund H. Durfee , V. R. L. (1989). Negotiating task decomposition and allocation using partial global planning. Distributed Artificial Intelligence San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. . **2:** 229-243.

Faltings, B. (2000). "Intelligent Agents: Software Technology for the new Millennium." Informatik Informatique **1**: 2-5.

Ferguson, I. A. (1992). TouringMachines: An Architecture for Dynamic, Rational, Mobile Agents. Ph.D., University of Cambridge.

Garcia, A. (2003). Software engineering for large-scale multi-agent systems : research issues and practical applications. Berlin ; New York, Springer.

Georgeff, M., B. Pell, et al. (1999). The belief-desire-intention model of agency, Paris, France, Springer-Verlag.

Georgeff, M. P. (1989). Decision-making in an embedded reasoning system the Eleventh International Joint Conference on Artificial Intelligence Morgan Kaufmann.

Gilad Zlotkin, J. S. R. (1993). A domain theory for task oriented negotiation. IJCAI'93 Proceedings of the 13th international joint conference on Artifical intelligence, Morgan Kaufmann Publishers Inc. San Francisco, CA, USA

He, M. H., H. F. Leung, et al. (2003). "A fuzzy-logic based bidding strategy for autonomous agents in continuous double auctions." IEEE Transactions on Knowledge and Data Engineering **15**(6): 1345-1363.

Heckerman, D. (2008). A tutorial on learning with Bayesian networks. Innovations in Bayesian Networks. **156:** 33-82.

Hernandez, A. G., A. El Fallah-Seghrouchni, et al. (2004). "Distributed learning in intentional BDI multi-agent systems." Proceedings of the Fifth Mexican International Conference in Computer Science: 7.

Ho, F. and M. Kamel (1998). "Learning coordination strategies for cooperative multiagent systems." Machine Learning **33**(2-3): 155-177.

Iglesias, C. A., M. Garijo, et al. (1999). "A survey of agent-oriented methodologies." Intelligent Agents V **1555**: 317-330.

J Deneubourg, S. G., G Sandini, F Ferrari (1990). Self-organizating collection and tranport of objects in unpredictable environments. Janpan-U.S.A. Symposium on Flexible Automation, Kyoto, Japan.

Jennings, N. R., P. Faratin, et al. (2001). "Automated negotiation: Prospects, methods and challenges." Group Decision and Negotiation **10**(2): 199-215.

Jennings, N. R., K. Sycara, et al. (1999). "A roadmap of agent research and development." Autonomous Agents and Multi-Agent Systems **1**(1): 7-38.

Keeney, R. L. and H. Raiffa (1976). Decisions with multiple objectives : preferences and value tradeoffs. New York, Wiley.

Klavins, E. (2002). Communication complexity of multi robot systems. Algorithmic Foundations of Robotics V, volume 7 of Springer Tracts in Advanced Robotics, Springer**:** 275-292.

Klavins, E. (2004). Communication Complexity of Multi-robot Systems. Algorithmic Foundations of Robotics V. J.-D. Boissonnat, J. Burdick, K. Goldberg and S. Hutchinson, Springer Berlin Heidelberg. **7:** 275-292.

Koller, D. and B. Milch (2003). "Multi-agent influence diagrams for representing and solving games." Games and Economic Behavior **45**(1): 181-221.

Konolige, K. (1986). A Deduction Model of Belief. San Mateo, CA, Pitman

Korf, R. E. (1990). "Real-time heuristic search." Artificial Intelligence **42**(2-3): 189-211.

Koulinitch, A. S. and L. B. Sheremetov (1998). "Coordination and communication issues in multi-agent expert system: concurrent configuration design advisor." Expert Systems with Applications **15**(3-4): 295-307.

Kube, C. R. and H. Zhang (1996). The use of perceptual cues in multi-robot box-pushing. IEEE International Conference on Robotics and Automation, Minneapolis, MN , USA

Kweon, I. S., Y. Kuno, et al. (1992). Behavior-based Intelligent Robot In Dynamic Indoor Environments. lEEE/RSJ International Conference on Intelligent Robots and Systems.

Lesser, V. R. (1999). "Cooperative multiagent systems: A personal view of the state of the art." IEEE Transactions on Knowledge and Data Engineering **11**(1): 133-142.

Levesque, H. J. (1984). "A Logic of Implicit and Explicit Belief." Proceedings of the Fourth National Conference on Artificial Intelligence: 5.

Liu, B. (2003). "Scheduling via reinforcement." Artificial Intelligence in Engineering **3**(2): 9.

Luck, M., V. Mark, et al. (2001). Multi-agent systems and applications. New York, Springer.

MacNab, Y. C. (2003). "A Bayesian hierarchical model for accident and injury surveillance." Accident Analysis and Prevention **35**(1): 91-102.

Maes, P., Ed. (1990). Designing Autonomous Agents -Theory and Practice from Biology to Engineering and Back, MIT Press.

Maes, S. and P. Leray (2006). Multi-agent causal models for dependability analysis. First International Conference on Availability, Reliability and Security.

Maes, S., J. Reumers, et al. (2003). Identifiability of causal effects in a multi-agent causal model, Halifax, NS, Canada, IEEE Comput. Soc.

Mataric, M. (1998). Coordination and learning in multirobot systems. **13:** 6-8.

Mataric, M. J. (1998). "Behavior-based robotics as a tool for synthesis of artificial behavior and analysis of natural behavior." Trends in Cognitive Sciences **2**(3): 82-87.

McConnell, J. B. (2003). Technical and Policy Issues Surrounding the Use of Autonomous Maneuverable Earth Observing Satellites. Master of Science, Massachusetts Institute of Technology.

Meganck, S., S. Maes, et al. (2005). Distributed learning of multi-agent causal models, Compiegne, France, IEEE Comput. Soc.

Moreira, A. F., R. Vieira, et al. (2004). "Extending the operational semantics of a BDI agent-oriented programming language for introducing speech-act based communication." Declarative Agent Languages and Technologies **2990**: 135-154.

Moreno, A. and J. L. Nealon (2003). Applications of software agent technology in the health care domain. Basel, Switzerland ; Boston, Birkhäuser Verlag.

Muller J. P., P. M., Thiel M. (1995). Modelling reactive behaviour in vertically layered agent architectures. Intelligent Agents: Theories, Architectures and Languages, Heidelberg, Germany, Springer-Verlag.

Newell, A. and H. A. Simon (1976). "Computer Science as Empirical Inquiry: Symbols and Search." Communications of the ACM **19**(3): 13.

Norman Carver, Z. C., Victor Lesser (1991). Sophisticated cooperation in FA/C distributed problem solving systems. AAAI'91 Proceedings of the ninth National conference on Artificial intelligence AAAI Press ©1991.

Nwana, H. S. (1996). "Software agents: An overview." Knowledge Engineering Review **11**(3): 205-244.

Panait, L. and S. Luke (2004). A pheromone-based utility model for collaborative foraging, New York, NY, United States, Association for Computing Machinery, New York, NY 10036-5701, United States.

Parker, L. E. (1998). ALLIANCE: an architecture for fault tolerant multirobot cooperation. IEEE Transactions on Robotics and Automation.

Parker, L. E. (2000). "Lifelong adaptation in heterogeneous multi-robot teams: Response to continual variation in individual robot performance." Autonomous Robots **8**(3): 239-267.

Parunak, H. V., A. C. Ward, et al. (1999). "The MarCon algorithm: A systematic market approach to distributed constraint problems." Ai Edam-Artificial Intelligence for Engineering Design Analysis and Manufacturing **13**(3): 217-234.

Parunak, R. V. (2000). "Agents in overalls: Experiences and issues in the development and deployment of industrial agent-based systems." International Journal of Cooperative Information Systems **9**(3): 209-227.

Peng, Y. B., J. Gao, et al. (2008). "An Extended Agent BDI Model with Norms, Policies and Contracts." International Conference on Wireless Communications, Networking and Mobile Computing **1**(31): 5014-5017.

Rao, A. S. and M. P. Georgeff (1991). Modeling Rational Agents Within a BDI-architecture, Australian Artificial Intelligence Institute.

Rao, A. S. and M. P. Georgeff (1995). BDI agents: from theory to practice, San Francisco, CA, USA, AAAI Press.

Ren, Z. and A. B. Williams (2003). Lessons learned in single-agent and multiagent learning with robot foraging, Washington, DC, United States, Institute of Electrical and Electronics Engineers Inc.

Ronald A. Howard , J. E. M., Ed. (1989). Readings on the principles and applications of decision analysis Menlo Park, Calif, Strategic Decisions Group

Sahin, F. (2000). A bayesian network approach to the self-organization and learning in intelligent agents. P.h D., Rochester Institute of Technology.

Sandholm, T. (1993). An Implementation of the Contract Net Protocol Based on Marginal Cost Calculations. . National Conference on Artificial Intelligence, Menlo Park, Calif, American Association for Artificial Intelligence.

Scerri, P., R. Vincent, et al. (2005). Coordination of large-scale multiagent systems. New York, Springer.

Schumacher, M. (2001). Objective coordination in multi-agent system engineering : design and implementation. Berlin ; New York, Springer.

Smith, R. G. (1980). "The contract net protocol: High-level communication and control in a distributed problem solver." IEEE Transactions on Computers **29**(12): 9.

Spiegelhalter, D. J., A. P. Dawid, et al. (1993). Bayesian Analysis in Expert Systems, JSTOR. **8:** 219-247.

Steels, L. and P. Hanappe (2006). "Interoperability through emergent semantics - A semiotic dynamics approach." Journal on Data Semantics Vi **4090**: 143-167.

Steels, L. and M. Tokoro (1995). "Artificial life and real world computing." Computer Science Today **1000**: 15-28.

Stilwell, D. J. and J. S. Bay (1993). Toward the development of a material transport system using swarms of ant-like robots. IEEE International Conference on Robotics and Automation, Atlanta, GA , USA.

Stirling, W. C. and R. L. Frost (2005). "Social utility Functions-part II: applications." IEEE Transactions on Systems, Man and Cybernetics, Part C (Applications and Reviews) **35**(4): 533-543.

Stone, P. and M. Veloso (2000). "Multiagent systems: A survey from a machine learning perspective." Autonomous Robots **8**(3): 345-383.

Sycara, K. (1997). Using Option Pricing to Value Commitment Flexibility in Multiagent Systems, School of Computer Science, Carnegie Mellon University.

Sycara, K. P. (1990). "Negotiation Planning - an Ai Approach." European Journal of Operational Research **46**(2): 216-234.

Theraulaz, G., J. Gervet, et al. (1991). "Social Regulation of Foraging Activities in Polistes Dominulus Christ - a Systemic Approach to Behavioral Organization." Behaviour **116**: 292-320.

Tian, F., H. Wang, et al. (2002). Inference and modeling of multiply sectioned Bayesian networks, Beijing, China, IEEE.

Tian, J. and J. Pearl (2002). A new characterization of the experimental implications of causal Bayesian networks. National Conference on Artificial Intelligence (Aaai-02)/Fourteenth Innovative Applications of Artificial Intelligence Conference (Iaai-02).

Tresp, V. and K. Yu (2005). An introduction to nonparametric hierarchical bayesian modelling with a focus on multi-agent learning, Maynooth, Ireland, Springer Verlag, Heidelberg, D-69121, Germany.

Undeger, C. and F. Polat (2010). "Multi-agent real-time pursuit." Autonomous Agents and Multi-Agent Systems **21**(1): 69-107.

Wang, J. (1994). On sign-board based inter-robot communication in distributed robotic systems. IEEE International Conference on Robotics and Automation, San Diego, CA.

Weiss, G., Ed. (1999). Multiagent Systems. Intelligent Agents, The MIT Press.

Weiss., G. (1999). Multiagent systems : a modern approach to distributed artificial intelligence, MIT Press.

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

Wooldridge, M., N. R. Jennings, et al. (2000). "The Gaia methodology for agent-oriented analysis and design." Autonomous Agents and Multi-Agent Systems **3**(3): 285-312.

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

Xiang, Y., X. An, et al. (2003). "Simulation of Graphical Models for Multiagent Probabilistic Inference." Simulation **79**(10): 545-567.

Zhang, Z. Y., W. Y. Liu, et al. (2002). Dynamic multi-agent influence diagrams for modeling multistage games. International Conference on Machine Learning and Cybernetics.

Zheng, Z. Q. and P. A. Pavlou (2010). "Toward a Causal Interpretation from Observational Data: A New Bayesian Networks Method for Structural Models with Latent Variables." Information Systems Research **21**(2): 365-391.

Zhongyan L., W. L. a. L. L. (1999). Agent-based Information Processing System Architecture. Proceedings of FUSION’99, Sunnyvale, USA, Omnipres.

Appendix A Graphical Models of MAS

Since MAS are very complex systems with many uncertainties, it is important to find a suitable architecture to build it. In this section, the literatures of probabilistic networks for MAS architecture are reviewed. A probabilistic network, such as the Bayesian Network, Influence Diagram and Markov network, is an ideal tool to model uncertainty.

Probabilistic network has become an established framework for representing and reasoning with uncertain knowledge. A probabilistic network consists of a dependency structure coupled with a corresponding set of probabilistic tables. The dependency structure is a graphical representation of the conditional independencies that are known to hold in the problem domain. These conditional independencies are needed to provide an economical representation of a joint probabilistic distribution over the problem domain. It may be interesting and challenging to investigate MAS with a probabilistic network approach.

Multiply-sectioned Bayesian Networks

Multiply-sectioned Bayesian networks (MSBNs) were motivated by the development of a medical diagnostic system under the single agent paradigm, and was then extended to the multi-agent paradigm. An MSBN forms the core of a cooperative MAS for diagnosis of a large system. Each agent is equipped with private knowledge about a subsystem and acts autonomously and cooperatively with other agents.

MSBN is first developed by (Klavins 2004) for single-agent oriented and modular knowledge representation and more efficient inference. It was then extended to

MAS(Xiang 1996). The MAS referred to consists of autonomous and cooperative agents with self-interest. The MSBN provides a framework in which multiple agents can estimate the state of a domain effectively with exact and distributed probabilistic inference. It aims to tackle a large and complex knowledge domain by dividing the domain into several subsets each of which is related with an intelligent agent. With a distributed fashion, an MSBN allows the privacy protection of intelligent agents and the active communication in a Multi-agent system.

The structure of an MSBN is defined to have the following properties: a hypertree organization is used for the subnets; each subnet is a connected DAG; the union of all subnets is also a connected DAG; nodes shared by adjacent subnets for d-sepset; the adjacent agents in the hypertree are conditionally independent and consistent. Figure A-1 provides the hypertree organization of subnets.

An MSBN M is a collection of Bayesian subnets that together defines a BN(Tian, Wang et al. 2002). M represents probabilistic dependence of a total universe of variables partitioned into multiply sub domains. Variables in each sub domain model a subsystem, which is represented by a subnet. Each subnet corresponds to an agent. For agents to reason exactly through message passing, their communication pathways need to satisfy a condition known as a hypertree, defined as follows (Xiang, An et al. 2003):

Definition A-1: Let $G_i = (V_i, E_i)(i = 0,1,..., n-1)$ be a set of graphs where V_i is the set of nodes and E_i is the set of edges in graph G_i. The graph $G = (\bigcup_i V_i, \bigcup_i E_i)$ also denoted by $\bigcup G_i$, is referred to as the union of $\mathrm{G}_0, \mathrm{G}_1,...,$ and $\mathrm{G}_{n\text{-}1}$.

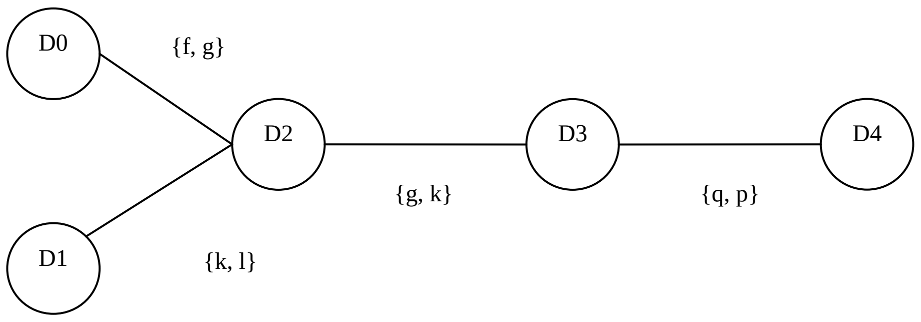


Figure A-1 Hypertree organization of subnets

Definition A-2: Let G = (V ,E) be a connected graph sectioned into subgraphs $\{G_i = (V_i,E_i)\}$. Let the G_is be organized as a connected tree Ψ, where each node is labeled by a G_i and each link between G_k and G_m is labeled by the interface $V_k \cap V_m$ such that for each i and j , $V_i \cap V_j$ is contained in each subgraph on the path between G_i and G_j in Ψ. Then Ψ is a hypertree over G. Each G_i is a hypernode, and each interface is a hyperlink.

The following definition of an MSBN specifies how the numerical distributions are associated with the structure (Xiang 1996).

Definition A-3: An MSBN M is a triplet (V,G, P). $V = \cup_i V_i$ is the total universe, where each V_i is a set of variables called a subdomain. $G = \cup_i G_i$ (a hypertree MSDAG) is the structure where nodes of each subgraph G_i are labeled by elements of V_i . Let x be a variable and π(x) be all parents of x in G. For each x, exactly one of its occurrences (in a containing $\{x\} \cup \pi(x)$) is assigned $P(x|\pi(x))$, and each occurrence in other subgraphs is assigned a unit constant potential. $P = \cup_i P_i$ is the joint probability distribution (JPD), where each P_i is the product of the potentials associated with nodes in G_i . Each triplet S_i

= (V_i,G_i, P_i) is called a subnet of M. Two subnets S_i and S_j are said to be adjacent if G_i and G_j are adjacent in the hypertree.

It can be seen that an MSBN comprises a set of Bayesian networks that share some common nodes. The common nodes compose an interface *S* between adjacent Bayesian networks associated to individual agents. One important property of the interface in an MSBN is stated as follows: the adjacent agents are independent conditioned on the observation of states in the interface which is the only channel for all their communication.

Ensuring the correctness of agent communication also includes that nodes shared by agents should form a d-sepset.

Definition A-4: Let G be a directed graph such that a hypertree over G exists. A node x contained in more than one subgraph with its parents π(x) in G_is is a d-sepnode if there exists one subgraph that contains π(x). An interface I is a d-sepset if every x $\in$ I is a d-sepnode.

In an MSBN, the intersection between each pair of subnets must satisfy the d-sepset condition because the semantics of joint probability distribution of a cooperative MAS is defined based on this condition. With this condition, in order to bring two adjacent subnets up-to-date, it is sufficient to pass the new probability distribution on the d-sepset between them and nothing else.

Using an MSBN, multiple agents can perform probabilistic reasoning by local inference and inter-agent message passing. To make the local inference efficient, each subnet must be sparse. To make message passing efficient, the d-sepset between each pair of adjacent agents should be small compared with the corresponding subdomains. The d-sepset should also be sparsely connected so that an efficient run time representation can be derived. That is to say probabilistic inference and reasoning in MSBN works better in a sparse network environment.

There are several advantages to represent cooperative MAS with MSBN. It can measure the exact probability of belief; it communicates by belief over small sets of shared variables; its organization of agents is simpler; it uses DAG for domain structuring; its joint belief admits agents' belief on internal variables and combining their beliefs on shared variables; its agents are cooperative and trustful to each other while the internal know-how is protected, that is it protects agents privacy; it ensures disciplined communication.

Methods to simulate an MSBN are available. Because of its special model requirements, a back track simulation process is used. The procedure is as follows: firstly to simulate a hypertree topology; then simulate hypernodes and hyperlinks as junction trees in a breadth-first fashion; after that, convert the junction tree at each hypernode into a connected DAG; finally simulate probability parameters.

With its distributed framework and efficient inference methods, an MSBN provides a good solution for a Multi-agent reasoning problem. Since MSBN is built under strict requirements, some extensions and relaxations of the MSBN framework can be made in the future. Less fundamental constraints can be relaxed, such as, if subdomain structures and observation patterns are less than general, the d-sepset restriction can be relaxed.

Multi-agent Causal Model

Multi-agent causal models (MACM) (Maes, Reumers et al. 2003; Meganck, Maes et al. 2005) are an extension of causal Bayesian Networks (CBN) (Tian and Pearl 2002)to a distributed domain. In this setting, it is assumed that there is no single database containing all the information of the domain. Instead, there are several sites holding non-disjoint subsets of the domain variables. At each site, there is an agent capable of learning a local causal model.

A causal model consists of 3 elements M =<V,G,P($v_i|pa_i$)>, where (i) V = {V_1, . . . V_n} is a set of variables, (ii) G is a directed acyclic graph (DAG) with nodes corresponding to the elements of V , and (iii) P($v_i|pa_i$), i =1, . . . , n, is the conditional probability of variable V_i given its parents in G.

The arrows in the graph G have a causal interpretation, which means that they are viewed as representing autonomous causal relations among the variables they connect. P ($v_i|p_{ai}$) represents a stochastic process by which the values of V_i are chosen in response to the values p_{ai}, which stays invariant under changes in processes governing other variables. The presence of a bi-directed arrow between two variables represents the presence of a confounding factor between the corresponding variables. The causal effect of a variable *X* on a set of variables S is noted as $P_x(s)$. $P_x(s)$ is identifiable from a graph G if the quantity

$P_x(s)$ can be computed uniquely from any positive probability of the observed variables - that is, if $P^{M1}{}_x (s) = P^{M2}{}_x (s)$ for every pair of models M1 and M2 with $P^{M1} (v) = P^{M2} (v) > 0$ and G (M1) = G (M2) = G.

Let a path composed entirely of bi-directed edges be called a bi-directed path. In a causal model the set of variables can be partitioned into disjoint groups by assigning two variables to the same group if and only if they are connected by a bi-directed path. Such a group is called a confounded component (c-component).

In multi-agent causal models, there is no central controller having access to all the observable variables, but instead there is a collection of agents each having access to non-disjoint subsets of V = {V1, . . . , Vn}. A multi-agent causal model consists of 4 elements $M_i = <V_{Mi},G_{Mi}, P_{Mi} (v_j |pa_j),K_i>$.

V_{Mi} is the subset of variables agent i has access to.

G_{Mi} is the causal diagram over variables V_{Mi} .

$P_{Mi} (v_j |pa_j)$ are the conditional probabilities over the V_{Mi} .

K_i stores the intersections with other agents j, $\{V_{Mi} \cap V_{Mj} \}$.

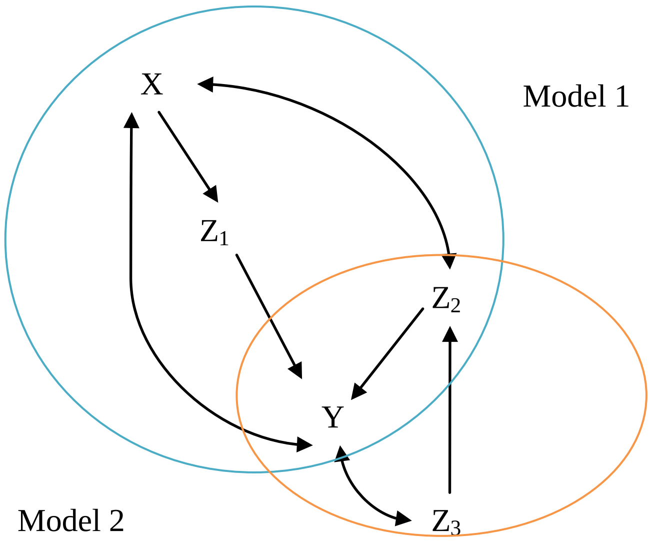


Figure A-2 Example of a multi-agent causal model with two agents.

In , an example of a multi-agent causal model is depicted. The two models are:

$M_1 = <V_{M1},G_{M1}, P_{M1} (v_j |pa_j),K_1>$

$M_2 = <V_{M2},G_{M2}, P_{M2} (v_j |pa_j),K_2>$

where, $V_{M1} = \{X, Y,Z_1, Z_2\}$, $V_{M2} = \{Y,Z_2, Z_3\}$, and $K_1 = K_2 = \{Y,Z_2\}$

The research work on MACM are mainly carried out by (Maes and Leray 2006), beginning from around 2003. It consists of a collection of agents each having access to a non-disjoint subset of the variables constituting the domain. In MACM, each agent models its own subdomain with a semi-Markovian causal model, and each of the individual agent models in an MACM has private variables that it keeps confidential and public variables that it shares in an intersection with other agents. MACM supposes agents are cooperative, which means they will not deliberately provide each other with false information.

MACM allows for multi-agent, privacy-preserving quantitative causal inference in models with hidden variables. Algorithms for performing probabilistic inference in MACM without hidden variables are well studied in theory, though applicable algorithms are still under research. Techniques to learn part of the structure of MACM from data have been developed while the completely directed structure of an MACM has not been investigated.

Hierarchical Bayesian Model

Hierarchical Bayesian Model (HBM) is a powerful and principled solution to the situation where agents need to learn from one another by exchanging learned knowledge (MacNab 2003) (Tresp and Yu 2005) .

Assume that there are M data sets $\{D_j\}_{j=1}^{M}$ for related but not identical settings and M different models with parameters $\{\theta_j\}_{j=1}^{M}$ are trained on those data sets. Each data set is sufficiently large such that $P(\theta_j | D_j, h_{prior})$ is sharply peaked at the maximum likelihood (ML) estimate θ_j^{ML} . Let $\{\theta_k^{ML}\}_{k=1}^{M}$ denote the maximum likelihood estimates for the M models. If a new model concerns a related problem, then it makes sense to select new hyperparameters h_{hb} such that $P(\theta | h_{hb})$ approximates the empirical distribution given by the maximum likelihood parameter estimates instead of using the original uninformed prior $P(\theta | h_{prior})$ In this way the new model can inherit knowledge acquired not only from its own data set but also from the other models. Figure A-3 illustrates a Hierarchical Bayesian Model.

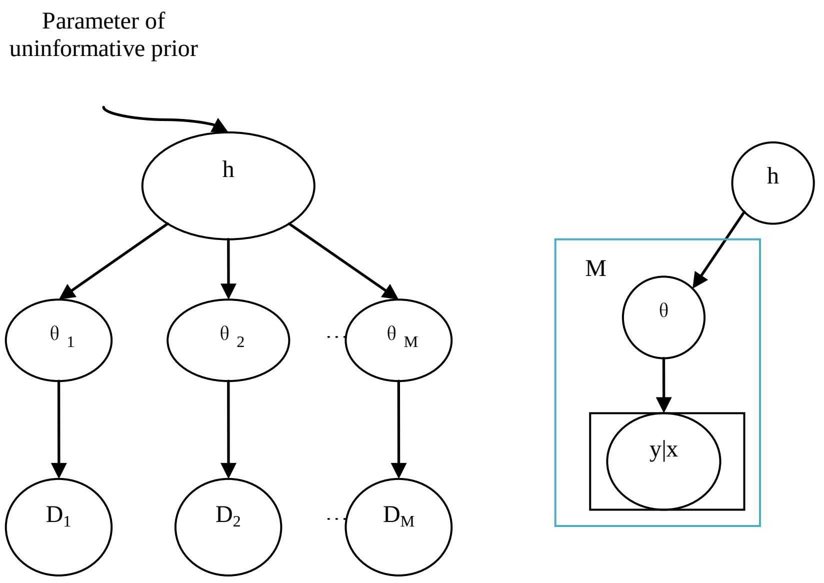


Figure A-3:The left is Hierarchical Bayesian Model, and the right is a plate model for HB. The large plate indicates that M samples from $P(\theta/h)$ are generated; the smaller plate indicates that, repeatedly, data points are generated for each θ

HBM is a powerful and principled solution to MAS where agents need to learn from one another by exchanging learned knowledge. Parametric hierarchical Bayesian modeling and non-parametric hierarchical Bayesian modeling are powerful tools to collaborative filtering in MAS.

Praxeic Networks

Praxeic network (PN) is a 2N-dimensional Bayesian network which is used to represent MAS. The praxeic network for an N-agent system consists of 2N nodes, with each participant having two nodes associated with it: one for its selectability persona and one for its rejectability persona (Stirling and Frost 2005).

The variables associated with these nodes are the options available to the decision maker and the edges represent the influence that one persona has on another persona. These linkages consist of conditional selectability or conditional rejectability functions.

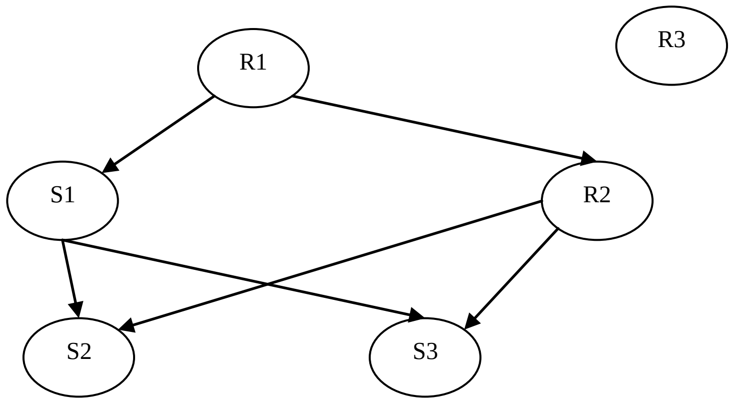


Figure A-4 Praxeic network for a three-agent system

Consider the graph displayed in Figure A-4, which corresponds to a three-agent system whose interdependence function is given by

$$P_{S_1S_2S_3R_1R_2R_3} = P_{S_1S_2|S_3R_2} \cdot P_{S_3R_2|R_1} \cdot P_{R_1} \cdot P_{R_3} \qquad \text{A-1}$$

Persona does not influence and is not influenced by any other personas because of the structure of the interdependence function.

Praxeic network is a directed acyclic graph using graph theory to express a MAS. Praxeic network employs social utilities which satisficing game theory is based on. Satisficing game theory provides a mechanism by which cooperative social systems may be synthesized according to a systematic concept of rational behavior involving social utilities that count for the interests of others as well as of the self. Compared with conventional methods of multiple-agent decision making, it replaces individual rationality with satisficing rationality, replace single utility functions with dual utility functions that separate the desirable and undesirable attributes of the options, replaces unconditional utilities with conditional utilities.

Praxeic Networks outperforms conventional methods of multi-agent decision making. It permits explicit modeling of situational altruistic behavior, provides a natural framework for negotiation, can be solved using standard Bayesian network techniques.

Multi-agent Influence Diagram

The Multi-agent Influence Diagrams (MAID) extend the influence diagram framework to the multi-agent case to represent non-cooperative games (Koller and Milch 2003).

In a MAID, there is a set A of agents. The world in which the agents act is represented by the set X of chance variables and a set D_a of decision variables for each agent a $\in$A. Chance variables correspond to decisions of nature and they are represented in the diagram as ovals. The decision variables for agent *a* are variables whose values a gets to choose, and are represented as rectangles in the diagram. D is used to denote $\bigcup_{a \in A} D_a$. Edges into a decision variable are drawn as dotted lines. The agents' utility functions are specified using utility variables: each agent a $\in$A has a set U_a of utility variables, represented as diamonds in the diagram. The domain of a utility variable is always a finite set of real numbers. U is used to denote $\bigcup_{a \in A} U_a$ and V is used to denote $X \cup D \cup U$.

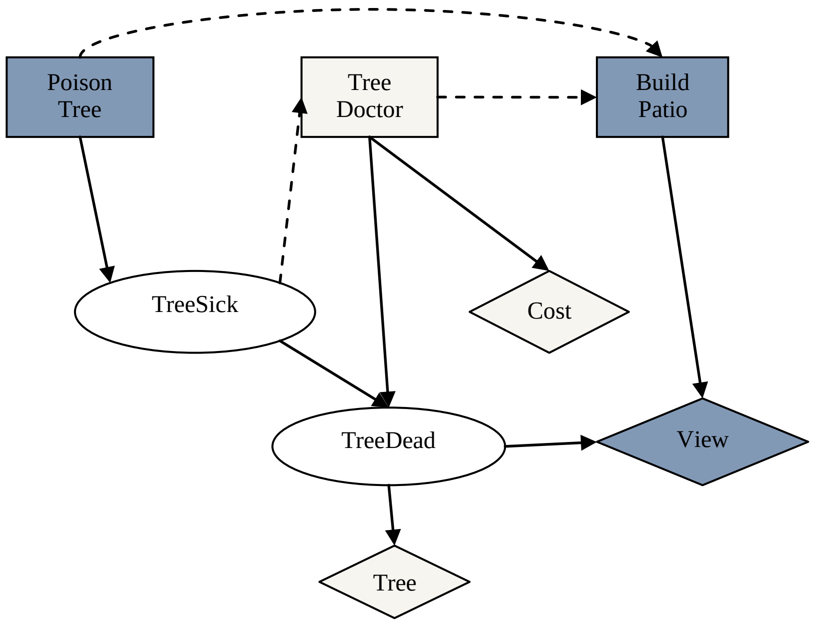


Figure A-5 A MAID for the Tree Killer example; Alice's decision and utility variables are in dark gray and Bob's in light gray.

Figure A-5 shows a tree killer example of MAID. Alice and Bob are neighbors and they share the same garden. Alice would like to build a patio and she thinks the tree in the garden blocks the view and decision to poison the tree. Poison the tree will lead it to sick. Bob cares about the tree, and he need to decision whether to bring a tree doctor which will cost some money or leave the tree there and let it die. If the tree dead Bob will grow a new tree. On the other hand, the dead tree and patio will give Alice a better view of the garden. Alice and Bob's decision and utility variable combine to build a MAID model.

Like a BN, a MAID defines a directed acyclic graph with its variables as the nodes, where each variable X is associated with a set of parents $Pa(X) \subset X \cup D$. The value of a utility variable is required to be a deterministic function of the values of its parents. The total utility that an agent derives from an instantiation of V is the sum of the values of U_a in this instantiation. Thus, by breaking an agent's utility function into several variables, it is only needed to define an additive decomposition of the agent's utility function((Ronald A. Howard 1989) (Keeney and Raiffa 1976)).

MAID is considered as a milestone for the research work on Multi-agent decision making. It focuses on the representation of games and tries to find Nash Equilibrium in the games with some strategies. A MAID extends the formalisms of Bayesian networks and influence diagrams to represent game problems involving multiple agents. To take advantage of independence structures in a MAID, a qualitative notion of strategic relevance is defined to find a global equilibrium through a series of relatively simple local computations. However, since the goal of MAID is to represent and solve games, it is predetermined that it lacks the ability to handle a general Multi-agent decision problem, not to mention dealing with a more complex and large problem domain. Its solution strategies specify only on game problem that is a subset of decision problem.

Dynamic Multi-agent Influence Diagrams

Dynamic multi-agent influence diagram (DMAID) is an extension of MAID, which helps to represent multistage games in a dynamic way (Zhang, Liu et al. 2002). A multistage game is one which players choose their actions sequentially. The framework of DMAID consists of pieces of multi-agent influence diagram fragments (MAIDF), which is a slightly form-changing of MAID. Figure A-6 shows the DMAID model of an indirectly financing game: deposit and fetch money.

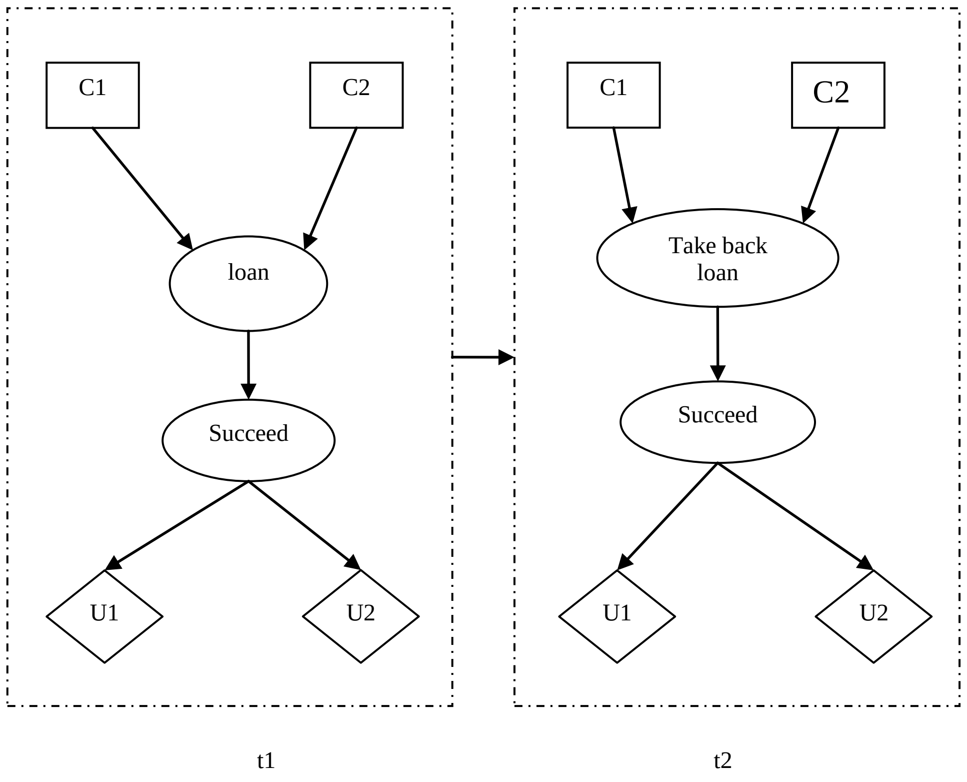


Figure A-6: A DMAID model of indirectly financing game. G(t_1) denotes deposit game, G(t_2) denotes fetch money game.

<u>Definition A-5</u>: A MAIDF is a tuple <A,X,D,U,P>, where A: a set of all agents; X: chance variables set; D=$\bigcup_{a\in A} D_a$ is decision variables set; U=$\bigcup_{a\in A} U_a$ is utility variables set; P: a joint probability distribution. For each agent a:

$$a \in A, D \in D_A, D = \bigcup_{a\in A} D_a, x \in X, Pa(x) \subset X \cup D, U \subset U_A, U = \bigcup_{a\in A} U_a$$

<u>Definition A-6</u>: M (T) is a DMAID, if $M(t_k) \in M(T)$ is a MAIDF at a certain time point,

$t_k \in T$ (k=0,1,2,…,n); T is a set of discrete time points. The MAIDF of a DMAID at time t_k depends only on its immediate past: its MAIDF at t_{k-1}.

MAID is a graphical representation for non-cooperative games. Unlike the previous two methods that are more relevant under MAS, MAID is focused on game theory. MAID can represent complex games in a natural way, whose size is no larger than that of the extensive form, but which can be exponentially more compact in (Koller and Milch 2003). Its relevance graph data structure provides a natural decomposition of a complex game into interacting fragments and provides an algorithm that finds equilibrium for these sub-games in a way that is guaranteed to produce a global equilibrium for the entire game. The divide-and-conquer algorithm generalizes the standard backward induction algorithm for game trees and it can be exponentially more efficient than an application of standard game-theoretic solution algorithm in (Koller and Milch 2003).

MAID right now only handles one-stage games, which means one agent plays while others do not, then the game stops. It is thus not suitable for multi-stage context. However it is possible to extend MAID to dynamic setting and DMAID may be a good choice. DMAID It extends MAID in a dynamic way such that it is more suitable to analyze multi-stage games, represent them in semantics of time and compute its Nash equilibrium efficiently. It simplifies the dependency relations between variables in MAID of multi-agent context. Both DMAID and MAID can be converted into game tree. However, there is no algorithm to compute global Nash equilibrium in dynamic games yet.

Analysis and Comparisons

In the previous sections, six important probabilistic models in MAS has been introduced. They are multiply-sectioned Bayesian Networks (MSBN), Multi-agent Causal Models (MACM), Praxeic Networks (PN) and Hierarchical Bayesian Models (HBM), Multi-agent Influence Diagrams (MAID), Dynamic Multi-agent Influence Diagrams (DMAID). When there are many tools available, it is important to know when it is suitable to use which tool, what capabilities they have, what are their strengths and weaknesses, why these tools come to life, whether they can be made to an even better one, and so on. In this section, these issues will be discussed in detail and some comparisons and analysis will be made.

The models considered in this chapter fall into two categories. One is focused on probabilistic learning and reasoning, like MSBM, MACM, HBM and PN; the other is on finding global Nash equilibrium, like MAID and DMAID. It is clear from the analysis that each method has its own application environment, its strengths and its limitations. So when applying these models into problem solving, it is important to carefully examine the problem settings and choose the best suitable modeling method.

Table A-1 Comparisons of surveyed models

Models	Extend from	Basic settings	Advantages	Limitations
MSBN	BN	Sparse cooperative MAS	Exact probabilistic measure, efficient probabilistic reasoning, communicate by belief over small sets of shared variables, DAG for domain structures	More suitable for sparse environment, Strict requirements
MACM	CBN	Cooperative MAS	privacy-preserving quantitative causal inference, handle hidden variables	Algorithms for probabilistic inference and complete structure learning not available yet
HBM	BN	MAL	Learning ability	The learning ability greatly affected by the correctness of assumed model.
PN	Nil	Satisficing game theory , multi-agent decision making	Model altruistic behavior, framework for negotiation	Not appropriate when more considerations on individual rationality
MAID	ID	Static games	Computational efficiency and guarantee global equilibrium	Could not handle dynamic environment
DMAID	MAID	Multistage games	Model time and simplify dependence relations in MAID	no algorithm to compute global Nash equilibrium yet