MECHANISM DESIGN FOR DISTRIBUTED TASK AND RESOURCE ALLOCATION AMONG SELF-INTERESTED AGENTS IN VIRTUAL ORGANIZATIONS

A Dissertation

by

LINLI HE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

May 2006

Major Subject: Computer Science

MECHANISM DESIGN FOR DISTRIBUTED TASK AND RESOURCE ALLOCATION AMONG SELF-INTERESTED AGENTS IN VIRTUAL ORGANIZATIONS

A Dissertation

by

LINLI HE

Submitted to the Office of Graduate Studies of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Approved by:

Chair of Committee,	Thomas R. Ioerger
Committee Members,	Riccardo Bettati
	Jianer Chen
	Goong Chen
Head of Department,	Valerie E. Taylor

May 2006

Major Subject: Computer Science

ABSTRACT

Mechanism Design for Distributed Task and Resource Allocation Among Self-Interested Agents in Virtual Organizations. (May 2006)

Linli He, M.S., Southwest Petroleum Institute

Chair of Advisory Committee: Dr. Thomas R. Ioerger

The aggregate power of all resources on the Internet is enormous. The Internet can be viewed as a massive virtual organization that holds tremendous amounts of information and resources with different ownerships. However, little is known about how to run this organization efficiently.

This dissertation studies the problems of distributed task and resource allocation among self-interested agents in virtual organizations. The developed solutions are not allocation mechanisms that can be imposed by a centralized designer, but decentralized interaction mechanisms that provide incentives to self-interested agents to behave cooperatively. These mechanisms also take computational tractability into consideration due to the inherent complexity of distributed task and resource allocation problems.

Targeted allocation mechanisms can achieve global task allocation efficiency in a virtual organization and establish stable resource-sharing communities based on agents' own decisions about whether or not to behave cooperatively. This high level goal requires solving the following problems: synthetic task allocation, decentralized coalition formation and automated multiparty negotiation.

For synthetic task allocation, in which each task needs to be accomplished by a virtual team composed of self-interested agents from different real organizations, my approach is to formalize the synthetic task allocation problem as an algorithmic mechanism design optimization problem. I have developed two approximation mechanisms that I prove are incentive compatible for a synthetic task allocation problem.

This dissertation also develops a decentralized coalition formation mechanism, which is based on explicit negotiation among self-interested agents. Each agent makes its own decisions about whether or not to join a candidate coalition. The resulting coalitions are stable in the core in terms of coalition rationality. I have applied this mechanism to form resource sharing coalitions in computational grids and buyer coalitions in electronic markets.

The developed negotiation mechanism in the decentralized coalition formation mechanism realizes automated multilateral negotiation among self-interested agents who have symmetric authority (*i.e.*, no mediator exists and agents are peers).

In combination, the decentralized allocation mechanisms presented in this dissertation lay a foundation for realizing automated resource management in open and scalable virtual organizations.

ACKNOWLEDGMENTS

I would like to thank my advisor Dr. Thomas Ioerger for directing my Ph.D. study at Texas A&M University. It was impossible to complete this dissertation without his inspiration, encouragement and support. His scientific approach toward research taught me to be a serious scholar. I would like to express my gratitude to Dr. Thomas Ioerger and Dr. Richard Volz for generously providing financial support for my study. Without that, a study here would have been impossible. I would like to thank Dr. Janier Chen for being my committee member. I also benefited a lot from many courses taught by him. It has always been a privilege and joy to discuss with Dr. Chen about different algorithms used in my study. Dr. Riccardo Bettati and Dr. Goong Chen are also appreciated for being committee members. They found time in their busy schedules to read this dissertation and provide valuable suggestions and comments.

Appreciation is also extended to the staff in the Department of Computer Science. They have provided all kinds of help during my study. I am also thankful to the friendship developed with many of the faculty and staff members. Their support and encouragement will always be in my heart.

Finally, I appreciate my parents and my husband for their constant and unconditional love and support, without which, it would have taken me much longer or even be impossible to pursue and finish this doctoral study.

TABLE OF CONTENTS

Page

CHAPTER Page

CHAPTER Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

CHAPTER I

INTRODUCTION AND MOTIVATION

The main theme of this dissertation is to design decentralized mechanisms for distributed task and resource allocation among self-interested agents who are trying to maximize their own benefit without concern of the global good in a multiagent system. The developed mechanisms are targeted to establish virtual organizations through interactions among self-interested agents. Within such a virtual organization, self-interested agents must have incentives to behave cooperatively if global efficiency of distributed task and resource allocation is to be achieved without a centralized controller.

MOTIVATION

Distributed task and resource allocation have been fundamental research topics in distributed computer science (Zweben and Fox 1994; Clearwater 1996; Kraus and Plotkin 2000). Multiple agents need to work together due to an inherent distribution of resources such as knowledge, capability, information, and expertise among the agents. Agents are often unable to accomplish their own tasks alone, or they might be able to accomplish tasks better when working with others (Weiss 1999). Traditionally, the designers of distributed task and resource allocation algorithms and protocols have made an implicit assumption that the participating agents will act as instructed – except, perhaps, for faulty or malicious

---

This dissertation follows the style of *Applied Artificial Intelligence*.

ones (Nisan and Ronen 1999; Ronen 2000). The main concerns of designing distributed allocation algorithms are algorithmic complexity and communication load (network complexity).

This assumption can no longer be taken for granted with the emergence of the Internet as the platform of computation. The aggregate power of all resources on the Internet is huge. Ideally, this aggregate power would be optimally and dynamically allocated online to appropriate users (Nisan and Ronen 1999; Ronen 2000). The resources on the Internet have many new features that do not exist in traditional computational platforms. Notably, the resources are heterogeneous with different ownerships, dynamic availability, and geographical dispersion. Thus, the Internet has the characteristics of an economy as well as those of a computational system (Feigenbaum et al. 2001; Feigenbaum and Shenker 2002). The majority of the participating agents are neither obedient nor adversarial, but rational or self-interested in the sense that each agent has its own goals and preferences. The decision for a self-interested agent to be cooperative is based upon whether the cooperation can bring greater benefit than working alone. They will not necessarily follow prescribed algorithms but will respond to punishments and incentives.

Existing distributed task and resource allocation mechanisms typically have been constructed from the top down (Rosenschein and Zlotkin, 1994), namely, by imposing fixed allocation rules to handle all possible situations. This design philosophy does not work well in a computational platform like the Internet because there does not exist an omniscient designer who can develop a task and resource allocation mechanism that satisfies the preferences of all self-interested resource users and suppliers and maximizes the global efficiency. Instead, task and resource allocation mechanisms need to be

established from the bottom up (Clearwater 1996, Tesfatsion 2002), meaning that every participating agent makes individual decisions based on local knowledge and preferences (most likely with incomplete information about other agents) without considering the global efficiency. The global efficiency is generated from the bottom up through interactions among self-interested agents. A decentralized mechanism for a distributed allocation problem is indeed an interaction mechanism for self-interested agents who normally have incomplete information about others (Rosenschein and Zlotkin 1994). The objective of designing such an interaction mechanism is to establish a stable system where resources are shared through interactions among self-interested agents. Computational grids and Peer-to-Peer systems are typical examples of such a system.

Interactions among self-interested agents have been studied intensively in game theory (Kahan and Rapoport 1984; Fudenberg and Tirole 1991; Osborne and Rubinstein 1994). The focus of game theory is to analyze whether a given solution is stable in terms of the willingness of self-interested agents to behave cooperatively. Game theorists have developed a number of solution concepts such as dominant strategy, core, bargain set, Sharply value, Nash equilibrium, *etc*. to evaluate this stability. The preferred solution concept is dominant strategy, which means that each agent can maximize its own utility no matter what kind of strategies that other agents are using (Fudenberg and Tirole 1991; Osborne and Rubinstein 1994).

Mechanism design is a sub-field of game theory that has focused on problems where the goal is to satisfactorily aggregate privately known preferences of several agents towards a "social choice" (Jackson 2001). In other words, it studies how "rules of a game" constrains the public behavior of the participating agents by providing proper incentives or

punishments. The desired mechanisms are incentive-compatible in the sense that the designed mechanisms can provide incentives to self-interested agents to behave cooperatively in a multiagent system.

However, game theory does not provide actual methodologies to design such a mechanism. For a decade, distributed artificial intelligence has been studying how to develop automated interaction mechanisms for self-interested agents in multi-agent systems (Rosenschein and Zlotkin 1994; Sandholm 1996; Parkes 2000). Game theory plays an irreplaceable role in providing a mathematic foundation to evaluate developed mechanisms. Researchers soon recognized that game theory downplays the complexity aspect of these mechanisms. Self-interested agents in game theory are assumed to be perfectly rational in the sense that the computational cost of rational reasoning is not taken into account (Fudenberg and Tirole 1991; Osborne and Rubinstein 1994). This assumption is invalid in the real world. There are many optimization allocation problems that are computationally intractable (Sandholm 1996). Yet, it has been proved that some approximation algorithms might cause a mechanism to be no longer incentive-compatible (Parkes 2000). Therefore, to solve distributed task and resource allocation problems among self-interested agents, incentive compatibility and computational tractability need to be jointly addressed.

Researchers in distributed artificial intelligence have put a great deal of attention into developing theoretic models that consider incentive compatibility and computational tractability jointly. Established by Nisan and Ronen (1999), algorithmic mechanism design (AMD) is a formal model of centralized computation that combines incentive compatibility (the "mechanism design" part) with computational tractability (the "algorithmic" part). Feigenbaum et al. (2001) extended this model to distributed algorithmic mechanism design

(DAMD), in which the same goals of incentive compatibility and computational tractability are considered. In addition, the agents, the relevant information, and the computational model are all inherently distributed. Network complexity also needs to be taken into consideration. Sandholm and Lesser (1995, 1997) developed a domain-independent model for coalition formation in a situation where the rationality of self-interested agents is bounded by computational complexity. These theoretic models bring game theory and complexity theory together and provide tangible criteria for evaluating whether a mechanism is both incentive-compatible and computationally tractable.

The focus of this dissertation is not to invent a new incentive-compatible and computationally tractable theoretic model, but to use the existing models (Rosenschein and Zlotkin 1994; Sandholm and Lesser 1997; Nisan and Ronen 1999; Papadimitriou 2001; Azoulay-Schwartz and Kraus 2004; Porter 2004) to analyze the mechanisms that I develop for solving distributed task and resource allocation problems that have a variety of applications in virtual organizations, grid and peer-to-peer computing and electronic commerce. This dissertation also focuses on designing decentralized mechanisms where a system-wide solution for a distributed allocation problem consists of agreements made by multiple participating agents. How to achieve those agreements in a distributed and dynamic environment is one of the major focuses of this dissertation.

Negotiation is essential to achieving an agreement among self-interested agents (Kraus 2001). Social scientists have studied negotiation from many different perspectives (Raiffa 1982). Game theorists are interested in how self-interested agents divide the payoff of cooperation through negotiation (Raiffa 1982; Fudenberg and Tirole 1991; Osborne and Rubinstein 1994). Bargain theory (Harsanyi 1967, 1977) is a typical example. Researchers

in distributed artificial intelligence have put a great deal of effort into implementing automated negotiation among self-interested agents. Most previous studies focus on bilateral negotiation between two agents (Rosenschein and Zlotkin 1994; Jennings et al 2001; Larson and Sandholm 2001). As for how to achieve an agreement among multiple agents, little work has been done. In this dissertation, implementing automated multiparty negotiation among self-interested autonomous agents is one of the most important methods for solving distributed allocation problems.

EXAMPLE DISTRIBUTED TASK AND RESOURCE ALLOCATION SCENARIOS

To address the motivation of this research from a practical perspective, the following section describes some example scenarios in real applications where the distributed task and resource allocation mechanisms developed in this dissertation are needed. The formal distributed allocation problems that this dissertation will address are abstracted from these real applications.

Outsourcing/Virtual Teamwork

As information and communication technologies overcome the constraints of time and distance, it becomes a necessary to create virtual organizations that consists of a temporary network of independent companies linked by IT infrastructure to share skills, costs, and access to one another's markets. One of the most important advantages of a virtual organization is executing synthetic tasks by forming temporary teams composed of

experts from different fields and independent organizations (which are most likely geographically dispersed) through the Internet.

One typical example is to construct temporary offshore software development teams to accomplish multiple projects (Hatch 2005). The problem is how to construct the most efficient offshore teams from various outsourcing service vendors to finish these projects as soon as possible. One of the major challenges of this problem is that the actual capabilities of software engineers from different outsourcing service vendors are private information that cannot be accessed directly from outside. The project manager has to construct virtual software development teams and allocate tasks based on the reported capabilities.

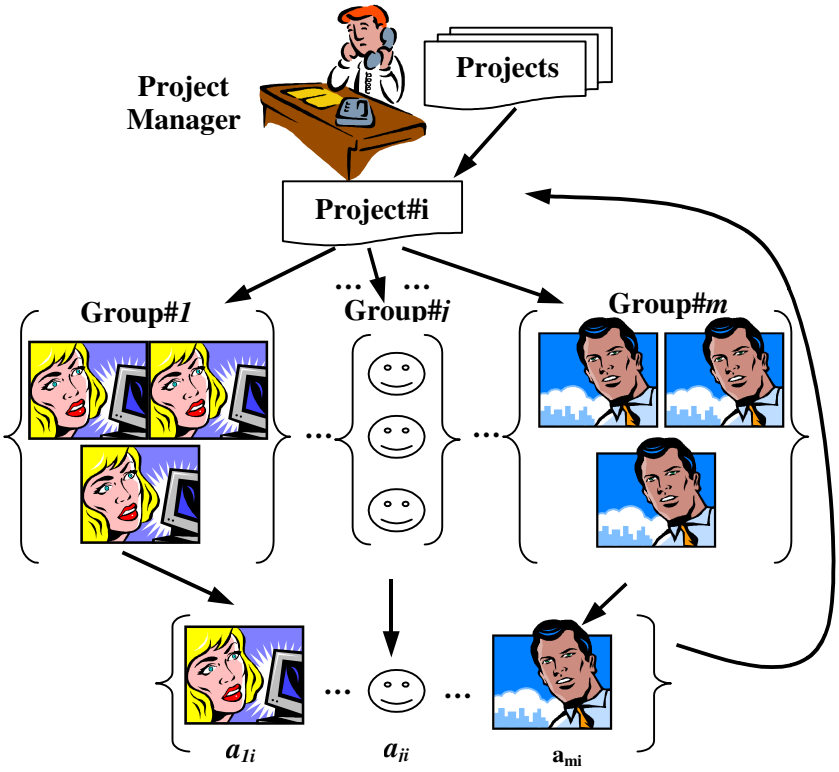

FIGURE 1.1. Synthetic Task Allocation

Obviously, each vendor is self-interested in the sense that its goal is to maximize its own profit. Therefore, an incentive-compatible allocation mechanism is required to induce outsourcing service vendors to be willing to report the true capability of their software engineers.

Suppose there is a project manager who has a number of projects at hand, and each project cannot be accomplished by only one of software engineers due to multiple skills required (e.g., one project might need a GUI engineer and a database designer working together). People with different expertise might belong to different groups that are self-interested, and experts in one group might have different capabilities in terms of how efficient they are. How does the project manager determine the true capabilities of all the experts in different groups and compose a temporarily appropriate team for each project? Intuitively, a senior software engineer might be slowed down because of working with a junior graphic designer. Therefore, the problem becomes how does the project manager build efficient teams for synthetic tasks without knowing the true capabilities of agents in different groups so that he can accomplish all projects in the minimal amount of time?

Figure 1.1 illustrates the synthetic task allocation problem considered in this dissertation. A set of projects needs to be done. Each of these projects requires cooperation among agents from different groups with different expertise. Within each group, different agents require different amounts of time to finish identical tasks. I use the capability model in (He and Ioerger 2003) to represent the capability differentiation among agents in each group. Without loss of generality, I assume that each project needs exactly one agent from each group. I call these kinds of projects synthetic tasks. The project manager needs to build a temporary team for each project. The amount of time required to finish a project

depends on the capability of the most inefficient member of that team. The objective of the project manager is to minimize the total amount of time required to accomplish all projects.

As shown in Figure 1.1, the number of possible teams that need to be considered is n^m in the worst case, where m is the number of groups and n is the number of members in each group. In fact, even if there is only one group and agents are cooperative, the task allocation problem is NP-hard (Coffman et al. 1987). Complexity cannot be ignored in solving this synthetic task allocation problem. Therefore, an efficient synthetic task allocation mechanism is needed that is both incentive-compatible and computationally tractable.

Indeed, the synthetic task allocation problem described above is essential to conducting efficient task allocation in a virtual organization. The Internet is becoming a large-scale virtual organization that holds a tremendous amount of information and resources with different owners. Little is known about how to run this organization efficiently. There does not exist a well-understood organizational structure that can model this system. In this system, allocating tasks to appropriate computational resources is analogous to allocating synthetic tasks to offshore teams. Computational tasks might need different resources from different resource owners on the Internet. Also, computational resources are most likely heterogeneous. Resources of the same type may have different capabilities (e.g. CPUs with different speeds). Incentive-compatible allocation mechanisms are required to induce resource owners to reveal the true capabilities of their computational resources.

Resource Sharing in Computational Grids and Peer-to-Peer Systems

Many scientific and commercial computational applications require increasingly powerful computational resources to satisfy computational performance requirements. Large amounts of computational resources connected via the Internet are idle most of the time. Utilizing these idle resources that are owned by different organizations or individuals and are geographically dispersed to satisfy intensive computational power requirements from numerous scientific and commercial applications has become a major goal of distributed computer science. The emergent areas of grid computing and peer-to-peer computing are aimed at overcoming this challenge (Foster and Kesselman 1998; Buyya 2002; Buyya et al. 2002; Milojicic et al. 2002; Berman et al. 2003).

The recent developments in grid and peer-to-peer computing have positioned them as promising next-generation computing platforms. They enable the creation of virtual enterprises for sharing computational resources distributed across the world. Both of these two research areas in distributed computing are aimed at addressing the problem of organizing large-scale computational societies for resource sharing within virtual communities where resources may not be controlled by any single organization.

The participating agents are inherently self-interested in computational grids and peer-to-peer systems. Providing incentives for those self-interested agents to participate in a computational grid or peer-to-peer system is a key to making these computational systems feasible. A basic assumption is that agents in a computational grid or a computational peer-to-peer system have peak workloads at different times so that they can utilize others' resources at idle times. That necessitates distributed load balancing (Lan et al. 2002; Shan

et al. 2003) among self-interested agents. Agents share their resources with their partners. The question is how to establish such partnerships?

In current computational grids, community standards are represented via explicit policies (Foster and Iamnitchi 2003; Foster et al. 2004). Resources owned by various administrative organizations are shared under locally defined policies that specify what is shared, who is allowed to share, and under what conditions. Normally, a small number of sites are connected in collaborations engaged in complex scientific applications. As system scale increases, grid developers are now facing problems relating to autonomic configuration and management. How to automatically adjust system level policy to be adaptive to system updates (both hardware and software) and user requirement changes remains a major challenge in grid computing. As Foster and Iamnitchi (2003) pointed out: "Over all, scalable autonomic management remains a goal, not an accomplishment, for Grid computing."

The explicit grid policy implemented in the existing computational grids can be viewed as agreements achieved through negotiation among participating organizations. This dissertation proposes a resource sharing mechanism that is based on a distributed coalition formation mechanism through automated multiparty negotiation among self-interested agents. This approach aims to establish resource management policies in computational grids autonomously and dynamically through automated negotiations among participating agents.

Combinatorial Trade in Electronic Markets

In electronic markets, the distance between producers, wholesalers, distributors, retailers, and consumers has practically disappeared (Ye and Tu 2003). There are many more choices faced by all parties involved in electronic combinatorial trade than in a traditional trade system. The relationship between suppliers and customers is under-going revolutionary change.

Buyers vary a great deal in the quantity of goods they want to purchase, in customer service requirements, in income, in time constraints and in many other dimensions. Different purchasing goals can cause widely varying production and transaction costs. Suppliers have their own "buyer selection" strategies to achieve better profitability. Quickly differentiating the supplier's marketing strategy based on the difference of purchasing goals among various buyers plays a key role in improving the sellers' competitive capabilities in electronic markets (He and Ioerger 2004a, 2005a and 2005b).

In traditional markets, it is impractical for buyers to build such purchasing strategies because of the expense of access to product information. However, in the age of electronic commerce, buyers can access product information easily and inexpensively. Small buyers with little or zero bargaining power in traditional markets now can build collaborative purchasing strategies to minimize their cost in electronic markets. A well-known example of building such a purchasing strategy for buyers is to form buyer coalitions (*e.g.*, Buyer Club) to enlarge the total quantity of goods purchased in each transaction (Lerman an Shehory 2000; Yamamoto and Sycara 2001; Li and Sycara 2002). Buyers can obtain lower prices without buying more than their real need. If the buyers are heterogeneous in the

sense that they need to buy different goods in a combinatorial market, the mechanism is the so-called combinatorial coalition formation.

Buyer coalition formation is a distributed combinatorial optimization problem, which is a highly non-trivial problem that needs to be solved by considering incentive compatibility and computational tractability jointly. In contrast to the previous work, this dissertation focuses on solving this problem in a pure decentralized manner, which is more realistic than solving the problem through a centralized mediator. I apply my distributed coalition formation mechanism that is based on automated multiparty negotiation. This approach allows agents to make their own decisions, and significantly reduces the computational complexity by distributing the computational costs among all participating agents.

DISTRIBUTED METHODS FOR TASK AND RESOURCE ALLOCATION

I have developed the following distributed methods for task allocation and resource sharing in virtual organizations.

Incentive-Compatible Mechanisms for Synthetic Task Allocation

In this dissertation, designing incentive-compatible mechanisms for synthetic task allocation (He and Ioerger 2005c) among self-interested agents is one of the major contributions. I develop both incentive-compatible and computationally intractable mechanisms for synthetic task allocation problems, in which each task needs to be

accomplished by a virtual team composed of self-interested agents from different real organizations. I formalize the synthetic task allocation problem as an algorithmic mechanism design optimization problem. I have developed two incentive-compatible mechanisms for the synthetic task allocation problem. It shows that designing both incentive-compatible and computationally tractable mechanisms is feasible for synthetic task allocation problems in virtual organizations.

Forming Resource Sharing Coalition through Multiparty Negotiation

Another major contribution of this dissertation is that I develop a decentralized coalition formation mechanism, which is based on explicit negotiation among self-interested agents (He and Ioerger 2004a, 2005a and 2005b). The developed coalition formation mechanism achieves decentralization through explicit negotiation among self-interested agents. Each agent makes its own decisions on whether or not to join a possible coalition. The resulting coalitions are stable in the core in terms of coalition rationality. Compared with the centralized approaches, this mechanism significantly reduces the complexity of coalition formation processes. The communication load caused by negotiation is very low due to a properly designed multiparty negotiation protocol.

The multiparty negotiation mechanism itself is unique because existing negotiation mechanisms in distributed artificial intelligence are bilateral. The developed negotiation mechanism extends the Monotonic Concession Protocol (Rosenschein and Zlotkin 1994) for negotiation between two agents and realizes automated multilateral negotiation among

self-interested agents who have symmetric authority (i.e., no mediator exists and agents are peers) in a multiagent system.

I have applied this mechanism to form resource sharing coalitions in computational grids and buyer coalitions in electronic markets. The simulation results show that the coalition formation process is successful in the sense that automated multiparty negotiation processes can lead agents to find appropriate coalitions and a coalition formation process can end properly. The communication load is practical in the sense that the number of messages received by each agent is much less than the worst case. For buyer coalition formation, my distributed coalition formation mechanism can result in nearly optimal results.

OVERVIEW OF CONTRIBUTIONS

This dissertation develops decentralized mechanisms for distributed task and resource allocation among self-interested agents in virtual organizations. These mechanisms are both incentive-compatible and computationally tractable. Based on these mechanisms, stable virtual organizations (or communities) can be established through interactions among self-interested agents without a centralized controller. These mechanisms are also easy to implement, so that they can be built in autonomous agent systems. This has the potential to dramatically change the current reality in which most resource management policies in virtual organizations (e.g. computational grids) are manually enforced.

CHAPTER II

RELATED WORK

Many researchers in distributed artificial intelligence (Rosenschein and Zlotkin 1994; Sandholm 1996, 2003; Parkes 2000) have reviewed related game-theoretic aspects in terms of incentive compatible issues in multiagent systems. This chapter mainly focuses on the related work that has been done in distributed artificial intelligence because I am interested in applying existing theoretic models to formalize distributed task and resource allocation problems for self-interested agents. The goal of this dissertation is to design mechanisms for distributed task and resource allocation among self-interested agents that can result in stable resource-sharing communities. This chapter reviews existing theoretic models that jointly address incentive compatibility and computational tractability and the methodologies for distributed task and resource allocation problems in distributed artificial intelligence.

ALGORITHMIC MECHANISM DESIGN

Established by Nisan and Ronen (1999), algorithmic mechanism design (AMD) is a formal model that considers both incentive compatibility (the "mechanism design" part) and computational tractability (the "algorithmic" part).

Mechanism Design

A mechanism design problem has two components: the algorithmic output specification and descriptions of what kind of benefits the participating agents can obtain. These components are given as utility functions over the set of possible outputs.

Definition 2.1: Mechanism Design Problem

A mechanism design problem is given by an output specification and by a set of agents' utilities. There is a collection of agents $A = \{a_1, \ldots, a_n\}$. Each agent a_i has some private information termed as its type $t_i \in T_i$. The output specification maps to a type vector $t = (t_1, \ldots, t_n)$. Each agent a_i's preferences are given by a real valued function: $v_i(o, t_i)$ in terms of some common currency. If the mechanism's output is o and the mechanism hands this agent p_i units of this currency, then its utility will be $u_i = p_i + v_i(o, t_i)$ (termed quasi-linear utility). The agent aims to optimize this utility.

Definition 2.2: Mechanism Design Optimization Problem

This is a mechanism design problem where the output specification is given by a positive real valued objective function $g(o, t)$ and a set of feasible output F. The objective is to find an output $o \in F$ that optimizes $g(o, t)$.

A mechanism solves a given problem by assuring that the required output occurs even as agents choose their strategies so as to maximize their own utilities. The formal definition of a mechanism is given as the following:

Definition 2.3: A Mechanism

There is a collection of agents $A = \{a_1, \ldots, a_n\}$. The mechanism defines a family of strategies S_i for each agent a_i. The agent can choose any $s_i \in S_i$. A mechanism $m = (o, p)$ is

composed of two elements: An output function $o = o(s_1 \ldots s_n)$ and an n-tuple of payments $p_1(s_1 \ldots s_n), \ldots, p_n(s_1 \ldots s_n)$. A mechanism is poly-time computable if the output and payment functions are computable in polynomial time.

The Revelation Principle

Definition 2.4: Dominant Strategy

A mechanism is an implementation of dominant strategies if for each agent a_i and each type t_i there exists a strategy $s_i \in S_i$, termed as dominant strategy, such that for all possible strategies of the other agents a_{-i} (*i.e.* $\{A/ a_i\}$), s_i maximizes agent a_i's utility.

Definition 2.5: Truthful Implementation

A mechanism is truthful if for each agent a_i and all its t_i, $S_i = T_i$, *i.e.*, its dominant strategy is to report its real type.

Definition 2.6: Strongly Truthful Implementation

A mechanism is a strongly truthful implementation if truth-telling is the only dominant strategy for agents.

Vickrey-Groves-Clarke Mechanisms

The most positive result in mechanism design is the generalized Vickrey-Groves-Clarke (VGC) mechanism (Vickery 1961; Clarke 1971; Groves 1973). The VGC mechanism applies to mechanism design problems where the objective function is simply the sum of all agents' valuations.

Definition 2.7: Utilitarian Functions

A maximization mechanism design problem is called utilitarian if its objective function satisfies $g(o, t) = \sum_i v_i(o, t_i)$.

Definition 2.8: A VGC Mechanism

A direct revelation mechanism $m = (o(t), p(t))$ belongs to the VGC family if

- $o(t) \in argmax_o \sum_{i=1}^{n} v_i(o, t_i)$
- $p_i(t) = \sum_{j \neq i} v_j(o(t), t_j) + h_i(t_{-i})$ where $p_i(t)$ is the payment that agent a_i obtains when the type vector is t, $\sum_{j \neq i} v_j(o(t), t_j)$ is the sum of values of all agents except agent a_i when the type vector is t, $h_i()$ is an arbitrary function of t_{-i} and $t_{-i} = (t_1, \ldots t_{i-1}, t_{i+1}, \ldots, t_n)$. A very important theorem of VGC mechanism has been proved by Groves (1973):

Theorem 2.1: A VGC mechanism is truthful.

Feigenbaum *et al*. (2001) extended this model to distributed algorithmic mechanism design (DAMD), in which the same goals of incentive compatibility and computational tractability are presented. In addition, the agents, the relevant information, and the computational model are all inherently distributed. Network complexity also needs to be considered. In Chapter III, I will study the synthetic task allocation problem by formalizing this problem as the algorithmic mechanism design optimization problem.

COALITION FORMATION

Sandholm (1996) probably did the most complete survey of literature related to the theory of coalition formation among self-interested agents in his dissertation. He points out

that coalition formation includes three activities. The first is coalition structure generation, that is, formation of coalitions by the agents such that agents within each coalition coordinate their activities. Mathematically, it means partitioning a given set of agents into disjoint coalitions.

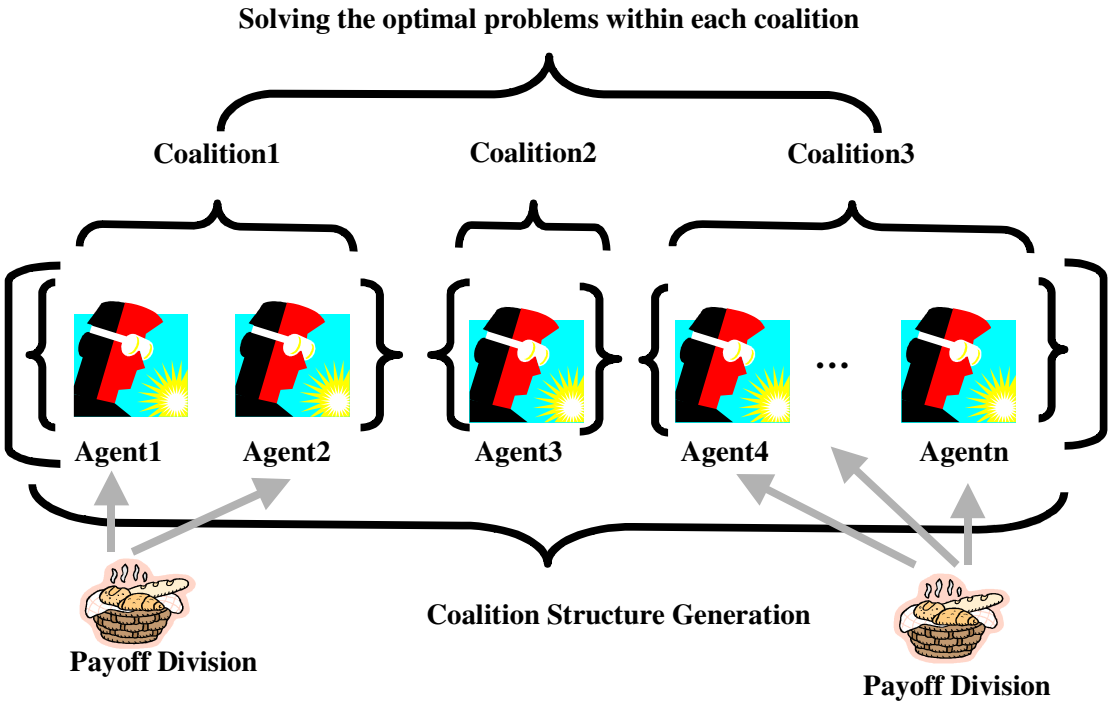

FIGURE 2.1. Activities in Centralized Coalition Formation

The partition is called a coalition structure. Different subsets have different values (*e.g.* due to cost savings, or synergies of capabilities), and the goal is to find a partition of the agents that maximizes this value (summed over each group). The second is solving the optimization problems within each coalition. In a task allocation problem, this decides how to distribute tasks among the member agents of a coalition. The third activity involves payoff division. It deals with how to divide the gain of a coalition to its member agents.

The stability of the coalition depends on whether each member agent agrees with the actual payoff division. These activities interact with each other. Figure 2.1 illustrates these three activities.

The first two activities described above are difficult in terms of their complexities. Sandholm *et al*. (1999) has proved that the complexity of coalition structure generation is $O(n^n)$. Normally, most allocation optimization problems are NP-hard problems (Sandholm 1996; Parkes 2000). Payoff division is also a hard problem in terms of its complexity. The stability of a coalition depends on whether the payoff division is incentive-compatible in the sense that each member agent will not obtain more benefit by leaving the coalition. Therefore, coalition formation also needs to be addressed by jointly considering incentive compatibility and computational tractability.

Game theory provides many solution concepts for evaluating the stability of a coalition (Fudenberg and Tirole 1991; Osborne and Rubinstein 1994) under the assumption that agents involved in coalition formation have perfect rationality (*i.e*., algorithms can find the optimal solution with zero computational cost), which is not realistic in the real world. Sandholm and Lesser (1997) extended coalition formation in game theory to a normative theory of coalitions in combinatorial domains based on a domain classification for bounded rational agents.

Game theorists did not provide actual methods of forming coalitions in real applications. Researchers in distributed artificial intelligence have put a great deal of effort into developing feasible algorithms (Sandholm and Lesser 1997; Sen and Dutta 2000; Caillou *et al.* 2002; Li and Sycara 2002) for all three activities of coalition formation among self-interested agents. Most of these works take centralized approaches by

formalizing coalition formation as a set of optimization problems. Generally, a group leader is chosen for organizing the coalition formation process and is in charge of payoff division. It is unclear how to select such an unselfish leader who is fair and acts in each member's and the group's best interests. The computational intractability of the centralized approaches also makes these algorithms only applicable for a small number of agents.

There are only a few works on coalition formation that adopt distributed approaches. Shehory and Kraus (1995, 1998) developed distributed any-time algorithms of forming coalitions for cooperative agents for task allocation problems. They proposed two additional distributed algorithms for coalition formation among self-interested agents in non-super-additive games (Shehory and Kraus 1999). A merging process of coalition configurations from all agents is required. Voting is suggested to be one of the possible decision-making methods. Lerman and Shehory (2000) developed a distributed buyer coalition formation mechanism for a large-scaled electronic market, where a buyer coalition may form when buyers encounter other buyers or existing coalitions randomly.

This dissertation proposes a distributed coalition formation mechanism that is based on conducting explicit negotiation among self-interested agents. By "explicit", I mean that there are clear procedures for either accepting or rejecting a proposed coalition. An automated multiparty negotiation protocol is developed for autonomous agents. This distributed coalition formation mechanism allows self-interested agents to make their own decision about whether or not to join a coalition. A coalition is formed if all members agree to join this coalition. All agents are symmetric in terms of their equal roles in a coalition formation process. There does not exist any group leader or matchmaker to mediate the coalition formation process. Each agent only knows its own benefits of joining a coalition,

which is private information that cannot be accessed by other agents directly. An agreement about forming a coalition is achieved by negotiation among participating agents.

This decentralized approach to coalition formation is more realistic in practice than previous approaches. Coalitions in the real world are generally formed through explicit negotiation among multiple agents. Agents make their own decisions on whether or not to join a coalition. Most likely, agents have no complete information about other agents' preferences. My mechanism tries to capture these features in a coalition formation process among human agents. Furthermore, by letting agents make their own decisions, my mechanism removes two computationally intractable activities of coalition formation: coalition structure generation and payoff division. The ultimate objective of this research is to maximizing social welfare (optimal coalition structure) through providing appropriate incentive to agents so that their decisions on joining a coalition will lead to an optimal coalition structure under constraints that agents are self-interested and willing to maximize their own utilities.

NEGOTIATION AMONG SELF-INTERESTED AGENTS

Negotiation has been studied in many different disciplines such as politics, economics, business and public relations. Rosenschein and Zlotkin (1994) pointed out that the world functions through interacting agents. Each person pursues his own goals through encounters with other people or machines. The process of negotiation takes place in both formal and informal contexts. It is part of our daily life.

The importance of studying negotiation is straightforward for designing decentralized mechanisms for distributed task and resource allocation problems (Rosenschein and Zlotkin 1994; Sandholm 1996; Kraus 2001; Jennings *et al* 2001). Negotiations are initialized when agents need to make agreements on how to allocate a shared resource, how to do distributed load balancing, how to exchange resources *etc*. In this dissertation, negotiation is used for coalition formation, namely, agents making agreements about whether or not to form a coalition through multiparty negotiation.

There are two main research issues on negotiation in multiagent systems. The first is how to develop practical negotiation strategies (Kraus 2001), *i.e*., what kind of strategies that agents should use to maximize their own good during a negotiation process. The second is to develop protocols that allow automated negotiation agents (Jennings *et al* 2001) to negotiate with each other. Game theory tools (*e.g*., Nash equilibrium, dominant strategy *etc*.) are used to evaluate negotiation strategies and protocols.

The Art and Science of Negotiation

Although I mainly focus on introducing the related work on negotiation in multiagent systems, I do not want to ignore Raiffa's book (1982) "The Art and Science of Negotiation", which is probably the most popular reference book for research work on negotiation. "Art" means dealing with the human element. "Science" means those aspects of the negotiation process that are capable of being analyzed in a fairly structured manner. The book conveys an idea that the "zero-sum" way of thinking, according to which one side must lose if the other wins, often makes both sides worse off than they would be when

bargaining for joint mutual gains. Raiffa is mainly interested in determining which outcomes to negotiation are optimal for both parties. Much of his analysis is based on the premise that both parties will act in an ultimately rational manner and make decisions that will be optimal. These analyses may not be practical in negotiation among human subjects due to the assumption that the negotiation subjects are always rational (*i.e.*, each subject always tries to maximize its own utility). I believe it is the main reason why the researchers in multiagent systems would like to borrow ideas from this book because autonomous agents can be built to be rational. As with most work in game theory, this book does not focus on developing actual negotiation strategies and protocols. It instead analyzes existing cases.

Rules of Encounter

"Rules of Encounter: Designing Conventions for Automated Negotiation among Computers" is a very well-known book about negotiation among autonomous agents written by Rosenschein and Zlotkin (1994), who are pioneers of introducing game theory to distributed artificial intelligence. The book presents the beginnings of the theory of designing interaction protocols among computers with different designers and owners by using game theory tools. It identifies three distinct domains (task oriented, state-oriented and world oriented domain) where negotiation is applicable and addresses different strategies for each domain.

There are two main points that the book tries to make about designing negotiation strategies. The first is that by appropriately adjusting the rules of public behavior (rules of

the game) by which agents must interact, the private strategies of each agent will be influenced. This can ensure desirable global attributes for a distributed and heterogeneous system. The second is that the participating agents must agree on the rules of the game. This can result in stable interaction protocols. Game theory tools are mainly used to evaluate these two criteria.

Throughout the book, only two-agent domains and encounters are analyzed. The book presents a famous two-agent negotiation protocol, which is called "Monotonic Concession Protocol" (MCP). In the MCP protocol, agents start by simultaneously proposing one deal from the space of possible deals. An agreement is reached if one of the agents matches what the other one asked for in terms of utility. My focus in this dissertation is to let multiple self-interested agents achieve agreements through a multiparty negotiation protocol, which extends the MCP protocol to multiple-agent domains.

Strategic Negotiation in Multiagent Systems

Kraus (2001) presented a strategic-negotiation model, which is based on Rubinstein's model of alternating offers where agents exchange offers until they reach an agreement or until one of them opts out of the negotiation. The goal of developing this model is to resolve conflicts among agents by reaching agreements through negotiation. The applications of this model include the data allocation problem in information server, common resource allocation problem, task delegation within a team, and the pollution allocation problem. The objective of this strategic-negotiation model is to provide the

agents with ways to reach mutually beneficial agreements without delay. In this model, there are *n* agents who need to reach an agreement on a given issue.

This dissertation embeds negotiation into coalition formation. Agents might be involved in multiple negotiation processes during a coalition formation process. An agent's decision on whether or not to accept an offer may not only depend on the decisions of other members in a proposed coalition but also depend on the decisions of agents outside of the proposed coalition.

Automated Negotiation

Jennings *et al*. (2001) proposed a generic framework for classifying and viewing automated negotiations. This framework was then used to discuss and analyze the three main approaches that have been adopted to automated negotiation: game theoretic, heuristic and argumentation-based approaches. For each approach, a brief appraisal of its relative merits and drawbacks is presented. They pointed out that much research still needs to be performed in the area of automated negotiation. The aim of this work is not to develop actual automated negotiation strategies or protocols for particular domains. It analyzes existing automated negotiation implementation.

Other Work Related to Negotiation in Multiagent Systems

Negotiation has been studied in distributed artificial intelligence both in distributed problem solving (DPS) where agents are cooperative and in Multiagent Systems (MAS)

where agents are self-interested. Negotiation is used in DPS for solving conflicts, distributed planning and distributed search (O'Hare and Jennings 1996). This dissertation does not focus on cooperative agents.

Sycara (1987, 1990) developed a model of negotiation that combines case-based reasoning and optimization of multi-attribute utilities. Zeng and Sycara (1998) embedded learning into negotiation. Agents can learn from previous encounters about their opponents' negotiation strategies so that they choose corresponding strategies to influence their opponents or to obtain a better deal. Sierra *et al*. (1997) presented a model of negotiation for autonomous agents, which is distilled from intuitions about good behavioral practice in human negotiation. Sandholm and Lesser (2002) explored issues such as levels of commitment that arise in automated contract among self-interested agents whose rationality is bounded by computational complexity. I will discuss about their research more in the related work about distributed task and resource allocation.

DISTRIBUTED TASK AND RESOURCE ALLOCATION IN MULTIAGENT SYSTEMS

Distributed task and resource allocation is a central theme of distributed computer science (Clearwater 1996). I do not address the allocation issues of distributed computer systems when those systems have been centrally designed to pursue a single global goal such as a distributed operating system. I am interested in cooperative task allocation and resource sharing problems in systems that are established through interactions among multiple self-interested agents that are developed by different designers and belong to

different owners. This type of allocation problems has a variety of applications in grid and peer-to-peer computing, electronic commerce and virtual organizations.

Contract Net and Levels of Commitment

The most influential distributed task allocation mechanism in distributed artificial intelligence is the Contract Net protocol (Davis and Smith, 1983), which can be used for both cooperative agents and self-interested agents. The basic idea is that a task manager auctions a group of tasks, agents bid on these tasks based on their local marginal cost calculations. The original Contract Net does not take computational tractability into consideration, even though the marginal cost calculation for combinatorial problems are most likely intractable. Sandholm and Lesser (2002) extended the Contract Net protocol to allow it to work among self-interested computationally limited agents. Agents can reallocate tasks to each other for dynamically constructed charges. As a result, a more profitable global task allocation is reached than the initial one, while not executing a centralized task allocation algorithm.

This dissertation studies the distributed synthetic task allocation problem. A synthetic task must be accomplished by an agent team, which is composed of self-interested agents from different organizations. The objective to minimize the completion time of a set of synthetic tasks even though the true capabilities of agents are not known by the task allocation mechanism.

Methods for Task Allocation via Agent Coalition Formation

As mentioned before, Shahory and Kraus (1995, 1998) developed distributed any-time algorithms for forming coalitions among cooperative agents for task allocation problems. They considered situations where it is necessary to execute a task by a group of agents because it is more efficient or a single agent is not able to perform the task. The objective of this work is to improve the efficiency by allocating tasks to cooperative agent coalitions, which are formed through distributed algorithms. There is not an explicit negotiation protocol among agents. Each agent calculates the costs of coalitions it involves by itself and joins the coalition with the lowest cost for a certain task. The procedure is executed iteratively until there are no more tasks or no existing coalition is beneficial.

This dissertation addresses distributed task and resource allocation among self-interested agents. By forming resource sharing coalitions, agents with both tasks and resources can improve the efficiency of task execution (*e.g.* reduce task execution time) without increasing the amount of resources owned by each agent individually. The coalitions are formed through explicit negotiation among multiple agents.

Auction and Market Based Resource Allocation Mechanisms

Auctions and markets represent two ends of a spectrum of market formulations (Wolski *et al* 2001, 2003). On the market end, an attempt is made to satisfy all bidders and sellers at a given price. At the auction end, one bidder and seller is satisfied at a given price.

Market based resource allocation mechanisms are decentralized and no direct communication is needed. The balance between supply and demand decides the actual resource allocation. When the supply and demand for a certain resource reaches equilibrium, the price becomes stable. How long it will take to reach equilibrium is normally unpredictable. Hence, price setting is a big obstacle for developing a market based resource allocation mechanism (Wolski *et al* 2001, 2003).

An auction (Krishna 2002) is the simplest resource allocation mechanism for self-interested agents in terms of its implementation. Auction-based distributed resource allocation mechanisms have been successful in many real distributed allocation applications. The most positive result about auction is that the second sealed price auction belongs to VGC family and is a truthful implementation. Also, an auction has no problem with price setting. The bad news about auctions is that deciding the winner of a combinatory auction is a computationally intractable problem, but the distributed allocation problems in this dissertation are all combinatory optimization problems.

These two types of allocation mechanisms are basically monetary approaches (Buyya *et al*. 2002). When computational resource allocation among self-interested agents is considered, agents do not explicitly buy others' resources but use them when they are idle. It is very hard for agents to decide whether they should buy the time slots of using a resource or buy the resource itself.

This dissertation focuses on bartering approaches (He and Ioerger 2005b) for developing computational resource allocation mechanisms. The idea is to let agents exchange their idle time slots of their computational resources.

CHAPTER III

SYNTHETIC TASK ALLOCATION IN VIRTUAL ORGANIZATIONS

In traditional computational organization models, the designers of task allocation mechanisms generally make an implicit assumption that organizational agents will report their true capabilities for achieving certain tasks. This assumption might not be true for virtual organizations. Organizational agents with self-interests are typical in virtual organizations. Incentive-compatible task allocation mechanisms must be designed for synthetic task allocation in virtual organizations. In this chapter, I study a synthetic task allocation problem by formalizing this problem as an algorithmic mechanism design optimization problem (Nisan and Ronen 1999).

SYNTHETIC TASK ALLOCATION

Recalling the synthetic task allocation problem illustrated in Figure 1.1, a number of tasks need to be done. Each of these tasks requires cooperation among different groups, which belong to different organizations. Members in different groups have different expertise. Within each group, all members have the same specialty, but they have different capabilities in the sense that different members might require different amounts of time to finish an identical task (or with a different quality, accuracy *etc*.). A member in a group is referred as an agent. A task needs to be done by a team that is composed of agents from different groups. It is assumed that there is no dependency among tasks and that tasks can be executed in parallel.

It is assumed that each task needs exactly one agent from each group. These kinds of projects are referred as synthetic tasks. The project manager needs to build a temporary team for each task. The amount of time required to finish a task depends on the capabilities of members in that team. The objective of the project manager is to minimize the total amount of time required to accomplish all tasks. Definition 3.1 gives the formal definition of synthetic tasks.

Definition 3.1: Synthetic Task

A task, t, is called a synthetic task if t requires a set of activities $X = \{ x_1, \ldots, x_m \}$. Each of these activities needs to be performed by an agent with special skills. The synthetic task t, needs to be executed by an agent team, T, composed of m agents. Each of these agents is only able to perform one of the activities in X. Different agents in T differ in the activities that they can perform.

A typical example of a synthetic task is a software development project that needs to be accomplished by a software development team. For instance, to build such a development team, a software project manager might need to have a software architect, a database administrator, a networking administrator and a few software developers who are good at different programming languages. These experts might actually belong to different groups in an organization. How to allocate the project to the most efficient team to accomplish the project is a big challenge.

Before formally defining the synthetic task allocation problem, I need to define how efficient an individual agent is for executing a single activity (which needs to be done by only one agent) and how efficient an agent team is for executing a synthetic task (which needs to be done by a team of agents with different specialties). Here, I only consider the

amount of time required to finish a task as the evaluation criterion[1] for both individual and team efficiency. Greater efficiency means less time required to finish a task.

The concept of individual efficiency is used to distinguish the different capabilities of agents in one group. They have the same specialty, but they differ in their efficiencies in the sense that they need different amounts of time to finish the same single activity.

Definition 3.2: Individual Efficiency

Individual efficiency is defined as a function $f_I(a_i, x)$: $I \times \Omega \rightarrow \Re^+$, where I is a set of agents and Ω is a set of activities. For $\forall\ a_i \in I$, it can accomplish an activity $x \in \Omega$. The value of $f_I(a_i, x)$ is equal to the amount of time required by agent a_i to finish a single activity x. Given a set $G = \{a_1, \ldots, a_n\}$ of n agents and a single activity x, an agent a_i is individually more efficient than a_j if $f_I(a_i, x) < f_I(a_j, x)$.

My goal is to design a reward mechanism that will get a distributed group of self-interested agents to automatically solve the synthetic task allocation problem. Individual efficiency has a strong effect on team efficiency. Besides individual efficiency, the amount of time that the team needs to finish the task might be also affected by some other factors such as how often these agents work together and how good these agents are at teamwork. If only individual efficiency is known, it is still unclear how to evaluate how efficient an agent team is. Therefore, I define the following concept of team efficiency to distinguish the different capabilities of agent teams, which are composed of agents from different

[1] To evaluate the efficiency of an individual agent or an agent team, the performance quality of accomplishing a task can be another criterion, such as cost or accuracy. However, higher quality performance usually requires more time. Here, assume the performance criteria for each task is the same. More efficient agents or agent teams spend less time to achieve the quality requirement than less efficient agents or agent teams.

groups. The composed teams differ in their efficiencies in the sense that they need different amounts of time to finish a same synthetic task.

Definition 3.3: Team Efficiency

Team efficiency is defined as a function $f_T(T_i, y)$: $V \times \Phi \rightarrow \Re^+$, where V is a set of agent teams and Φ is a set of synthetic tasks. For $\forall\ T_i \in V$, $T_i = (a_{i1}, \ldots, a_{im})$ is an agent team. For $\forall\ y \in \Phi$, y requires a set of activities $X = \{ x_1, \ldots, x_m \}$. a_{ij} in T_i is able to perform activity x_j. The value of $f_T(T_i, y)$ is equal to the amount of time required by agent team, $(a_{i1}, \ldots, a_{im})$, to finish synthetic task y. Let $G_1, \ldots, G_m$ denote m groups of agents and $y \in \Phi$ denote a synthetic task. Agents in group G_j are able to perform activity x_j. Synthetic task y needs to be accomplished by a team composed of m agents. Agent team $(a_{i1}, \ldots, a_{im})$ is more efficient than agent team $(a_{j1}, \ldots, a_{jm})$ if $f_T(T_i, y) < f_T(T_j, y)$.

Now, how does individual efficiencies affect team efficiencies? Currently, I assume that the efficiency of a team composed of more efficient agents is not worse than the efficiency of a team composed of less efficient agents. This kind of teamwork is defined as monotonic teamwork.

Definition 3.4: Monotonic Teamwork

Given a set G = $\{G_1, \ldots, G_m\}$ of m groups of agents and a synthetic task, y, y needs to be accomplished by a team composed of m agents and different agents in a team are from different groups in G. For any two teams A_1 and A_2 that differ by only one agent, *i.e.*, $A_1 \backslash A_2 = \{a_{j1}\}$ and $A_2 \backslash A_1 = \{a_{j2}\}$, where a_{j1} and a_{j2} to the same group G_j (meaning that a_{j1} and a_{j2} can do the same type of single task x), agent teamwork is monotonic teamwork iff $f_T(A_1, y) \geq f_T(A_2, y)$ when $f_I(a_{j1}, x) \geq f_I(a_{j2}, x)$.

In many real applications, the most inefficient team member may slow down the efficiency of the whole team significantly. For example, if a software development team has an inefficient network administrator, the efficiency of the network administrator might determine the whole team efficiency. If the network administrator cannot recover the computer network from some disaster events on time, the other team members may not be able to do any of their jobs. Let's consider another assumption: for the most efficient team, if one agent is replaced by the second most efficient agent in the corresponding group, the efficiency of the team is immediately decreased to the same level of the team in which every member is the second most efficient team from each group. The same rule applies to the transformation from the second most efficient team to the third one, and so on. This type of teamwork is defined as strongly monotonic teamwork.

Definition 3.5: Strongly Monotonic Teamwork

Given a set $\mathcal{G} = \{G_1, \ldots, G_m\}$ of m groups of agents and a synthetic task, y, y needs to be accomplished by a team composed of m agents and different agents in a team are from different groups in $\mathcal{G}$. Assume that each group has n agents and the agents in any group G_j are ordered by efficiency for any given activity x: $f_I(a_{j1}, x) \leq f_I(a_{j2}, x) \leq \ldots \leq f_I(a_{jn}, x)$. Let A_i denote the agent team $(a_{1i}, \ldots, a_{mi})$ and A_k denote that agent team $(a_{1k}, \ldots, a_{mk})$. Let $A_i^{'}$ denotes an agent team in which $m\text{-}1$ members are the same as A_i except for one member from A_k and $i < k$. Agent teams are strongly monotonic teamwork iff $f_T(A_i, y) \geq f_T(A_k, y)$ and $f_T(A_i^{'}, y) = f_T(A_k, y)$.

Based upon the concept of team efficiency, the synthetic task allocation problem is defined as follows:

Definition 3.6: Synthetic Task Allocation Problem

The synthetic task allocation problem is denoted by a tuple $Q_{STA} = <\Phi, \mathcal{G}, f_T>$, where Φ is a set of synthetic task $\{ t_1, \ldots, t_k \}$; $\mathcal{G}$ is a set of agent groups $\{G_1, \ldots, G_m\}$; and f_T is the team efficiency function for tasks in Φ. Within a group G_j, there are n agents $\{a_{j1}, \ldots, a_{jn}\}$, who have the same specialty but with different capabilities. Namely, they can accomplish the same activity in different amounts of time. To accomplish a synthetic task t_i, a team composed of $a_{1i}, \ldots, a_{ji}, \ldots, a_{mi}$ is required. There is no pair in these m agents from the same group in $\mathcal{G}$. Assume that every synthetic task in Φ needs to be accomplished by a team composed of m agents[2] and each task is independent from other tasks in Φ. The amount of time d_i required to finish task t_i is a function of vector $A_i = (a_{1i}, \ldots, a_{ji}, \ldots, a_{mi})$: $d_i = f_T(A_i; t_i)$. Furthermore, if an agent has been allocated to a team for a certain task, it cannot start a new task before the whole task has been finished. The objective of this problem (where agents are assumed to be implicitly cooperative, not self-interested, and all individual capabilities are known) is to minimize the total amount of time for finishing all tasks (makespan). The designed allocation algorithm for this problem can be represented as follows:

- A feasible output of the allocation algorithm is a set of agent vectors $\mathcal{A} = \{A_1, .., A_k\}$ where synthetic task t_i is assigned to A_i.
- Without loss of generality, assume that each group has the same size of n. Then, the total number of possible teams is n^m, but there are only n teams that can exist in parallel. Assume that the total number of tasks is much larger than the size of each

[2] This assumption requires that all synthetic tasks in Φ are the same type of task in the sense that these tasks need to be finished by an agent team composed of m different experts.

group, *i.e.*, $k >> n$[3]. Since there are only n teams that can exist in parallel, each agent in a group G_j might be needed to participate in executing multiple tasks. Hence, from the view of group G_j, an output of the allocation algorithm results a task partition $X_j = (x_{j1}, \ldots, x_{jn})$. Agent a_{jr} will participant in tasks in x_{jr}. X_j is the function of the allocation output $\mathcal{A}$: $X_j = X_j(\mathcal{A})$. Agents of group x_j can execute their tasks in parallel.

- The objective of the allocation algorithm is to minimize $TE(\mathcal{A}, \Phi)$, which is the total completion time of all synthetic tasks in Φ. The objective function can be as:

$$TE(\mathcal{A}, \Phi) = \mathbf{max}_j(\mathbf{max}_r \sum_{l \in x_{jr}} f_T(A_l; t_l))$$

where j is the index of a group, r is the index of an agent in group G_j, and l is the index of a task in a task subset, x_{jr}, in which agent a_{jr} participates. $\sum_{l \in x_{jr}} f_T(A_l; t_l)$ is the total amount of time that agent a_{jr} (who belongs to group G_j) has to spend on participating in executing tasks in x_{jr}. The amount of time that agent a_{jr} needs is equal to the amount of time that tasks in x_{jr} need to be finished because agent a_{jr} cannot start to participating in another new task until its current task has been finished. From the view of group G_j, all n agents in G_j execute task subsets in X_j in parallel. Since the task allocation process is offline, there is no idle time for each agent between any two tasks in which it participates[4]. Therefore, the maximum amount of time that an agent participates in the

[3] This assumption makes the synthetic task allocation problem nontrivial. If k=1, the allocation problem is trivial because the project manager only needs to allocate this task to the most efficient agent team.

[4] **Proof:** Since no team member can quit from a task execution, after a task ends, all involved agents are free for other tasks. The scheduling process is offline. The project manager can construct proper teams before the scheduling. Since all tasks are the same type and independent, the project manager can build fixed n teams that are the best combinations based on the reported types. Then, the project manager schedules all tasks to proper teams. In other words, during the scheduling, the project manager does not reshuffle teams. As a result, there is no idle time for each agent between any two tasks in which it participates.

whole task execution process can be used to represent the minimum amount of time for completing all tasks in Φ.

If agents in each group have the same capability, which is known by an allocation algorithm in advance, the synthetic task allocation problem would be equivalent to the Makespan problem (Coffman *et al*. 1987), which is how to minimize the completion time of a set of single tasks by scheduling them to multiple processors with the same capacity. The Makespan problem is a well-known NP-hard problem (Chen 2004).

Lemma 3.1: The synthetic task allocation is an NP-hard problem.

Proof: Even though the total number of possible teams is n^m in the synthetic task allocation, there are only n teams than can exist in parallel because any agent cannot join two teams at the same time. Therefore, the synthetic task allocation problem can be simplified to be the Makespan problem if the capabilities of all agents in each group are the same and known by the allocation mechanism in advance. The mechanism only needs to allocate k tasks to n teams with the same capability and tries to minimize the completion time of those k tasks.

The NP-hard Makespan problem (Garey and Johnson 1979) is polynomial reducible to the synthetic task allocation problem. The Makespan problem is defined as follows (Chen 2004): given the set of tuples $T = \{c_1, \ldots, c_n; m\}$, where c_i is the processing time for the ith job and m is the number of identical processors. Can $\{c_1, \ldots, c_n\}$ be partitioned into m subsets, $P_1, \ldots, P_m$, such that the processing time of the largest subset can be minimized?

Given an instance $T = \{c_1, \ldots, c_n; m\}$ of the Makespan problem, an instance $\alpha = < \Phi, G, f_T >$ for the synthetic task allocation problem can be constructed. There are n tasks in $\Phi = \{t_1, \ldots, t_n\}$ and only one group G of m agents $(a_1, \ldots, a_m)$. Each task in Φ needs to be executed by one agent in G. For task t_i, the amount of time, c_i, is calculated by function f_T.

Therefore, *T* is a yes-instance for the Makespan problem if and only if an optimal solution to the instance α of the synthetic task allocation problem is minimizing the maximum amount of time that an agent in *G* participates in executing tasks assigned to the agent. □

There exist efficient approximation algorithms for the Makespan problem that can be extended to solve the synthetic task allocation problem. For example, Coffman *et al*. (1987) developed a 2-approximation algorithm for the Makespan problem. I extend it to a 2-approximation algorithm for the synthetic task allocation problem if the teamwork among different groups is strongly monotonic.

SYNTHETIC TASK ALLOCATION FOR SELF-INTERESTED AGENTS

If the project manager knows the capabilities of agents in each group in advance, he can build a proper agent team for each task. The synthetic task allocation problem can be solved as a Makespan problem (Chen 2004). We only need to consider the issue of complexity (*e.g*., use an approximation algorithm for efficiency). However, in virtual organizations, the project manager most likely does not know the true capability of agents in each group in advance. Each agent group belongs to different real organizations. How to give each group incentive to report the true capabilities of their agents also needs to be considered.

Intuitively, if the project manager would pay each team in proportion to its efficiency, each group might have incentive to send its most efficient agent to assigned tasks. However, this policy gives each group incentive to lie about the efficiencies of its agents because reporting higher efficiency than the real efficiency can bring better payment

for each group. Therefore, the project manager needs to have a better payment policy to induce each group to always tell the truth.

My approach is to extend the synthetic task allocation problem as an algorithm mechanism design problem (AMD) (Nisan and Ronen 1999). The idea is to give each group incentives to report the true capabilities of their agents by providing appropriate payments.

EXTENDING SYNTHETIC TASK ALLOCATION TO AN AMD PROBLEM

Assume that all members in one group are cooperative in the sense that all members want to maximize the benefit of the group. The self-interested entities in the synthetic task allocation domain do not refer to group members but to groups. Each group is self-interested in the sense that its goal is to maximize its own utility. For example, a law firm that has lawyers to outsource and a CPA firm with accountants for hire. Typically, there is no communication between groups. Each group does not know about other groups' strategies and states. I also assume that the project manager does not know the true capability of each agent in a group. He builds a team for each synthetic task by using available agents as reported by agent groups. The only way that the project manger can affect the truth-telling of each group is by giving an appropriate amount of payment to each group. I extend synthetic task allocation to the following algorithmic mechanism design optimization (AMDO) problem:

Definition 3.7: Synthetic Task Allocation AMDO Problem

The synthetic task allocation AMDO problem is denoted by a tuple $Q_{STA\text{-}AMDO} = < \Phi, \mathcal{G}, f_T(T), P_{MAX}>$, where Φ, $\mathcal{G}$, $f_T(T)$ are the same as the synthetic task allocation problem (Definition 3.5). P_{MAX} is the total reward available that can be distributed among the groups for tasks accomplished. The objective of this problem is to design a payment policy that will cause each self-interested group to report the true capabilities of its agents (assuming each group is rational). Then the project manager can use an algorithm for the synthetic task allocation problem (definition 3.5) to minimize the total amount of time for finishing all tasks in Φ (makespan).

The synthetic task allocation problem can be extended as the following algorithmic mechanism design optimization problem**:**

- The objective of the synthetic task allocation algorithmic mechanism design optimization problem is to search for an optimal payment policy π^* in a payment policy space such that each group has incentive to reveal the true capabilities of its agents. Let π denote a payment policy that maps a feasible synthetic task allocation $\mathcal{A}$ to a set of payment vectors, $\{p_1, \ldots, p_m\}$. p_j is the payment vector, $(p_{j1}, \ldots, p_{jk})$, of group G_j for tasks in Φ. p_{ji} is the payment that group G_j obtains by sending an agent to execute task t_i. In real applications, a payment function must satisfy that the sum of the payments of all agents in a team for a certain synthetic task must be less than or equal to P_{MAX}, which is the maximum payment that a project manager can afford for all synthetic tasks.

- p_{ji} is the payment group G_j can obtain from sending agent a_{ji} to participate executing task t_i. p_{ji} can be defined as a function $f(t_i, a_{ji}, P_{MAX})$.
- The utility of group G_j is the sum of pay for all tasks its agents help accomplishing

$$u_j(\mathcal{A}) = \sum_{i=1}^{k} p_{ji}$$

The objective of group G_j is to maximize $u_j(\mathcal{A})$ (*i.e.* group G_j is assumed to be rational.)

Solving this problem requires building proper teams for all synthetic tasks and to schedule tasks for those teams. There are totally n^m possible teams. In the worst case, the complexity could be $O(n^m)$. In order to give each group incentive to report the true capabilities of their agents, I need to develop an algorithm for calculating payments for possible teams for each task.

INCENTIVE-COMPATIBLE SYNTHETIC TASK ALLOCATION

In this section, I present my incentive-compatible mechanisms for the above synthetic task allocation AMD problem. The incentive compatible mechanism is the one that gives each group the proper amount of payment that can induce each group to report the true capabilities of its agents so that the group can maximize its utility. Assume that teamwork is monotonic teamwork (see Definition 3.4). This assumption results in the following heuristic: picking the most efficient agent from every group and putting them in one team will create the most efficient or the fastest team**.** If the efficiency of a team depends on the most inefficient team member, then having the most efficient agents from some groups cannot improve the efficiency of a team within which there are some

inefficient agents from other groups. Based on this assumption, I propose the following MinTeamwork Mechanism.

MinTeamwork Mechanism

Based upon this assumption, a simple approximation mechanism for the synthetic task allocation problem can be developed similar to the MinWork mechanism that was described by Nisan and Ronen (1999) for the single task scheduling problem. Since I am dealing with synthetic tasks, I call it the MinTeamwork mechanism.

MinTeamwork Mechanism

- Allocation algorithm: Task t_i is allocated to the most efficient team that can finish task t_i with minimum amount of time.
- Payment policy: By participating in executing task t_i, group G_j can receive payment $P_{ji}(\mathcal{A}) = c(\min_{A_i \in A_I} f_T(A_i; t_i) - f_T(A_i^*; t_i))$ such that $A_i \neq A_i^*$, where $f_T(A_i^*; t_i) = \min_{A_i \in A_I} f_T(A_i; t_i)$ and A_I is the set of all possible agent team for task t_i; c is the factor that maps time to payment unit of P_{MAX} and $c = \frac{P_{MAX}}{m(\sum_i \min_{A_i \neq A_i^*} f_T(A_i; t_i) - \sum_i f_T(A_i^*; t_i))}$. In other words, for each task, the payment is proportional to the value that is equal to the amount of time that the second most efficient team requires to finish this task minus the optimal execution time of task t_i.

The MinTeamwork mechanism lets each team member receive the same amount of payment for executing a task because the corresponding synthetic task cannot be finished if lacking any of the team members.

Theorem 3.1: The MinTeamwork mechanism is truthful for the synthetic task allocation problem.

Proof**:** I can show that truth-telling is the only dominant strategy. Since the synthetic tasks are assumed to be independent, I only need to show the case of one synthetic task, *y* (Varian 1995; Nisan and Ronen 1999). Consider the case where there are only two groups of agents, G_1 and G_2. Let *A* denote the most efficient team and *A'* denote the second most efficient team for the given task *y* according to the reported capabilities of group G_1 and G_2. For group G_1, its utility is

$$U_j = f_T(A'; y) - f_T(A; y),$$

where $f_T(A; y)$ refers to the amount of time that *A* needs to finish the given task and $f_T(A'; y)$ refers to the amount of time that *A'* needs to finish task *y*. The problem is that G_1 does not know G_2's capabilities, so it cannot directly evaluate $f_T(A'; y)$ or U_j. Hence it must make a decision based on the possible values of $f_T(A'; y)$.

If $f_T(A; y) < f_T(A'; y)$, group G_1 wants to make the difference between $f_T(A'; y)$ and $f_T(A; y)$ as large as possible. Making $f_T(A; y)$ as small as possible is the best way to maximize U_j. Therefore, the best strategy of group G_1 is to tell its true type, which is its most efficient agent for this task.

If $f_T(A; y) > f_T(A'; y)$, then group G_1 wants to make the difference between $f_T(A'; y)$ and $f_T(A; y)$ as small as possible. Making $f_T(A; y)$ as small as possible is the best way to maximize U_j. Again, the best strategy for group G_1 is to tell its true type. □

Theorem 3.2: For the synthetic task allocation problem, MinTeamwork is an n-approximation mechanism.

Proof: Let $\mathcal{A}_{opt}$ denote the optimal allocation. The following statements are true:

$TE(\mathcal{A}, \Phi) \leq \sum_{i=1}^{k} f_T(A_i^*; t_i)$

and $TE(\mathcal{A}_{opt}, \Phi) \geq \frac{1}{n} \sum_{i=1}^{k} f_T(A_i^*; t_i)$.

n is the total number of teams that can exist in parallel. In other words, all tasks are executed sequentially by the most efficient team. Since there can be n teams existing in parallel, in the worst case, these n teams have the same efficiency, the MinTeamwork mechanism needs n times of the optimal amount of time to complete all tasks.

$TE(\mathcal{A}, \Phi) \leq n\ TE(\mathcal{A}_{opt}, \Phi)$

Therefore, the MinTeamwork mecanism is an n-approximation. □

MinCompletion Mechanism

Since the MinTeamwork mechanism needs to pay the most efficient team with the payment that is based on the performance of the second most efficient team and the teamwork is assumed to be monotonic teamwork, then the complexity could be $O(2^m)$ in the worst case. The reason is that all possible teams composed of team members who are the first and second most efficient agents in each group should be considered.

The MinTeamwork is not a poly-time computable mechanism because it needs to find out the second most efficient team for each task in order to calculate the payment for each task. Also, *n*-approximation is not a very good allocation algorithm. This mechanism just allocates all tasks to the most efficient team that includes the most efficient agent from each group. Other agents in each group are idle.

In many real applications, no group would be happy if only its most efficient agent is working and others are idle. There is a more realistic payment function for group G_j in $\mathcal{G}$ as follows:

$$P_j(\mathcal{A}) = -c'(\max_{r=1}^{n} \sum_{l \in x_{jr}} f_T(A_l; t_l))$$

Namely, group G_j in $\mathcal{G}$ values the execution of all tasks by the makespan as the shorter the better. This means that all groups have the same objective as the project manager. If the teamwork is strongly monotonic teamwork, there is another new mechanism for this special case:

MinCompletion Mechanism

- Allocation algorithm: Task t_i is allocated to the currently most efficient team that can finish task t_i with minimum completion time (completion time is equal to the duration from the starting time of the first task to the finishing time of task t_i).
- Payment policy: The payment of group G_j is equal to

  $P_j(\mathcal{A}) = -c'(\max_{r=1}^{n} \sum_{l \in x_{jr}} f_T(A_l; t_l))$

  Where $\max_{r=1}^{n} \sum_{l \in x_{jr}} f_T(A_l; t_l)$ is the completion time of all tasks and *c'* is the factor that maps time to payment unit of P_{MAX} and $c' = \frac{P_{MAX}}{-m(\max_{r=1}^{n} \sum_{l \in x_{jr}} f_T(A_l; t_l))}$.

When conditions satisfy strongly monotonic teamwork, the best way to construct teams is to make agents the same level of individual efficiency as a team because mixing agents with different levels of individual efficiency does not help improving the team efficiency.

Theorem 3.3: MinCompletion mechanism is a truthful implementation.

Proof: If group G_j declares a more efficient agent than the real most efficient agent, it does not help to decrease the completion time of a task, *i.e.*, $u_j(\mathcal{A})$ will not increase. If group G_j commits a less efficient agent than its real most efficient agent to a task, it will increase the completion time of a task, *i.e.*, $u_j(\mathcal{A})$ will be less than the one by telling truth. Therefore, MinCompletion is truthful. □

Theorem 3.4: For the synthetic task allocation problem, MinCompletion is a *2*-approximation mechanism and poly-time computable.

Proof: Under the above assumption, the MinCompletion mechanism automatically constructs n teams for k tasks in the following way: every member of the most efficient team is the most efficient agent from each group; every member of the second most efficient team is the second most efficient agent from each group and so on. The synthetic task allocation is equivalent to a task scheduling problem in which k tasks are assigned to n agents with different capacities. Let $\mathcal{A}_{opt}$ denote the optimal allocation. Let D_1 denote the time period within which all n team are executing tasks in parallel and D_2 denote the time period from the end of D_1 to the completion of all tasks by the MinCompletion mechanism.

$$D_1 = \min_{r=1}^{n} \sum_{l \in x_{jr}} f_T(A_l; t_l)$$

$$D_2 = \max_{r=1}^{n} \sum_{l \in x_{jr}} f_T(A_l; t_l) - D_1$$

Therefore, $TE(\mathcal{A}, \Phi) = D_1 + D_2$

where $TE(\mathcal{A}, \Phi)$ is the completion time for all tasks. Since all teams are executing tasks during D_1,

$TE(\mathcal{A}_{opt}, \Phi) \geq D_1$,

where $TE(\mathcal{A}_{opt}, \Phi)$ is the optimal completion time of all tasks. During D_2, there is at least one team idle. Since the MinCompletion mechanism allocates each task to the team that can finish the task within minimum completion time, in the worst situation, there is one big task left after D_1. This task_should be allocated to the most efficient team. In this worst case, the optimal completion time is longer than D_2. Therefore,

$TE(\mathcal{A}_{opt}, \Phi) \geq D_2$.

Then, $2TE(\mathcal{A}_{opt}, \Phi) \geq TE(\mathcal{A}, \Phi)$. □

The running time of the allocation algorithm of the MinCompletion mechanism is $O(nk)$, where k is the total number of synthetic tasks and n is the total number of teams. For each task, which team can finish the task within the minimum completion time needs to be computed. The allocation algorithm allocates each task to the team that can finish it earliest based on the reported types.

SUMMARY

In this chapter, I studied a synthetic task allocation problem by formalizing this problem as the algorithmic mechanism design optimization problem (Nisan and Ronen 1999). Each synthetic task needs to be accomplished through the cooperation among agents

who belong to different groups that are self-interested and have different specialties. If the capabilities are known, this problem can be solved as a makespan problem. But with self-interested agents, our goal is to design a payment mechanism that gives agents incentive to tell the truth and form optimal teams automatically. The problem is extremely hard in the sense that there are $O(n^m)$ possible teams (n is the size of a group, and m is the number of groups.) and $k>>n$ tasks needs to be executed. Indeed, even the individual task allocation problem is NP-hard (Coffman *et al*. 1987). Therefore, the synthetic task allocation problem needs to jointly address incentive compatibility and computational tractability.

For self-interested agents, I have developed two incentive-compatible mechanisms for this problem. The MinTeamwork is an n-approximation mechanism and a strongly truthful implementation for monotonic teamwork. By changing the valuation function and having a more restrictive assumption, the MinCompletion mechanism is a truthful implementation with 2-approximation for strongly monotonic teamwork. I have shown that incentive-compatible mechanism design is applicable for synthetic task allocation problems in virtual organizations.

CHAPTER IV

DECENTRALIZED COALITION FORMATION THROUGH EXPLICIT NEGOTIATION

Distributed task and resource allocation are intertwined problems. Resource users need to allocate their tasks to the most efficient resources with the lowest cost. Resource providers want to allocate their resources to the most profitable tasks. In many real applications such as computational grids and peer-to-peer systems, the participating agents are both resource users and providers. The basic idea behind these systems is that agents have peak workloads at different times so that they can utilize the resources of others at idle time. The task and resource allocation mechanisms can be used to build a virtual community, in which participating agents (who might belong to different organizations) can share their computational resources to satisfy their excess resource capacity demands without purchasing more actual resources individually. In other words**,** distributed load-balancing crosses the boundaries of ownership. The aim of this study is to develop a decentralized approach to enable coalition formation among self-interested agents through automated negotiation.

As Sandholm (1996) points out, coalition formation processes for self-interested agents include three activities: coalition structure generation, solving the problem optimally within each coalition and payoff division. The computational cost of this approach has been proved extremely expensive. Sandholm *et al*. (1999) have proved that the complexity of coalition structure generation is $O(n^n)$ in the worst case. The most important reason for the computational complexity is that this approach uses centralized methods (*e.g*. via a group

leader) to search for globally optimal solutions (Yamamoto and Sycara 2001; Li and Sycara 2002; Ye and Tu 2003). Also, the centralized decision-making approach is not applicable in the real world where self-interested agents would like to make their own decisions. Ideally, the globally optimal solutions should be achieved through the interaction among agents. Toward this goal, I develop a distributed coalition formation mechanism called the decentralized coalition formation through explicit negotiation (DCF-EN) mechanism, in which self-interested agents reach agreements on whether or not to join a coalition through explicit negotiation. The DCF-EN mechanism can dramatically reduce the complexity of the coalition formation process by decentralizing two activities in centralized coalition formation approaches: coalition structure generation and payoff division.

FORMALIZING COALITION FORMATION

Definition 4.1 (Coalition Formation Problem)

Let $A = \{a_0, a_1, \ldots, a_{n-1}\}$ be a set of agents in a multiagent system. Each agent is willing to find partners in A to form a coalition so that it can obtain some benefit (*e.g.* saving cost) that it cannot gain when it acts alone. If no such a coalition exists, agents will act alone. Agents can communicate with each other symmetrically.

Definition of Terms

- Coalition: A coalition (*CL*) is a subset of the agent set A that is committed to cooperating for a certain purpose (*e.g.* accomplishing a task).

- Coalition Structure: A coalition structure (*CS*) is defined as a partition of the agent set, (*i.e.*, an exhaustive and disjoint set of coalitions, where each agent is in exactly 1 coalition.)
- Utility Function: The utility of an agent to join in a coalition is the benefit that the agent can obtain from being a member of the coalition. It is defined as a function of the corresponding coalition. Without loss of generality, assume that the value of a utility is a real number. For example, for agent a_i to be a member of a coalition CL_j, its utility is defined as: $U_i(CL_j)$: $2^A \rightarrow \Re$. The goal of agent a_i is to maximize its U_i. The utility of joining a coalition is computed by each agent itself, and it is based on the agent's preferences and the domain features of the application.
- Payoff Function: The payoff that agent a_i obtains by joining coalition CL_j is defined as: $P_i(CL_j) = U_i(CL_j) - U_i(\{a_i\})$. Let $P(CL)$ denote a vector of payoffs of all agents in a coalition CL. Let $P(CS)$ denote a payoff configuration that is a vector of $P(CL)$s of all coalitions in a coalition structure CS.
- Coalition Value: The coalition value of a coalition CL is defined as the sum of the utilities that all members obtain through joining the coalition. It can be described with the following equation:

$$V_{CL} = \sum_{a_x \in CL} U_x(CL)$$

- Value of Coalition Structure: The value of a coalition structure CS is defined as the sum of the values of all coalitions in the coalition structure. It can be described as the following equation:

$$V_{CS} = \sum_{CL_y \in CS} V_{CL_y}$$

The objective of the coalition formation problem is to find the partition that maximizes V_{CS} under the constraint that agents want to maximize their own utilities.

DECENTRALIZED COALITION FORMATION THROUGH EXPLICIT NEGOTIATION (DCF-EN)

Given the above coalition formation problem, in this section, I present the DCF-EN mechanism, which is a decentralized coalition formation mechanism based on multiparty negotiation. There are two main stages in coalition formation. The first is that each agent calculates the utilities of all possible coalitions that it could join. The second is that agents negotiate to achieve agreements on joining coalitions. The first stage is relatively simple. Each agent does exhaustive search for possible coalitions for itself. The second stage is much more complex, in the sense that it requires a multiparty negotiation mechanism, which involves negotiation strategies, negotiation protocol, and message handling methods. Since my goal is to develop a decentralized coalition formation mechanism to be used by autonomous agents, communication costs and conflicts need to be taken into consideration.

Constructing Possible Coalition Space

The first stage of coalition formation in the DCF-EN mechanism is constructing a possible coalition space for each agent in the agent set *A*. Each agent computes the utilities of all possible coalitions (subsets) in *A* that also include the agent itself. Then the agent removes all possible coalitions for which the utilities are less than the utility obtained by

the agent acting alone. It sorts other possible coalitions in a descending order and stores these possible coalitions (*PCF*) as its potential coalition space. In the worst case, for each agent, the complexity of computing its *PCF* is $O(2^{n-2})$. Figure 4.1 illustrates this process.

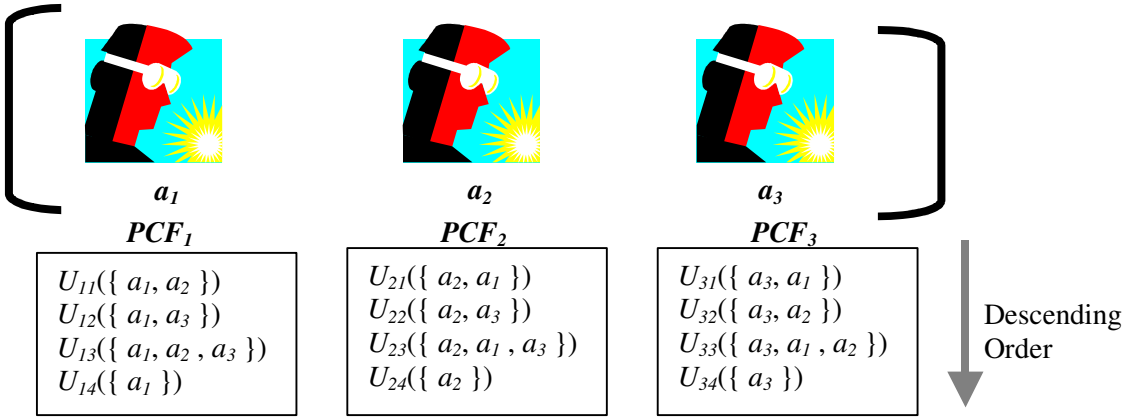

FIGURE 4.1. Constructing Possible Coalition Spaces

The main purpose of this process is to search for possible coalition proposals for the next negotiation stage. The sorting process is used to prioritize all the possible proposals. The approach in this work is based on an assumption under which each agent is able to compute the utility of joining a coalition according to its local knowledge. Although agents do not necessarily know the preferences of others at this stage, it is reasonable to have some expectation. In the real world, any agent who wants to join a proper coalition will have a prior expectation before it enters negotiations. This expectation could be calculated according to prior knowledge or the belief of others' preferences in a given domain. The utility of joining in a possible coalition for an agent represents the expected benefit that this

agent would obtain from joining in this coalition. I will give examples on how to compute the utility of various coalitions in real applications.

Multiparty Negotiation Mechanism

The purpose of developing a multiparty negotiation mechanism in this work is to facilitate coalition formation among self-interested agents. Therefore, it is not a single bargaining process within one agent group, but multiple bargaining processes within multiple agent groups. Assume that all agents have symmetric negotiation abilities in the sense that each agent can independently propose a possible coalition and no agent can force others to accept a proposal. The communication channels are also assumed to be symmetric. There are three main details that need to be defined in such a negotiation mechanism (Rosenschein and Zlotkin 1994): the space of possible deals, the negotiation strategy, and the negotiation process.

The Space of Possible Deals

In the DCF-EN mechanism, the space of possible deals for each agent is its PCF. In the worst case, the size is 2^{n-1}-1. For an entire coalition formation process, the space of possible deals is the union of all agents' *PCF*s. In the worst case, the size is $n2^{n-1}$ for each agent. The reason why the DCF-EN mechanism is still exponential is that the number of possible coalitions is exponential in the number of agents n. Agents have to calculate their own utilities of joining a possible coalition. As Shehory and Kraus (1998) pointed out that the complexity can be reduced to polynomial time by restricting the maximum number of agents to participating coalition formation to a constant number, since large groups are

unreasonable (*e.g.* inefficient) in many domains (*e.g.* buyer coalition formation). If the number of agents is restricted to a small constant k, the total number of negotiation deals is 2^{k-1}. In this situation, the DCF-EN mechanism runs in polynomial time, given k as a constant.

Negotiation Strategy

Negotiation strategy defines how an agent decides which deals to choose. The goal of DCF-EN mechanism is to form coalitions among n self-interested agents. The negotiation strategy of each agent should help it join a coalition so that its utility can be maximized. There are two types of negotiation strategies involved in a coalition formation process. The first is a bargaining strategy for members in a possible coalition. The second is a coalition strategy for finding the best coalition among all possible coalitions. These two types of strategies interweave with each other.

For each possible coalition, all members need to agree on whether or not to join it. When an agent proposes a possible coalition to all other members, bargaining can be used to determine payoff division. To evaluate all feasible coalitions and to decide which one is the best, each agent needs to compare the bargaining results of all possible coalitions and find the one that maximizes its utility.

Since I assume each agent knows the utility that it will obtain by joining a coalition, in the current DCF-EN mechanism, the bargaining strategy is that if every member agrees on the proposal, the deal is closed; otherwise the agents move on to other possible proposals. Because each agent has prioritized its possible coalitions, the DCF-EN mechanism lets each agent do greedy searching, namely, always proposing the possible coalition with maximal utility first. This way, each agent can end its negotiation procedure

whenever it has decided to join a coalition without going through the remaining possible coalitions. As a summary, I list the negotiation strategies for an agent in DCF-EN mechanism as follows:

- For each proposed coalition, agents can accept it if it is the best current possible coalition, put it into a waiting list if it is better than staying alone or reject it if it is worse than staying alone.
- Agents greedily accept the best coalition they can get. If there are multiple choices, they will join a coalition with the smallest size because forming smaller size of coalitions through negotiation will result in lower communication cost.
- Each agent might have multiple negotiation processes going on simultaneously, but all of the involved coalitions should result in the same utility for the agent at that time.
- Each agent terminates a negotiation process when the coalition proposal is formed (accepted) or failed (refused).

Negotiation Process

A negotiation process defines how negotiation among coalition members is conducted. To do so, the following questions need to be answered: What kind of information needs to be exchanged among agents? Where does the information flow go during a negotiation process? Because agents communicate with each other in a decentralized manner, how could they avoid communication conflicts during negotiation? Since multiple negotiation processes might execute in parallel during coalition formation, how do they affect each other?

In the DCF-EN mechanism, agents negotiate with each other by sending messages. Three types of messages are needed: offering a proposal, accepting/rejecting a proposal

offered by others and closing a deal. Table 4.1 lists all messages exchanged in the DCF-EN mechanism. Negotiation processes for an agent require processing negotiation messages from other agents and updating its *PCF* based on negotiation results in previous negotiation processes. A message handling method is called "message handler". Any negotiation decision is made based on the negotiation strategy of buyers. Therefore, the message handlers implement the negotiation strategies of agents.

TABLE 4.1. Messages in the DCF-EN Mechanism

Message Type	Message Purpose
CFPropose	Proposing a possible coalition to a corresponding member
CFAccept	Accepting a received proposal
CFReject	Rejecting a proposal for a possible coalition
CFFailed	Informing an agent who has sent CFAccept that the corresponding proposal has been withdrawn
CFConfirm	Informing an agent who has sent CFAccept that all members have accepted the corresponding proposal
CFNoNeed	Informing an agent who proposes a new coalition that the decision on joining a coalition at this round has been made

The message handlers also embed the negotiation protocol in the DCF-EN mechanism. The negotiation protocol is presented as follows:

- An agent proposes a coalition by sending the proposal to all other members.
- When an agent receives a proposal, it will send back its decision on whether or not to accept it. If it has found a coalition, it will send back its status.
- After every member agrees to a proposal, the agent who proposed it will send confirmation to every member. If the proposal is rejected, the agent who proposed it will send failure message to the members who have accepted.

- At any given time, each agent can only accept to join in one coalition.
- Each agent can propose multiple best current coalitions simultaneously. However, all these coalitions that have been sent out in parallel should result in the same utility.
- Each agent terminates a negotiation process on a coalition proposal 1) if the coalition is accepted by all members, 2) if the proposal is rejected by anyone of the members, and 3) if the expected ending time of the corresponding coalition proposal has been approached. I will address the reason why this is one of reason to terminate a negotiation process in the next section.

Handling Deadlocks in the DCF-EN Mechanism

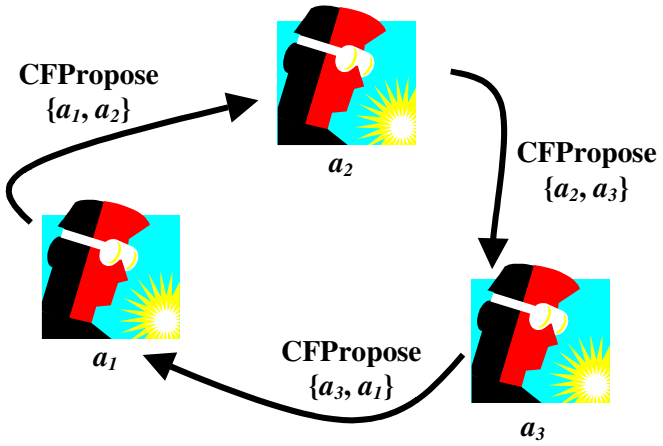

FIGURE 4.2. An Example of Communication Deadlock

There exist multiple decentralized negotiation processes simultaneously because every agent can send out its proposals. Hence the last item of this protocol is designed to avoid conflicts and to end negotiations properly. One potential problem is deadlock (Figure

4.2). A deadlock occurs when an agent proposes to another agent who is waiting for a response from a third agent. Figure 4.2 illustrates a simple example. Agent a_1 sends a current best proposal to agent a_2; agent a_2 sends a proposal to agent a_3 and agent a_3 sends a proposal to agent a_1. Each agent is waiting for feedback. A circle is formed and a deadlock happens.

Indeed, the reason why deadlocks exist is that agents' individual preferences conflict with each other. If deadlocks do not happen during coalition formation, the DCF-EN mechanism is able to find a coalition structure in which each agent can maximize its utility. As a result, the value of the coalition structure is also the maximum. In other words, a coalition structure does not exist when a deadlock happens. Resolving a deadlock is equivalent to letting each agent know the current proposal in which it is interested might not exist. In this case, the agent would move to the next possible coalition proposal. Handling these types of deadlocks includes two steps: detecting the deadlock and breaking the deadlock.

There exist many excellent deadlock detecting algorithms in both distributed operating systems (Silberschatz *et al*. 2002) and database concurrency control mechanisms (Bernstein *et al*. 1987). Typical examples are time-out, token-passing and dependent graph. Any of them can be extended to detect deadlocks in coalition formation.

However, breaking deadlocks in coalition formation is different from breaking deadlocks caused by multiple processes competing for a common resource in distributed operating systems or breaking deadlocks caused by writing to the same variable in a distributed database. In these two situations, breaking a deadlock means the common resource or the variable will eventually be assigned to one process. Breaking a deadlock in

coalition formation means that this coalition structure does not exist. None of the agents can be guaranteed to keep its current proposal because in order to break a deadlock, some agents need to move to other choices. There are more than two agents in coalition formation. The next possible proposal might not involve the same agents as the current proposal at all. Therefore, in order to break a deadlock, deciding whether to remain committed to the current proposal or to move to the next possible proposal is the key decision that each agent needs to make. In the following two subsections, I will address what kinds of strategies that the DCF-EN mechanism uses to detect and break deadlocks in detail.

Detecting Deadlocks

I have considered three approaches to detect deadlocks: time-out, token passing and dependent graph. Each has positive and negative consequences for detecting deadlocks in coalition formation. Time-out is the simplest method by which each agent detects a deadlock if the amount of time it waits for feedback on a proposal exceeds a reasonable amount of time. There are several benefits to the time-out approach (Bernstein *et al*. 1987). It is a pure distributed approach in the sense that each agent does not need extra information to make a decision**.** It does not require that any other agents reveal their private information, and it does not introduce any extra communication load. The down side of the time-out approach is that each agent needs to wait a certain amount of time that is long enough to detect the situation.

Token passing approach (Holliday and El Abbadi, 2005) is also a distributed method by which each agent sends out a token with its proposed possible coalition. The agents that are involved in the coalition will propagate the token by attaching the token to

the proposals that they send out. If the original agent who starts the token receives the token later, it will detect that there is a deadlock. The benefit of this approach is that even though agents have to wait for a while to detect the deadlock, it is not necessary for each agent to wait the same amount of time every time. The down side of this approach is that agents need to reveal their private information about who wants to be their partners to their desired partners, and it will cause extra communication load. The number of tokens passed to agents is equal to the number of messages for proposing all possible coalitions.

Dependent graph (Bernstein *et al*. 1987) is a centralized approach by which there exists a centralized deadlock detector who knows the proposals that are sent out by all agents. Based on the global information, this centralized detector can detect deadlocks by constructing a dependent graph and analyzing the dependencies between different proposals. In order to let the detector obtain information about all proposals sent out, I assume that each agent sends a proposal copy to this centralized detector whenever they send out a coalition proposal. The benefit of this approach is that agents do not have to wait for a long time to detect whether they are in a deadlock. The obvious down side of this approach is that self-interested agents have to reveal their private information to the third party and trust the detector absolutely. Centralized construction of the dependent graph and detecting deadlocks are not trivial tasks but NP-hard problems (Bernstein *et al*. 1987). Furthermore, this approach also causes extra communication load. The number of messages sent to the centralized detector is equal to the number of proposals sent out. After the detector detects deadlocks, it will inform every agent who is involved in the deadlock.

Functionally, all three of these approaches are able to detect deadlocks. However, token-passing and dependent graph approaches require that agents reveal their private

information and incur extra communication load. Since agents are self-interested in coalition formation, in the current DCF-EN mechanism, we use the time out strategy for detecting deadlocks. Furthermore, another reason to choose the time out strategy is that each agent needs to decide whether to move to other possible coalition proposals with less utility or wait for a while to see if the current proposal can be achieved because others might move to new proposals first after a deadlock has been detected. Therefore, agents still need to decide how long they would like to wait before they move to other possible coalition proposals. I will describe this in detail in the next section.

Breaking Deadlocks

During coalition formation, for each agent, breaking a deadlock is equivalent to deciding whether to shift to another possible coalition proposal with less utility or to wait a while to see if the current proposal can still be made because other agents shift to other coalition proposals. If an agent decides to move to the next possible coalition proposal with less utility, it risks that it might lose the opportunity to achieve the current proposal with better utility if other agents move to their next proposals first. On the other hand, if the agent decides to stick with its current proposal, it risks that the current proposal might never be achieved. As a result, it also loses the opportunity to join a coalition with a better utility because other agents might form coalitions that do not include the agent at all. Hence, estimating how long an agent needs to wait to make a proper decision about whether to give up this proposal or not is a critical issue for breaking a deadlock. Agents need to consider the risks they will take to make a proper decision.

In the DCF-EN mechanism, an agent calculates how long it should wait to move to the next possible coalition proposal by the following equation:

$$d = c_d + c_b(1 - \frac{U_b - U_c}{U_b - U_s})\frac{U_c - U_n}{U_c - U_s}$$

where d refers to the amount of time that an agent should wait to move to the next possible coalition proposal, c_d refers to the amount of time that an agent needs to wait to detect a deadlock, c_b refers to the maximal amount of time that an agent is willing to wait for joining a coalition (except the amount of time it needs to detect deadlocks), U_b refers to the utility that the agent can obtain if it joins the best possible coalition in its possible proposal list, U_c refers the utility that the agent can obtain if the current coalition is formed, U_n refers to the utility that the agent can obtain if the next possible coalition is formed and U_s refers to the utility that the agent can obtain if it does not join any coalition.

Sandholm and Lesser (1995) extended the Contract Net Protocol to be used among bounded rational self-interested agents in a production scheduling domain. In this work, the authors pointed out that a more advanced agent should use a risk taking strategy to decide whether to accept the current offer now or wait longer to see if there are better offers later. Agents need to take the risk of missing opportunities due to others making related contracts first. In the DCF-EN mechanism, the current proposal is always better than future proposals. Agents need to decide whether or not to wait longer because if other agents give up earlier, the current proposal might not be made. The basic idea behind the above waiting time equation is that the amount of time that an agent should wait is proportional to the difference between U_c and U_n (utility of the best and the next best proposal). The larger the difference, the longer an agent would like to wait. In other words, if U_c is much larger than U_n, an agent would be willing to take the risk of waiting longer. If there is no big difference between U_c and U_n, the agent should move to the next possible coalition proposal in order

to avoid missing the opportunity to achieve it. $\frac{U_c - U_n}{U_c - U_s}$ is a risk-taking factor. The bigger this factor is, the more risk of waiting longer that an agent is willing to take. The reason why I adopt this strategy is that agents do not know other agents' possible coalition proposal spaces and they have to make decisions according their local information. If the difference between U_c and U_n is significant, an agent would be willing to take more risk of losing the opportunity to achieve U_n because it is possible that it could achieve U_c if it waits long enough so that the involved agents make the agreement.

Another important issue about this equation is how to determine c_d (the amount of time for deadlock detection) and c_b (the difference between the total amount of time that each agent is willing to wait in order to find a coalition to join and c_d). In the DCF-EN mechanism, setting c_d is tricky but manageable (Bernstein *et al*. 1987). If it is too short, an achievable transaction will be aborted. If it is too long, the deadlock will not be detected until the timeout period has elapsed. The timeout period is therefore a parameter that needs to be tuned. It should be long enough so that most deadlocks are detected, but short enough that deadlocks are noticed without waiting too long. The c_d should be longer than the maximal amount of time by which an agreement can be made without deadlocks happening. This maximal amount of time should be equal to or longer than the multiplication of 2^k (k is the total number of agents and it is restricted as a small constant number) and the amount of time of making an agreement without waiting for any other proposals. In the DCF-EN mechanism, each agent itself decides c_d. Each agent can set c_d based on its own preference (*e.g*. a deadline). In different domains, agents may have different preferences.

The DCF-EN Agent Structure and Algorithms of Message Handlers

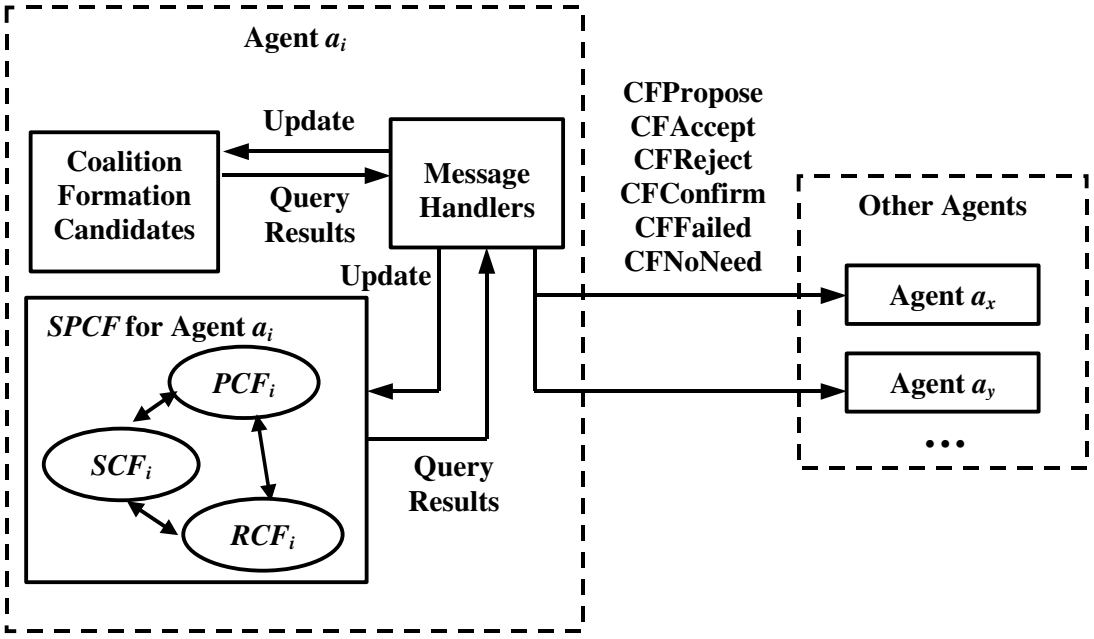

FIGURE 4.3. Agent Structure in the DCF-EN Mechanism

In my negotiation protocol in the DCF-EN mechanism, all possible coalitions construct the space of possible deals (*SPCF*) for an agent. An *SPCF* includes three sets of possible coalitions for each agent: the proposals an agent has sent out (*SCF*); the proposals it has received (*RCF*); and all other possible proposals (*PCF*) that are not in *SCF* and *RCF*. The relationship among the above coalition sets is given by the following two equations:

```
If CFSearch is not done
Then
  If  All other members in the CF have accepted And not send out any CFAccept
  Then Send CFConfirm to all other agents in the CF;
       Send CFNoNeed to all agents who are not members in the CF;
       If  The size of SCF > 1
       Then Send CFFailed to all members of the other CFs in SCF
Else Send CFNoNeed to the message sender
```

FIGURE 4.4. Message Handler for CFAccept

```
If  CFSearch is not done
Then
   If  the proposed CF not in RCF, SCF and PCF
   Then  If  the CF is optimal
         Then  Send CFAccept
         Assert the CF to RCF.
   If  the CF in SCF
   Then   If  has not sent CFAccept
         Then Send CFAccept
   If  the CF in PCF
   Then  Remove it from PCF;
         Assert it to RCF;
         If  the CF is optimal
         Then Send CFAccept
Else  Send CFNoNeed to the sender
```

FIGURE 4.5. Message Handler for CFPropose

- $SPCF = SCF \cup RCF \cup PCF$;
- $SCF \cap RCF \cap PCF = \varnothing$.

Figure 4.3 shows the agent structure in the DCF-EN mechanism. The pseudo code of all the message handlers is presented from Figure 4.4 to Figure 4.9

If CFSearch is not done
Then Accept new available CF again;
Send CFFailed to other members in the *CF*;
If No more new Candidates
Then Remove the *CF* from *SCF*

FIGURE 4.6. Message Handler for CFReject

If CFSearch is not done
Then
If Has accepted the *CF*
Then Accept new available *CF* again;
If No more new candidates
Then Remove the *CF* from *RCF*

FIGURE 4.7. Message Handler for CFFailed

Clean all *CFs* in *SPCF*;
End the current *CF* search process.
Send CFNoNeed to any new message sender.

FIGURE 4.8. Message Handler for CFConfirm

The diagram in Figure 4.10 illustrates what kind of information needs to be exchanged and where a message should go during negotiation. Suppose that the best possible coalition for agents a_1, a_2, a_3 and a_4 are (a_1, a_2), (a_1, a_1), (a_3, a_2, a_4) and (a_4, a_2, a_3) respectively. Agent a_1 and a_3 send CFProposes to agent a_2 and a_4 first. After agent a_2

receives the CFPropose(a_1, a_2) from agent a_1, it accepts that proposal and sends CFReject to agent a_3 because agent a3 has sent CFPropose(a_3, a_2, a_4).

If CFSearch is not done
Then
For the agent who send the message
Remove *CFs* in *PCF*, which include the agent;
If Has accepted a *CF* in *RCF*, which include the agent
Then Accept new available *CF* again
Remove *CF* in *RCF*, which include the agent;
Send CFPropFailed to all members in *CFs* in *SCF*, which include the agent;
Remove *CF* in *SCF*, which include the agent.

FIGURE 4.9. Message Handler for CFNoNeed

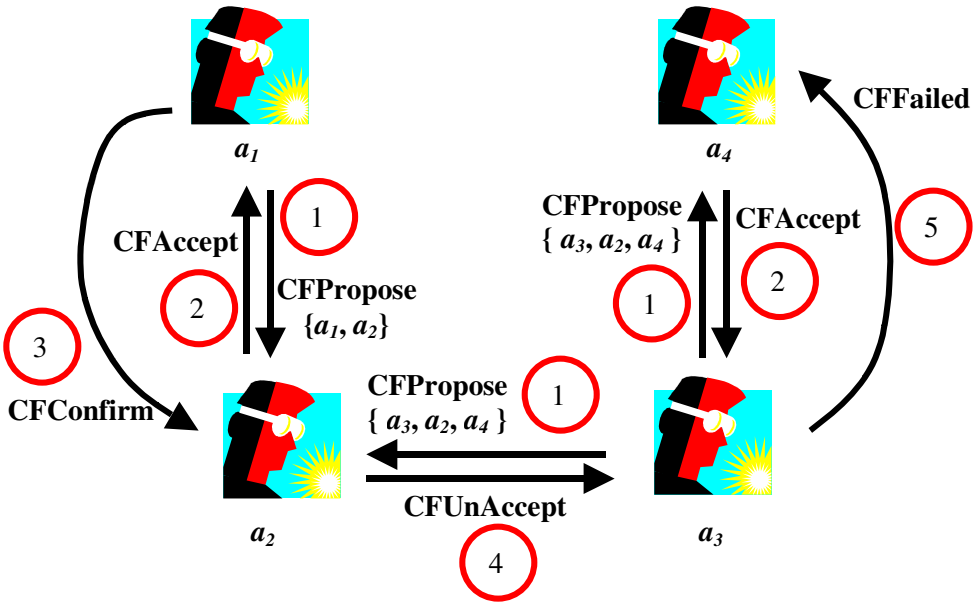

FIGURE 4.10. Information Flows in Coalition Formation

After agent a_3 accepts CFReject, it sends CFFailed to agent a_4 because agent a_4 sent CFAccept before. Agent a_1 sends CFConfirm to agent a_2 to confirm their coalition. Then

agent a_1 and agent a_2 end their coalition formation process. Agent a_3 and a_4 may continue their processes if they can find other partners to form coalitions such that they can obtain better utilities than they act alone. The numbers in the circles represent the order of the messages being sent out.

Properties of the DCF-EN Mechanism

Compared with the traditional centralized coalition formation approach (Sandholm 1996, Yamamoto and Sycara 2001; Li and Sycara 2002; Ye and Tu 2003), the DCF-EN mechanism has the following advantages:

Complexity analyses

- Coalition structure generation is not necessary any more. The DCF-EN mechanism significantly reduces the complexity of coalition formation. In the worst case, the total number of negotiation deals is 2^{n-1} for each agent. The number of agents can be restricted to a small constant k (Shehory and Kraus 1998) so that the total number of negotiation deals is 2^{k-1}. In this situation, the DCF-EN mechanism runs in polynomial time, given k as a constant.
- Since the DCF-EN mechanism relies on communication among agents, the communication complexity needs to be examined. Practically, the communication costs are not expensive because there are at most three types of messages between two agents for each negotiation process: CFPropose, CFAccept/CFReject, CFConfirm/CFFailed. In the worst case, the total number of messages for negotiation in coalition formation is $3n \bullet 2^n$. Again, if the number of agents is restricted to a small number k, the total number

of messages is reasonable. In practice, since agents would select the smallest size of coalition to join if multiple coalitions result in the same utility, the communication load can be further reduced.

- In traditional centralized approach for coalition formation, payoff division is another big challenge in terms of not only stability of formed coalitions but also computational complexity (Sandholm 1996). Since each agent makes its own decision on whether or not to join a coalition, payoff division is excluded through explicit negotiation processes in the DCF-EN mechanism.

Stability

The DCF-EN mechanism is a core-stable mechanism in terms of coalition rationality. The formal proof is given in the following text.

The classical core is the strongest of the solution concepts in coalition formation (Sandholm 1999). The core of a game is a set of payoff configurations $P(CS)$, where each $P(CL)$ is a vector of the payoff of a coalition CL in a coalition structure, CS, to the agents, and no subgroup is motivated to depart from the CS. The purpose of this concept is to maximize the value of coalition structure, V_{CS} (group rationality) and to motivate agents to stay with the coalition structure that maximizes the social welfare (individual rationality). Furthermore, every subgroup of agents in a coalition is better off staying within this coalition than forming a coalition of their own (coalition rationality).

The core concept is so strong that the core of a coalition game can be empty in many cases. Here, I relax the classical core concept to emphasize only the coalition rationality. After a coalition is formed, no subgroup of the coalition is willing to form its

own coalition. I define a coalition as stable in the core in terms of coalition rationality as the following:

Definition 4.2: Core-Stable in Terms of Coalition Rationality

A coalition CL is stable in the core in terms of coalition rationality iff for $\forall C \subset CL$ and $\forall a_i \in C$, $P_i(C) \leq P_i(CL)$ is always true.

Claim 4.1: The coalitions formed through the DCF-EN mechanism are stable in the core in terms of coalition rationality.

Proof: Each agent always tries to join the best coalitions that it can find by using the DCF-EN mechanism. The best coalition for an agent is the one that maximizes its own utility. Before an agent starts any coalition formation process, it will calculate the utilities of all possible coalitions to which it could belong. Then it sorts all these coalitions in descending order based on the corresponding utilities. The agent greedily proposes or accepts the best possible coalition that has not been rejected currently. Any coalition that has been accepted by all of its members must be the best coalition for all members that they can find. For each agent, if there are multiple coalitions with same utilities, it chooses the one with smallest size. Therefore, the value of the best coalition that an agent could join cannot be worse than the values of the coalitions composed of any subset of members in this best coalition. □

Lower Bound

Another important attribute of the DCF-EN mechanism is how good it is in terms of the value of resulting coalition structure.

If the optimal coalition structure is either all agents staying alone or all agents staying in the same coalition (grand coalition), the DCF-EN mechanism can find the

optimal solution after one proposal has been finished being negotiated. If during an entire coalition formation, a deadlock has not occurred, the resulting coalition structure of the DCF-EN mechanism is also an optimal solution.

Definition 4.3: Singleton Coalition Structure

The coalition structure is called a singleton coalition structure if it only includes agent subsets with a size of one.

For other cases, the DCF-EN mechanism can guarantee a lower bound because an agent will not join any coalition that results in a utility for the agent less than the utility that the agent can obtain by staying alone. Therefore, the lower bound of the value of resulting coalition structure by the DCF-EN mechanism is equal to the value of the singleton coalition structure. Is there a better lower bound that the DCF-EN mechanism can provide? Since the utility that each agent can obtain by joining a coalition is arbitrary, there is no better lower bound that the current DCF-EN mechanism can provide. However, given randomly generated utilities for each agent, the results of my experiments will show that the DCF-EN mechanism can always result in a non-trivial coalition structure, which is better than a coalition structure in which every agent stays alone. Thus, I believe that the DCF-EN mechanism can provide a better lower bound if it allows agents to adjust their strategies when they recognize that they will end up staying alone.

In order to guarantee that the resulting coalition structure is better than the singleton coalition structure, I extend the current DCF-EN mechanism to a repeated DCF-EN mechanism, which is called the RDCF-EN mechanism. The basic idea is to allow the original DCF-EN mechanism run multiple times. The first time, each agent runs the full original DCF-EN mechanism completely. If it ends up a coalition structure in which every

agent stays alone, all agents start the second run in which only coalitions of size two are considered. Starting from this turn, agents are no longer greedy. Each agent will accept the first coalition that results in better utility than the one it gets by staying alone. If some coalitions of size two are formed, the involved agents will stop their coalition formation procedure. All agents that cannot find a coalition to join at this turn will start the next turn, in which only possible coalitions with size three will be considered. Again, the agents who have found a coalition to join stop their coalition formation procedure and the agents who still stay alone will keep searching, and so on. The coalition formation ends when all agents find a coalition to join or the n^{th} run has been finished.

```
Calculate PCF
If the singleton coalition structure is not optimal
Then
    Start the DCF-EN negotiation procedure
    If the resulted coalition structure is not a singleton coalition structure
    Then
      Counter = 2
      While (Counter<=n)
          Build a PCF with possible coalitions of size Counter
          Start the DCF-EN negotiation procedure on new PCF
          If there is a coalition which can be formed
          Then
              Join the coalition
              Break
          Else
              Counter++
Else
    End coalition formation
```

FIGURE 4.11. Agent algorithm in the Repeated DCF-EN Mechanism

Figure 4.11 depicts the formal description of the RDCF-EN mechanism. The RDCF-EN mechanism can provide a better lower bound by which the resulting_coalition structure is always better than the one in which all agents stay alone, if such coalition structures exist.

Claim 4.2: If the singleton coalition structure is not the optimal solution of coalition formation, the RDCF-EN mechanism can always find a coalition structure that is better than singleton coalition structure in the sense that the value of the resulting coalition structure is larger than the value of the singleton coalition structure.

Proof: If the singleton coalition structure is the optimal solution of coalition formation, the RDCF-EN mechanism can find it right after it finishes calculating the possible coalition proposals because there do not exist any possible coalition proposals for every agent. The best coalition for each agent is to stay alone.

If the singleton coalition structure is not the optimal solution of coalition formation, there must exist some coalitions that result in better utilities for some agents. The size of those coalitions should be larger than one. If during the first turn, the DCF-EN mechanism could not find any of such coalitions, from the second turn, agents will accept any coalition that results in better utility. Since the RDCF-EN mechanism eventually enumerates all sizes of possible coalitions, it will form at least one coalition that results in better utilities for involved agents than they stays alone. Therefore, the value of the resulting coalition structure is always better than the singleton coalition structure. □

Besides providing better lower bound than the DCF-EN mechanism, the RDCF-EN mechanism does not increase the computational complexity significantly. In the worst case,

each agent only has $2 \cdot 2^{k-1}$ negotiation deals to be processed. Also, in the worst case, the total number of messages for negotiation in coalition formation is $2 \cdot 3k \cdot 2^{k-1}$.

SUMMARY

In this chapter, I presented a coalition formation mechanism (DCF-EN) that achieves decentralization through explicit negotiation among self-interested agents. Each agent makes its own decisions about whether or not to join a possible coalition. Through a properly designed multiparty negotiation protocol, the negotiation mechanism allows automated multilateral negotiation among self-interested agents who have symmetric authority (*i.e.*, no mediator exists and agents are peers) in a multiagent system.

The resulting coalitions are stable in the core in terms of coalition rationality. Compared with the centralized approaches, this mechanism significantly reduces the complexity of coalition formation processes. In the next two chapters, I apply the DCF-EN mechanism to two different applications for forming coalitions through explicit negotiation. The DCF-EN mechanism can provide a trivial lower bound for the value of the resulting coalition structure that is the value of the singleton coalition structure. I extend the DCF-EN mechanism to the RDCF-EN mechanism that can always guarantee to find a better coalition structure than the singleton coalition structure, if it exists.

CHAPTER V

FORMING RESOURCE SHARING COALITIONS THROUGH DCF-EN MECHANISM

In Chapter IV, I proposed a decentralized coalition formation (DCF-EN) mechanism to enable self-interested agents to form coalitions through explicit multiparty negotiation. In this chapter, I will show the experimental results of applying the DCF-EN mechanism to forming resource sharing coalitions in computational grids.

In computational grids, the participating agents are organizations that own a large amount of computational resources. Even so, these organizations sometimes need extra computational capacity to satisfy their computational requirements (*i.e*. at peak times). Meanwhile, most of their resources are idle at other times. The resource sharing mechanisms in existing computational grids consist of agreements made through negotiation among human representatives belonging to these organizations. Afterwards, if there are internal or external changes (*e.g*. hardware or software upgrades), the resource sharing agreements have to be changed by additional negotiations. By applying the DCF-EN mechanism, autonomous agents can execute these negotiation processes.

FORMING RESOURCE SHARING COALITIONS IN COMPUTATIONAL GRIDS

Applying the DCF-EN mechanism to computational grids is essential for automatically making resource sharing agreements among different organizations. The basic idea is to build resource sharing coalitions through multilateral negotiation among

self-interested agents. Resources owned by all members in a formed coalition are inter-connected via a network and can be accessed by every member.

A key step in the DCF-EN mechanism is that each agent needs to compute the utilities of all its possible coalitions. The utility for an agent of joining a resource sharing coalition should reflect the benefit obtained from the coalition by the agent. Previously, it was assumed that a general currency existed (denoted by grid dollars (Buyya 2002)) for expressing the cost of using a computational resource. However, it is unclear how to consistently map the values of different type of resource usages to different amounts of grid dollars. Therefore, I propose a task-oriented mechanism for measuring the value of resource usage in computational grids.

Economic Value of Computational Resource Usage

Let us start from a simple observation. Suppose there are three processors P_1, P_2, and P_3 that have different speeds from the highest to the lowest respectively. Here, I do not specify what the exact meaning of the speed of a CPU is. It could be measured by MIPs, clock rates, or any other kind of standard units. Given an identical job, these three processors would finish it in different amounts of time. Figure 5.1 shows the performance of each processor, assuming all other conditions are the same (*e.g.* same amount of RAM associated with each processor). The equivalent performance line depicts the fact that these three processors P_1, P_2, and P_3 finish an identical job within H_1, H_2, and H_3 CPU hours respectively.

If a resource user gives the same job to P_1, P_2, and P_3, how should the processor owner charge the user for using different processors? If all processors can satisfy the deadline of a job, the resource user would prefer not to pay extra for using P_1. But if the deadline of the job is tight, he may be willing to pay more for using P_1. Therefore, in order to set proper prices for using P_1, P_2, and P_3 to execute an identical job, the resource suppliers need to consider both the different capabilities of the processors and the users' performance preferences.

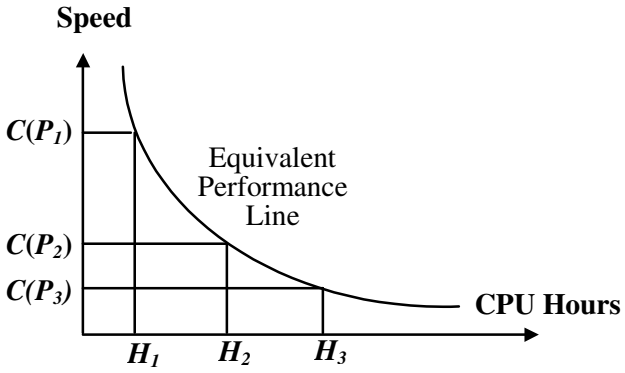

FIGURE 5.1. Equivalent Performance by Different Processors

Modeling Resource Capabilities

For simplicity, I consider time constraints as the only performance requirement for modeling the capabilities of computational resources (*e.g*. CPU, storage, bandwidth *etc*.). I define the capability of a group of heterogeneous resources as the following:

Definition 5.1: Let K be a set of tasks, $\{k_1, \ldots, k_n\}$, with a given duration, D (which is the total amount of time for accomplishing all tasks in K), and a group of resources, $G = \{R_1, \ldots, R_m\}$. The capabilities of a group of resources in G for executing the tasks in K is

denoted by *CanGroup*(G, K, D). *CanGroup*(G, K, D) is true if and only if the resources in G can finish all tasks in K within duration D.

Pragmatically, to determine if G is capable, there needs to be some scheduling algorithms for resources in G that can schedule the tasks in K properly so that the resources in G can accomplish them within duration D. The quality of the scheduling algorithm has a strong impact on the practical assessment of the capabilities of a group of resources (He and Ioerger 2003). The scheduling algorithm used for assessment should be transparent to resource users. This definition encapsulates the physical differentiation of resources to users. They allow users to ignore the different physical capabilities of resources and only consider their own performance preferences.

Economic Value of Resource Usage

From a user's perspective, regardless of the type of resources that are provided to execute a task, the economic values of using these resources are equivalent if they can accomplish the task while satisfying the same performance requirements. The following claim addresses the equivalent values of the usages of two groups of resources:

Claim 5.1: Let K a set of tasks $K = \{k_1, \ldots, k_n\}$ with duration D and groups, G_1 and G_2, with resources, $\{R_{11}, \ldots, R_{m1}\}$ and $\{R_{11}, \ldots, R_{r1}\}$, respectively. The economic value of using G_1 to execute the tasks in K is denoted by $V_1(K)$. $V_1(K) = V_2(K)$ if and only if both *CanGroup*(G_1, K, D) and *CanGroup*(G_2, K, D) are true. In other words, if *CanGroup*(G_1, K, D) is true and *CanGroup*(G_2, K, D) is not true, $V_1(K) > V_2(K)$; if *CanGroup*(G_1, K, D) is not true and *CanGroup*(G_2, K, D) is true, $V_1(K) < V_2(K)$.

Note that no agent in a computational grid has the power to set the true economic value of using a resource. The value is determined by the interaction among the resource

users and suppliers. Whether the established economic value of the usage of a resource is stable depends on the relationship between the supply of resources and the demand (Mas-Colell *et al*. 1995). The value of resource usage is not the price of using the resource, but a fair price should reflect the real value of using the resource.

Based upon the above analyses, any mechanism for measuring the economic value of a computational resource usage should obey the following principles:

- The economic value of using a resource is determined by the task executed by the resource.
- The values of using two groups of resources relative to a given set of tasks are equal if they can accomplish the set of tasks with satisfying the same performance requirements.
- The real economic value of using a resource is established through the interactions among agents in computational grids.

Task-Oriented Mechanism for Measuring the Economic Value of Resource Usage

Based upon the above capability model of computational resources, I establish a new mechanism for measuring the economic value of resource usage. Note that the economic value of a resource usage is not the intrinsic value of a resource itself, but the value of using the resource.

I use CPUs as an example to illustrate the mechanism. Referring to Figure 5.1, three processors with different speeds can be used to execute an identical task. The usage of each processor for the task is the same. Mathematically, “the usage of each processor” refers to the area of each rectangle in Figure 5.1. The formal definition of the usage of a processor to execute a computational task is given as follows:

Definition 5.2: Given a task, k, and a processor, P, the speed of P is $C(P)$. P needs H hours to finish k. The usage, $S(k)$, of P for executing k is the following:

$$S(k) = C(P) \times H$$

This definition reflects the amount of processor usage to execute a task no matter what kind of processors are used. The following claim is obviously true:

Claim 5.2: $S_x(k) = S_y(k)$ for given two processors P_x and P_y with different speeds.

Based on this claim, a mechanism can be established to translate the usage of CPUs with different speeds to a common measurement. The idea is to establish a standard speed and convert real CPU usage to the usage of a virtual CPU with the standard speed. Hence there are two directions to convert the CPU usage of a task, one is changing the duration and the other is changing number of CPUs with the standard speed.

Users in computational grids generally expect to finish their tasks as soon as possible. Given a task with certain duration, I can measure the usage of CPU for executing the task by calculating how many standard CPUs should be used to execute the task while satisfying the time constraints given by the user. I also define a standard time unit to measure the expected duration of a task. Thus, the definition of the usage of processors to execute a computational task is modified as follows:

Definition 5.3: Given a task, k, with a certain expected duration, D, the standard CPU speed is $C(P_s)$ and the standard time unit is D_s. $D = m \times D_s$, where m is the number of standard time units that D includes. In order to finish k within D, there must be at least n processors with speed $C(P_s)$ working simultaneously (assume k can be divided into n subtasks evenly) or a processor with speed $n \times C(P_s)$. The usage $S(k)$ of CPUs for executing k is the following:

$$S(k) = n \times C(P_s) \times m \times D_s$$

This definition implies that the CPU usage of any task can be measured through a standard speed and a standard time unit. The standard unit of CPU usage is given by the following equation:

$$S_s = C(P_s) \times D_s$$

Therefore, the CPU usage for executing a task can be measured through S_s by changing the number of CPUs with the standard speed or the number of the standard time units to finish a task. Thus,

$$S(k) = n \times m \times S_s$$

If the economic value of S_s is V_s, it is easy to calculate the corresponding economic value $V(k)$ of CPU usage for executing the task.

$$V(k) = n \times m \times V_s$$

However, it does not reflect the common sense that the CPU usage for executing a task in a shorter duration might have higher value than the one for executing the same task with a longer duration. In order to account for this,

$$V(k) = (1+\lambda_1 n) \times (1+\lambda_2 m) \times V_s$$

where, n and m refer to the increasing number of CPUs with the standard speed and the number of the standard time units respectively. The coefficients λ_1 and λ_2 imply that changing the number of CPUs with standard speed and the number of the standard time units to finish the same task could result in different economic values of CPU usage. If $\lambda_1 \neq \lambda_2$, then changing the number of CPUs with standard speed and the number of the standard time units to finish the same task causes different economic values of CPU usage.

Now, a task-oriented mechanism for measuring the economic value of computational resource usage for executing a task has been established.

Forming Resource Sharing Coalitions in Computational Grids

Agents in a computational grid generally have peak workloads at different times, so they can potentially utilize others' resources at their idle time. Thus, I need to define the relationship between the workload of each agent and time (Tulga and Sheridan 1980; O'Donnell and Eggemeier 1986; Tsang and Wilson 1997). It is called a workload distribution function.

Workload Distribution Function

Based on the measure of resource usage in the last section, I can define the computational capability of an agent (representing the computational resources of an organization on the grid) by converting all its CPUs with different speeds to the CPUs with the standard speed. The entire CPU capacity of a group of CPUs is defined as follows:

Definition 5.4: Let U be a group of processors, $\{P_1, \ldots, P_m\}$, where different P_i in U could have different speeds. The standard CPU speed is $C(P_s)$. The speed $C(P_i)$ of P_i is equal to $n_i \times C(P_s)$. The entire CPU capacity $C(U)$ of a group of processors in U within time interval $[t_1, t_2]$ is:

$$C(U) = \sum_{i=1}^{m} n_i \times C(P_s) \times | t_1 - t_2 |$$

Hence, the workload of an agent at a certain time point is decided by how many standard CPU P_ss are needed to run its tasks at that time.

Definition 5.5: The workload of an agent at certain time point t is defined as $m \times C(P_s)$, if the agent needs m standard CPU P_ss to run its tasks at time t.

Definition 5.6: The workload distribution function is defined as a function of time, $w(t)$: $T \rightarrow R$, that is the workload of the agent at time t (T is the set of time points).

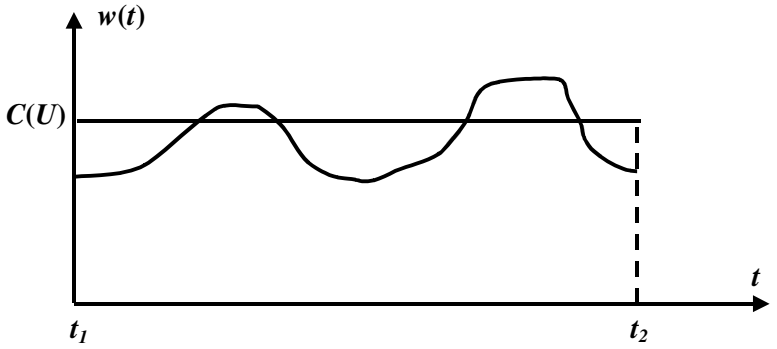

FIGURE 5.2. Workload Distribution Function

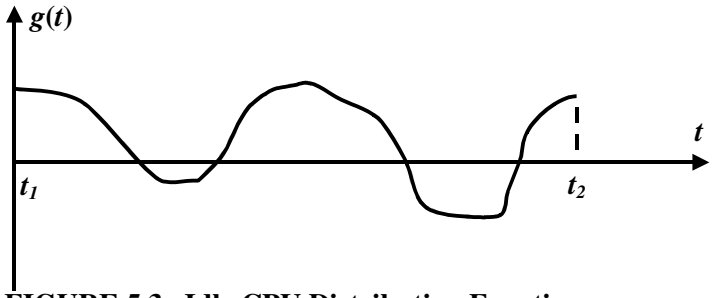

FIGURE 5.3. Idle CPU Distribution Function

The workload at a certain moment in a workload distribution function is larger than the entire computational capability of an agent because the tasks require more resources than it has available. Figure 5.2 shows an example of workload distribution function. In this work, it is assumed that the workload distribution function is known (or can be estimated

ahead of time). So negotiation can be done prior to execution of any tasks. (The problem could be more difficult if future workloads were unknown, and agents had to adapt and adjust cooperation dynamically.)

To evaluate the cooperation among agents, an idle resource distribution function also needs to be defined because resource sharing in grids is mainly related to how much idle resource capacity is available for involved agents.

Definition 5.7: Given a workload distribution function $w(t)$ of an agent within time interval $[t_1, t_2]$ and the entire CPU capacity of the agent is $C(U)$. The idle CPU distribution function $g(t)$ is defined as:

$$g(t) = C(U) - w(t), \quad t \in [t_1, t_2]$$

Figure 5.3 shows an example of $g(t)$ when $w(t)$ is given by Figure 5.2. When $g(t)>0$, the agent has idle CPUs. When $g(t)<0$, the agent is overloaded. Agents are self-interested in computational grids. Considering only the idle resource distribution is not enough to form a resource-sharing coalition. Each agent needs to consider the economic value of contributing its idle resource to others and the cost of using others' idle resource.

Definition 5.8: Given the idle CPU distribution function $g(t)$ of agent a within time interval $[t_1, t_2]$, the amount C_R of CPU capacity required for agent a is:

$$C_R(a, g(t)) = \int_{t_2}^{t_1} g^-(t)$$

where, $g^-(t)$ is equal to $g(t)$ when $g(t) < 0$ and $g^-(t)$ is equal to 0 otherwise. The economic value V_R of using the idle CPUs of others to finish agent a's tasks is:

$$V_R(g(t)) = \frac{C_R(a, g(t))}{S_s} \times V_S$$

Definition 5.9: Given the idle CPU distribution function of agent a within a time interval, $[t_1, t_2]$, the amount C_I of CPU capacity of agent a is idle:

$$C_I(a,\ g(t)) = \int_{t_2}^{t_1} g^+(t)$$

where, $g^+(t)$ is equal to $g(t)$ when $g(t) > 0$ and $g^+(t)$ is equal to 0 otherwise. The economic value V_I of agent a's idle CPUs is:

$$V_I(g(t)) = \frac{C_I(a,\ g(t))}{S_s} \times V_S$$

Based on the above analyses, the formal problem definition of forming resource-sharing coalitions in a computational grid is as follows:

Definition 5.10: Resource Sharing Coalition Formation Problem

Let A be a set of agents in a computational grid, $\{a_1, a_2, \ \ldots, a_r\}$. Each agent a_i also has its CPU capacity $N_i \times C(P_s) \times | t_1 - t_2|$ within time interval $[t_1, t_2]$. Each agent a_i needs to run a set of tasks $K_i = \{ k_{i1}, k_{i2}, \ \ldots, k_{il} \}$ which generates its idle resource distribution function $g_i(t)$ within $[t_1, t_2]$. The objective is to form appropriate coalitions such that each agent a_i can have the optimal idle resource capacity exchanging with other agents in A within time interval $[t_1, t_2]$.

For each agent in A, the best exchange is to contribute a minimal amount of its own idle resource capacity and to obtain a maximal amount of idle resource capacity from a resource sharing coalition while finishing as many of its own tasks as possible. To determine the optimal idle resource capacity exchange for each agent, the utility function of an agent needs to be defined. The utility is composed of two parts. One is the difference between the resource capacity the agent obtains from the coalition and the resource capacity

it contributes to the coalition. The other is the economic value of finishing tasks by using the resource capacity it obtains.

Definition 5.11: Suppose that agent a_i obtains idle CPU capacity $C_{gain}(a_i, CL_j)$ by joining a resource coalition, CL_j, and it contributes its own idle CPU capacity, $C_{give}(a_i, CL_j)$. By obtaining $C_{gain}(a_i, CL_j)$, agent a_i can finish a subset of tasks K_i' in K_i while satisfying the time constraints. The utility, $U(a_i, CL_j)$, that agent a_i obtains by joining coalition CL_j is given by:

$$U(a_i, CL_j) = \frac{C_{gain}(a_i, CL_j) - C_{give}(a_i, CL_j) + S(K_i')}{S_s} \times V_S$$

```
Start Message Listener
FinishCF = false
AcceptCF = false
While true
    If receive a new message
    Then process the message by calling corresponding message handler
    If FinishCF = false
    Then If startCF=false
            Then Calculate all possible resource-sharing coalitions
                 Sort all possible coalitions in a descending order in PCF
                 StartCF = true
             Else If PCF=Φ and SCF=Φ
                 Then There does not exist possible coalition
                      FinishCF = true
                 Else If RCF=Φ
                      Then If SCF=Φ and PCF!=Φ
                           Then Send the current best possible coalition proposal in PCF
                                Assert the send-out CF into SCF
                                Remove it from PCF
                           Else If (SCF!=Φ and SCF∈ RCF) or (SCF=Φ and PCF.elementAt(0) ∈ RCF)
                                Then Send CFAccept message to the corresponding agent
                                     AcceptCF = true
```

FIGURE 5.4. Agent Algorithm for Resource Sharing Coalition Formation

where $S(K_i')$ is the amount of CPU capacity that agent a_i obtaining from CL_j for finishing tasks in K_i'. Each agent a_i uses this utility function to evaluate its decision on which coalitions in the grid it should join by maximizing its utility.

Agent Algorithm for Forming Resource Sharing Coalitions

After defining the utility function of each agent, I can now apply the DCF-EN mechanism to solve the problem of forming resource sharing coalitions in computational grids. Figure 5.4 gives the agent algorithm, in which *PCF*, *SCF* and *RCF* denote the possible coalition proposal set, sent-out coalition proposal set and received coalition proposal set, respectively. In the following, I present the experimental results that aim to show how well the DCF-EN mechanism performs in this application domain.

EXPERIMENTS

The main goal of my experiments is to evaluate how well the DCF-EN mechanism performs on the resource sharing coalition formation in computational grids.

Experimental Objective

I mainly focus on the values of resulting coalition structures and the communication load caused by negotiation among agents. As for the value of resulting coalition structure, I will show that the average lower bound of the DCF-EN mechanism is better than the value of the singleton coalition structure (the lower bound was established in Chapter IV). Since the DCF-EN mechanism uses a time-out strategy to detect and break deadlocks, Student's

t-test will be used to compare two different time-out strategies: one that waits for a constant amount of time, and the other where the amount of wait time is proportional to the difference between the utility of the current coalition proposal and that of the next proposal. As for the communication load, I will examine that the total number of messages that each agent receives during a coalition formation. In other words, I will test the following hypotheses:

- If there exist better coalition structures than a singleton coalition structure, through the DCF-EN mechanism, the value of a resulting coalition structure will be higher than the value of the singleton coalition structure.
- The communication load caused by the negotiation among agents is much lower than the one in the worst case. (I believe that this hypothesis is valid because agents prefer smaller size of coalitions if possible coalitions result in the same utility, and whenever a coalition is formed, all members in that coalition will exit the negotiation process.)
- By using proportional time-out strategies, the DCF-EN mechanism will generate better values of resulting coalition structures. Intuitively, I believe that associating wait time with utilities of the corresponding coalition proposals is a better strategy, compared to waiting for constant amount of time for all possible coalition proposals. Here, a better strategy is a strategy that results in higher coalition structure value or causing less communication load.

In real computational resource sharing applications, different workload distributions of agents will result in different strategies for resource sharing. If an agent always has a heavy workload or mostly has a nearly full workload, it could be hard for this agent find a partner to share resources, as it cannot provide much help to others. If an agent always has a

low workload at most time and it is only overloaded occasionally, the organization could easily find a partner as it can help others a lot. However, the agent might choose not to help if it does not see any benefit to itself. For example, a restaurant web server might be only busy at lunch or dinnertime and be idle at most other time in a day. If an agent always has a regular workload distribution in the sense that it is sometimes overloaded and has idle capacity at other times, then the agent may easily find partners because it is able provide help to others when it has idle capacity and also needs help from others when it is overloaded.

Since the utilities of possible coalition proposals are determined by the workload distributions in resource sharing coalition formation, I need to examine whether different workload distributions will result in different coalition structures and different communication load. Hence, additional hypotheses need to be tested:

- Agents with nearly full workload distribution are more likely to choose not to join any coalition than agents with the other two types of workload distributions.
- Different workload distributions will cause different communication load.

Experimental Settings

I consider 10 time units for each experiment. For simplicity, instead of randomly generating workload distributions, I directly generate idle resource distribution for resource sharing coalition formation. Since I need to test how different workload distributions affect coalition formation, I design three types of idle resource distributions: FULL, SPIKE and TRIANGULAR.

FULL idle resource distribution represents the situation in which an agent has a nearly full workload at most time (*i.e.*, has a low utilizable idle resource capacity). SPIKE idle resource distribution implies the situation in which an agent is overloaded for short durations and has idle capacity at most time. TRIANGULAR implies the situation in which an agent is overloaded at some time and has idle capacity at other times.

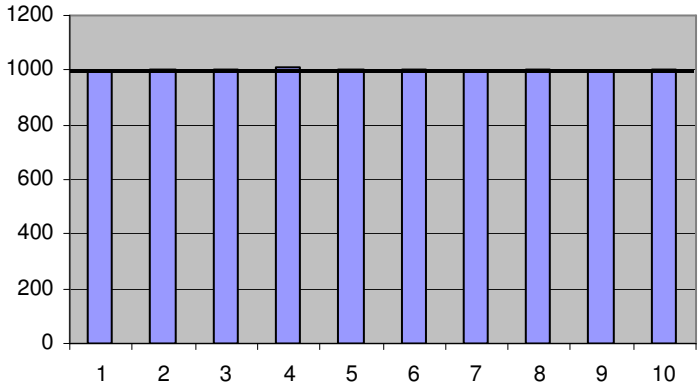

FIGURE 5.5. FULL Workload Distribution

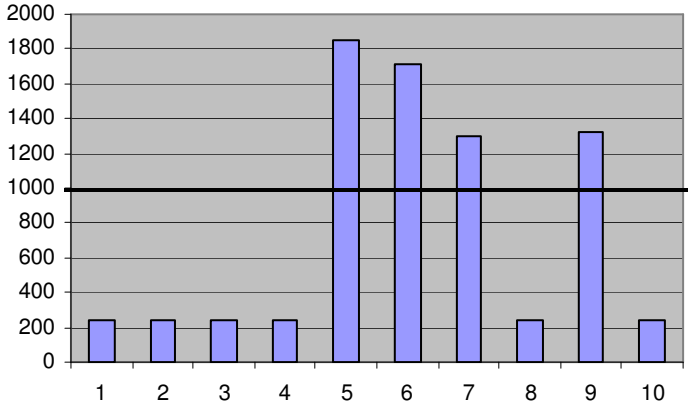

FIGURE 5.6. SPIKE Workload Distribution

In order to let the values of the resulting coalition structures be comparable, the idle capacity is generated randomly using a uniform distribution from 0 to 10 for FULL idle resource distribution; from 0 to 1000 for both SPIKE and TRIANGULAR distribution, assuming the maximal capacity of each agent is 1000. Figures 5.5 to 5.7 illustrate the corresponding workload distributions of these three types of idle resource distributions.

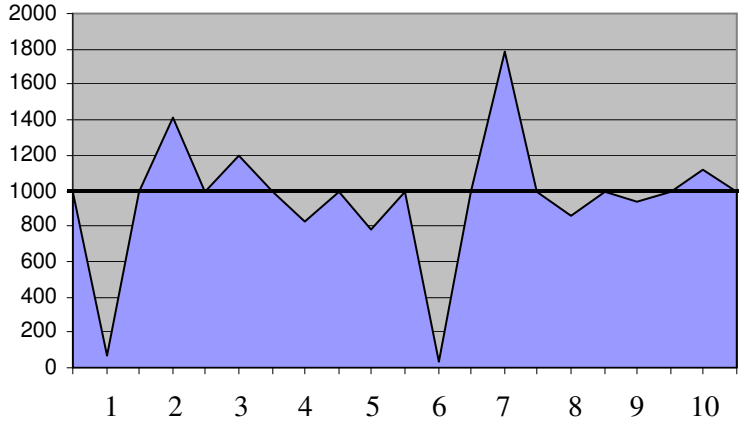

FIGURE 5.7. TRIANGULAR Workload Distribution

I chose 5 agents for each experiment. For each type of idle resource distribution and each type of time-out strategy, I ran 20 experiments. Hence, there are a total of 120 experiments. Table 5.1 lists all values that I collected from each experiment. U_{CL} is defined as $U_{CL} = -C_P + C_{NO}$ and U_a is defined as $U_a = -C_P$, where C_P is the total amount of positive idle capacity that the agent owns and C_{NO} is the total amount of idle capacity that the agent obtains from other agents. The basic idea is that idle capacity owned by the agent itself decreases its utility and idle capacity obtained from others increases its utility.

TABLE 5.1. Values Collected from Experiments

Values	Description
CS	The resulting coalition structure
U_{CL}	The utility that each agent obtains by joining the coalition it chooses
U_a	The utility that each agent obtains without joining any coalition
numMsgs[]	Numbers of messages that each agent receives for coalition formation

TABLE 5.2. Values Needed to be Computed

Values	Description
V_{CS}	The value of the resulting coalition structure, which is equal to the sum of the utilities that all agents can obtain in the coalition structure
V_a	The value of the singleton coalition, which is equal to the sum of the utilities that all agents obtain without joining any coalition
V_{CL-a}	The difference between the value of the resulting coalition structure and the singleton coalition
N_S	The number of agents who still stay alone after coalition formation
A_{NM}	The average number of messages that an agent receives for coalition formation

Based on the data that I collected from each experiment, I computed the following values in Table 5.2.

Experimental Results

In order to examine my hypotheses, V_{CL-a}, A_{NM} and N_S are three main values that I am interested in. Table 5.3 lists the experimental results by time-out strategy of proportional wait time. PT_Full, PT_Spike and PT_Triangular refer to experiments with FULL, SPIKE and TRIANGULAR idle resource distribution respectively. Table 5.4 lists the experimental results when the time-out strategy is to wait for constant amount of time.

CT_Full, CT_Spike and CT_Triangular refer to experiments with FULL, SPIKE and TRIANGULAR idle resource distribution respectively.

TABLE 5.3. Experimental Results for Proportional Wait Time

	PT_Full			PT_Spike			PT_Triangular		
	A_{NM}	$V_{CL\text{-}a}$	N_S	A_{NM}	$V_{CL\text{-}a}$	N_S	A_{NM}	$V_{CL\text{-}a}$	N_S
Mean	17.42	28.53	2.9	25.26	3887.05	2.25	29.1	2527.64	1.65
StdV	5.06	18.88	1.17	6.26	2371.41	0.97	3.18	1469.92	1.31

TABLE 5.4. Experimental Results for Constant Wait Time

	CT_Full			CT_Spike			CT_Triangular		
	A_{NM}	$V_{CL\text{-}a}$	N_S	A_{NM}	$V_{CL\text{-}a}$	N_S	A_{NM}	$V_{CL\text{-}a}$	N_S
Mean	20.84	31.4	2.9	23.75	3967.35	2	28.05	2439.6	1.55
StdV	6.21	25.98	0.97	7.51	2000.88	0.97	6.24	1275.27	1.1

I ran 120 experiments with 5 agents. Only 7 of them resulted in singleton coalition structures. 6 of these 7 experiments are based on FULL idle resource distribution. Therefore, the DCF-EN mechanism can generally result in coalition structures that are better than singleton coalition structure. Based on the experimental results, the following hypothesis is true:

If there exist better coalition structures than a singleton coalition structure, through the DCF-EN mechanism, the value of a resulting coalition structure will be higher that the value of the singleton coalition structure.

The second conclusion that I can draw is that the real communication load is much less than the theoretic worst case. For 5 agents, in the worst case, there could be around 240

messages for each agent. In my 120 experiments, the highest number of messages that an individual agent receives is 43. The mean of the numbers of messages that each agent receives during coalition formation is from 18 to 30. Therefore, the following hypothesis is also true:

The resulting communication load caused by the negotiation among agents is much lower than the one in the worst case.

In order to compare two different time-out strategies for detecting and breaking deadlocks during coalition formation, I did Student's t-test[5] to test the values of resulting coalition structures with the same type of idle resource distribution. In my t-test, the values generated by PT is group A and values generated by CT is group B. PT refers to the time-out strategy in which the wait time is proportional to the utility difference between the current possible coalition and the next possible coalition. CT refers to the time-out strategy by which the wait time is a constant.

FULL (A-PT, B-CT)
The results of an unpaired t-test

---

t = -0.399
sdev = 22.7
$\Delta \cong -2.9$
degrees of freedom = 38
The probability of this result, assuming the null hypothesis, is 0.69

FIGURE 5.8. T-Test for FULL Idle Resource Distribution

[5] The t-test results are generated at http://www.physics.csbsju.edu/stats/t-test_bulk_form.html

My hypothesis is that the proportional wait time strategy should outperform the constant wait time strategy in the sense that the value of resulting coalition structure should be higher. Therefore, the hypothesis of the t-test is that the mean of group A is larger than the mean of group B.

SPIKE (A-PT, B-CT)
The results of an unpaired t-test

---

t=-0.116
sdev= 0.219E+04
$\Delta \cong$ -100.3
degrees of freedom = 38
The probability of this result, assuming the null hypothesis, is 0.91

FIGURE 5.9. T-Test for SPIKE Idle Resource Distribution

TRIANGULAR (A-PT, B-CT)
The results of an unpaired t-test

---

t= 0.202
sdev= 0.138E+04
$\Delta \cong$ 88.04
degrees of freedom = 38
The probability of this result, assuming the null hypothesis, is 0.84

FIGURE 5.10. T-Test for TRIANGULAR Idle Resource Distribution

Figures 5.8 to 5.10 show the results of unpaired[6] t-tests for FULL, SPIKE and TRIANGULAR idle resource distribution respectively. In the t-test results, for both FULL and SPIKE idle resource distribution, the t values are all near zero, *i.e.*, the mean for PT is not larger than the one for CT. Only the t value for the TRIANGULAR idle resource

[6] I use unpaired t-tests because we want to see the significance of the difference between the means of two independent groups of results.

distribution is positive. The *t* values for all idle resource distributions implies that the probabilities of the corresponding null hypotheses (*i.e.*, there is no big difference between the mean for PT and the mean for CT.) being true is high, 0.69, 0.91 and 0.84 for FULL, SPIKE and TRIANGULAR respectively.

The results of the t-tests do not support the hypothesis by which the performance of PT is better than that of CT. To further evaluate the t-test results, I need to test whether the DCF-EN mechanism is sensitive to two important parameters, c_d (the amount of time that an agent needs to wait to detect a deadlock) and c_b (the maximal amount of time that an agent is willing to wait to join a coalition except the amount of time it needs to detect deadlocks). In the above experiments, I let $c_d = 2^n c_n$, where c_n is the amount of time required to process a possible coalition proposal without deadlock. In my experiments, c_n is estimated manually through prior empirical experiments. I also let $c_b = c_d$.

TABLE 5.5. Experimental Results with Different c_b and c_d

	ST_Triangular			PT_Triangular			LT_Triangular		
	A_{NM}	$V_{CL\text{-}a}$	N_S	A_{NM}	$V_{CL\text{-}a}$	N_S	A_{NM}	$V_{CL\text{-}a}$	N_S
Mean	27.22	2316.48	1.9	29.1	2527.64	1.65	28.78	2261.75	2.05
StdV	4.42	1345	1.3	3.18	1469.92	1.31	2.97	1275.97	1.15

In order to test whether the DCF-EN mechanism is sensitive to c_b and c_d, I conducted the following experiments: I ran two groups of experiments with proportional wait time strategy and TRIANGULAR idle resource distribution. In one group, I reduced c_d by half and doubled it in another group. I ran 20 experiments for each group. Let ST denote

the experiments with shorter c_b and c_d. LT denotes the experiments with longer c_b and c_d. Table 5.5 shows the experimental results.

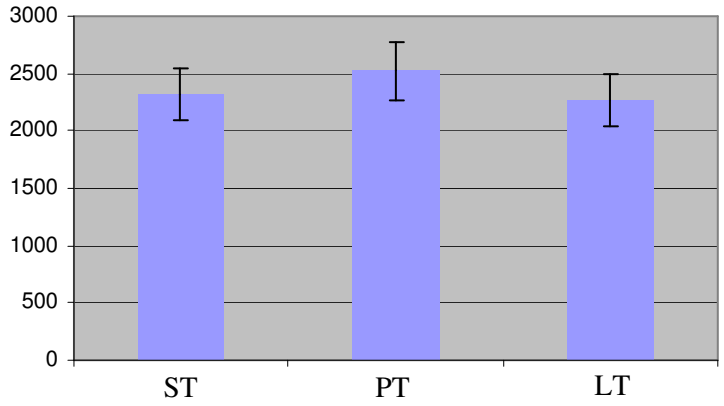

FIGURE 5.11. Error Bar of $V_{CL\text{-}a}$

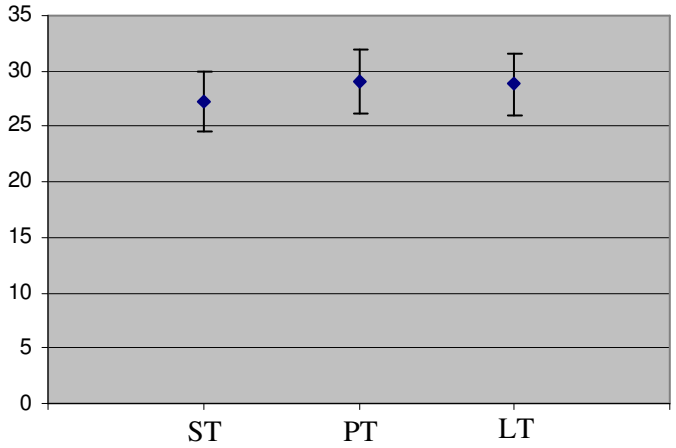

FIGURE 5.12. Error Bar of A_{NM} with Different c_b and c_d

Figure 5.11 shows that the error bars of $V_{CL\text{-}a}$ by experiment ST, LT and PT which is the experiment with original c_b and c_d. Figure 5.12 shows that the error bars of A_{NM} by experiment ST, LT and PT. Figure 5.13 shows the error bars of the average number (N_S) of agents who choose not join any coalition after coalition formation. The experimental results

do not show a big difference among ST, LT and PT. Therefore, the previous t-test results are not very sensitive to parameters c_b and c_d.

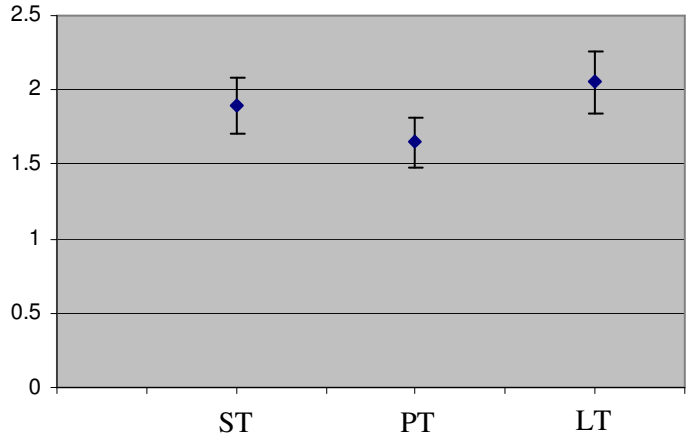

FIGURE 5.13. Error Bar of N_S with Different c_b and c_d

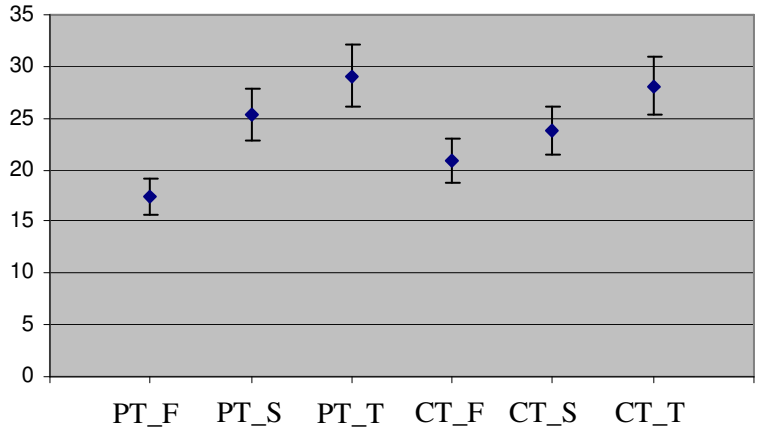

FIGURE 5.14. Error Bar of A_{NM} with Different Workload Distributions

Another observation is that the different types of idle resource distribution cause different communication load. Figure 5.14 shows the error bar of average number of messages for each agent in experiments with different types of idle resource distribution.

The error bars of the average number of received messages with the same type of idle resource distribution fall into a similar range no matter what kind of time-out strategy used. FULL idle resource distribution causes the lowest communication load because agents have fewer chances to find partners. TRIANGULAR idle resource distribution results in the highest communication load as agents have more opportunities to find partners.

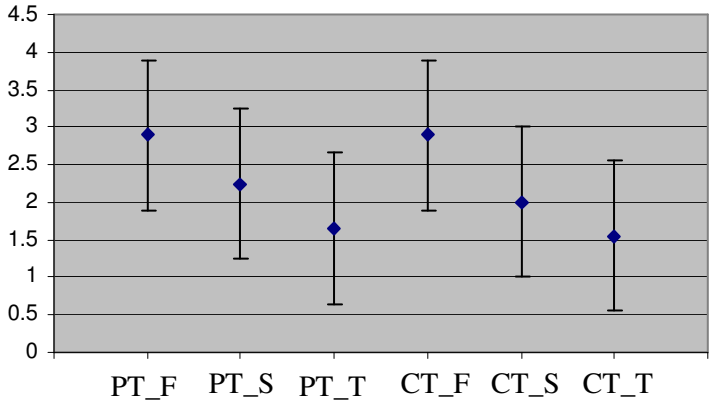

FIGURE 5.15: Error Bar of N_S with Different Workload Distributions

By examining the number of agents who choose not join any coalition after coalition formation, I found that FULL idle resource distribution produces the maximum agents who choose not join any coalition after coalition formation, and TRIANGULAR idle resource distribution results in minimum agents who choose not join any coalition after coalition formation. Figure 5.15 shows the error bars of the average number (N_S) of agents who stay alone after coalition formation with different idle resource distribution.

Since the different types of idle resource distribution implies different types of workload distribution, the following two hypotheses are also valid:

- *Different workload distributions will cause different communication load.*

- *Agents with nearly full workload distribution are more likely to choose not join any coalition than agents with the other two types of workload distributions.*

SUMMARY

In this chapter, I applied the DCF-EN mechanism to resource sharing coalition formation in computational grids. To compute the utility that an agent can obtain by joining a coalition, I developed a task-oriented mechanism to measure the economic value of computational resource usage. Based on this measurement, agents can compute the utilities of sharing others' idle resource capacity as well as the utilities of contributing their own idle resource capacity to other agents in a grid.

To examine how well the DCF-EN mechanism performs on resource sharing coalition formation, I designed experiments with three types of idle resource distribution, FULL, SPIKE and TRIANGULAR. The experimental results support the following hypotheses:

- If there exists better coalition structures than a singleton coalition structure, through the DCF-EN mechanism, the value of the resulting coalition structure will be higher than the value of the singleton coalition structure.
- The communication load caused by the negotiation among agents is much lower than the one in the theoretic worst case. Different workload distributions will cause different communication load.
- Agents with nearly full workload distributions will have more chances to choose to not join any coalition than agents with other two types of workload distributions.

Before I ran the experiments, I expected that by using different time-out strategies, the DCF-EN mechanism would generate different values of resulting coalition structures. However, the t-test results showed that there was not much difference between the time-out strategy with proportional wait time and the time-out strategy with constant wait time. Overall, the experimental results show that the DCF-EN mechanism can generally generate better coalition structures than the singleton coalition structure (*i.e.*, working alone). The communication load of the DCF-EN mechanism is also practical.

CHAPTER VI

COMBINING BUNDLE SEARCH WITH BUYER COALITION FORMATION

The DCF-EN mechanism is not only applicable for establishing virtual communities for computational resource sharing, but it is also practical for other applications such as buyer coalition formation in electronic markets. Small buyers without bargaining power (Proter 1980) individually can form a coalition as a virtual big buyer (He and Ioerger 2004b, 2005a). As a result, sellers could be willing to give a greater discount to the whole group.

Combinatorial trade in electronic markets is becoming more and more important. In electronic markets, buyers are able to access an incredible amount of product information through the Internet. This advantage allows buyers to build better purchasing strategies to save costs. Forming buyer coalitions among small buyers is one such purchasing strategy that allows small buyers with little individual bargaining power to form a virtual big buyer that can obtain better discounts.

Another very interesting buyer strategy is called the "bundle search" that addresses the situation where a buyer needs to buy different goods as a bundle. A typical example is the travel package search problem (Chang *et al*. 2003). Because of the different retail prices and discount policies of different suppliers, different bundles result in different discounts. The problem is to find the optimal bundle that results in minimum cost. Actually, searching for the maximal discount of a buyer coalition can be viewed as a bundle search problem if the discount policies of sellers are based on the total cost to all buyers in the buyer

coalition. Under this kind of discount policy, it is valuable for buyers to use both bundle search and buyer coalition formation to obtain better discounts.

In this Chapter, I consider both combinatorial coalition formation and bundle search together when the discount policies of sellers depend on the total cost of all goods sold in each transaction. I apply the DCF-EN mechanism to solve the following purchasing problem:

Definition 6.1: A Purchasing Problem

Let $G = \{g_0, g_1, \ldots, g_{l-1}\}$ denote a collection of goods. Let B be a group of buyers, $\{b_0, b_1, \ldots, b_{m-1}\}$, where each of them has a shopping list denoted by vector $Q_i = (q_{i0}, q_{i1}, \ldots, q_{i,l-1})$, where q_{ik} refers to the quantity of each item, g_k, buyer b_i needs to buy ($i = 0, 1, .., m-1; k = 0, 1, .., l-1$). There is a set of sellers, $S = \{s_0, s_1, \ldots, s_{n-1}\}$, who can supply some or all of the goods in G. Each seller, s_j ($j = 0, 1, .., n-1$), has its own discount function $\delta_j(c)$: $\Re^+ \rightarrow \Re^+$, that is the discount a buyer obtains when the cost of his purchase from seller, s_j, is c. $P_j = (p_{j0}, p_{j1}, \ldots p_{j, n-1})$ is a retail price (per unit) vector for each seller, s_j. The cost of buyer, b_i is denoted by c_i: $c_i = \sum_j (\sum_{k=0}^{l-1} p_{jk} \times q_{ik} - \delta_j (\sum_{k=0}^{l-1} p_{jk} \times q_{ik}))$. If seller, s_j, has no good, g_k, available, $P_{jk} = 0$. The objective of the problem is to minimize the cost to each buyer in the buyer set, B, *i.e.* $min \sum_{i=0}^{m-1} c_i$.

Definition 6.2: Discount Ratio

The discount ratio is defined as the ratio of the discount to the corresponding cost: $r_d = \delta(c)/c$, where $\delta(c)$ is the amount of discount that a buyer can obtain by spending c dollars in one transaction. The discount ratio must have an upper bound in a real market, because sellers need to guarantee that their profit is positive. Hence, searching the

maximum discount could be interpreted as finding the highest discount ratio the buyers can obtain.

The discount ratio function may not be monotonic increasing along with the amount of cost. In other words, more cost will not necessarily result in a higher discount ratio. For example, J.C. Penney provides $10 off for purchases over $50, $15 off for purchases over $75 and $35 off for purchases over $150 on certain sale days. Figure 6.1 shows the corresponding discount ratio function.

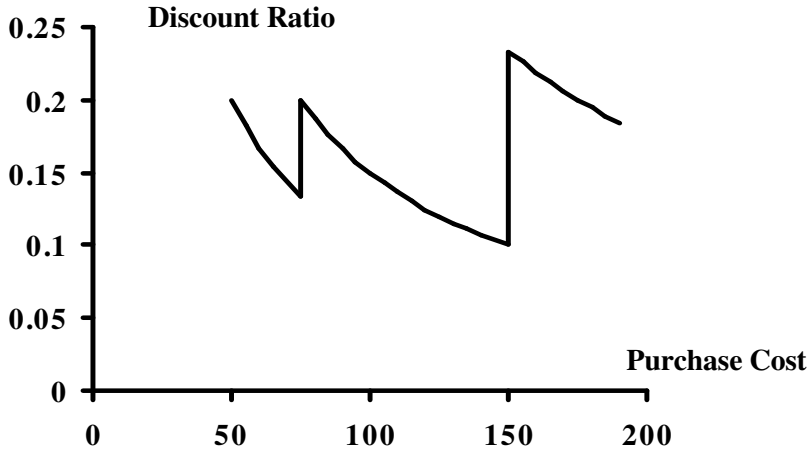

FIGURE 6.1. Discount Ratio Function

Definition 6.3: Buyer Utility

The utility of a buyer is defined as the difference between the discount he can obtain by shopping alone and the one he can get by using a purchase strategy based on buyer coalition formation: $u = d_{CL} - d_b$, where d_b is the discount that a buyer can obtain by shopping alone and d_{CL} is the discount that the buyer can obtain by shopping with partners. Each buyer tries to maximize its own utility.

Note that d_{CL} for each buyer depends on how the corresponding shopping group divides the total discount that it obtains. Assume that the discount of each buyer is proportional to the cost it contributes to the group, *i.e.*, for buyer b_i, $d^i_{CL} = \delta_j(c) \times \frac{c_i}{c}$ where c is the total cost of the group, c_i is the cost of buyer b_i and $\delta_j(c)$ is the total amount of discount the group obtains from seller s_j,

TRADITIONAL CENTRALIZED APPROACH

Traditionally, the above purchase problem can be solved by a centralized approach. Suppose there is a buyer leader who has all information about buyers and sellers. The buyer leader searches for the optimal strategies for buyers (Li and Sycara 2002). The simplest centralized approach is to enumerate all possible coalition structures of all goods that buyers need to purchase and find the optimal coalition structure that minimizes the costs of all buyers.

However, the computational cost of this approach is prohibitively expensive. The total number of all possible coalitions is $2^{NQ} - 1$. Sandholm *et al* (1999) have already proved that the total number of all possible coalition structures is $O(NQ^{NQ})$, which is so huge that not all coalition structures can be enumerated unless the number of all goods is extremely small (below 15 or so in practice). Also, for each coalition, an optimal bundle search needs to be done. The time complexity of the optimal bundle search is $O(NM)$ in the worst case, where M is the number of goods, and N is the number of sellers (Chang *et al*. 2003).

Another costly computation of this centralized approach is evaluating whether the resulting coalitions are stable in the core (Kahan and Rapoport 1984; Li and Sycara 2002). The reason to do this is because buyers will refuse to join a coalition that causes higher cost than the cost of joining other coalitions. A buyer_leader needs to calculate the total cost to each buyer in every coalition structure and determine whether the coalitions are stable (Li and Sycara 2002).

In real electronic markets, buyers are self-interested and geographically distributed. They make purchase decisions based on their local information and on minimizing their own cost. The incentive of buyers to join a buyer coalition is to obtain a greater discount than they would from purchasing individually. It is more realistic to let buyers make their own decisions (Shehory and Kraus 1999; Lerman and Shehory 2000) and form coalitions through negotiation than it is to set a coalition formation leader to evaluate the coalition value and distribute the payoff. I propose a distributed approach based on the DCF-EN mechanism for solving the purchasing problem. It is much more efficient and practical than the centralized approach for real applications.

A DISTRIBUTED APPROACH

My approach to solving this purchasing problem involves two steps. First, buyers do their individual bundle searches to find the optimal bundle for their own shopping lists. If the discount ratios obtained from the sellers involved in the optimal bundle are the maximal discount ratios that the sellers can offer, buyers do not have incentive to form or join any buyer coalition, since they cannot increase the amount of discount they can gain.

Otherwise, buyers start the second step by either searching for coalitions proposed by other buyers or proposing new coalitions to related buyers.

The reason for a buyer to do a bundle search first is that the discount resulting from the individual bundle search belongs to the buyer for sure. The possible discount obtained from joining a buyer coalition is uncertain because it depends on whether buyers can achieve a consensus. It is possible to miss the optimal result by separating the bundle search and the coalition formation into two steps, but it ensures that buyers obtain the discount at least as much as they can get individually.

The objective for buyers to join a coalition is to obtain a greater discount from one seller. I set an independent buyer club agent for each seller in *S*. If buyers are interested in joining the coalitions related to a specific seller, they register in its buyer club and obtain information about other buyers who need to join coalitions from the buyer club. Buyer club agents have nothing to do with the negotiation process among buyer agents except for providing the initial information about possible partners.

BUNDLE SEARCH PROBLEM

Compared with buyer coalition formation, the bundle search problem targets a single buyer who has a shopping list that includes a set of desired items.

Definition 6.4: Bundle Search Problem

let Q_i be a shopping vector ($q_{i0}, q_{i1}, \ldots, q_{i,l-1}$), where q_{ij} is equal to 1 or 0. Let S be a set of sellers $\{s_0, s_1, \ldots, s_{n-1}\}$, who can supply some or all of the goods in Q_i. Each seller s_j ($j = 0, 1, .., n-1$) has its own discount function $\delta_j(c)$: $R^+ \rightarrow R^+$. The problem is to find an optimal

purchasing strategy (*i.e.*, partitioning goods in Q_i by the targeting sellers in S) in the sense that it takes buyer b_i the minimal cost to purchase all goods with $q_{ij} = 1$ in Q_i.

The optimal algorithm for the bundle search problem is trivial. The idea is to enumerate all partitions of goods on the buyer's shopping list based on different combinations of sellers. The partition that results in minimal cost is the optimal bundle. The time complexity of this algorithm in the worst case is $O(N^M)$ (Chang *et al*. 2003). If the prices of items on the shopping list are not affected by adding more items, dynamic programming can be used to solve this problem with time complexity $O(N^2)$. However, this is not the case for the purchasing problem here. The price of each item is changed by the discount ratio that the buyer obtained. The discount ratio is a function of the total cost to a buyer. If a dynamic programming algorithm is used, when a new item is added, the total cost may change and the discount ratio may also change. In this case, the price of items that have already been calculated may change. Hence, the dynamic programming algorithm is not appropriate for this problem.

I have developed an efficient heuristic algorithm to solve this bundle search problem. It is called Maximal Gain Bundle Search (MGBS) algorithm. The algorithm is based on the following three heuristic rules:

Rule #1 Maximal Bundle: The problem of bundle search comes from the general economic situation where the more one spends with a single seller, the more discount one gets from that seller. I call the bundle purchase from one seller that provides the maximal discount the "Maximal Bundle". If the cost of a bundle with the maximal discount from every available seller is larger than the sum of the corresponding minimal retail prices for the goods in the

bundle, then it is not necessary to continue the bundle search. The buyer just needs to buy all goods in the bundle at the lowest retail prices.

Rule #2 Maximal Gain Ratio: If the costs of maximal bundles from multiple sellers are less than the sums of the corresponding minimal prices, the seller with the best "Maximal Gain Ratio" is picked as the candidate seller.

To define this term, I need to define the gain of each bundle purchase from one seller. In this paper, the gain of each bundle purchase is not defined by the amount of the discount. If the prices provided by a seller are too high, even if it gives a large discount, the purchase cost could still be very high. So, the gain of each bundle purchase is defined to be the difference between the final cost of this bundle purchase and the sum of the corresponding minimal prices. The gain ratio is defined to be the ratio of the gain of a bundle purchase to the sum of the corresponding minimal prices; so the maximal gain ratio is the ratio of the gain of the maximal bundle purchase to the sum of the corresponding minimal prices.

Suppose that there is a set of sellers, $Sb = (Sb_0, Sb_1, \ldots, Sb_k)$ for a bundle of goods, $Gb = (Gb_0, Gb_1, \ldots, Gb_k)$. The gain ratio $g(Gb, Sb)$ of the bundle of Sb is defined by the following equation:

$$g(Gb, Sb) = \frac{\sum P_{\min}^{Gb} - (\sum P_{Sb}^{Gb} - \sum D_{Sb}^{Gb})}{\sum P_{\min}^{Gb}}$$

$\sum P_{Sb}^{Gb}$ denotes the sum of the prices of all goods in Gb of the sellers in Sb. $\sum P_{\min}^{Gb}$ denotes the sum of the minimal prices to purchase all goods in Gb. denotes the sum of the discounts obtained from all of the sellers in Sb for purchasing goods in Gb as a bundle. Maximal gain ratio is defined as a ratio of the difference between the sum of minimal retail prices of the

bundle and the cost of the bundle after applying a discount to the sum of minimal retail prices of the bundle.

Begin MGBS (*G*, *S*, *Mp, Mr*)
  $|PV| = |G|$, $|P_{min}| = |G|$, $|SV_{min}| = |G|$
  (P_{min} is the minimal price vector and SV_{min} is the corresponding seller Vector)
  $MaxCost = \sum P^i_{min} - \sum \delta_i(P^i_{min})$
  For each seller S_j
    $MaximalBundle[j] = \sum Mp_j$
    $MaxDiscount[j] = \delta_j(MaximalBundle[j])$
    $BundleCost[j] = MaximalBundle[j] - MaxDiscount[j]$
  For all *BundleCost[j]*
    If All of *BundleCost[j]*s > *MaxCost[j]*
    Then $PV = SV_{min}$, $PC = MaxCost$
      Return *PV, PC*
    Else
      For each *BundleCost[j]*
        $MinBundleCost[j] = \delta_j^{-1}(MaxDisicount[j])$
        MinBundle[j] = FindMinBundle (*MinBundleCost[j], j, Mp*)
        $g(MinBundle[j], S_j) = (MaxSubCost[j]-BundleCost[j])/MaxSubCost[j]$
  Pick *MinBundle[k]* of seller S_k with argmax $g(MinBundle[k], S_k)$
  For each item *m* in *MinBundle[k]*
    $PV_m = S_k$
  $PC = PC + MinBundle[k]$ - discountFunction(*MinBundle[k]*, S_k)
  Set the entries of the corresponding rows of this *MinBundle[k]* to *0* in *Mp* and *Mr*
  If All entries in *Mr* are *0*
    Return *PV, PC*
  Else MGBS(*Mp, Mr*)
End MGBS (*Mp, Mr*).

Begin Procedure FindMinBundle (*MinBundleCost*, *j*, *Mp*)
 Fetch the column *j* in Mp to be the good vector GV_j of seller S_j
 Sort GV_j according to the price increasingly
 $s = |GV_j|$
 While (s>0)
   Remove the last element *L* in GV_j
   If the sum of prices in GV_j < *MinBundleCost*
   then $GV_j \leftarrow GV_j + L$
   $s \leftarrow s - 1$
 Return GV_j
 End FindMinBundle (*MinBundleCost*, *j*, *Mp*)

FIGURE 6.2. Maximal Gain Bundle Search Algorithm

Rule #3 Bundle Regression: Since the discount ratio may not be monotonic increasing, through the inverse function of the discount function, the minimal cost to get the same amount of discount of a maximal bundle can be found. Based on this minimal cost, I can search for the cheapest bundle purchase with the same amount of discount from this seller, and leave the other goods for another round of searching. Before calculating the maximal gain ratio, the maximal bundle for each seller should be refined to the "minimal bundle" from the seller with the same amount of discount as the maximal bundle. This rule provides a method to refine the search results already obtained from the two rules above. The heuristic goal here is to achieve a higher discount ratio for each partial bundle purchase.

```
DonePurchase = false
Run the MGBS algorithm to get the optimal bundle with a seller vector SV
For each s_i in SV
   If  The cost in the optimal bundle does not cause the seller to offer its highest discount ratio
  Then Send CFRequire to the corresponding buyer club agent
If  All sellers in SV do not need to send CFRequire
Then DonePurchase = true
Else While true
        If  receive a new message
        Then process the message by calling corresponding message handler (defined in Chapter IV)
        If  DonePurchase = false
        Then  If  startCF=false
                Then  Calculate all possible resource-sharing coalitions
                      Sort all possible coalitions in a descending order in PCF
                      StartCF = true
                Else  If PCF=Φ and SCF=Φ
                      Then There does not exist possible coalition
                           DonePurchase = true
                      Else If RCF=Φ
                           Then If SCF=Φ and PCF!=Φ
                                Then Send the current best possible coalition proposal in PCF
                                     Assert the send-out CF into SCF
                                     Remove it from PCF
                           Else If (SCF!=Φ and SCF∈ RCF) or (SCF=Φ and PCF.elementAt(0)∈ RCF)
                                Then Send CFAccept message to the corresponding agent
                                     AcceptCF = true
```

FIGURE 6.3. Buyer Agent Algorithm

The outputs of the bundle search for buyer b_i is a seller vector $SV_i = (sv_{i0}, sv_{i1}, \ldots, sv_{i,l-1})$, where sv_{ij} = null if $q_{ij} = 0$, otherwise sv_{ij} is equal to the corresponding seller's ID. With the seller vector, buyer b_i can calculate whether he has obtained the highest discount ratios from sellers in SV_i. Figure 6.2 gives the algorithm (He and Ioerger 2003).

COMBINING BUNDLE SEARCH AND BUYER COALITION FORMATION

Buyers start their coalition search based on the results of their bundle search in the first step. Each buyer only registers with the buyer clubs involved in the individual optimal bundle result. When buyers' bundle searches do not include a particular seller, or when they have already obtained the optimal discount ratio from a seller, they do not register with the corresponding buyer clubs. I apply the DCF-EN mechanism to the buyer coalition formation processes. The complete buyer agent algorithm is in Figure 6.3.

SIMULATION RESULTS

To evaluate the results of my solution to the purchase problem and the efficiency of the distributed coalition formation mechanism, I need to evaluate the costs of buyers using different purchasing strategies. The main goal of my simulation is to evaluate the average cost to each buyer and the total cost to all buyers with different purchasing strategies.

I will compare the cost to each buyer and the total cost to all buyers using the following four purchasing strategies:

- Purchasing each of the goods with the lowest price in the market;

- Combining the optimal bundle search and buyer coalition formation through the DCF-EN mechanism;
- Combing the MGBS bundle search and buyer coalition formation through the DCF-EN mechanism;
- The optimal solution of the purchasing problem (Since the computational cost is too high to run an optimal algorithm for the purchasing problem, to compare the results with the optimal results for a certain purchase problem, I use the lower bound of the optimal cost for a buyer, which is the sum of the minimal retail prices of all his goods with obtaining the highest discount ratio in the market. In real markets, it is impossible for buyers to obtain this lower bound cost).

TABLE 6.1. Parameters Used for Simulation

Parameters	Experimental Values
$N \times M$	3×3, 4×4, 5×5
Numbers of Buyers	3, 4, 5, 6, 7, 8, 9
Discount Function	$\delta_l(c) = \begin{cases} 10, \text{ if } 50 \le c < 100; \\ 20, \text{ if } 100 \le c < 150; \\ 35, \text{ if } 150 \le c < 200; \\ c \times 20\%, \text{ if } c \ge 200. \end{cases}$

Based on the purchasing problem definition, the input parameters and the discount function used in my simulation are given in Table 6.1, where G refers to the vector of goods and S refers to the vector of the corresponding sellers, $N = |G|$ and $M = |S|$. Indeed, the discount function defined in Table 6.1 is used by JCPenny, Dillards and Foleys *etc*. department store very often. Since, in a real market, the buyers with a small amount of

purchase cost are more likely join a buyer coalition, I did not use large numbers of items and sellers in my simulation.

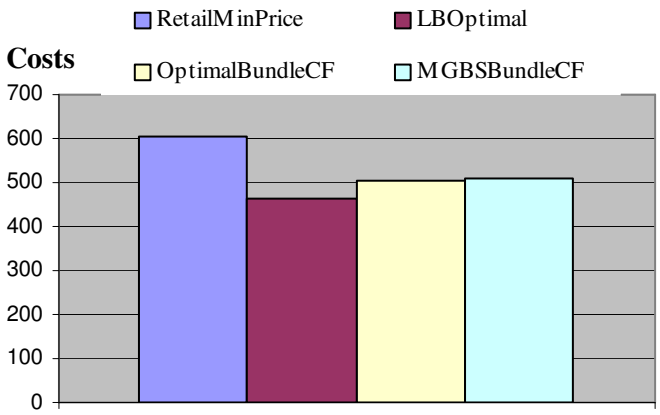

FIGURE 6.4. Total Cost to All Buyers Using Different Purchasing Strategies

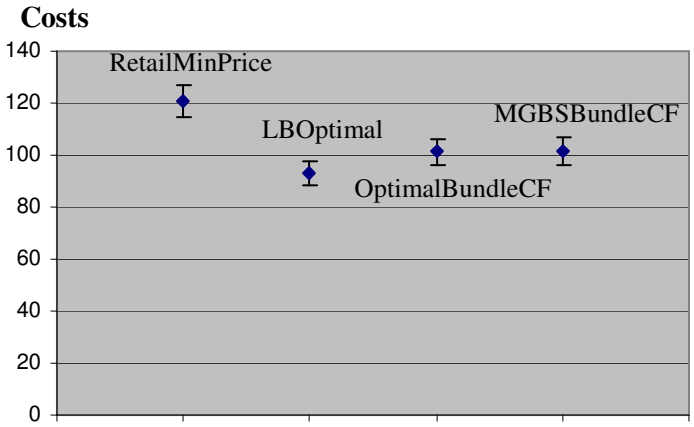

FIGURE 6.5. Error Bar of the Average Cost to Each Buyer

In real markets, sellers adjust retail prices very often. For example, Walmart and HEB change food prices every week. However, JCPenny does not adjust its discount policy very often. Thus, I assume that sellers have the same discount policies, but the retail prices offered by different sellers are different (I will test different discount policies later). I

generated the seed price of each goods from each seller randomly using a uniform distribution from 0 to 50. For the same goods, based on the corresponding seed price, I generated the price for each seller by adding a random number, which is generated using a uniform distribution form 0 to 10. Therefore, for same goods, the price difference between two sellers is less than 10.

A-LBOptimal, B-RetailMinPrice
The results of an unpaired t-test

---

t = -3.66
sdev = 24.0
$\Delta \cong 27.78$
degrees of freedom = 38
The probability of this result, assuming the null hypothesis, is 0.0008

FIGURE 6.6. T-Test for Comparing RetailMinPrice and LBOptimal

A-LBOptimal, B-OptimalBundleCF
The results of an unpaired t-test

---

t = -1.18
sdev= 21.9
$\Delta \cong -8.15$
degrees of freedom = 38
The probability of this result, assuming the null hypothesis, is 0.25

FIGURE 6.7. T-Test for Comparing OptimalBundleCF and LBOptimal

Given different combinations of $N \times M$, I ran the experiments for different number (3-9) of buyers. For each different number of buyers, I ran 20 experiments.

Figure 6.4 shows the mean of the total cost to all buyers using different purchasing strategies when $N \times M$ is 4×4 and the total number of buyers is 5. Figure 6.5 shows the error bars of the average cost of each buyer. RetailMinPrice denotes the cost to a buyer using the

strategy of only searching for the minimal retail price in the market for each item needed. OptimalBundleCF denotes the cost to a buyer of doing an optimal bundle search first and then trying to join buyer coalitions. MGBSBundleCF denotes the cost to a buyer of doing an MGBS bundle search first and then trying to join buyer coalitions. LBOptimal is the lower bound of the optimal cost that a buyer has to pay, which is equal to the sum of the minimal retail prices of all the buyer's goods with obtaining the highest discount ratio.

A-LBOptimal, B-MGBSBundleCF
The results of an unpaired t-test

---

t = -1.22
sdev = 22.0
$\Delta \cong$ -8.45
degrees of freedom = 38
The probability of this result, assuming the null hypothesis, is 0.23

FIGURE 6.8. T-Test for Comparing MGBSBundleCF and LBOptimal

Based on the above experimental results, both OptimalBundleCF and MGBSBundleCF are very close to the lower bound of the optimal value and much better than using the strategy of only searching for the minimal retail price in the market. For each buyer, the difference between using MGBSBundleCF and LBOptimal is 8.54, and the difference between using OptimalBundleCF and LBOptimal is 8.15**.** However, the difference between using RetailMinPrice and LBOptimal is 27.78. Figures 6.6 to 6.8 show the Student t-test results[7] of comparing RetailMinPrice and LBOptimal, OptimalBundleCF and LBOptimal, MGBSBundleCF and LBOptimal respectively. The t-tests results also support the above conclusion.

[7] The null hypothesis is that the mean of group A is not less than the mean of group B.

Figure 6.9 shows the total cost to all buyers for using different purchasing strategies with different number of buyers when $N \times M$ is 4×4. For different numbers of buyers, both OptimalBundleCF and MGBSBundleCF are also close to the lower bound of the optimal value and much better than using the strategy of only searching for the minimal retail price.

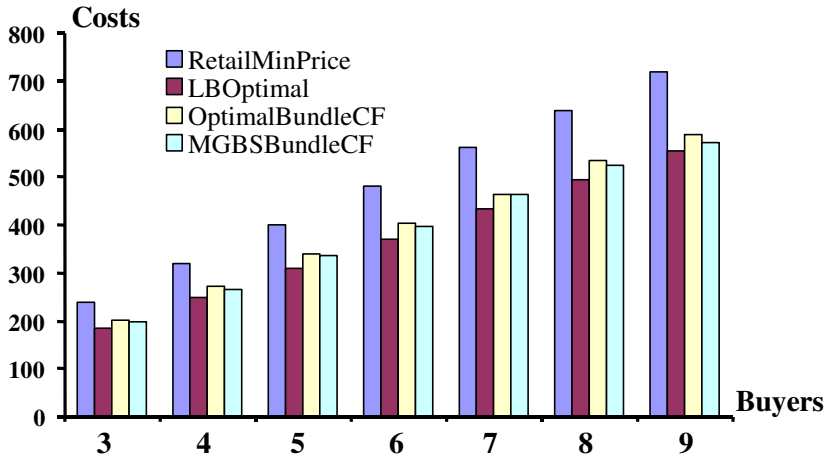

FIGURE 6.9. Total Cost to All Buyers Using Different Purchasing Strategies

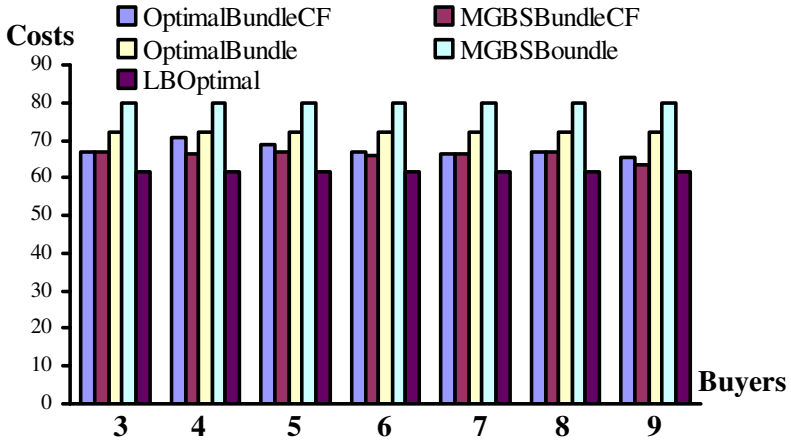

FIGURE 6.10. Average Cost to Each Buyer

Figure 6.10 shows the average cost to each buyer. The difference between the average cost to each buyer does not increase or decrease_significantly as the number of buyers increases in the market. OptimalBundle denotes the cost to a buyer of doing an optimal bundle search without joining any buyer club. MGBSBundle denotes the cost to a buyer of doing a MGBS bundle search without joining any buyer club. It also shows that combining bundle search strategy and buyer coalition formation strategy can reduce the cost more than just doing a bundle search and that the cost is very close to the optimal cost. Indeed, buyer coalition formation can be viewed as an extension of the bundle search strategy by extending the shopping list of one buyer to multiple buyers. In order to guarantee to obtain the discount that each buyer can obtain by himself, my current algorithm makes each buyer do bundle search first. Then, based on the result of bundle search, each buyer decides whether to participant in buyer coalition formation or not. Therefore, the results show that combining bundle search strategy and buyer coalition formation strategy can reduce the cost more than just doing a bundle search in every single experiment.

In order to test how my algorithm perform under different discount policies, I considered the following two discount policies: $\delta_2(c) = 10$ if $c \geq 50$ and $\delta_3(c) = 35$ if $c \geq 150$. It is very easy to find real examples for these two discount policy. At many online stores (*e.g*. Amazon.com), if you spend $50, they normally give you free shipping that may cost you $10 in one transaction. In many fashion stores, it is very normal to obtain a $35 gift card if you spend over $150 in one transaction. Another reason to consider these two discount policies is that they are comparable to the previous discount policy $\delta_1(c)$ (defined in Table 6.1). I want to test the following hypotheses:

- Since discount policy $\delta_1(c)$ allows buyers have more choices to obtain discounts, each buyer will obtain more discount under discount policy $\delta_1(c)$ than under the other two discount policies.
- For all three discount policies, combining buyer coalition formation and bundle search can reduce the cost more than just doing a bundle search.
- For discount policy $\delta_2(c)$, there are few chances for buyers to form coalitions because as long as the total cost is larger than \$50, no matter how much more a buyer spends, the total amount of discount is \$10. For discount policy $\delta_3(c)$, there are more chances (than $\delta_2(c)$) for buyers to form coalitions because the input price for each goods is less than \$50. In order to obtain a discount, a buyer needs to spend \$150.
- For discount policy $\delta_2(c)$, the communication load is lower than the one under another two discount policies because few negotiation processes were required.

TABLE 6.2. Total Cost to All Buyers

	OptimalBundle	MGBSBundle	MGBSBundleCF	OptimalBundleCF
$\delta_1(c)$	518.367	520.85	507.97	506.02
$\delta_2(c)$	530.47	533.30	528.30	505.81
$\delta_3(c)$	590.99	590.99	554.97	530.98

Given $N \times M$ is 4×4 and the total number of buyers is 5, for each combination of discount policy (including $\delta_1(c)$, $\delta_2(c)$ and $\delta_3(c)$) and purchasing strategy (including MGBSBundle, OptimalBundle, MGBSBundleCF and OptimalBundleCF), I ran 20 experiments. Table 6.2 lists the total cost for all 5 buyers using different type of purchasing

strategies under different discount policies. The results show that $\delta_1(c)$ has the best performance in the sense that it costs the buyer the least.

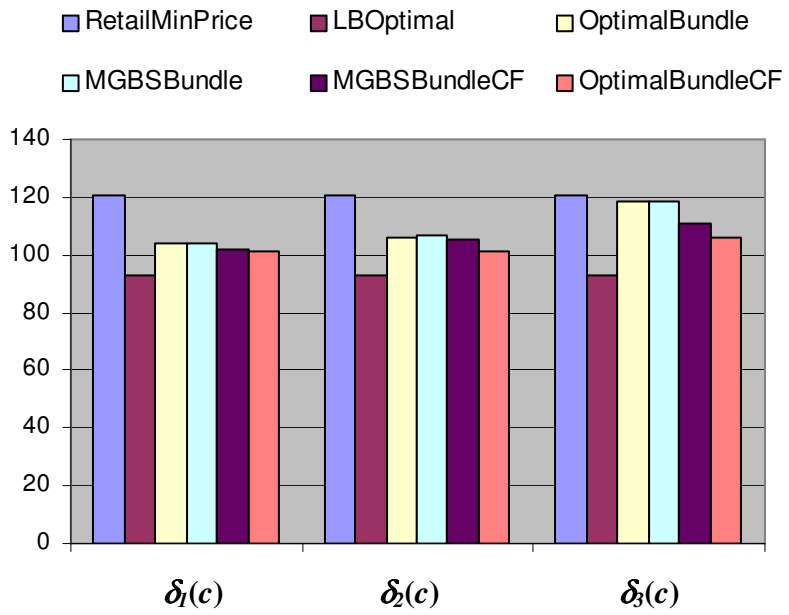

FIGURE 6.11. Average Cost to Each Buyer Using Different Purchasing Strategies

Figure 6.11 depicts the average cost to each buyer for using different purchasing strategies under different discount policies. Figure 6.11 also shows the costs to buyers using RetailMinPrice and LBOptimal. All purchasing strategies, MGBSBundle, OptimalBundle, MGBSBundleCF and OptimalBundleCF can reduce costs more than RetailMinPrice. MGBSBundleCF and OptimalBundleCF can reduce costs more than MGBSBundle and OptimalBundle. Thus, the experimental results support hypothesis 1 and 2. However, the improvement in savings for MGBSBundleCF is less pronounced under $\delta_3(c)$, compared to $\delta_1(c)$ and $\delta_2(c)$.

TABLE 6.3. Numbers of Experiments without Buyer Coalitions Formed

	MGBSBundleCF	OptimalBundleCF
$\delta_1(c)$	2	1
$\delta_2(c)$	15	12
$\delta_3(c)$	4	2

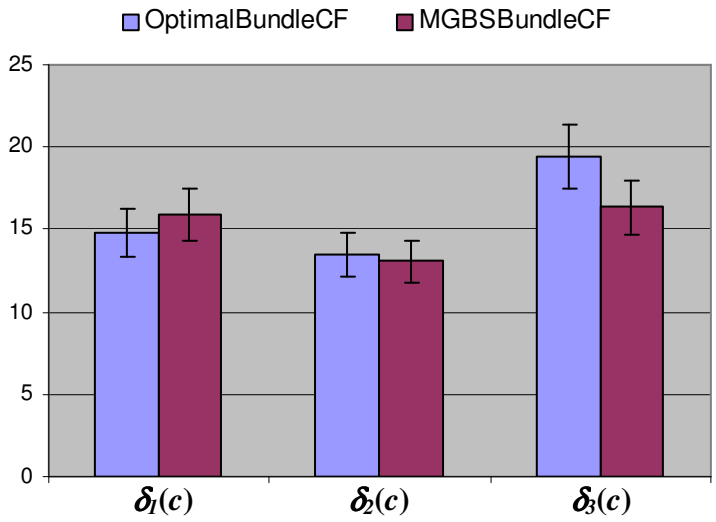

FIGURE 6.12. Average Number of Messages Received by Each Buyer

Table 6.3 lists the numbers of experiments in which no buyer coalitions were formed under different discount policies and different purchasing strategies (For each combination of a discount policy and a purchasing strategy, I ran 20 experiments). The results show that buyers did not form coalitions very often under discount policy $\delta_2(c)$. Under discount policy $\delta_1(c)$ and $\delta_3(c)$, buyers formed coalitions very frequently. Therefore, hypothesis 3 is valid.

As for communication load, Figure 6.12 shows the average number of messages that each buyer received in each experiment and the error bars. Under discount policy $\delta_2(c)$, each buyer received the least number of messages. Hence, hypothesis 4 is valid.

SUMMARY

In electronic markets, both bundle search and buyer coalition formation are profitable purchasing strategies for buyers who need to buy small amount of goods and have no individual bargaining power. It is very valuable to combine these two purchasing strategies for buyers to obtain greater discounts based on the different discount policies of multiple sellers. In this chapter, I presented a distributed mechanism that allows buyers to use both purchasing strategies. The mechanism includes a very efficient heuristic bundle search algorithm and a distributed coalition formation scheme that is based on the DCF-EN mechanism. The simulation results show that the cost to buyers is close to the optimal cost. I also tested how the DCF-EN mechanism performs under different discount policies. The simulation results show that combining bundle search and buyer coalition formation can always outperform only searching for the minimal retail price or only doing bundle search. Also, buyers participate in buyer coalition formation more often when the discount policy requires more cost.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

This dissertation studies the problems of distributed task and resource allocation among self-interested agents in virtual organizations. The developed solutions are not allocation mechanisms that can be imposed by a centralized designer but decentralized interaction mechanisms that provide incentives to self-interested agents to behave cooperatively in a multiagent system. Besides incentive compatibility, these mechanisms also take computational tractability into consideration due to the inherent complexity of distributed task and resource allocation problems. The virtual organizations are not established through centralized administrative regulations but formed through the interaction among self-interested agents. The developed mechanisms can easily be implemented in autonomous agent systems. These decentralized task and resource allocation mechanisms can eventually make automated resource management in virtual organizations a reality.

During the last decade, researchers in distributed artificial intelligence endeavored to establish theoretic models that can address incentive compatibility and computational tractability together when evaluating interaction mechanisms among self-interested agents in multiagent systems. Their approaches are normally developing theoretic models that consider both the classic game-theoretic solution concepts for noncooperative games and computational complexity analysis. Typical example problems of these theoretic models are distributed task and resource allocation problems. However, little work has focused on developing practical solutions for those problems. This dissertation applies these models to

formalizing the distributed task and resource allocation problems that have a variety of applications in grid and peer-to-peer computing, electronic commerce and virtual organizations. The developed solutions for these problems are decentralized interaction mechanisms among self-interested agents that can lead to global task allocation efficiency in a multiagent system or stable resource sharing virtual communities based on agents' own decisions on whether or not to behave cooperatively. This dissertation contributes to the following research areas in multiagent systems: synthetic task allocation, decentralized coalition formation and automated multiparty negotiation.

SYNTHETIC TASK ALLOCATION

This dissertation presents two incentive compatible mechanisms for synthetic task allocation problems in which each task needs to be accomplished by a virtual team composed of self-interested agents from different real organizations. Compared with traditional task allocation problems (Zweben and Fox 1994; Clearwater 1996), the synthetic task allocation needs to consider both individual efficiency and team efficiency. The participating agents have different owners and are designed by different designers. Agents who have tasks that need to be accomplished have no authority to force task-executing agents to disclose their true capabilities. Providing incentives to task executing agents to report their true capabilities is the key for developing efficient synthetic task allocation mechanisms.

My approach is to formalize the synthetic task allocation problems as an algorithmic mechanism design optimization problem. I have developed two incentive

compatible mechanisms for the synthetic task allocation problem. The MinTeamwork is *n*-approximation mechanism and a strongly truthful implementation for monotonic teamwork. Through changing the valuation function and having a more restrictive assumption, the MinCompletion mechanism is a truthful implementation with *2*-approximation for strongly monotonic teamwork. It shows that designing both incentive compatible and computationally tractable mechanisms is feasible for synthetic task allocation problems in virtual organizations.

DECENTRALIZED COALITION FORMATION

The inherent complexity of coalition formation among self-interested agents makes the traditional centralized approaches for coalition formation computationally intractable. Another major contribution of this dissertation is developing a decentralized coalition formation mechanism that is based on explicit negotiation among self-interested agents. Compared with the centralized approaches, the developed mechanism significantly reduces the computational cost of the coalition formation process. The communication cost caused by negotiation processes is low due to the properly designed multiparty negotiation protocol. Each agent makes its own decisions about whether or not to join a possible coalition. The resulting coalitions are stable in the core in terms of coalition rationality.

I have applied this mechanism to form resource sharing coalitions in computational grids and to form buyer coalitions in electronic markets. The simulation results show that the coalition formation process is successful in the sense that explicit negotiation processes can lead agents to find the appropriate coalitions and that a coalition formation process can

end properly. The communication load increases almost linearly with the total number of agents. Particularly, the simulation results reflect that self-interests can still lead to resource sharing in computational grids. This invalidates the need to assume that agents are cooperative in the existing systems. For buyer coalition formation, my distributed coalition formation mechanism can result in nearly optimal cost savings.

AUTOMATED MULTIPARTY NEGOTIATION

The multiparty negotiation mechanism in the decentralized coalition formation mechanism is itself a very unique contribution of this dissertation.

In this negotiation mechanism, the negotiation protocol is designed to allow multiple agents to make agreements among multiple choices. An agent's decision about whether or not to accept an offer may not only depend on the decisions of other members in a proposed coalition but may also depend on the decisions of agents outside of the proposed coalition. The negotiation protocol also includes a conflict handling method that can detect and break deadlocks caused by parallel negotiation processes. The simulations show that the negotiation protocol can conduct the negotiation processes properly.

FUTURE APPLICATIONS

Designing both incentive compatible and computationally tractable mechanisms for distributed task and resource allocation among self-interested agents is one of the most difficult and important themes in multiagent systems. It is essential to realizing a world

where the aggregate power on the Internet will be optimally and dynamically allocated online to appropriate users. The methods we described can be applied in either of the following two ways toward this vision.

Resource Management in Virtual Organizations

Little is known about how to run a virtual organization efficiently. There does not exist a well-understood organizational structure that can represent a virtual organization (Hatch 2005). There are many open problems and new research directions. The techniques developed in this dissertation can be adapted to perform resource management in virtual organizations. The allocation of proper resources to proper tasks so that the global performance of a virtual organization can be maximized even if the participating agents are self-interested. This will require improving the current task and resource allocation mechanisms for more complex scenarios and embedding those mechanisms into existing systems such as computational grids, wireless sensor network and supply chain management in electronic commerce.

Task and Resource Allocation in Scalable Multiagent Systems

One of the major research trends in multiagent systems is to make systems open and scalable. Decentralized task and resource allocation in such a system is highly non-trivial because the system dynamically changes and the scalability invalidates effective communication among agents. How to get self-interested agents to behave cooperatively so

that a global efficiency is achieved by the local decision of each participating agents is the key issue for task and resource allocation in open and scalable multiagent systems. I believe that the decentralized allocation mechanisms presented in this dissertation have laid a foundation for overcoming this challenge in the future.

REFERENCES

Azoulay-Schwartz, R. and S. Kraus. 2004. Stable Repeated Strategies for Information Exchange between Two Autonomous Agents. *Artificial Intelligence* 154: 43-93.

Berman, F., G. Fox, and T. Hey (Editors). 2003. *Grid Computing: Making The Global Infrastructure a Reality*. Chichester, England: John Wiley & Sons.

Bernstein, P.A., V. Hadzilacos, and N. Goodman. 1987. *Concurrency Control and Recovery in Database Systems*. Reading, MA: Addison-Wesley Publishing Company.

Buyya, R., 2002. Economic-Based Distributed Resource Management and Scheduling for Grid Computing. Ph.D. Dissertation, Monash University, Melbourne, Australia.

Buyya, R., D. Abramson, J. Giddy, and H. Stockinger. 2002. Economics Paradigm for Resource Management and Scheduling in Grid Computing. *Concurrency and Computation: Practice and Experience*. 14: Grid Computing Environments Special Issue 13-14.

Caillou, P., S. Aknine, and S. Pinson. 2002. A Multi-Agent Method for Forming and Dynamic Restructuring of Pareto Optimal Coalitions. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems*. Pages 1074-1081, Bologna, Italy, July 15-19.

Chang, Y., C. Li, and J. R. Smith. 2003. Searching Dynamically Bundled Goods with Pairwise Relations. In *Proceedings of ACM Electronic Commerce*. Pages 135-143, San Diego, CA, June 9-12.

Chen, J., 2004. *Introduction to Tractability and Approximability of Optimization Problems*. Book Manuscript, Department of Computer Science, Texas A&M University.

Clarke, E. H. 1971. Multipart Pricing of Public Goods. *Public Choice* 2: 17-33.

Clearwater, S.H. (Editor). 1996. *Market-Based Control: A Paradigm for Distributed Resource Allocation*. Singapore: World Scientific.

Coffman,E.G., L. Flatto, M.R. Garey and R.R. Weber. 1987. Minimizing Expected Makespans on Uniform Processor Systems. *Advances in Applied Probability* 19:177-201.

Davis, R. and R.G. Smith. 1983: Negotiation as a Metaphor for Distributed Problem Solving. *Artificial Intelligence* 20 (1): 63-109.

Feigenbaum, J., A. Krishnamurthy, R. Sami, and S. Shenker. 2001. Approximation and Collusion in Multicast Cost Sharing. In *Proceedings of the Third Annual ACM Conference on Electronic Commerce*. Pages 253-255, Tampa, Florida, October 14-17.

Feigenbaum, J. and S. Shenker. 2002. Distributed Algorithmic Mechanism Design: Recent Results and Future Directions. In *Proceedings of the Sixth International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications*. Pages 1-13, Atlanta, GA, September 26 – 28.

Foster, I. and C. Kesselman. 1998. *The Grid: Blueprint for a New Computing Infrastructure* (First Edition). San Francisco, CA: Morgan Kaufmann.

Foster, I. and A. Iamnitchi. 2003. On Death, Taxes, and the Convergence of Peer-to-Peer and Grid Computing. In *Proceedings of the Second International Workshop on Peer-to-Peer Systems*. Pages 118-128, Berkeley, CA, February 20-21.

Foster, I., N.R. Jennings, and C. Kesselman. 2004. Brain Meets Brawn: Why Grid and Agents Need Each Other. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-agent Systems*. Pages 8-15, New York , July 21-23.

Fudenberg, D and J. Tirole. 1991. *Game Theory*. Cambridge, MA: MIT Press.

Garey, M. R. and D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco, CA: W. H Freeman and Company.

Groves, T., 1973. Incentives in Teams. *Economica* 41: 617-631.

Harsanyi, J. C., 1967. Games with Incomplete Information Played by "Bayesian" Players. *Management Science* 14(3):159-182.

Harsanyi, J. C., 1977. *Rational Behavior and Bargaining Equilibrium in Games and Social Situations*. Cambridge, UK: Cambridge University Press.

Hatch P.J., 2005. Offshore 2005 Research Preliminary Findings and Conclusions. Vers.1.2.5. *Ventoro Report*. Portland, OR, Ventoro Corporation.

He, L. and T.R. Ioerger 2003. A Quantitative Model of Capabilities in Multi-Agent Systems. In *Proceedings of the International Conference on Artificial Intelligence*. Pages 730-736, Las Vegas, NV, June 23-26.

He, L. and T.R. Ioerger. 2004a. An Efficient Heuristic Bundle Search Algorithm for Buyers in Electronic Markets. In *Proceedings of the 2004 International Conference on Artificial Intelligence*. Pages 729-735, Las Vegas, NV, June 23-26.

He, L. and T.R. Ioerger. 2004b. Combining Bundle Search with Buyer Coalition Formation in Electronic Markets: A Distributed Approach through Negotiation. In *Proceedings*

of the Sixth International Conference on Electronic Commerce. Pages 95-104, Delft, The Netherlands, October 25-27.

He, L. and T.R. Ioerger. 2005a. Combining Bundle Search with Buyer Coalition Formation in Electronic Markets. *Electronic Commerce Research and Applications Journal* 4(4): 329-344.

He, L. and T.R. Ioerger. 2005b. Forming Resource-Sharing Coalitions: A Distributed Resource Allocation Mechanism for Self-Interested Agents in Computational Grids. In *Proceedings of the Twentieth Annual ACM Symposium on Applied Computing*. Pages 84 – 91, Santa Fe, NM, March 13 – 17.

He, L. and T.R. Ioerger. 2005c. Incentive Compatible Mechanism Design for Synthetic Task Allocation in Virtual Organizations. In *Proceedings of the 2005 IEEE International Conference on Integration of Knowledge Intensive Multi-Agent Systems*. Pages 205-210, Waltham, MA, April 18 – 21.

Holliday, J. and A. El Abbadi. 2005. Distributed Deadlock Detection. *Encyclopedia of Distributed Computing*. Kluwer Academic Publishers, Norwell, MA, accepted for publication.

Jackson, M.O., 2001. A Crash Course in Implementation Theory. *Social Choice and Welfare* 18: 655-708.

Jennings, N., P. Faratin, A. Lomuscio, S. Parsons, C. Sierra, and M. Wooldridge. 2001. Automated negotiation: Prospects, Methods and Challenges. *International Journal of Group Decision Negotiation* 10(2): 199-215.

Kahan, J.P. and A. Rapoport 1984. *Theories of Coalition Formation*. London, UK: Lwarence Erlbaum Associates, Inc.

Kraus, S. and T. Plotkin. 2000. Algorithms of Distributed Task Allocation for Cooperative Agents. *Theoretical Computer Science* 242(1-2): 1-27.

Kraus, S., 2001. *Strategic Negotiation in Multiagent Environments*. San Francisco, CA: The MIT Press.

Krishna, V., 2002. *Auction Theory*. London, UK: Academic Press.

Lan, Z., V. Taylor and G. Bryan. 2002. A Novel Dynamic Load Balancing Scheme for Parallel Systems. *Journal of Parallel and Distributed Computing* 62(12):1763-1781.

Larson, K. and T. Sandholm 2001. Bargaining with Limited Computation: Deliberation Equilibrium. *Artificial Intelligence*. 132(2): 183-217.

Lerman, K. and O. Shehory. 2000. Coalition Formation for Large Scale Electronic Markets. In *Proceedings of the International Conference on Multi-Aent Systems*. Pages 167-174, Boston, MA, July 10-12.

Li, C. and K. Sycara. 2002. Algorithm for Combinatorial Coalition Formation and Payoff Division in an Electronic Marketplace. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multi-agent Systems*. Pages 120-127, Bologna, Italy, July 15-19.

Mas-Colell, A., M.D. Whinston and J.R. Green 1995. *Microeconomic Theory*. New York: Oxford University Press.

Milojicic, D.S., V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu 2002. Peer-to-Peer Computing. Technique Report, HPL-2002-57, HP Laboratories, Palo Alto, CA.

Nisan, N. and A. Ronen. 1999. Algorithmic Mechanism Design. In *Proceedings of the 31st Annual ACM Symposium on Theory of Computing*. Pages 129-140, Atlanta, GA, May 1-4.

O'Donnell, R.D. and F.T. Eggemeier. 1986. Workload Assessment Methodology. In: *Handbook of Perception and Human Performance: Volume II. Cognitive Processes and Performance*, eds. K.R. Boff, L. Kaufman, and J. Thomas, Chap. 42, New York: John Wiley and Sons.

O'Hare G. and N. Jennings. 1996. *Foundations of Distributed Artificial Intelligence.* Sixth-Generation Computer Technology Series. Soucek B. (Series Editor). New York: John Wiley and Sons.

Osborne, M.J. and A. Rubinstein. 1994. *A Course in Game Theory*. New York: MIT Press.

Papadimitriou, C., 2001. Algorithms, Games, and the Internet. In *Proceedings of the 33rd Symposium on Theory of Computing*. Pages 749-753, Heraklion, Crete, Greece, July 6-8.

Parkes, D.C., 2000. Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency. Ph.D. Dissertation, University of Pennsylvania, Philadelphia, PA.

Porter, R., 2004. Mechanism Design for Online Real-time Scheduling. In *Proceedings of the ACM Conference on Electronic Commerce*. Pages 61–70, New York, May 17-20.

Proter, M. E., 1980. *Competitive Strategy*. New York: The Free Press.

Raiffa, H., 1982. *The Art and Science of Negotiation: How to Resolve Conflicts and Get the Best out of Bargaining*. Cambridge, MA: Harvard University Press.

Ronen, A., 2000. Solving Optimization Problems among Selfish Agents. Ph.D Dissertation, Hebrew University, Jerusalem.

Rosenschein, J. S. and G. Zlotkin 1994. *Rules of Encounter: Designing Conventions for Automated Negotiation among Computers*. Cambridge, MA: The MIT Press.

Sandholm, T. and V. Lesser. 1995. Issues in Automated Negotiation and Electronic Commerce: Extending the Contract Net Framework. In *Proceedings of the First International Conference on Multi-Agent Systems*. Pages 328-335, San Francisco, CA, June 12 -14.

Sandholm, T., 1996. Negotiation among Self-Interested Computationally Limited Agents. Ph.D. Dissertation, University of Massachusetts at Amherst.

Sandholm, T. and V. Lesser. 1997. Coalitions among Computationally Bounded Agents. Special issue on Economic Principles of Multiagent Systems, *Artificial Intelligence* 94(1): 99-137.

Sandholm, T., K. Larson, M. Andersson, O. Shehory, and F. Tohme. 1999. Coalition Structure Generation with Worst Case Guarantees. *Artificial Intelligence Journal* 111: 209-238.

Sandholm, T. and V. Lesser. 2002. Leveled Commitment Contracting: A Backtracking Instrument for Multiagent Systems. *AI Magazine* 23 (3): 89-100.

Sandholm, T., 2003. Making Markets and Democracy Work: A Story of Incentives and Computing. *In Proceedings of the International Joint Conference of Artificial Intelligence*. Pages 1649-1671, Acapulco, Mexico, August 9-15.

Sen, S. and P. S. Dutta. 2000. Searching for Optimal Coalition Structures. In *Proceedings of the Forth International Conference on Multiagent Systems*. Pages 286-292, Boston, MA, July 7-12.

Shan, H., L. Oliker, and R. Biswas. 2003. Job Superscheduler Architecture and Performance in Computational Grid Environment. In *Proceedings of Super Computing'03*. Pages 44, Phoenix, AZ, November 15-21.

Shehory, O. and S. Kraus. 1995. Task Allocation via Coalition Formation among Autonomous Agents. In *Proceedings of the International Joint Conference of Artificial Intelligence*. Pages 655-661, Quebec, Canada, August 20-25.

Shehory, O. and S. Kraus. 1998. Methods for Task Allocation via Agent Coalition Formation, *Artificial Intelligence Journal* 101 (1-2): 165-200.

Shehory, O. and S. Kraus. 1999. Feasible Formation of Coalitions among Autonomous Agents in Non-Super-Additive Environments. *Computational Intelligence* 15(3): 218-251.

Sierra, C., P. Faratin, and N.R. Jennings. 1997. A Service-Oriented Negotiation Model between Autonomous Agents. In *Proceedings of the Eighth European Workshop on Modeling Autonomous Agents in a Multi-Agent World*. Pages 17-35, Ronneby, Sweden, May 13-16.

Silberschatz, A., P.B. Galvin, and G. Gagne. 2002. *Operating System Concepts*. Fifth Edition, Palo Alto, CA: Sun Microsystems Press.

Sycara, K.P., 1987. Resolving Adversarial Conflicts: An Approach to Integrating Case-Based and Analytic Methods. Ph.D. Dissertation, Georgia Institute of Technology, Atlanta, GA.

Sycara, K.P., 1990. Persuasive Argumentation in Negotiation. *Theory and Decision* 28: 203-242.

Tesfatsion, L., 2002. Agent-based Computational Economics: Growing Economies from the Bottom Up. *Artificial Life* 8:55--82.

Tsang P. and G.F. Wilson. 1997. Mental Workload. In: *Handbook of Human Factors and Ergonomics* (Second Edition), 417-449, New York: John Wiley & Sons.

Tulga, M.K. and T.B. Sheridan. 1980. Dynamic Decision and Workload in Multitask Supervisory Control. IEEE Trans. *On Systems, Man, and Cybernetics* 10(5):217-232.

Varian, H.R., 1995. Economic Mechanism Design for Computerized Agents. In *Proceedings of the First Usenix Workshop on Electronic Commerce*. Pages 13-21, New York, July 11-12.

Vickery, W., 1961. Counterspeculation, Auctions and Competitive Sealed Tenders. *Journal of Finance* March:8-37.

Weiss, G.(Editor), 1999. *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. Cambridge, MA: The MIT Press.

Wickens C.D., 1999. Attention, Time-sharing, and Workload. In: *Engineering Psychology and Human Performance*, eds. C.D. Wickens and J.G. Hollands, 439-479, Upper Saddle River, NJ: Prentice Hall.

Wolski, R., J.S. Plank, J. Brevik, and T. Bryan. 2001. Analyzing Market-based Resource Allocation Strategies for the Computational Grid. *International Journal of High Performance Computing Applications* 15(3): 258-281.

Wolski, R., J. Brevik, J. Plank, and T. Bryan. 2003. Grid Resource Allocation and Control Using Computational Economies. In: *Grid Computing: Making the Global Infrastructure a Reality*, eds. F. Berman, G. Fox, and T. Hey, 747-772, Chichester, England: John Wiley and Sons.

Yamamoto, J. and K. Sycara. 2001. A Stable and Efficient Buyer Coalition Formation Scheme for E-Marketplaces. In *Proceedings of the Fifth International Conference on Autonomous Agents*. Pages 576-583, Montreal, Canada, May 28 - June 1.

Ye, Y. and Y. Tu. 2003. Dynamics of Coalition Formation in Combinatorial Trading. In *Proceedings of International Joint Conference of Artificial Intelligence*. Pages 625-632, Acapulco, Mexico, August 9-15.

Zeng, D. and K. Sycara. 1998. Bayesian Learning in Negotiation. *International Journal of Human-Computer Studies* 48:125-141.

Zweben, M. and M.S. Fox (Editors). 1994. *Intelligent Scheduling*. San Francisco, CA: Morgan Kaufmann Publishers.

VITA

Linli He received her Master of Science degree in computer science from Southwest Petroleum Institute, P. R. China in 1999. She entered the computer science program at Texas A&M University in September 2000, and she received her Doctor of Philosophy degree in May 2006. Her research interests include artificial intelligence, multiagent systems, game theory, information fusion and operations management.

Linli He may be reached at Knowledge Based Systems, Inc., 1408 University Drive East, College Station, TX 77840. Her email address is lhe@kbsi.com.