2015

Agent-based task allocation in dynamic and open grid and cloud environments

Yan Kong
University of Wollongong

Agent-based Task Allocation in Dynamic and Open Grid and Cloud Environments

A thesis submitted in fulfillment of the
requirements for the award of the degree

Doctor of Philosophy

from

UNIVERSITY OF WOLLONGONG

by

Yan Kong

School of Computer Science and Software Engineering
August 2015

Dedicated to
My family and friends

Declaration

This is to certify that the work reported in this thesis was done by the author, unless specified otherwise, and that no part of it has been submitted in a thesis to any other university or similar institution.

Yan Kong
August 23, 2015

Abstract

Task allocation is an important problem in grid/cloud environments in both research and applications. With the rapid development of grid/cloud environments, the features of openness and dynamism of environments put two new challenges to the development of task allocation approaches and strategies in such environments. Firstly, the participants in the environments normally have only local views about the environments due to the administrative independencies between the participants and the limited communication abilities of the participants. Secondly, task allocation methods/approaches have to handle the dynamism and openness of the environments. In particular, task allocation methods/approaches have to respond to and be resilient from the unpredicted changes in the environments in a quick manner.

This thesis aims

1. to study the approaches of single task allocation with time constraints in open and dynamic grid/cloud environments where both the resource consumers and providers can enter and leave freely. A decentralised negotiation-based method for such a type of task allocation is proposed. In the proposed method, both consumers and providers only have local views about the environment. Consumers and providers trade with each other through negotiations in which they make their offer (count-offer) decisions strategically through taking the issues that they are concerned with into account. The experimental results show that the proposed method can achieve desirable performances in terms of the success rate of and profit obtained from the task allocation;

2. to investigate the problem of group task allocation with time constraints, in which a task consists of more than one independent subtask, in open and dynamic grid/cloud environments. To solve this problem, a decentralised combinatorial auction-based approach is proposed. In the proposed approach, the combinatorial auction is used to select a group of resource providers to perform all the subtasks

need to be allocated. An indicator is also designed to help a consumer to select the most suitable group of providers; and

3. to study the problem of group task allocation, in which a task consists of more than one interdependent subtask with dependency constraints, in open and dynamic grid/cloud environments. A strategic max-sum belief propagation-based approach is proposed for such a type of task allocation. In the proposed approach, providers quote for the subtasks that they are interested in, and are allowed to dynamically and strategically update their quote prices, according to the constant changes in the environment. In addition, aiming at accelerating the online response to, improving the resilience from the unpredicted changes in general grid/cloud environments, and mitigating the requirement of message passing, a max-sum belief propagation-based method is also proposed.

Acknowledgement

First and foremost, I express my deepest gratitude to my parents, Lingfeng Kong and Yuling Mao, for their everlasting love and care for me over the past 28 years. I also want to give heartfelt thanks to my grandparents who passed away during my PhD study, my younger brother, and my other family members. My family members are the strongest supports during my PhD study.

I am indebted to my supervisor, Professor Minjie Zhang. Having Professor Minjie Zhang to be my supervisor is one of the most fortunate things in my life. Professor Minjie Zhang has been so supportive, patient, and keen in guiding me in my PhD study and research. Without her help, it is impossible for me to accomplish my PhD study. I also express my deep gratitude to Dr Dayong Ye, who is both my co-supervisor and good friend, for his valuable advices and patient guidance about my research. My thanks are extended to Dr Madeleine Cincotta, who helped so much in proofreading my papers. Additionally, I wish to thank Mr Xing Su, who always discussed with me about the research in our laboratory.

Studying abroad alone is a hard time and tiring journey. I want to express my gratitude to the accompany and help of my friends: Sarah Grant, Amy Jones, and Carmal Shimpson. Sarah Grant, one of my best Australian friends, helped me very much in English and Australian culture. She also introduced many kind Australian friends to me, who made me feel the real life of Australian people. Amy Jones, my housemate, always drove me to the beach after dinner to have a walk and to help me relax whenever I felt too stressed in my PhD study. Carmal Shimpson, also my housemate, is the funniest girl I ever met. Carmal gave a lot happiness and fun to me. I felt so relaxed and can forget all the stress when I chatted with her home.

I would like to express my heartfelt thanks to China Scholarship Council and University of Wollongong for the financial support for my PhD study. Finally, thanks to all the anonymous reviewers of my research papers and examiners of my thesis, and all my other dear friends and relatives who have helped me during my PhD study.

Publications

The following is a list of my research papers, which have been already published (accepted) or currently is under review, during my PhD study that ends with the completion of this thesis.

Refereed Journal Articles:

1. **Yan Kong**, Minjie Zhang and Dayong Ye, A Negotiation-based Method for Task Allocation with Time Constraints in Open Grid Environments. *Concurrency and Computation: Practice and Experience*, **Vol. 27**, **No. 3**, pp. 735-761, 2015. (ERA Journal List A)

2. **Yan Kong**, Minjie Zhang and Dayong Ye, An Auction-based Approach for Group Task Allocation in An Open Network Environment. *The Computer Journal*. (This paper was accepted in July of 2015, and will be published soon.) (ERA Journal List A*)

Scholarly Book Chapters:

1. **Yan Kong**, Minjie Zhang and Dayong Ye, A Group Task Allocation Strategy in Open and Dynamic Grid Environments. In *Recent Advances in Agent-based Complex Automated Negotiation, Studies in Computational Intelligence*, Fujita et al. (Eds.), Springer, accepted in February 2015 and in press.

2. **Yan Kong**, Minjie Zhang and Dayong Ye, A Negotiation Method for Task Allocation with Time Constraints in Open Grid Environments. In *Next Frontier in Agent-Based Complex Automated Negotiation, Studies in Computational Intelligence*, Fukuta et al. (Eds.), Springer, **Vol. 596**, pp. 19-36, 2014.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years, with the development of the technologies of grids and clouds, more and more organizations have built platforms for resource exchange or rent their platforms to other organisations for other purposes, using such technologies. Amazon's EC2 (Elastic Compute Cloud) [EC2] and EBS (Elastic Block Store) [EBS] are two of the first examples of such platforms. Amazon rents its idle computing resources and storage resources through EC2 and EBS, respectively, to gain profits. Some other platforms, which play the role of mediums, encourage resource providers and consumers to rent or exchange resources through them. The grid-based platform, Global Environment for Network Innovations (GENI) [GEN], is a typical example of such a type of platforms. In addition, a variety of other market-oriented grid/cloud systems [FCC+03, ICG+06, ACSV04, LRA+05] used for resource exchange and renting have also been explored. There is a common problem to be solved in the above mentioned platforms and systems, that is, how to allocate the tasks of resource consumers to resource providers. In other words, the tasks of resource consumers need to be allocated to suitable resource providers [MSRJ11]. Therefore, task allocation in grids and clouds, which helps regulate the resource supply and demand [ALIZ10], is an important research problem in grid/cloud environments.

The purpose of this thesis is to study the agent-based task allocation problem in grid/cloud environments and provide agent-based solutions to the problem. This chapter is structured as follows. The background knowledge about task allocation, grid/cloud environments, and the roles of agents in task allocation in such environments are briefed in Section 1.1. In Section 1.2, the motivations of this research is described. The research issues and objectives of this thesis are outlined in Section 1.3. The contributions of this thesis are presented in Section 1.4, and the structure of this thesis is given in Section 1.5.

1.1 Background

1.1.1 Task Allocation

Task allocation is a problem where a resource consumer selects a suitable resource provider/providers, which the consumer wishes to allocate its tasks to. With the development of grid/cloud technologies, task allocation has been becoming an important problem in both research [ZK08, MSRJ11] and applications in many domains, such as RoboCup rescue [RFMJ10, FJDSB+10, NITM02, RPF+10, CMKJ09], radar predictions of weather situations [ALWZ11], supply chain formation [WWY00, PA13], e-trading [Eba], and distributed computing systems [MLT82]. The purpose of task allocation depends on many factors such as the types of tasks and environments where task allocation takes place.

1.1.1.1 Common Environments of Task Allocation

There are mainly three types of environments where task allocation takes place, which are *cooperative environments* [TPVS07, ZK08], *competitive environments* [ACSV04, KN09], and *semi-competitive environments* [LJS+03].

Cooperative environments: In cooperative environments, multiple resource providers generally cooperate to finish tasks with the goal of maximising the global profit or minimizing the global cost.

Competitive environments: In competitive environments, resource consumers allocate their tasks to resource providers to earn profits, and both the providers and consumers normally try to maximise their individual profits while overlooking others'.

Semi-competitive environments: In semi-competitive environments, both the providers and consumers are self-interested. However, none of the providers can gain profits without collaboration with others. Therefore, self-interested providers generally collaborate to finish consumers' tasks to gain mutual profits but meanwhile try to maximise their individual profits.

1.1.1.2 Types of Task Allocation

From the perspective of the tasks to be allocated, task allocation mainly include:

Single task allocation: The task to be allocated is atomic, which means that the task cannot be divided into subtasks and thus can be only allocated to one provider.

Group task allocation: The execution of a task needs a group of providers to collaborate.

Continuous task allocation: The task to be allocated consists of multiple subtasks which have to be finished in a certain sequence or new subtasks keep appearing.

This thesis mainly focuses on the study of task allocation problems in both competitive or semi-competitive environments due to the selfishness of both the resource providers and consumers. All of the three types of task allocation are considered in this thesis.

1.1.2 Important Features of Grid/Cloud Environments

This thesis addresses the task allocation in grid/cloud environments. This Subsection provides a brief introduction of the grids and clouds.

Since the technologies of grids/clouds just appeared and were spurred in recent years, there have not been universally accepted definitions for them. However, out of the diverse definitions, there are several definitions which can highlight the general and essential features of grids and clouds and thus are broadly accepted. According to the definition given by Cheey and Buyya [CB02], a grid is *a distributed and parallel system which can dynamically enable the sharing, selection, and aggregation of distributed and autonomous resources, depending on the resources' capability, availability, cost, performance, and the quality-of-service requirements of grid users.*

Another well-known and widely accepted definition of the grid is the one given by Ian Foster [FKT02]. In their definition, a grid is *a system that coordinates resources which are not subject to any centralised control, using open, standard, and general-purpose protocols and interfaces to deliver nontrivial qualities of services.*

In the past years, grid technology has been widely studied and employed into real applications. Many projects studying the grid technology have been founded, such as NSFs National Technology Grid [Che04], NASAs Information Power Grid [JGN99], GriPhyN [DKM+02], NEESgrid [PDJ+04], Particle Physics Data Grid [LCLM+01], the European Data Grid [SRG+00], Globus and EGI-InSPIRE [EGI]. DemoGrid [GLO], the first release of Globus, is a tool which built an instructional grid environment that can be deployed using virtual machines on physical resources or on a cloud [SK11]. The goal of DemoGrid is to provide easy access to an environment with various grid tools, without the need to install the tools or requiring an account on an existing grid [SK11]. EGI-InSPIRE is a project that aims to establish a sustainable European Grid

Infrastructure (EGI) to join together the new Distributed Computing Infrastructures (DCIs) such as desktop grids, supercomputing networks and clouds, for the benefits of user communities within the European Research Area [SK11]. DDGrid [DDG] is a platform built for drug discovery using grid computing technology. Due to the data intensive scientific applications and large-scale computation in the field of medicine chemistry, DDGrid provides significant help in the drug discovery.

Like grids, there has not been an universally accepted definition of clouds. The definition of a cloud given by Buyya is widely accepted. According to Buyya, a cloud is *a distributed and parallel system which consists of inter-connected and virtualized computers that are presented and provisioned dynamically as unified computing resource(s), based on some service-level agreements established between service providers and consumers* [Buy99].

A large number of projects and applications of the technology of clouds have been founded in recent years. TCloud [CLOb] is a project that aims to prototype an advanced cloud infrastructure, which delivers scalable, cost-efficient, and simple computing and storage services, with a new level of security and privacy. The applications of the cloud technology include web hosting, media hosting, multi-tenant service, HPC, distributed storage, multi-enterprise, etc [SK11]. Panda Cloud antivirus [CLOa], is the first free antivirus from the cloud. It uses collective intelligence servers for simple interface, fast detection and protects PC offline.

From both the formal definitions and real world applications of grids and clouds, three major features of grid/cloud environments are summarised.

1. ***Openness***: resource providers and consumers can decide when to leave or enter into the grid/cloud environments autonomously.

2. ***Dynamism***: in both grid/cloud environments, resources are dynamically provisioned and presented.

   The dynamism mainly results from three reasons: (1) openness of the environments, (2) the constant varying of the capabilities and performances of resources, and (3) the constant changing of resoruce requirements of tasks.

3. ***Distribution***: the involved resources are owned by administratively independent organizations/owners and thus, they are not subject to any central controller.

The above three features put great challenges for developing effective and efficient task allocation methods/approaches in grid/cloud environments.

1.1.3 Roles of Agents in Task Allocation in Grid/Cloud Environments

Breaking interests in the research of agents appeared in late last century from many motivations such as artificial intelligence [RNI95], object-oriented programming [RBP+91] and concurrent object-based systems [Agh85], and human-computer interface design [M+94]. Even though agents have been widely studied and used in related areas, there has not been universally accepted definitions of agents. To date, the mostly used definition of an agent is the one that was normally summarised by Michael Wooldridge and Nicholas R. Jennings based on the various versions of definitions of an agent [WJ+95]. In their definition, an agent is a hardware or a software-based (more usually) computer system which possesses the following four properties [WJ+95]:

1. ***Autonomy***: agents can control their own actions and behaviours, and can operate without direct guidance of humans or others [Cas95].

2. ***Reactivity***: agents can perceive the environment and react the changing of the environment in a timely manner.

3. ***Social ability***: agents are capable of interacting other agents using agent communication language [GK94].

4. ***Pro-activity***: agents do not just simply act in response to the environment, rather, they can exhibit goal-directed behaviours by taking initiatives.

The above four properties are the essential ones but not all of the properties of an agent. An agent may have more than these four properties, and given an application, the four properties could be in different importance levels [JSW98]. Agents are being used in more and more applications ranged from comparatively small systems such as personalized email filters to complex and large systems such as the medical care systems and the air-traffic control systems. In fact, however, agents are not always used individually in applications, rather, Multi-Agent Systems (MASs) are used the most. A MAS is a team of agents that work in collaboration pursuing assigned tasks to achieve the overall goal of the system [PFR09]. In principle, a MAS may be conceptualised in terms of agents, but implemented without any software structure that corresponds to agents. A MAS can operate without human interventions and thus is *autonomous*. In addition, MAS*s* are *social-able* in that the agents in a MAS can interact with other

agents via some kind of agent communication languages or protocols. In addition, the agents can perceive and react to the environments. Moreover, a MAS is *proactive* since it is capable of exhibiting goal-oriented behaviours by taking the initiatives [PFR09]. MAS*s* have evolved into increasingly powerful tools, which have been used to develop complex systems by taking advantages of the properties of an agent (i.e., autonomy, sociality, reactivity and pro-activity) [FG97]. Specially, MAS technology is ideally suited to represent the problems that have multiple problem solving entities, multiple problem solving methods, and multiple perspectives [JSW98].

MAS*s* have been used in a wide range of applications such as air traffic control [LL92], mobile robots [LBY03], traffic and transportation management [JMC+96, FMlP96], unmanned air vehicles (UAVs) [FM04], power system [PFR09], and smart grid system [YZS11, RZS13]. For instance, in the area of traffic and transportation management, a MAS designed to implement a future car pooling application was introduced in [BHM97]. There are two types of agents in the system: one representing the customers and the other representing the stations. In particular, the customer agents require or offer transportation, and the station agents determine whether the requests of customer agents (e.g., when and where the customers want to go) can be accommodated. In the power system, Pipattanasomporn et al. designed and implemented a multi-agent system used in the Intelligent Distributed Autonomous Power System (IDAPS), which is a concept of distributed smart grid proposed by Advanced Research Institute of Virginia Tech [PFR09]. With the built-in multi-agent system, IDAPS can be perceived as an intelligent microgrid. Ye et al. proposed a multi-agent framework with a Q-learning algorithm to support rapid restoration of power grid systems when catastrophic disturbances happen. By using the technology of multi-agent, their framework can achieve accurate decision making and quick responses when potential cascading failures are detected in power systems [YZS11]. In order to dynamically and efficiently manage the power dispatch in a distribution network to balance the power supply and demand by considering the variability of distributed generators and loads, Ren et al. proposed a MAS, which works through introducing five types of autonomous agents, the agent communication ontology, the electricity management mechanisms, and the agent cooperation strategy [RZS13].

MAS methods/approaches have been widely employed in open and dynamic grid/cloud environments. Jun et al. devised an agent-based grid resource discovery algorithm, which was used to dynamically assemble agents to supply the information about the distributed network resources [JBPM00]. In their algorithm, each agent exchanges grid

resource information with other agents periodically. In [Sim06], negotiants are modeled as market-driven agents, which negotiate for resources on behalf of their owners in grids. The agents take the dynamism of the grid environment into account when negotiating, and are programmed to slightly relax their negotiation criteria to enhance the success rates of negotiations. In [CSJ+03], an agent-based management infrastructure was proposed to balance the work for task allocation in grid environments. In such an infrastructure, homogeneous agents are adopted at a higher level to represent the grid resources. Agents cooperate with each other to balance the workload in a global grid environment using mechanisms of service advertisement and discovery [CSJ+03]. Another example applying techniques of agents into clouds is the agent-based cloud resource management testbed proposed by Sim in [Sim09]. The testbed is devised to simulate cloud resource management and consists of a set of physical machines, virtual machines, resource consumers, resource consumer agents, provider agents, and broker agents that connect the requests from consumers to the advertisements from providers. From the perspective of the cloud, this testbed contributes a novel approach of resource negotiation and discovery to the cloud. From the perspective of agents, this testbed demonstrated that agents could be used in the application domains which require interacting components to be designed with self-managing capabilities [Sim09].

1.2 Importance of Task Allocation in Grid/Cloud Environments

Task allocation in grids and clouds can be motivated by the fields of both research and applications. In this section, some examples of grids and clouds in real world from both ***research*** and ***applications*** are briefly introduced, respectively.

PlanetLab [PL] and GENI [GEN] (Global Environment for Network Innovations) are two of the typical examples initially developed for ***research*** use. PlanetLab supports the development of new network services through pooling together computational resources that can be shared among sites or peers to finish the computing-intense works [LRA+05]. Therefore, resource sharing and task allocation rules are necessary for PlanetLab. PlanetLab is critical to the foundation of GENI which is a facility concept being explored by the United States Computing Community with support from the National Science Foundation. The purpose of GENI is to enhance the experimental research in distributed systems and computer networking and to accelerate the transition of this

research into services and products that can improve the economic competitiveness of the United States. GENI was initially developed as a non-profit platform to allocate research tasks to the resources assembled from a wide range of administratively independent entities, such as research departments and universities. Figure 1.1 is a simple illustration of the racks connection of GENI.

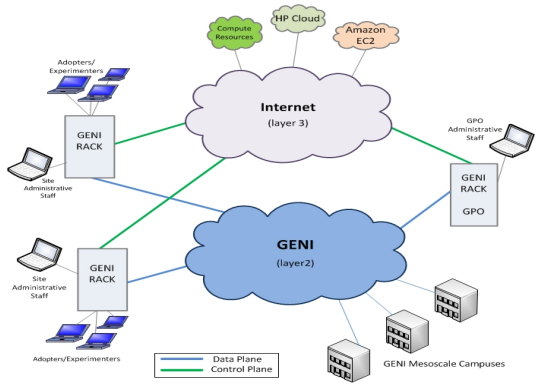


Figure 1.1: GENI-Racks-Connection

Amazon Elastic Compute Cloud (EC2) [EC2] and HP cloud [HPC] are two well-known examples of grids and clouds in the ***application*** fields. In EC2 and HP Cloud, self-interested resource providers and consumers lease, rent or exchange resources to gain profits. As a consequence, a market-oriented task allocation mechanism is required. In such environments, resource providers provide and withdraw their resources, and tasks appear and expire dynamically and unpredictably over time. As a consequence, a system where providers can respond to the dynamism quickly and resiliently is required [NPC+04]. To solve this problem, a number of projects have been promoted for the study of task allocation and resource scheduling in such environments, such as CONOISE-G (Constraint-Oriented Negotiation in an Open Information Services Environment (Grid)) [NPC+04], MASSYVE (Multiagent Manufacturing Agile Scheduling Systems for Virtual Enterprises) [RCMA98], and X-CITTIC (a planning and control system for semi-conductor virtual enterprises) [SdSAB97]. CONOISE-G provides a model of virtual organisation management which operates in a resilient and agile manner and focuses particularly on the virtual organisation formation to finish big tasks. MASSYVE focuses on agile resource scheduling, while X-CITTIC concentrates on the planning and controlling the resources of virtual organizations to deliver services to tasks.

All of the above mentioned projects demonstrate the importance of task allocation in open and dynamic grid/ cloud environments.

1.3 Research Issues and Objectives of This Thesis

The purpose of this thesis is to study the task allocation in grid/cloud environments and develop solutions to solve the challenging research issues related to the task allocation problem. To do this, the characteristics of the tasks and the features of the grid/cloud environments cannot be ignored to study the research questions of the task allocation. From the introduction of the characteristics and features earlier in Subsections 1.1.1 and 1.1.2, respectively, this PhD study aims to solve the following three research issues.

Issue 1: Task allocation methods have to react to and be resilient from the dynamism and openness of the environments in a timely manner.

Resources in grid/cloud environments can be owned by administratively independent organizations that can decide when to join and leave the environments autonomously, that is, the environment is open. Due in part to the openness of the environments, the resources in such environments are dynamically provisioned and presented. In addition, the constant changing of the capabilities and performances of the resources and the varying of resource requirements of tasks also contribute to the dynamism of the environments. Consequently, in the dynamic and open environments, task allocation have to be resilient from and react to the unpredicted changes in a timely manner.

Issue 2: Task allocation methods should be designed based on local views of agents.

Due to both the independence of all the participants (i.e., consumers and providers) and the distribution of resources, a participant in the grid/cloud environments normally has a local view about the environments. Consequently, task allocation methods have to be designed based on local views of agents.

Issue 3: Task allocation methods should be designed with both the distribution of resources and the large scale requirement of the grids/clouds in mind.

As introduced in Subsection 1.1.2, the resources in grid/cloud environments are distributed over administratively independent organizations. In addition, the large scale requirement of the grid/cloud environments has also to be taken into account when task allocation methods are designed. Therefore, the distribution of resources and the requirement of large scale of grid/cloud systems have to be considered when designing task allocation methods.

Focusing on the above three research issues, this thesis sets the following three

objectives.

Objective 1: to develop methods for the allocation of single tasks without central controllers in open and dynamic grid/cloud environments under time constraints.

Objective 2: to propose and develop decentralised task allocation approaches for the allocation of the complex tasks where each task may consist of more than one subtask. The required resources of the subtasks are distributed over multiple resource providers in open and dynamic grid/cloud environments.

Objective 3: to study the decentralised allocation of the task of which the subtasks have dependency constraints, which means that a subtask cannot be executed until some other(s) is (are) finished, in open and dynamic grid/cloud environments.

1.4 Contributions of This Thesis

Focusing on the three objectives set in the previous section, this thesis has the following contributions.

1. **A Negotiation-based Method for Single Task Allocation with Time Constraints in Open Grid/Cloud Environments**

   In this thesis, a negotiation-based method for the allocation of single tasks, in which each task has to be allocated to only one resource provider under a time constraint, is proposed. The main contributions of the proposed method include that (1) each agent only has a local view of the environments, and (2) both the resource providers and consumers are allowed to enter and leave the environments freely. The proposed method was experimentally evaluated. The experimental results demonstrate that the proposed method can outperform some well-known methods in terms of the success rate of task allocation and the total profit obtained by agents under different time constraints. This negotiation-based method can achieve **Objective 1** of this thesis.

2. **An Auction-based Approach for Group Task Allocation with Time Constraints in Open Grid/Cloud Environments**

   To solve the problem of allocating tasks when each task consists of more than one independent subtask in open and dynamic grid/cloud environments, a combinatorial auction-based approach is proposed. This approach takes the challenges of the decentralization, and the dynamism and openness of the environments into

account through encoding such challenges into an indicator to help consumers comprehensively make the winning providers decision. The experimental results reveal that the proposed approach outperforms two well-known approaches in terms of the success rate of task allocation, the individual utility distribution of the participants, the speed of task allocation, and the scalability, so as to achieve **Objective 2** of this thesis.

3. **A Strategic Max-Sum Belief Propagation-based Method for Group Task Allocation with Dependency Constraints in Dynamic Grid Networks**

   Focusing on the allocation of tasks, each of which consists of more than one interdependent subtask with dependency constraints, a strategic max-sum belief propagation-based task allocation method (SLBP) is proposed. Compared with other max-sum belief propagation-based methods, the distinguishing contribution of SLBP is the quoting strategy, which allows providers to strategically update their quotes according to the continuous changing of the environment. This is important in market-based grid/cloud environments because resource providers always do not want to quote truthfully due to the selfishness of the participants and competitiveness of the environments, rather, they intend to quote strategically to try to gain as many profits as possible.

4. **A Max-sum Belief Propagation-based Method for Efficient Group Task Allocation through Decomposing and Pruning the Grid/Cloud Network**

   Besides SLBP method, another max-sum belief propagation-based task allocation method is developed. This method has the same background but different objectives with SLBP. Unlike SLBP, which focuses on the strategic quoting of providers, this method focuses on the simplification of the belief propagation through decomposing and pruning the grid/cloud network formed by all the resource consumers and providers. The pruning phase can reduce the search space through pruning the providers, which will never optimise the task allocation. The decomposition phase addresses decomposing the grid/cloud network into independent parts where belief propagation can be operated in parallel and thus converge quickly. The distinguishing features of this method include (1) mitigating both the communication and computation requirements for task allocation

in the highly dynamic environments, (2) accelerating the convergence of belief propagation, which is important to the task allocation under time constraints, and (3) improving the online response to and resilience from the unpredicted changing of the environments.

Both **Contributions 3** and **4** can achieve **Objective 3** of this thesis from different perspectives and with different goals.

1.5 The Structure of This Thesis

The remaining chapters of this thesis are organised as follows.

Chapter 2 reviews the current literature, in particular, in regard to the allocation of single tasks, the allocation of tasks when each task consists of a group of independent subtasks, and the allocation of tasks when each task consists of multiple interdependent subtasks with dependency constraints.

Chapter 3 proposes a negotiation-based task allocation method in which resource consumers and providers trade with each other through negotiations to study the allocation of the atomic task that cannot be allocated to more than one resource provider.

Chapter 4 goes further from the work of Chapter 3 to study the allocation of the task, which consists of more than one independent subtask. An auction-based task allocation approach is proposed where the consumer obtains all the required resources from multiple resource providers through a combinatorial auction.

Chapter 5 presents two max-sum belief propagation-based task allocation approaches with the same background but different objectives to study the allocation of tasks when each task consists of more than one interdependent subtask with dependency constraints. The dependency constraint between two subtasks means that a subtask cannot start to be executed until its depending subtask is finished.

Chapter 6 concludes this thesis and outlines the future work.

Chapter 2

Literature Review

Task allocation has been thoroughly studied in grid/cloud environments and there are a large number of task allocation methods developed in such environments.

In the early stage of the grid/cloud environments, task allocation was viewed as a family of problems, depending on the way how the problem is formalised (simulated vs. realistic), task dependencies (independent vs. dependent), the environments (dynamic vs. static), processing mode (batch vs. immediate), the number of objectives to optimised (multi-objective vs. single-objective), etc [KXK09]. From the perspective of objective functions, task allocation mainly includes time optimisation allocations (such as minimising the average latency, and minimising the total execution time makespan) [Fri09], cost optimisation allocations [Fri09], resource utilisation optimisation allocations, etc. Typical allocation mechanisms have four types as follows: First Come First Service (FCFS) allocations [LFC+03], Heuristic allocations [MAS+99, CLZB00], Resource Migration allocations [ZL05], and Load Balancing allocation [MMB03]. For instance, Lee et al. proposed a task allocation algorithm which can adapt to the dynamic grid environments. In their algorithm, each processor has its own waiting queue in which the waiting tasks will be executed by the FCFS metric. When the waiting queue of a processor is empty and the processor is idle, the processor starts to execute the replicas of a task, which is being executed in one of other processors until there is a new task in its own waiting queue [LLC06]. Some well-known heuristic algorithms were also developed for task allocation, including DFPLTF (Dynamic Fastest Processor to Largest Task First) algorithm [DSCB03], Suffrage-C [CLZB00], and Min-min and Max-min algorithms [FGA+98, MAS+99]. Kolodziej et al. proposed a Hierarchic Genetic Strategy (HGS) for independent task allocation on computational grids. In HGS, makespan and flowtime are simultaneously optimised [KXK09].

In this chapter, the literature review concentrates on the recent task allocation methods/approaches in current and developed grid/cloud environments, considering

the new features and challenges introduced in Section 1.1.2 of Chapter 1. As mentioned in Chapter 1, there are three objectives in this thesis, i.e., 1) to develop agent-based methods for the allocation of single (indivisible) tasks in open and dynamic grid/cloud environments, 2) to propose and develop agent-based task allocation approaches for the allocation of the complex (group) tasks when each task may consist of more than one independent subtask, and 3) to study the agent-based allocation of a group of tasks with dependency constraints. Therefore, the related research regarding these three objectives will be reviewed in detail.

The outline of this chapter is as follows. Section 2.1 provides a detailed review of the recent task allocation methods/approaches of single tasks. Section 2.2 gives a detailed review of the current research on group task allocation where independent subtasks belong to a task. Section 2.3 gives a detail review of a group task allocation with dependency constraints.

2.1 Agent-based Allocation Approaches for Single Tasks

A single task can be either a single-issue task or a multi-issue task. An issue here can be either an attribute of a resource/service (such as delivery time, price, and quality) [FSJ02, LLS06, LSL08] or a factor that affects the success of the task allocation. Consequently, task allocations could deal with either single-issue tasks or multi-issue tasks. The multi-issue task allocation is more challenging and complex than the single-issue task allocation in that the solution space of the former is multi-dimensional and it is often difficult to reach a Pareto-efficient solution [LLSG04]. The multi-issue task allocation can happen commonly in grid/cloud environments, such as service-oriented systems and market-based trading systems, and is usually characterised by the situations in which two or more parties recognise that the differences of interests over the issues exist. Meanwhile, the trading parties recognise that the value of cooperation also exists. Consequently, they want to seek a compromised agreement in such situations [Rai82]. For example, a resource provider and a consumer agree on the delivery time and the expected quality of the service, but they may have different prices on the service. In order to gain some values through the cooperation, both the consumer and the provider would like to seek for a compromised agreement over these three issues.

In the remainder of this section, two major categories of agent-based methods/approaches for single task allocation are reviewed in detail.

2.1.1 Learning-based Approaches for Single Task Allocation

Due to the requirement of scalability, the spatial and temporal constraints of the grid/cloud environments, and the lack of the accurate information of resource status at the global scale, resource consumers and providers always only have local views of the environments. In order to achieve high profits and success rates of task allocation, many learning and self-adaptation approaches have been employed for task allocation in grid/cloud environments. For instance, in [NJ06], if an agent comes to a negotiation with a poor reputation (e.g., it frequently reneged on its previous encounters), the agent has to pay a higher penalty fee than one that has shown itself to be more trustworthy when it initialises a decommitment. Consequently, the decision making of agents can also be improved since agents can make more accurate predictions about their opponents' decommitment strategies through learning from the past behaviours of their opponents. Klos et al. [KN09] proposed a similar approach. They developed an Agent-based Computational Economics model (ACE) for market-based task allocation in dynamic environments. In ACE, due to the dynamism of the environments and the competition among agents, agents can adapt themselves based on the trusts on their partners, in response to the changing in the environments.

In [GCL04], the consumer used reinforced learning to choose its desirable resources to maximise its utility. The utility of a consumer varies when being considered from different perspectives, such as the minimised waiting time and the minimised response time. Since consumer agents do not have the global information of resources, they use their past experiences (Q-value learning) to choose the resources to allocate their tasks to. The limitation of their work is that the Q-value learning algorithm is too simple to integrate most of the elements which can affect the execution performance of one resource, and consequently there is a large space to improve the learning algorithm and thus the utility of the resource in grid/cloud environments. Against this, Galstyan et al. further developed a system consisting of a large number of heterogeneous reinforcement learning agents that share common resources for their computational needs [GCL05].

Due to the requirements of decentralisation, local views, and the dynamism of grid and grid-like environments, a large number of heterogenous reinforcement learning agents were devised in the task/resource allocation algorithm, proposed by Galstyan et

al. [GCL05]. In their algorithm, there is not explicit communication between agents, and agents only have local views about the environments. The expected response time of a task is the only reinforcement signal that can be perceived by agents. The used reinforcement learning could achieve desirable load balanced task allocation in large scale heterogenous systems. However, due to the competitiveness and competition of agents, the expected response time of a task may be not truthful, rather, it may be a strategic one of the consumer. Consequently, if the only reinforcement signal that can be perceived by agents, i.e., the expected response time of a task, is not truthful, the execution of the task may not be satisfactory. This is a limitation of their algorithm.

Unlike the learning-based approach in [GCL05], which was mainly motivated by the features of the environments (such as decentralisation and dynamism), the learning-based task allocation approach proposed by Abdallah et al. [AL06] was mainly motivated by the truth that different resource providers might vary in the capabilities of executing the same task. Based on such a motivation, mediators were designed by Abdallah et al. to be responsible for allocating the tasks to the most suitable resource providers to maximise the global reward, with a gradient ascent learning algorithm. Even though the mediator design is useful, their task allocation approach is only suitable for cooperative environments, but not suitable for competitive ones which are the environments that this thesis studies. The reason is that in competitive environments, the self-interested providers may not be willing to be allocated to the most suitable tasks, with respect to their own utilities.

A learning-enhanced market-based task allocation approach for oversubscribed environments was proposed by Jones et al. [GJDS07]. In oversubscribed environments, all the tasks cannot be finished under deadline constraints due to the lack of resources. In their approach, tasks with different urgency and importance levels keep entering the environments. The penalty is charged when a commitment fails. Under these constraints, if a provider agent only handles the importance and urgency of tasks, but does a poor job of anticipating future tasks, a higher value of penalty might be charged. Therefore, before making task commitments, agents have to reason about the future events. To do this, their approach employed a regression-based learning to reduce the overall incurred penalties. Their approach is powerful but limited to oversubscribed environments.

2.1.1.1 Knowledge Learnt from of Learning-based Methods for Single Task Allocation

In competitive grid/cloud environments where the self-interested participants do not reveal their truthful information, learning-based methods are useful to help participants adjust their trading strategies when being used into task allocation methods. However, learning is not always used for task allocation individually, but coupled with other technologies such as negotiations and auctions. For example, in negotiation-based task allocation methods, negotiants can learn the opponents' baselines through their offerings, in order to adjust their own negotiation strategies. In auction-based task allocation methods, bidders can bid through learning the information about the environments and other bidders when such information is not known to them.

2.1.2 Negotiation-based Approaches for Single Task Allocation

Negotiation is a key form of interaction and thus widely used in task allocation of a wide research area encompassing economics, game theory, computer science, and artificial intelligence, and has widely applied in many domains such as electronic commerce, grid computation, and service composition [ALS11]. According to Sim et al. [Sim10], a negotiation is a form of decision making with two or more actively involved agents who cannot make decisions independently (or achieve their goals unilaterally), and therefore, the agents must make concessions to achieve a compromise. Negotiation strategies in multi-issue task allocation approaches mainly contain fixed (pre-defined) negotiation strategies [RAMN$^+$97, WWW98, CDGM97, GM99] and adaptive negotiation strategies [FSJ98, SC03, AGL09, ALIZ10, ALS11]. Fixed negotiation strategies may not be suitable for the open and dynamic grid/cloud environments, due to the constant changes of the environments. In fixed negotiation strategies, agents do not always consider many important issues (e.g., the task deadline, resource competition, etc) when relaxing their offers (count-offeres) at a constant rate [SC01]. In contrary, the adaptive negotiation strategies help agents dynamically make offers (count-offers) in response to the constant changes. The motivation of the agents with adaptive negotiation strategies is to help users make optimal negotiation strategies in the open and dynamic grid/cloud environments. This subsection focuses on the review of current negotiation-based technologies for task allocation along the dimension of the issues

taken into account by agents to allocate tasks.

In the negotiation model for task allocation with time constraints, which are task deadlines in [FSJ98], when doing the decision making of offer (count-offer), agents take both the time passing and behaviours of their negotiation opponents into consideration. In order to reach a consensus on the pricing and other issues of transactions before task deadlines, the two negotiation parties are willing to give more concessions in their bottom lines with the time passage, that is, the concessions are time-dependent. However, in the dynamic and competitive environments, the weakness of their approach is that agents do not take enough important elements into consideration, such as resource competition levels and the reserve prices of negotiation opponents.

The agents in the negotiation task allocation model proposed by Sim et al. [SC03] also make adaptive offer (count-offer) proposals in dynamic markets. Compared with the negotiation strategy in [FSJ98], the negotiation strategy of Sim et al. takes more issues into account. In order to control the ratios of the concessions and make prudent and appropriate compromises, negotiating parties take four issues into account: the eagerness to make a deal, remaining trading time, trading opportunity, and resource competition. In 2009, Sim et al. [SA09] proposed Aggregative Fitness Genetic Algorithms (AFGA), to evolve best response strategies of negotiation agents that optimise their utilities, success rates, and negotiation speed in different market situations. In dynamic grid/cloud environments, it is essential to take the dynamism of the environments into account, whereas most of the existing work only adopted the utility as the unique performance measure of task allocations.

An et al. [AGL09] also proposed a negotiation-based task allocation method. In their method, a consumer agent is allowed to make tentative agreements with more than one provider in order to decrease the risk of the failure of its task when decommitments of tentative agreements initialised by the committed provider agents happen. The consequent problem of this, i.e., how many tentative agreements the consumer should make, was studied. In their method, reserve prices of agents are not fixed, rather, reserve prices are dynamically determined with a time-dependent negotiation strategy. The reserve price of each negotiation thread is dynamically determined by (1) the likelihood that negotiation will not succeed (conflict probability), (2) the expected agreement price, and (3) the expected number of final agreements given the set of tentative agreements made so far. Therefore, the tentative agreement is an advantage because it can decrease the possibility of the failure of consumers' tasks when decommitments happen. However, how many tentative agreements should a consumer

make is a computationally expensive problem to solve, and thus a weakness for task allocation with time constraints.

Based on their previous work in [AGL09], in 2010, An et al. proposed another Negotiation-based Task Allocation method (NTA) by considering more factors [ALIZ10] such as the resource competition, the task deadlines, the reserve prices of the negotiation opponents and the costs of the providers' resources to finish the tasks. In NTA, decomittments are allowed in couple with penalties. The trading party that initialises a decommitment has to pay a penalty to the other party. Unlike many other research where the penalty is decided exogenously, the penalty is negotiated simultaneously with the contract price to adapt to the dynamic environments since it is difficult to decide the optimal contract prices and penalties that can maximise the social welfare in dynamic environments with multiple agents [ALIZ10].

Auction-based approaches can also provide an efficient way to resolve one-to-many negotiations [VJ00], and thus have been widely used in task allocations in grid/cloud environments. An auction is a market institution with an explicit set of rules that are used to determine the task/resource allocation and prices based on the bids from all the market participants [MM87]. Sarne et al. proposed an auction-based task allocation model in open MASs [SK05] where both the resource consumers and providers were self-interested. The allocator in their model is responsible for the allocation of the dynamically arriving tasks using a second price reverse auction as the allocation protocol. However, when the environments is highly dynamic, it always becomes computationally intractable for the allocator to handle the dynamism. To handle this problem, Schoenig et al. proposed sequential Single-Item auctions (SSI) [SP11], to allocate tasks in dynamic environments where not all of the tasks are known at the start of the auction. In their work, different auctioning and winner determination schemes, which include plan modification, re-planning, minimum cost, and regret clearing for winner determination, were used and evaluated. SSI with appropriate auctioning and winner determination schemes can work effectively for task allocation in dynamic environments, under some circumstances. In addition, the task allocation results of their approach in dynamic environments were even improved compared to the task allocation in static environments with the appropriate auctioning and winner determination schemes.

2.1.2.1 Knowledge Learnt from Negotiation-based Approaches for Single Task Allocation

As the two trading parties (i.e., the resource consumer and provider in this thesis) in a negotiation can directly bargain with each other about the issues that they are concerned with, it is flexible for them to dynamically adjust their trading strategies to accommodate to the changing situations in open and dynamic grid/cloud environments. In addition, different from auction-based protocols which always play a major role when the primary concern of participants is the determination of values for task allocations, negotiation-based protocols may be more appropriate when participants are not only concerned with the determining values, but also other factors such as success rates and inter-business relationships [Sim10]. In summary, negotiation-based methods/approaches suit well to the multi-issue (i.e., multi-attribute) task allocation for single tasks. Based on this, a negotiation-based task allocation methods for the allocation of single tasks in open and dynamic grid/cloud environments is proposed in this thesis, which will be introduced in Chapter 3.

2.2 Agent-based Allocation Approaches for a Group of Independent Tasks

It is necessary to allocate a task to a group of resource providers when the task cannot be performed by a single provider. When a task is allocated to a group of providers, the collaboration among these providers may be mutually beneficial even though the providers are selfish and try to maximise their own expected utilities [SK98]. For instance, An et al. addressed a problem of multi-agent task allocation where individual users intend to route traffic by requesting the help of entities across a network [AL10]. A cost will be incurred at each network node that depends on the amount of traffic to be routed. To do the route traffic, the complete assembly of a set of distinct resources is required. There may be more than one combination of distinct resources that can meet the resource requirement. Against this, they developed a contract-based network task allocation method. In their method, each agent takes a myopic best-response strategy to react and interact with other agents to dynamically form contracts. Agents do not predict how their contracting decisions might influence the future decisions of other agents or, more generally, how they might affect the future evolution of the task allocation. Such myopic behaviours are used a lot in large MASs

where agents are resource bounded or have limited information about the incentives of others.

Two types of mehtods/approaches have been widely used to allocate tasks in which each task consists of a group of independent subtasks. These two types of method/approaches are coalition formation-based methods/approaches and multi-resource negotiation/combinatorial auction-based methods/approaches. The following two subsections review these two types of methods/approaches, respectively.

2.2.1 Coalition Formation-based Allocation Approaches for Group Task Allocation

Coalition formation is important for the allocation of the group of independent (sub)tasks, since the collaboration of multiple resource providers, which can form a coalition to perform the task, is required.

Kraus and Shehory developed a coalition formation approach [KST03] in which a coalition is treated as an entirety. In their approach, no member in a formed coalition is allowed to leave until the task allocated to the coalition is finished. This restriction is not realistic in dynamic and open grid/cloud environments since in such environments, both the resource providers and consumers are autonomous and own the rights to decide when to leave or join the environments.

Aiming at improving the adaptiveness and flexibility of prevailing static web service composition, Mller et al. proposed a multi-agent-based coalition formation approach [KB+06] for service composition. Their approach can achieve emergent behaviours from a group of agents based on a light-weight interaction protocol and decentralised decision makings. In their approach, coalitions of multiple agents emerge from interactions between these agents based on limited knowledge and local autonomous decision-makings.

Yang et al. proposed a genetic algorithm-based coalition formation mechanism for group task allocation in MASs [YL07]. Their mechanism can achieve the efficient task allocation through the self-adaptation of agents. Ye et al. developed a self-adaptation-based dynamic coalition formation mechanism in dynamic networks [YZS13]. The agents in their mechanism are not fully connected but can communicate and form coalitions only with their neighbouring agents. In particular, agents can dynamically adjust their degrees of involvements in different coalitions to achieve efficient task allocation, due to the dynamism of the environments. Their mechanism is more flexible

to work in different domains, especially for large scale systems.

Besides the coalition formation, sometimes Virtual Organizations (VOs) also need to be formed emergently for the execution of unpredicted large-scale tasks, in the open and dynamic grid/cloud environments. Focusing on this problem, a mechanism of VO formation was designed by Mashayekhy et al. [MG14]. The contribution of their mechanism is that it can be guaranteed that the formed VOs are stable, which means that the participants in a VO do not have incentives to break away from the current VO to join some other one. However, the stability of the formed coalition is also a limitation of their approach in that it may make their approach not suitable for task allocation in open and dynamic environments, where new tasks and providers keep coming into and thus more profitable coalitions need to be formed.

Carroll et al. modelled the VO formation into the coalition formation problem [CG10]. In their work, before making the coalition formation decisions, grid resource providers evaluate the values of each coalition that they can form with others, using a heuristic scheme called MinCost. MinCost is a fully polynomial time approximation scheme (FPTAS) which is used to map tasks to service providers to satisfy the constraints of both the deadlines and costs. Apparently, their approach is suitable for cooperative environments but not for competitive ones because resource providers only consider the global value when making decisions.

2.2.1.1 Knowledge Learnt from Coalition Formation-based Allocation Approaches for Group Task Allocation

As the resource providers in grid/cloud environments are autonomous, they can form coalitions with others according to their own preferences. In other words, a provider owns the right to decide which coalition to join. However, when there are too many coalitions that a provider can join, it might become computationally expensive or even intractable for the provider to decide which coalition to join, and this problem becomes more serious in dynamic environments where providers keep coming and leaving.

2.2.2 Negotiation/Auction-based Allocation Approaches for Group Task Allocation

Since resource providers and consumers may have different preferences, goals, policies, and interests, to negotiate an optimal allocation of tasks within a group of agents is always intractable in both computation and communication. Against this, An et

al. proposed a multi-resource negotiation-based task allocation approach [ALS08] in which the reserve price of each negotiation resource is dynamically determined by 1) the expected agreement price of the resource, 2) the likelihood that negotiation will not succeed (conflict probability), and 3) the expected number of final agreements given the set of tentative agreements made so far. An et al. further developed a simultaneous multi-resource negotiation approach to allocate tasks in market-based environments [ALS11]. In their approach, a consumer negotiates with providers for each of the required resources separately, and consequently, there are multiple concurrent negotiation threads. The consumer can adjust its negotiation strategy in a negotiation thread according to the statuses of other threads. However, in open and dynamic environments where providers keep coming and leaving, the adjustment becomes computationally expensive or even intractable, so this is a limitation of the separate negotiation.

Due to its perceived fairness and allocation efficiency, auction theory has been proved to be a useful and powerful tool in the distributed problems (such as communication and resource management, and networking [HBH06, PKCD07, FK10]) including the task allocation in grid/cloud environments. There are many different auction types such as English Auction, Vickrey Auction, First-Price Sealed-Bid Auction (FPSBA), and Dutch Auction.

A combinatorial auction-based protocol for resource/task allocation in grid environments was proposed by Das et. al [DG05]. In their protocol, resource consumers bid for the required resources of their tasks. The consumers place a value price for each of the combinations of all the required resources. An auctioneer was used to do the winner determination of the bids. The auctioneer can generally result in the centralisation of task allocation and thus the auctioneer is a limitation of the task allocation approach, especially when there are too many possible resource combinations for each task.

Among the various types of auctions, combinatorial reverse auctions are used commonly for the group task allocation in grid/cloud environments. In [PKCD07], the combinatorial reverse auction was used to formulate the resource scheduling problem in multi-rate wireless systems in which users compete against each other to sell a set of slots to the base station. Edalat et al. modeled the concurrent applications of wireless sensor networks into a scheme based on combinatorial reverse auctions [EXR+11]. In their scheme, bidders bid cost values for accomplishing the applications' tasks. The objective of their scheme is to maximise the network lifetime by sharing tasks and the network resources among applications while enhancing the overall QoS. However, their approach needs a public auctioneer to run the auction, and thus are centralised. The

public auctioneer always causes overloads of both the communication and computation and suffers from the single point of failure problem. It is always undesirable to introduce cental controllers into any system with a goal of high availability or reliability (e.g., business practices, software applications, or other industrial systems [Doo01]). In addition, in the competitive grid/cloud environments, it is hard for the self-interested bidders to trust the auctioneer and for the auctioneer to acquire truthful information about the self-interested bidders [ALIZ10].

Since the execution of a task needs the collaboration of multiple providers in a group task allocation, normally it is NP-hard to pick out a group of winning bidders (also known as 'group formation') in combinatorial auctions. Integer programming was adopted by Giovannucci et al. to solve this problem [GVRAC08]. However, the integer programming faces the problem of scalability. Even though Giovannucci et al. tried to improve the scalability problem caused by integer programming through taking the structural properties of networks into account, the improvement is limited and consequently, the scalability still remains an ongoing problem.

2.2.2.1 Knowledge Learnt from Negotiation/Auction-based Approaches for Group Task Allocation

The multi-resource negotiation-based task allocation methods/approaches generally have three limits. First, the separate negotiation treads always cause an invalid bundle of resources. The reason is that the obtained bundle of resources will become invalid when the follow-up negotiations for the complementary resources fail. Second, the separate negotiations always result in a large number of providers being selected to perform a task, and this may result in a communication overload among the selected providers in some situations. Third, since the negotiation threads in multi-resource negotiation are all in uncertainties and may affect one another, when the environment is highly dynamic, it may become computationally intractable for a consumer to adjust the interrelated negotiation threads.

Combinatorial reverse auction-based approaches are more suitable for and widely used in the allocation of the task, which consists of a group of independent subtasks, than negotiation-based approaches. The reason for this is that in combinatorial reverse auctions in which resource providers bid to win some subtasks to perform, resource providers always place bids for all-or-nothing bundles of subtasks. Consequently, the problem of bundles of invalid resources obtained by the consumer, which is a big

problem in multi-resource negotiation-based task allocation methods/approaches, can be mitigated in combinatorial (reverse) auction-based methods/approaches. Based on this, a combinatorial auction-based group task allocation approach is proposed in this thesis, which will be in detail introduced in Chapter 4.

2.3 Agent-based Allocation Approaches for Group Tasks with Dependency Constraints

In many applications such as supply chain formation and Distributed Constraint Optimisation Problems (DCOPs), a task can consist of a group of interdependent subtasks that have dependency constraints, which means that the subtasks have to be performed in a determined sequence. For simplicity, the allocation of such a type of tasks is referred to group task allocation with dependency constraints in the remainder of this thesis.

The widely developed methods/approaches for group task allocation with dependency constraints include combinatorial auction-based methods/approaches, DCOP-based methods/approaches, and belief propagation-based methods/approaches.

2.3.1 Combinatorial Auction-based Approaches for Group Task Allocation with Constraints

The technology of combinatorial auctions is widely used for group task allocations with dependency constraints. Walsh et al. developed a combinatorial auction-based approach for supply chain formation [WWY00]. Their approach aims at assembling suitable resource providers to form a supply chain to finish the group of subtasks with dependency constraints of a task. Their approach allows for strategic quoting of resource providers and concentrates on the quoting strategies.

In order to eliminate the central controller and improve the performance of their previous work, in 2003, Walsh and Wellman further proposed an asynchronous and decentralised market protocol for supply chain formation with resource scarcity [WW03]. In their protocol, the participants form the supply chain through negotiating in a bottom-up fashion, requiring only local views of the environments. The prices of agents are coordinated by the price system in which the price for each resource is determined through an ascending auction. Walsh and Wellman overcame the centralization of their

work in [WWY00] by auctioning for each of the required resources separately. However, the consequent decentralisation is acquired in compromise of consumers' risking in obtaining a bundle of invalid resources.

A Consensus-Based Auction Algorithm, CBAA, and its generalisation, the Consensus-Based Bundle Algorithm CBBA, were proposed by Choi et al. [CBH09] to address the task allocation problem in coordinating a fleet of autonomous vehicles. CBAA is a consensus-based auction algorithm in which only one task can be allocated to a single resource provider. CBAA was then extended to a multi-assignment problem in which a sequence of tasks can be allocated to a single provider by developing CBBA. Since the task allocation problem in both CBBA and CBAA is to find a conflict-free matching from a list of tasks to a given set of resource providers, the tasks to be allocated are considered to have resource constraints if a predefined global reward function needs to be optimised.

2.3.1.1 Knowledge Learnt from Combinatorial Auction-based Approaches for Group Task Allocation with Constraints

Combinatorial auctions in group task allocation with constraints always introduce central controllers (i.e., the auctioneers) which hinder the scalability of the global optimisation. In addition, it is hard for the auctioneer to decide when to start the auction in the open and dynamic environments where both the consumer and provider sets constantly change. Additionally, due to the dependency constraints among subtasks, an auctioneer has to coordinate the time and sequences of the winning providers to deliver services (i.e., to accomplish the subtasks that they win in the auction). However, in a large scale environment, it is computationally and communication expensive or even intractable for the auctioneer to do this. For these reasons, combinatorial (reverse) auctions do not suit well to the group task allocation with constraints among subtasks.

2.3.2 DCOP-based Algorithms for Group Task Allocation with Constraints

The group task allocation with dependency constraints is always modelled as DCOPs [OF08]. The DCOP is a general problem representation for task allocation in MASs. The DCOP has, for a long time, been considered as an important research area for group task allocation with constraints in MASs since a vast number of real-world

applications (e.g., sensor networks, traffic flow cooperation, and event scheduling) can be modeled by it.

The asynchronous distributed optimisation algorithm, ADOPT, was a complete DCOP algorithm proposed by Domi et al. [MSTY05] to solve the DCOP problem. Compared with existing DCOP algorithms, ADOPT can provide strong guarantees of global quality of solutions while allowing agents to execute concurrently and communicate asynchronously. As the agents in ADOPT can execute their tasks concurrently, ADOPT is more efficient than existing algorithms in terms of the required time to find the globally optimal solution for task allocation.

Compared with ADOPT in [MSTY05], the novel distributed task allocation algorithm LA-DCOP [SFOT05], which is an incomplete DCOP algorithm, works better in terms of the communication requirement. In order to reduce the communication requirement for task allocation, tokens were used to represent the tasks to be executed in LA-DCOP. In addition, tokens were created to deal with inter-task constraints of the simultaneous execution of tasks.

ADOPT was also improved from the perspective of the speed to obtain the solution, by Ali et al. who accelerated ADOPT by introducing different preprocessing techniques [AKT05]. The preprocessing techniques used the dynamic programming to calculate informed lower bound cost estimates for ADOPT. The work of Ali et al. can accelerate ADOPT by an order of magnitude, at a relatively low preprocessing cost.

Normally, DCOPs only focus on small (< 100 variables) and deterministic problems. Against this, Atlas et al. proposed an algorithm, Distributed Neighbour Exchange Algorithm (DNEA), to solve large-scale DCOPs [AD10]. They developed DNEA after finding that the existing algorithms for DCOPs can in fact run on large-scale problems, but always took many message passing cycles to converge to solutions. DNEA is a local neighbourhood search algorithm in which all neighbours exchange the potential gains for different task assignments (configurations) in each cycle. Agents in DNEA exchange current variable assignments with their neighbours, compute a maximisation function based on neighbouring assignments, and then accordingly update the local variable assignment information [AD10].

2.3.2.1 Knowledge Learnt from DCOP-based Algorithms for Group Task Allocation with Constraints

When a task consists of a group of interdependent subtasks with dependency constraints, the allocation of such a group of subtasks are always modelled as D-COPs. However, the DCOPs suit better to such a type of task allocation in cooperative environments (such as Robocup Rescue (RCR) [RFMJ10], sensor networks [RCJ09, SFRJ09]), compared to competitive grid/cloud environments. The main reason for this is that normally, the task allocation objective function in DCOPs is to maximise the global reward or to minimise the global cost, while ignoring the individual rewards or costs. Consequently, DCOPs do not suit well to the group task allocation with dependency constraints in competitive grid/cloud environments.

2.3.3 Belief Propagation-based Algorithms for Group Task Allocation with Constraints

Belief propagation, also known as sum-product message passing, is an algorithm used to perform an approximate inference through message passing on graphical models. Pearl expressed his concern that belief propagation in loopy networks, which was referred as Loopy Belief Propagation (LBP), might not converge [Pea88]. However, in the end of last century, LBP had been empirically shown to be a highly distributed competitive inference algorithm and had seen a great success as a general approximate inference algorithm [MWJ99] on Bayesian Networks, Markov Random Fields, etc [CP02]. Since this century, LBP has been becoming a leading algorithm for approximate inference due to both its strong empirical performance and ease of implementation [CP02]. As a variant of belief propagation, max-sum (the log-domain max-product) belief propagation is a local-message-passing algorithm guaranteed to converge to the neighbourhood maximum [WYM12]. Max-sum belief propagation has been a great success and widely used in group task allocation with constraints, such as Robocup Rescue (RCR) [RFMJ10], sensor networks [RCJ09, SFRJ09, CP02], and low-power embedded devices [FRPJ08]. For instance, in [FRPJ08], Farinelli et al. used an extension of the standard max-sum algorithm [AM00] to generate approximate solutions to the RCR problem through the decentralised local message passing between interacting agents.

In 2010, being inspired by the Task Dependency Network (TDN) model [WW03]

devised by Walsh and Wellman, Winsper and Chli encoded group task allocation with constraints in competitive environments into a TDN model where the belief propagation was adopted to allocate the interdependent tasks [WC10]. The usage of belief propagation (message passing) allows their approach to take full advantage of the graphical structure of the networks and thus can work the same as centralised approaches whilst still working in a decentralised manner. Similar to many other existing belief propagation-based task allocation approaches, the application of their approach is limited by the huge requirements of both memory and communication (message passing).

In 2012, the work of Winsper and Chli in [WC10] was improved by Penya et al. in both memory and communication requirements of belief propagation [PAVCRA12] through encoding the supply chain formation into a binary factor graph in which all the nodes only have two possible states. However, neither the work in [WC10] nor the work in [PAVCRA12] takes the important characteristics of the competitive environments (such as resource competition and agents' strategic offer (count-)) into account. Additionally, both of the two methods were designed for static environments where agents remain unchanged through the whole process of task allocation. However, this is not always realistic in real life applications.

Based on the existed fast-max-sum proposed by Ramchurn et al. in [RFMJ10], Macarthur et al. proposed a branch-and-bound fast-max-sum algorithm (BnB FMS) [MSRJ11] in the competitive and open environments where both the tasks and agents constantly change. BnB FMS consists of two interleaved sub-algorithms. The first sub-algorithm aims at narrowing the searching task space that an individual agent should consider, through on-line pruning some tasks using fast-max-sum algorithm. This accommodates well to the dynamism of the environments. The second sub-algorithm prunes the state space of an individual agent using branch-and-bound search. There are only two states in the pruned state space for each agent and task. The two states are 1) allocating the agent to the task, and 2) not allocating the agent to the task. BnB FMS significantly reduced the requirements of both the runtime and communication (i.e., message passing), compared to the work of Ramchurn et al. in [RFMJ10].

The decentralised supply chain formation method proposed by Penya-Alba and Vinyals in [PA13], Reduced Binary Loopy Belief Propagation-based (RB-LBP) task allocation method, focused on improving the performances of task allocation in terms of memory, communication, and computational requirements, and was experimentally proved successful. RB-LBP encoded the task allocation model into a factor graph where

all the variable nodes are binary ones. When the number of participants increases, the requirements of the communication, memory, and computation of RB-LBP scale linearly, whereas such requirements of the belief propagation-based method proposed by Winsper et al. in [WC13] scale exponentially when the market competition is very high. RB-LBP dramatically improved the scalability of the global optimisation performed by loopy belief propagation.

2.3.3.1 Knowledge Learnt from Belief Propagation-based Algorithms for Group Task Allocation with Constraints

Belief propagation is well suitable for the allocation of the task which consists of a group of interdependent subtasks with dependency constraints. One of the reasons for this is that it is easy to convert the dependency between subtasks, and the relationships between subtasks and resource providers into bipartite graphs. In the bipartite graphs, nodes represent the subtasks and providers, and each edge connecting a subtask and a provider represents allocating the subtask to the provider. In addition, the decentralisation and distribution of belief propagation in the bipartite graphs are important characteristics of the realistic relationships between administratively independent resource providers and consumers. Moreover, belief propagations can produce good approximation solutions even in loopy graphs in a quick and reliable manner [WC13]. Based on these reasons, two belief propagation-based task allocation approaches for the group tasks with dependency constraints are proposed in this thesis, which will be introduced in Chapter 5.

2.4 Summary

In this chapter, the current literature regarding the research concerns of this thesis was thoroughly reviewed. Specifically, the literature review concentrated on the studies on task allocation for single tasks, task allocation for group tasks, and task allocation for group tasks with dependency constraints.

Chapter 3

A Negotiation-based Method for Task Allocation with Time Constraints in Open Grid/Cloud Environments

In this chapter, a negotiation-based task allocation method (NTAL) for single (indivisible) tasks is proposed. The proposed method concentrates on the offer (count-offer) strategies of both the resource consumers and providers in the negotiation during task allocation. This chapter is organised as follows. Section 3.1 presents the problem description and related definitions, and the proposed task allocation method is in detail introduced in Section 3.2. In Section 3.3, the method is experimentally evaluated. A discussion is given in Section 3.4, and Section 3.5 summarises this chapter.

3.1 Problem Description and Definitions

The grid/cloud environments in the proposed negotiation-based task allocation method are open and dynamic environments in which nodes (grids) are connected in a network and tasks have time constraints. A node could be a computer, a computation station, a digital library, or a database, etc. Each node is modelled as an agent that can do the decision-making for itself. An agent sends a request message (see **Definition 3.3**) to some randomly chosen agents to construct neighbourhoods with them when it first enters the environment, and is limited to communicating with its neighbours. If an agent needs other agents' resources to execute its tasks, it is a consumer; and if it provides its resources to others, it is a provider. In the proposed method, it is possible that an agent can provide resources to other agents, and meanwhile, it needs others' resources to execute its own tasks, thus, an agent can be a consumer or a provider, or even both. In the environment, all of the consumers and providers are self-interested. When allocating its tasks to providers, a consumer should consider the time when its task can be finished and the reward that it can gain from the task when the task is

finished, and the provider should also consider the cost of its resources to finish the consumer's task and the reward it can gain from the consumer when it finishes the task. Thus, the key parameters that could drive the grid design include the numbers of providers, consumers and tasks in the grid environment, the deadlines of the tasks, the cost of the provider's resources and the rewards that the consumer and the provider can gain when the task is finished.

Definition 3.1 (Agent): Agent a_i is defined by a 3-tuple (ID_i, R_i, Set_i), where ID_i (a non-negative integer) is the unique identifier of a_i, R_i is the resource set that a_i owns, and $Set_i = \{a_{i1}, \ldots, a_{ik}\}$ is a_i's neighbour set where k (a positive integer) is the number of neighbours of a_i.

It should be noted that an agent has much more 'intelligence' than what is defined above. However, only the intelligence required and used in my work is defined in this thesis, including in Definition 3.1 and Definition 5.2.

In a grid environment, normally there is no central controller and all the agents are equal. When an agent enters the grid environment, it randomly picks up a certain number of agents from the environment to construct neighbourhoods (the construction of neighbourhoods will be described in detail later). Because a grid environment is dynamic and open, existing agents can leave the environment and new agents can enter the environment freely. Consequently, the neighbours of an agent keep changing. Each agent only has a local view of its neighbours. The agents that have possession of resources can provide their resources to other agents and require payments from those agents. The agents that need others' resources to execute their own tasks have to give payment to the resource providers.

Definition 3.2 (Task): A task, denoted as τ_k, is a 5-tuple $(R_k, t_g, t_{ls}, \overline{r}, t_d)$, where R_k is the resource set required by τ_k, t_g is the generation time of τ_k, t_{ls} is the deadline (i.e., the latest start time) of τ_k, $\overline{r}$ represents the maximal reward that τ_k's owner can gain after τ_k is completed successfully, and t_d is the duration time of τ_k.

Communication between any two agents is through passing messages. In general, messages can be classified into five types. (i) Request messages for constructing a neighbourhood (*ReqNeighbour*). (ii) Reply messages for accepting *ReqNeighbour*. (iii) Request messages for executing tasks (*ReqExecute*). (iv) Reply messages for executing tasks (*RepExecute*). (v) Heartbeat messages (*HeartBeat*). These five types of messages are formally defined as follows.

Definition 3.3 (Request message for constructing a neighbourhood): A request message $ReqNeighbour_{ij}$, sent from agent a_i to agent a_j for constructing a

neighbourhood is a 2-tuple ($ReqNeighbour_{ij}, ID_i$), where $ReqNeighbour_{ij}$ represents the message sent from agent a_i to a_j to request constructing a neighbourhood with a_j, and ID_i is the ID of agent a_i.

When agent a_j receives a request message ($ReqNeighbour_{ij}$, ID_i) from a_i, it replies a_i with a reply message defined as follows:

Definition 3.4 (Reply message for constructing a neighbourhood): A reply message $RepNeighbour_{ji}$, sent from a_j to a_i to reply to the message ($ReqNeighbour_{ij}, ID_i$) is a 2-tuple ($RepNeighbour_{ji}, ID_j$), where $RepNeighbour_{ji}$ represents that the message is sent by agent a_j to reply to the request message sent from a_i and ID_j is the ID of agent a_j.

Due to the nature of the dynamic and open grid environments, in the proposed method there is no central controller. The agents judge whether their neighbours are still active in the environment through heartbeat messages. A heartbeat message is defined as follows:

Definition 3.5 (Heartbeat message): A heartbeat message $HeartBeat_{ij}$, sent from a_i to a_j is a 3-tuple ($HeartBeat_{ij}$, ID_i, ID_j), where ID_i and ID_j are the ID numbers of a_i and a_j, respectively.

In particular, an agent keeps sending heartbeat messages to its neighbours once in each time period. If an agent has not received any heartbeat message from a neighbour in the past one unit time, it assumes that the neighbour has left the grid environment.

The above three definitions, i.e., Definitions 3.3, 3.4 and 3.5 are the definitions of messages for neighbourhood construction, while the following two definitions, i.e., Definitions 3.6 and 3.7 are the messages for executing tasks.

Definition 3.6 (Request message for executing a task): A request message $ReqExecute_{ij}$, sent from a_i to a_j for executing a_i's task is a 4-tuple ($ReqExecute_{ij}, ID_i$, τ_k, HL), where $ReqExecute_{ij}$ represents that the message is sent from agent a_i to request a_j to execute task τ_k of a_i. $HL \geq 1$ is a hop limitation, which prevents the request message from being transferred endlessly.

Definition 3.7 (Reply message for executing a task): A reply message $RepExecute_{ji}$ sent from a_j to a_i for replying to the message ($ReqExecute_{ij}, ID_i$, τ_k, HL) is a 4-tuple ($RepExecute_{ji}, ID_j$, t_s, τ_k), where $RepExecute_{ji}$ represents that the message is sent from a_j to a_i to reply to the message ($ReqExecute_{ij}, ID_i, \tau_k, HL$) from a_i, t_s is the start time for a_j to execute τ_k and t_s has to meet the condition that $t_s \leq t_{ls}$, where t_{ls} is the deadline of the task.

It is assumed that a new task denoted by τ_k is generated by agent a_i and a_j is

one of a_i's neighbours. a_i sends a request message $(ReqExecute_{ij}, ID_i, \tau_k, HL)$ to a_j. After receiving the request message, a_j will check whether its own resource set R_j can meet the resource requirement of τ_k. If $R_k \subseteq R_j$ (R_k is the requested resource set by τ_k, see **Definition 3.1**), a_j will send a reply message $(RepExecute_{ji}, ID_j, t_s, \tau_k)$ back to a_i. Otherwise, a_j will check whether HL equals to 0. If so, a_j does nothing to this request message, while if $HL \geq 1$, the request message will be transferred by a_j to a_j's neighbours. If a request message is transferred, it is re-assembled by the intermediate agents, which means that the intermediate agents will change some information in the request message. It is assumed that a_j is going to transfer the message $(ReqExecute_{ij}, ID_i, \tau_k, HL)$ to a_j's neighbour a_m. The receiver's ID, i.e., j, is replaced by the ID of the new receiver of the message, i.e., the ID of a_m. Rather, the sender's ID remains unchanged, that is, the sender's ID is still the consumer's ID, i.e., i. In addition, every time the request message is transferred, HL will be reduced by 1. Thus, before the request message $(ReqExecute_{ij}, ID_i, \tau_k, HL)$ is transferred by a_j to a_m, the request message will be changed into $(ReqExecute_{im}, ID_i, \tau_k, HL-1)$. The transfer process will be terminated once the value of HL becomes 0 or the receiver's resource set can meet the resource requirement of τ_k. If a provider can meet the resource requirement of a consumer, but is not the direct neighbour of the consumer, then the provider has to construct a neighbourhood relationship with the consumer first before sending a reply message straight to the consumer.

If a_i receives a reply message $(RepExecute_{ji}, ID_j, t_s, \tau_k)$ for executing a task from a_j, a_i will begin to negotiate with a_j. The negotiation method will be specifically described in Section 3.2. When the negotiation succeeds, it is possible that both the parties will sign a contract, which is an agreement between a_i and a_j for executing τ_k, and the signing of a contract will be discussed in Section 3.2.2.

Definition 3.8 (Contract): A contract con_{ij} between a_i and a_j is a 6-tuple $(\tau_k, pr_{con}, t_s, t_{con}, pro_i, pro_j)$, where pr_{con} is the price that a_i should pay to a_j, t_s is the start time for a_j to execute τ_k, t_{con} is the time that the contract is signed, and pro_i and pro_j are the profits that a_i and a_j gain from this contract, respectively.

In the proposed method, an agent is allowed to negotiate and sign contracts with more than one opponent in order to get a better allocation for the task. Because only one provider can eventually execute the consumer's task, thus, decommitment is allowed in the proposed method before the task is executed. The agent that initially decommits from the contract has to pay a penalty to the other party. Furthermore, the later the decommitment, the higher the penalty will be because of the pressure of

the deadline of the task.

Definition 3.9 (Penalty): The penalty pel_{ij}, that agent a_i pays to a_j if a_i decommits from the contract $con_{ij}(\tau_k, pr_{con}, t_s, t_{con}, pro_i, pro_j)$ is calculated by Equation (3.1).

$$pel_{ij} = \frac{t - t_{con}}{t_s - t_{con}} pro_j, \tag{3.1}$$

where t is the time of the decommitment, and $t_{con} \leq t \leq t_s$.

Because the role of an agent can be either a consumer or a provider or even both, the profit of an agent is defined as follows:

Definition 3.10 (Profit): The profit pro_i that agent a_i gains totally is calculated by Equation (3.2).

$$pro_i = \sum_{i=1}^{m}(r_i - p_i) + \sum_{l=1}^{n}(p_l - c_l) + \sum_{j=1}^{k} pel_{ji}, \tag{3.2}$$

where m is the number of successfully allocated tasks of a_i, r_i is the reward a_i gains from task τ_i and p_i is the price that a_i pays to the executor of τ_i; n is the number of tasks that are successfully allocated to agent a_i, p_l and c_l are the price that the owner of τ_l pays to a_i for executing τ_l and the cost that a_i spends to execute τ_l; pel_{ji} is the penalty that a_j pays to a_i, and k is the number of agents that pay penalties to a_i and a_i pays penalties to. If a_i pays a penalty to a_j, pel_{ji} is a negative number, while if a_j pays a penalty to a_i, pel_{ji} is a positive number.

Based on all the definitions given above, the task allocation procedure of the proposed method is described as follows.

Step 1: Constructing neighbourhoods of newly arrived agents.

The procedure of constructing a neighbourhood of newly arrived agents is described by Algorithm 3.1.

Algorithm 3.1: Neighbourhood Construction

a_i arrives the grid environment and selects 5 agents from the grid randomly.
It is assumed that the 5 selected agents are a_0, a_1, a_2, a_3, a_4.

1 while (j=0; j<5; j++)
2 a_i sends a *ReqNeighbour* (see **Definition 3.3**) to a_j.
3 a_j sends back a reply message *RepNeighbour* (see **Definition 3.4**) to a_i.
4 a_i and a_j become neighbours and keep the neighbourhood through heartbeat
5 messages (see **Definition 3.5**).
6 end while

In Algorithm 3.1, when an agent represented by a_i first enters the grid environment,

it selects several agents randomly to send request messages for constructing a neighbourhood (see **Definition 3.3**) to the selected agents (**Lines 1-2**). After receiving the request message for constructing the neighbourhood, the message receiver, denoted by a_j, sends back a reply message for constructing a neighbourhood (see **Definition 3.4**) to a_i (**Line 3**). Then a_i and a_j become neighbours and keep sending heartbeat messages (see **Definition 3.5**) to each other (**Lines 4-5**), until one of them leaves or both of them leave the grid environment.

Step 2: Looking for potential resources for consumers.

When a new task represented by τ_i is generated, the owner agent of τ_i which is represented by a_i, sends request messages for executing τ_k (see **Definition 3.6**) to all of its neighbours. It is assumed that a_j is one of the agents that receive the message. After receiving the request message, a_j checks whether its own resources can meet the resource requirement of τ_i. If so, a_j sends back a reply message (see **Definition 3.7**) to a_i. Otherwise, a_j transfers the request message to all of a_j's neighbours.

Step 3: Negotiation.

After a_i receives a reply message for executing task τ_i from a_j, a_i and a_j begin to negotiate with each other. The negotiation strategy will be specifically described and formulated in Section 3.2.

The task allocation procedure given above can be described from both the perspectives of a consumer and a provider, which are described by Algorithms 3.2 and 3.3, respectively.

Algorithm 3.2: Task Allocation Procedure From the Perspective of A Consumer

```
1 If agent a_i newly arrives in the grid environment then
2   a_i calls Algorithm 3.1, that is, the algorithm of neighbourhood construction.
3 for j=1 to n
4 |  a_i sends a request message, (ReqExecute_ij, ID_i, τ_k, HL), to a_j;
5 |  If a_i receives a reply message, (RepExecute_ji, ID_j, t_s, τ_k), from a_j then
6 |  a_i and a_j begin to negotiate;
7 end for
```

In Algorithm 3.2, it is assumed that τ_k is a task that is generated by agent a_i. R_k is the required resource set by τ_k and t_{ls} is the deadline of τ_k. If a_i newly comes to the environment, it constructs its neighbourhood first (**Lines 1-2**). Suppose that a_i has n neighbours, and $a_1, a_2, ..., a_n$ are used to represent the n neighbours of a_i. a_i sends a request message for executing τ_k (see **Definition 3.6**) to all of its neighbours (**Lines 3-4**). If a_i receives a reply message from some provider represented by a_j, a_i starts to negotiate with a_j (**Lines 5-6**).

Algorithm 3.3: Task Allocation Procedure From the Perspective of A Provider

```
1  If provider agent a_j newly arrives in the grid environment then
2    a_j calls Algorithm 3.1, that is, the algorithm of neighbourhood construction.
3   When a_j receives a request message (ReqExecute_ij, ID_i, τ_k, HL) from consumer a_i
4    a_j checks its resource set R_j;
5    if R_k ⊆ R_j and t < t_ls then
6    |  if a_j is not the direct neighbour of a_i then
7    |  |  a_j constructs neighbourhood with a_i;
8    |  end if
9    |  a_j sends back a reply message (RepExecute_ji, ID_j, t_s, τ_k) to a_i;
10   |  a_i and a_j begin to negotiate;
11 else if HL > 0 and t < t_ls then
12   |  HL = HL − 1;
13   |  a_j transmits the request message sent from a_i to all of its own neighbours;
14 end if
```

Similar to a_i in Algorithm 3.2, In Algorithm 3.3, if provider a_j newly comes to the environment, it also constructs its neighbourhood first (**Lines 1-2**). When a_j receives a request message $(ReqExecute_{ij}, ID_i, \tau_k, HL)$ from consumer a_i, it checks whether it can meet the resource requirement of τ_k (**Lines 3-4**). If yes, but a_j is not a direct neighbour of a_i, a_j constructs a neighbourhood with a_i before sending a reply message for executing τ_k (see **Definition 3.7**) to a_i (**Lines 5-8**). If a_j can meet the resource requirement of τ_k, and is a direct neighbour of a_i, it sends a reply message for executing τ_k to a_i (**Line 9**). a_i and a_j starts to negotiate (**Line 10**). When a_j cannot meet the resource requirement and HL in the message does not equal to 0, a_j transfers the request message to its own neighbours (**Lines 11-14**). It should be emphasised that when the request message is transferred by a_j, the ID of the request message's sender remains unchanged, which means that the ID of the message's sender is still the ID of the original sender of the message, i.e., a_i, while the ID of the destination agent of the message is replaced by the ID of a_j's neighbour. From **Lines 11-12**, it can be seen that Algorithm 3.3 is controlled by the value of HL, and stops once HL becomes 0.

3.2 The Negotiation-based Task Allocation Method

This section presents the proposed negotiation strategy for task allocation in detail. Each consumer in the proposed method is limited to communicating with its neighbours, and when a consumer has a task to be allocated, it only sends a request message (see **Definition 3.6**) to its neighbours. When an agent receives a request message for

executing tasks from one of its neighbours, it checks its own resources. If the agent cannot meet the resource requirement in the request message, and *HL* in the message does not equal to 0, the agent transfers the request message to its own neighbours. Thus, the eventual providers that a consumer negotiates with might be its direct neighbours or indirect neighbours, or both. Obviously, the number of providers that a consumer negotiates with equals to the number of reply messages (see **Definition 3.7**) that this consumer received, which is mainly affected by both the number of the consumer's neighbours and the value of HL.

The possible actions that an agent can take during the course of a negotiation are introduced in detail in Subsection 3.2.1. The offer generation method and negotiation strategies of providers and consumers are introduced in Subsection 3.2.2.

3.2.1 Actions of an Agent

The following are the possible actions that an agent can take during the course of a negotiation.

(i) Create an offer : A consumer gives its offer price to a provider.

In a negotiation thread, the consumer gives its offer price and the provider gives its counter offer price alternately. After the start of a negotiation, the consumer first sends a message which contains its offer price for the provider's resources to the provider, and the offer price is calculated according to some specific strategy, which will be described in Section 3.2.2.

(ii) Create a counter offer : If the provider refuses the consumer's offer, it will give its counter offer to the consumer.

After receiving the offer price of the consumer, the provider calculates the minimal price that it can accept, according to some specific strategy. In this research, the provider calculates the minimal price that it can accept according to an equation which will be described in Section 3.2.2. Meanwhile, the provider checks the offer price of the consumer. If the offer price is smaller than the minimal price it can accept, it rejects the offer price of the consumer and sends the minimal price it can accept to the consumer.

(iii) Accept an offer/a counter offer : When the provider accepts the offer price of the consumer or the consumer accepts the counter offer price of the provider, an agreement between the consumer and the provider has been achieved.

(iv) Refuse an offer/a counter offer: The provider refuses the consumer's offer

or the consumer refuses the provider's counter offer.

If the offer price of the consumer is smaller than the minimal price that the provider can accept, the provider refuses the consumer; if the counter offer price of the provider is larger than the maximal offer price of the consumer, the consumer refuses the provider.

(v) Sign a contract: If the provider accepts the consumer's offer or the consumer accepts the provider's counter offer, the negotiation succeeds and both parties may sign a contract (the conditions under which both parties will sign a contract when the negotiation succeeds will be discussed in Section 3.2.2).

(vi) Decommit from a contract: Both the consumer and the provider can decommit from the current contract.

From the above possible actions that an agent might take in a negotiation, it is known that the consumer makes an offer and the provider makes a counter offer alternately in the negotiation process. If one offer from the consumer and one counter offer from the provider are treated as one round of bargaining, then there may be more than one round of bargaining in a negotiation process before the negotiation process succeeds or is terminated. However, this would not result in a looping pattern that would be time consuming, nor would it result in time-sensitive tasks losing valuable negotiation time, and there are two main reasons for this. First, a consumer is allowed to negotiate with more than one provider simultaneously, and the consumer signs a contract with the provider as soon as one of the simultaneous negotiations succeeds; second, due to the time constraint of the task, both the consumer and the provider give concessions with the task's deadline approaching in order to reach an agreement before the deadline arrives, and this will be described in detail in Section 3.2.2.

The concessions of both parties in a negotiation cause the negotiation thread to converge to success quickly, otherwise, the arrival of the task's deadline will terminate the negotiation. Consequently, the number of rounds of bargaining in a negotiation is prevented from becoming large due to the time constraint of the task, which means that the messaging overhead is prevented from becoming large during a negotiation thread.

The negotiation will be terminated when at least one of the following situations occurs: **(i)** the task of agent a_i starts to be executed by another provider; **(ii)** the negotiation opponent terminates the negotiation; **(iii)** the negotiation succeeds or **(iv)** the deadline of the task arrives.

3.2.2 Offer Generation and Negotiation Strategies

When a consumer calculates its offer price $op_c(t, t_s)$ at time t, based on the start time t_s provided by the provider, the following five factors will be taken into account.

(i) The latest start time of a task.

With the pressure of the latest start time of the task, the consumer will give more concessions. Hence, the nearer t_s is, the higher $op_c(t, t_s)$ is, especially when $t = t_g$, $op_c(t, t_s)$ should be the lowest and equal to $c(R)$, which denotes the cost of the provider's resources to execute the task. However, this is an ideal situation that rarely happens because t_g is the generation time of the task (see **Definition 3.2**). If rep_{tc} is used to denote the number of reply messages for executing the task that the consumer has received in total till time t, when $t = t_s$ and $rep_{tc} = 1$, $op_c(t, t_s)$ should be the highest and its value should be the reserve price of the consumer, i.e., rp_c.

(ii) The number of reply messages.

The reason for taking into account the number of reply messages for executing the task, i.e., rep_{tc}, that the consumer has received is to consider the resource competition at the specific moment. The fewer reply messages for executing the task (*RepExecute*) that the consumer has received, the higher $op_c(t, t_s)$ will be. If a resource consumer a_i calculates its offer price to the provider a_j, it means that a_i has received at least one reply message *RepExecute*, i.e., the *RepExecute* sent from a_j. Thus, if $op_c(t, t_s)$ is calculated, the minimal value of rep_{tc} is 1. When the value of rep_{tc} is 1 and the time t equals to t_{st}, the offer price $op_c(t, t_s)$ is the highest and its value is the reserve price of the provider rp_c. Consequently, in order to meet these conditions about rep_{tc}, the logarithmic function '$log_2(rep_{tc} + 1)$' is introduced into Equation (3.4), which will be introduced later.

(iii) Reward that the consumer obtains.

The reward (denoted as $r(t_s)$), which the consumer will gain if its task can be executed at time t_s, can be calculated by:

$$r(t_s) = \frac{\overline{r}(t_{ls} - t_s)}{t_{ls} - t_g}, \tag{3.3}$$

where t_g is the generation time of the task, t_{ls} and $\overline{r}$ are the deadline and maximal reward of the task, respectively. The reward $r(t_s)$ also can be calculated in other ways, according to different specific situations, and Equation (3.3) is one of the methods to calculate the reward. The method used to calculate the reward of a consumer does not

affect the task allocation method.

It is assumed that a_i and a_j are the consumer and the provider in an individual negotiation, respectively. When a_i receives a reply message ($RepExecute_{ji}, ID_j, t_s, \tau_k$) from a_j, it calculates $r(t_s)$ first and then begins to negotiate with a_j. During the course of the negotiation, a_i gives its offer and a_j gives its counter offer alternately.

(iv) Reserve price of the consumer.

The reserve price of the consumer, i.e., rp_c, equals to $r(t_s)$. $r(t_s)$ is the reward that the consumer can gain from the task if the provider starts the task at time t_s.

(v) Provider's cost.

The cost of the provider's resources to execute the consumer's task, i.e., $c(R)$.

According to the above justification, $op_c(t, t_s)$ is given by:

$$op_c(t, t_s) = c(R) + (rp_c - c(R))\frac{t - t_g}{(t_s - t_g)log_2(rep_{tc} + 1)}, \tag{3.4}$$

where $t_g \leq t \leq t_s$.

When a provider calculates the counter offer price $op_p(t, t_s)$ at time t, it will consider the following three factors:

(i) The Latest start time of a task.

With the pressure of the latest start time t_s, the provider will also give more concessions. Thus, the nearer t_s is, the lower $op_p(t, t_s)$ should be.

(ii) The number of request messages for executing the task (*ReqExecute*) that the provider has received.

The more request messages ($ReqExecute$) the provider has received, the higher $op_p(t, t_s)$ is. If a resource provider a_j calculates its counter offer price to consumer a_i, it means that a_j has received at least one request message *ReqExecute*, i.e., the *ReqExecute* sent by a_i. Thus, if $op_p(t, t_s)$ is calculated, the minimal value of req_{tp} is 1. When the value of req_{tp} is 1 and the time $t = t_g$ ($t = t_g$ is an ideal situation which rarely happens), the counter offer price $op_p(t, t_s)$ is the highest. Consequently, in order to meet these situations about req_{tp}, the logarithmic function '$log_2(req_{tp} + 1)$' is introduced into Equation (3.5).

(iii) The cost of the provider's resource to execute the consumer's task, i.e., *c(R)*.

$c(R)$ is the reserve counter offer price of the provider. The higher the cost of resources is, the higher $op_p(t, t_s)$ is.

Accordingly, $op_p(t, t_s)$ is formulated as follows:

$$op_p(t, t_s) = c(R)\left(1 + \frac{t_s - t}{t_s - t_g} log_2(req_{tp} + 1)\right), \tag{3.5}$$

where $t_g \leq t \leq t_s$.

From Equation (3.5), it can be seen that when $t = t_g$, $op_p(t, t_s)$ is the highest and the value is $c(R)(1 + log_2(req_{tp} + 1))$. However, this is an ideal situation which rarely happens because t_g is the generation time of the task. When $t = t_s$, the value of $op_p(t, t_s)$ is the lowest, and $op_p(t, t_s) = c(R)$.

The negotiation strategies of consumers and providers are detailed in Algorithm 3.4 and Algorithm 3.5, respectively. In both Algorithms 3.4 and 3.5, it is assumed that a_i is the consumer and a_j is the provider. Let pro_i and pro_j be the current total profits of a_i and a_j, respectively. In addition, it is assumed that if the negotiation between a_i and a_j succeeds and they sign a contract, the total profits of a_i and a_j will become pro'_i and pro'_j respectively, after the new contract is successfully completed. Thus, $pro'_i - pro_i$ and $pro'_j - pro_j$ are the profits that a_i and a_j can gain from the contract, respectively.

Algorithm 3.4: Consumer's Strategy

After receiving a reply message $(RepExecute_{ji}, ID_j, t_s, \tau_k)$ from a_j, a_i calculates $op_i(t, t_s)$ by *Equation (3.4)* and sends $op_i(t, t_s)$ to a_j;

1 While $t < t_s$ **and** the task has not started to be executed by any other agent (because any agent is allowed to negotiate with more than one opponent, it is possible that a_i has signed a contract for τ_k with another provider, and there is a start time in that contract) **do**

2 **if** a_i receives $op_j(t, t_s)$ from a_j **then** a_i calculates $op_i(t, t_s)$ by *Equation (3.4)*;

3 **if** $op_j(t, t_s) \leq op_i(t, t_s)$ **then** negotiation succeeds; break;

4 **else**

5 a_i calculates $op_i(t, t_s)$ again by *Equation (3.4)*

6 and sends $op_i(t, t_s)$ to a_j;

7 **end**

8 end while

9 if negotiation succeeds **then** a_i calculates pro'_i by *Equation (3.2)*;

10 **if** $pro'_i > pro_i$ and a_j is also willing to sign a contract **then**

11 a_i and a_j sign a contract $con_{ij}(\tau_k, pr_{con}, t_s, t_{con}, pro'_i - pro_i, pro'_j - pro_j)$

12 end if

In Algorithm 3.4, when consumer a_i receives a count-offer price $op_j(t, t_s)$ from provider a_j, it checks whether it is willing to accept $op_j(t, t_s)$. If yes, the negotiation succeeds (**Lines 2-3**). Otherwise, a_i calculates a new offer price and sends it to a_j

(**Lines 4-7**). When the negotiation succeeds, a_i calculates whether signing a contract with a_j is profitable. If yes, a_i signs a contract with a_j (**Lines 9-12**).

Algorithm 3.5: Provider's Strategy

1 While $t < t_s$ and the task has not started to be executed by any other agent **do**
2 | **if** a_j receives $op_i(t, t_s)$ from a_i, **then** a_j calculates $op_j(t, t_s)$ by *Equation (3.5)*;
3 | | **if** $op_i(t, t_s) \geq op_j(t, t_s)$ **then** negotiation succeeds; break;
4 | | **else**
5 | | | a_j calculates $op_j(t, t_s)$ based on the synchronization time again by
6 | | | *Equation (3.5)* and sends $op_j(t, t_s)$ to a_i;
7 | | **end**
8 end while
9 if negotiation succeeds **then** a_i calculates pro'_j by *Equation (3.2)*;
10 | **if** $pro'_j > pro_j$ and a_i is also willing to sign a contract **then** a_i and a_j
11 | | sign a contract $con_{ij}(\tau_k, pr_{con}, t_s, t_{con}, pro'_i - pro_i, pro'_j - pro_j)$;
12 end if

In Algorithm 3.5, when provider a_j receives an offer price $op_i(t, t_s)$ from a_i, it checks whether it is willing to accept $op_i(t, t_s)$ (**Line 2**). If yes, the negotiation succeeds (**Line 3**). Otherwise, a_j calculates a new count-offer price and sends it to a_i (**Lines 4-7**). When the negotiation succeeds, a_j calculates whether signing a contract with a_i is profitable. If yes, a_j signs a contract with a_i (**Lines 9-12**).

3.3 Experiments and Analysis

Two experiments are conducted to evaluate the performance of the proposed method in task allocation. Experiment 1 is to exam the impacts of the task deadlines, the resource competitions, and the average number of the required resources per consumer. Experiment 2 is to test the impacts of task deadlines on the individual profit distribution of agents. This section demonstrates the detail of experimental results and gives analysis and discussions.

3.3.1 Experiment 1

3.3.1.1 Experimental Settings

The task allocation method proposed in this chapter aims at allocating as many tasks as possible under time constraints, thus, the deadline of the task is one of the key factors affecting the performance of the proposed method. In addition, the environment is competitive and all the agents are self-interested, thus, resource competition is also one of the important factors. In addition, a task can be allocated to a provider if and only if the provider has all the required resource types for the task. Consequently, the number of required resource types per task also plays an important role in the performance of the proposed method.

For the above reasons, Experiment 1 is conducted based on the following three different scenarios: (1) examination of the impact of the deadlines of tasks, (2) examination of the impact of resource competition and (3) examination of the impact of the average number of required resources per consumer.

In the experiment, the proposed method, i.e., NTAL, is compared with NTA proposed in [ALIZ10] and the Distributed Greedy Algorithm-based method (DGA) proposed in [dWZK07]. Since the parameter settings in the experiments of [ALIZ10] are reasonable and common in real life applications, the parameter settings in this experiment are similar to the experimental parameter settings of [ALIZ10]. However,

both the number of tasks and the number of providers in this experiment are higher than those in the experiment in [ALIZ10]. This is because NTAL is decentralised and this means that it should have good scalability, so it must be tested in an environment with more tasks and providers. Experiment 1 includes the following three scenarios, which include **1)** to test the impact of task deadlines, **2)** to test the impact of the resource competition, and **3)** to test the impact of the average number of required resource types per consumer.

Scenario 1: examination of the impact of task deadlines

The purpose of Scenario 1 is to examine the impact of task deadlines on the task allocation method. The parameters used in Scenario 1 are listed in Table 3.1.

Table 3.1: Parameters Setting for Scenario 1 of Experiment 1

Variables	meanings	values
N_τ	number of tasks	[0,100]
N_p	number of resource providers	[0,100]
$\psi(r)$	resource competition	1
N_{ave}	number of average required resource types of tasks	5

To control the maximum numbers of providers and tasks that enter the grid environment, both the maximum providers and tasks are set as 100, that is, both the numbers of providers and consumers are in the range of [0, 100] during the course of the experiment. This does not mean that the number of providers and the number of tasks in the experiment are randomly chosen from the range of [0, 100]. Rather, it means that both the number of providers and the number of tasks change between 0 and 100 over time during the course of the experiment. This is because there is a fixed probability for providers and tasks to enter and leave the environment during the course of the experiment. In each time unit, each provider or task has a 50 percent possibility to leave the environment or enter the environment. In addition, 100 providers are generated in total. Thus, the number of providers in the environment is normally distributed in [0, 100] during the course of the experiment. For the same reason, the number of tasks in the environment is normally distributed between 0 and the maximum number of tasks that can enter the grid environment. The parameters are set in this way due to the dynamism and openness of the grid environment. Based on this, the deadlines (i.e., latest start times) of tasks are changed, and different success rates, total profits, and total time used for the experiment are obtained.

Scenario 2: examination of the impact of resource competition

In fact, even though the purpose of Scenario 1 is to test the impact of different task deadlines on the performance of the proposed method, the resource competition is not fixed and can vary during the course of the experiment according to the parameter settings above. The resource competition varies over a small range and this cannot be avoided due to the dynamism and openness of the environment. The aim of Scenario 2 is to examine the impact of different levels of resource competition on the proposed task allocation method. The parameters used in Scenario 2 are listed in Table 3.2.

Table 3.2: Parameters Setting for Scenario 2 of Experiment 1

Variables	meanings	values
t_{ls}	deadlines of tasks	[400, 500]
t_g	generation times of tasks	0
$flex(\tau)$	allocation flexibility	[400, 500]
N_{ave}	average number of required resource types of tasks	5

The allocation flexibilities of tasks (i.e., $flex(\tau)$) are in the range of [400, 500], and the number of providers is set in-between [0, 100]. Change the maximum number of tasks involved in the task allocation between 20 and 600, different levels of resource competition from 0.2 to 6 can be obtained, according to *Equation (3.8)*.

Scenario 3: examination of the impact of the average number of required resource types per consumer

The purpose of Scenario 3 is to test the impact of the average number of resource types required by each agent on the performance of the proposed method. The parameters used in Scenario 3 are listed in Table 3.3.

Table 3.3: Parameters Setting for Scenario 3 of Experiment 1

Variables	meanings	values
N_τ	number of tasks	[0,100]
N_p	number of resource providers	[0,100]
$\psi(r)$	resource competition	1
$flex(\tau)$	allocation flexibility	[400, 500]

3.3.1.2 Experimental Results and Analysis

Before presenting and analysing the experimental results of the three scenarios in detail, it is necessary to make it clear that all the thresholds were not set in advance of the experiments. Moreover, in order to emphasize the uncertainties of the results that

are generated in dynamic environments, the experimental result data are accompanied with fluctuation ranges in all of the experimental result figures.

Experimental Results and Analysis of Scenario 1

In Scenario 1, the corresponding success rates when the allocation flexibilities of tasks change during the course of the experiment, are shown in Figure 3.1.

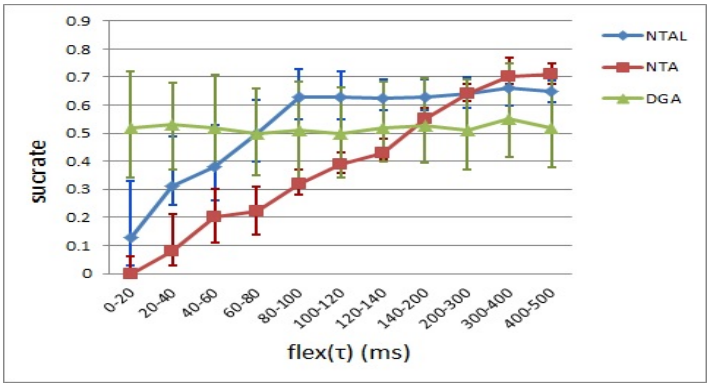


Figure 3.1: Success Rates based on Various Allocation Flexibilities

In Figure 3.1, it can be seen that the success rate of NTAL reaches a peak value earlier than that of NTA. This result can be explained from the view point of the negotiation strategies of both NTAL and NTA. In both NTA and NTAL, the consumer takes time t into account when calculating its offer, and gives concessions as time progresses even when the other factors that impact the offer remain unchanged. However, the negotiation strategy of the provider in NTAL is different from the strategy of the provider in NTA. It can be seen from Equation (3.5) in Subsection 3.2.2 that when the other parameters in Equation (3.5) are fixed, the larger t is, the lower the counter offer of the provider, i.e., $op_p(t, t_s)$, is. Therefore, when the other factors remain unchanged, the provider in NTAL still gives concessions as time progresses. However, in NTA, the provider does not take time t into account when calculating its counter offer. Thus, both the consumer and the provider in NTAL take time t into account in a negotiation process, while only the consumer in NTA considers time t. Consequently, a negotiation process in NTAL can reach success quicker than in NTA, which means that the success rate of NTAL reaches its peak value quicker than that of NTA, as it can be seen from Figure 3.1. For the same reason, when the allocation flexibility, i.e., $flex(\tau)$, is lower than 200 ms ($milliseconds$), the success rate of NTAL is higher than that of NTA. However, with the increase of $flex(\tau)$, the effect of the negotiation speed to the success rate decreases and consequently, the success rate of task allocation depends primarily

on the information that the consumer possesses. Compared with the global view of the consumer in NTA, the limitation of the local view of the consumer in NTAL results in the success rate of NTAL being lower than that of NTA when $flex(\tau)$ is over 200 *ms*. The results in Figure 3.1 demonstrate that compared with NTA, NTAL works better when tasks are urgent.

As can seen from Figure 3.1, the success rates of DGA are stable in different ranges of $flex(\tau)$. This is because there is no negotiation in DGA. Instead, only a greedy algorithm is adopted. Thus, the task allocation speed of DGA is less related to $flex(\tau)$ compared with both NTAL and NTA. Consequently, compared with both NTAL and NTA, the success rate of DGA is less related to $flex(\tau)$, even though the tasks in all of the three methods have strict time constraints. Because both NTAL and NTA need time for negotiation, while DGA does not, the success rate of DGA is higher than that of both NTAL and NTA when $flex(\tau)$ is lower than 80 *ms*. With the increase of $flex(\tau)$, however, the effect of the allocation flexibility on the success rate of task allocation decreases. In addition, compared with the greedy algorithm in DGA, the negotiation strategies adopted can make the consumers in both NTA and NTAL consider more factors, such as resource competition, task deadlines, etc, when allocating tasks. Consequently, the success rate of DGA is lower than those of both NTAL and NTA when $flex(\tau)$ is higher than 140 *ms*, and this can be seen from Figure 3.1.

It can also be seen from the fluctuation ranges in Figure 3.1 that the fluctuations of success rates of NTAL, NTA and DGA decrease with the increase of $flex(\tau)$. The reason for this is that with the increase of $flex(\tau)$, consumers have longer time to allocate tasks. Thus, the impact of the time constraints on the success rate is decreased. Meanwhile, other elements which affect the success rate remain unchanged. Consequently, success rates become steadier with the increase of $flex(\tau)$. Additionally, it can be seen that the success rate of NTA is steadier compared with those of both NTAL and DGA, and this can be explained from the view point of choice range. The consumers in both NTAL and DGA only have local views of providers, while the consumers in NTA have global views of the providers. Thus, the consumers in both NTAL and DGA choose providers from narrower ranges compared with the consumers in NTA. Consequently, the wider choice range of providers for consumers in NTA causes the steadier success rates compared with both NTAL and DGA, due to the dynamism of the grid environments.

Figure 3.2 illustrates the ratios of total profit between NTAL and NTA, and the

ratios of total profit between NTAL and DGA, based on different allocation flexibilities.

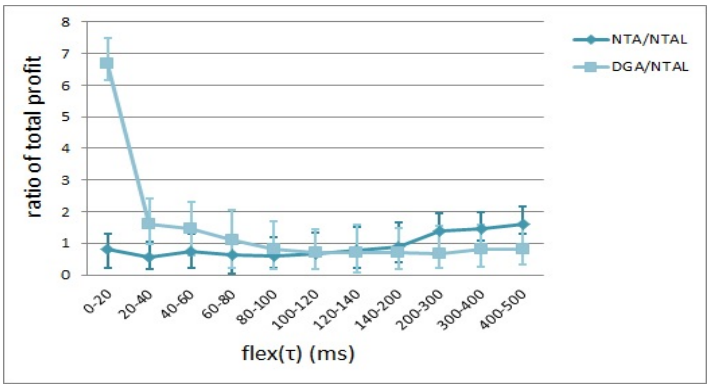


Figure 3.2: Total Profits Based on Various Allocation Flexibilities

As can be seen from Figure 3.2, when $flex(\tau)$ is lower than 200 ms, the ratios of total profit between NTA and NTAL are lower than 1. When $flex(\tau)$ is higher than 200 ms, the ratios of total profit between NTA and NTAL are higher than 1. This result can be explained from the view point of the corresponding success rates of both NTAL and NTA. As can be seen from Figure 3.1, the corresponding success rates of NTAL are higher than those of NTA when $flex(\tau)$ is lower than 200 ms, and the corresponding success rates of NTAL are lower than those of NTA when $flex(\tau)$ is higher than 200 ms. In addition, in the proposed method, it is stipulated that if a task fails to be allocated, the profit that this task's owner (i.e., the consumer) gains from this task is 0. Consequently, the total profit that an agent gains is closely related to the success rate of task allocation. Thus, compared with NTA, the higher success rates of NTAL result in the higher total profit when $flex(\tau)$ is lower than 200 ms, and the lower success rates of NTAL result in the lower total profit when $flex(\tau)$ is higher than 200 ms. When the task $flex(\tau)$ is lower than 100 ms, the ratios of total profits between DGA and NTAL are higher than 1. This is because NTAL needs time to negotiate when allocating tasks while DGA does not, which means that the success rate of NTAL is more greatly affected by $flex(\tau)$ as compared with that of DGA. Additionally, it has been analysed above that the total profit of agents is closely related to the success rate of task allocation. For this reason, when the flexibilities are low, the success rates of NTAL are lower than those of DGA, which causes the ratios of the total profit between DGA and NTAL to be higher than 1. When $flex(\tau)$ is higher than 100 ms, the ratios of the total profit between DGA and NTAL become lower than 1. This is because with the increase of $flex(\tau)$, the corresponding success rate of NTAL increases more sharply

than that of DGA, due to the fact that DGA is less sensitive to $flex(\tau)$ compared with NTAL.

It can also be seen from Figure 3.2 that the fluctuation of the ratio between the total profit of NTA and the total profit of NTAL remains steady with the increase of $flex(\tau)$, and the fluctuation of the ratio between the total profit of DGA and the total profit of NTAL also remains steady with the increase of $flex(\tau)$. This is because the tasks in NTAL, NTA and DGA have strict deadlines, and if a task cannot be finished before its deadline, the task fails. Meanwhile, the profit that the task's owner gains from the failed task is 0. Consequently, the increase of $flex(\tau)$ causes the increase of the success rate of task allocation, which results in the total profits in NTAL, NTA and DGA increasing. Consequently, the ratios of total profits remain steady.

Figure 3.3 presents the time (the time unit is *millisecond*) used for task allocation by the three methods.

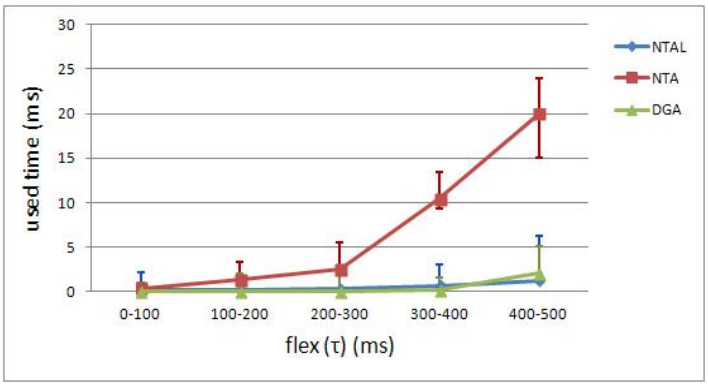


Figure 3.3: Used Time Based on Various Allocation Flexibilities

From Figure 3.3, it can be seen that the time used for task allocation by NTAL is shorter than that by NTA. The reason for this, as has already been shown in Figure 3.1, is that the negotiation strategy of the provider in NTAL is different from the negotiation of the provider in NTA. The provider in NTAL considers the time progress when calculating the counter offer and giving concessions in a negotiation process, as can be seen from Equation (3.5), while the provider in NTA does not. Thus, the negotiation process reaches success quicker in NTAL than in NTA, which results in the time used for task allocation by NTAL being shorter than for NTA.

It can also be seen from Figure 3.3 that the time used for task allocation by NTAL is not longer than that by DGA, even though the former needs negotiation when allocating tasks, while the latter does not. In NTAL, the consumer allocates

tasks to the providers through negotiations. It can be seen from Equation (3.4) that the start time, i.e., t_s, for the provider to start the task is taken into account by the consumer during the negotiation with the provider, and this can guarantee that the provider that is eventually allocated the task to can start the task soon. Thus, in NTAL, it is assumed that a task is successfully allocated as soon as it is allocated to a provider. In DGA, however, the consumer allocates its task to a provider as long as the provider can meet the resource requirement of the task. After being allocated to the provider, the task is inserted into the task queue to wait to be selected by the provider to execute. Consequently, it is probable that a task may wait a long time in the queue before being selected by the provider from the task queue to be executed. Therefore, it is assumed that a task in DGA is successfully allocated only when the task is selected by the provider from the task queue to be executed. According to the above analysis, it is reasonable that the time used for task allocation in NTAL is not longer than that in DGA. The longer time used in NTA results in the greater fluctuation of the time used compared with both NTAL and DGA.

Figure 3.4 shows the average number of rounds per negotiation thread of both NTAL and NTA, respectively.

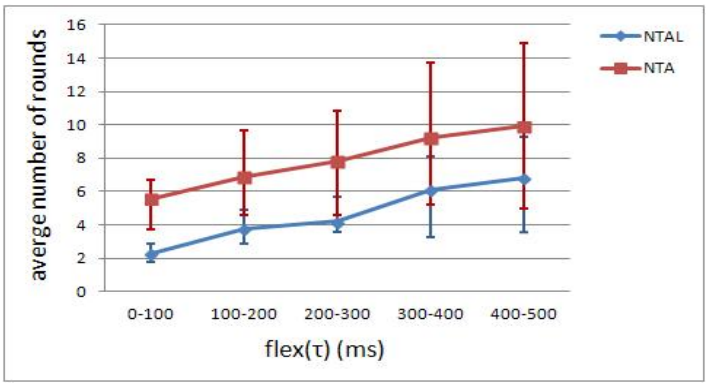


Figure 3.4: Average Number of Rounds Based on Various Allocation Flexibilities

From Figure 3.4, it can be seen that the average number of rounds of each negotiation (denoted by N_{roun}) of both NTAL and NTA increases when $flex(\tau)$ becomes higher. This is because with the increase of $flex(\tau)$, agents have a longer time to negotiate before the task deadline. It can also be seen from Figure 3.4 that N_{roun} of NTAL is smaller than that of NTA, and this can be explained by the differences in negotiation strategies of NTAL and NTA. It can be seen from Equation (3.4) and Equation (3.5) that the consumer and the provider in NTAL consider the factor of time progress, i.e., t, when calculating the offer and the counter offer. In NTA, however, only the consumer takes the time progress into account when calculating its offer. Thus, the negotiation process takes longer in NTA than in NTAL before it succeeds or is terminated, and this means that N_{roun} of NTAL is smaller than that of NTA. Because N_{roun} in NTAL is below 7, which can be seen from Figure 3.4, and each round contains two messages, that is, the message from the consumer to the provider about the consumer's offer and the message from the provider to the consumer about the provider's counter offer, the total messages in a negotiation process are fewer than 14 in NTAL, and the messaging overhead can be ignored. Additionally, compared with NTAL, the larger N_{roun} of NTA results in stronger fluctuations of N_{roun}.

From the experimental results of Scenario 1, it can be seen that the most suitable grid environment for DGA is that tasks in the grid environment have urgent deadlines. This is because DGA is a greedy algorithm-based task allocation method that only adopts a greedy algorithm when selecting tasks to execute. Consequently, compared with both NTAL and NTA, DGA saves time when allocating tasks, and this makes DGA suitable for urgent tasks. Compared with NTA, NTAL is more suitable for grid environments that require large scalability and when tasks have comparatively long deadlines. The situation that the deadlines are comparatively long does not include the extreme cases in which the deadlines are so long that consumers can adopt exhaustive methods to find suitable providers before the deadline.

Experimental Results and Analysis of Scenario 2

Figure 3.5 presents the success rates based on different levels of resource competition.

From Figure 3.5, it can be seen that when the resource competition, i.e., $\psi(r)$, is lower than 0.6 and higher than 2, the success rate of NTAL is higher than that of NTA. This can be explained as follows. When $\psi(r)$ is lower than 0.6, the low $\psi(r)$ indicates that there are sufficient providers for tasks, so the limitation of the local view of the consumer in NTAL to look for suitable providers can be ignored. When $\psi(r)$ is over

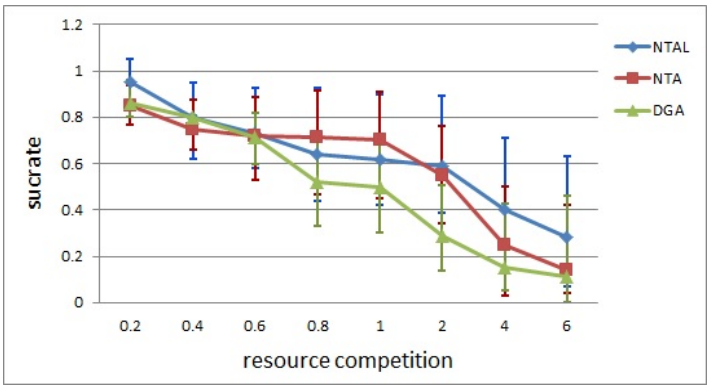


Figure 3.5: Success Rates Based on Various Resource Competitions

2, it indicates that there are insufficient providers in the environment, so it might still be hard for a consumer to look for suitable providers even though the consumer has a global view. In this situation, the advantage of the global view is no longer relevant. Thus, when $\psi(r)$ is lower than 0.6 and over 2, the success rate depends primarily on the negotiation strategy. As has been seen in Figure 3.1, from the view point of the negotiation strategies of the providers in both NTAL and NTA, a negotiation process reaches success quicker in NTAL than in NTA, and this results in the success rate of NTAL being higher than that of NTA, due to the time constraints of tasks.

As can be seen from Figure 3.5, when $\psi(r)$ is higher than 0.6, the success rate of DGA is lower than those of both NTAL and NTA. This is because the high resource competition indicates that the providers are insufficient in the environments, therefore when $\psi(r)$ is higher than 0.6, it is hard for consumers in DGA to successfully allocate tasks. Compared with the greedy algorithm adopted in DGA, the negotiation strategies employed in both NTAL and NTA can give flexibility to both providers and consumers to adapt to the high resource competition. Because the consumer in NTAL has no global view of the environment, it does not know the value of $\psi(r)$. Rather, the consumer in NTAL estimates the value of $\psi(r)$ according to the total number of reply messages (*Repexecute*) that it receives from the providers. The consumer assumes that the more reply messages it receives, the lower $\psi(r)$ is. It can be seen from Equation (3.4) that the consumer in NTAL takes the total number of the reply messages that the consumer receives, i.e., rep_{tc}, into account when calculating its offer, which means that the consumer can adapt to the changing $\psi(r)$ when calculating its offer in each round of the negotiation process. Similarly, the provider in NTAL takes the total number of the request messages (*Reqexecute*) that it receives, i.e., req_{tp}, into account when calculating its counter offer, which can be seen from Equation (3.5).

It can also be seen from the fluctuation ranges in Figure 3.5 that the fluctuations in the success rates of NTAL, NTA and DGA get stronger as $\psi(r)$ increases. This is because with the increase of $\psi(r)$, it becomes harder for consumers to find suitable providers, and this causes the success rate to fluctuate more. In addition, the success rate of NTA is the steadiest compared with both NTAL and DGA, and this can be explained by the choice range that has been analysed in Figure 3.1.

The ratios of total profit between NTA and NTAL, and the ratios of total profit between DGA and NTAL are shown in Figure 3.6.

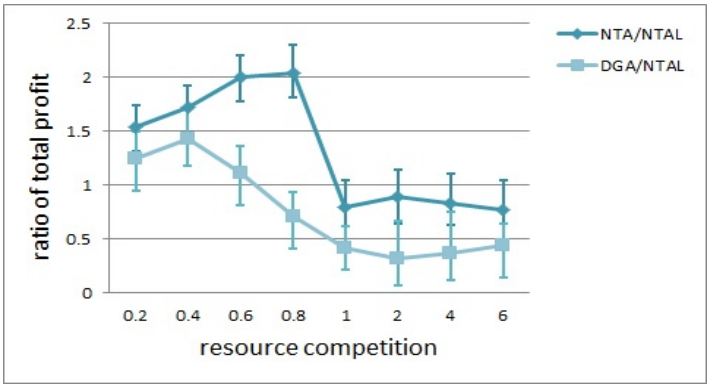


Figure 3.6: Total Profits based on Various Resource Competitions

In Figure 3.6, the top line represents the ratios between the total profits of NTA and the total profits of NTAL, and the other line is the ratio between the total profit of DGA and the total profit of NTAL. From Figure 3.6, it can be seen that when $\psi(r)$ is higher than 1, the ratios between the total profit of NTA and the total profit of NTAL are lower than 1, which means that the total profits of NTAL are higher than those of NTA. This can be explained by the corresponding success rates as shown in Figure 3.5. This is because if a task fails to be allocated, the profit that the task's owner, i.e., the consumer, gains from the task is 0. Thus, the profit that an agent involved in task allocation gains is closely related to the success rate of task allocation. Additionally, as can be seen from Figure 3.5, the corresponding success rates of NTAL are higher than those of NTA when $\psi(r)$ is higher than 1. Consequently, the total profits of the agents involved in task allocation in NTAL are higher than those in NTA, when $\psi(r)$ is higher than 1.

From Figure 3.6, it can be seen that the ratios between the total profits of DGA and the total profits of NTAL are lower than 1, when $\psi(r)$ is higher than 0.6. The reason for this is the same as that shown in Figure 3.5, that is, the negotiation approach employed in NTAL can provide both the consumer and the provider flexibility to make

them adjust to the high resource competition, and this can increase the probability of success of the negotiation processes.

The fluctuations in the success rates of NTAL, NTA and DGA get stronger with the increase of $\psi(r)$, which can be seen from Figure 3.5. In addition, the total profit is closely related to the success rate. Consequently, both the ratio between the total profit of NTA and the total profit of NTAL, and the ratio between the total profit of DGA and the total profit of NTAL remain steady, as shown in Figure 3.6.

Figure 3.7 demonstrates the total time used for the whole task allocation processes in the three methods.

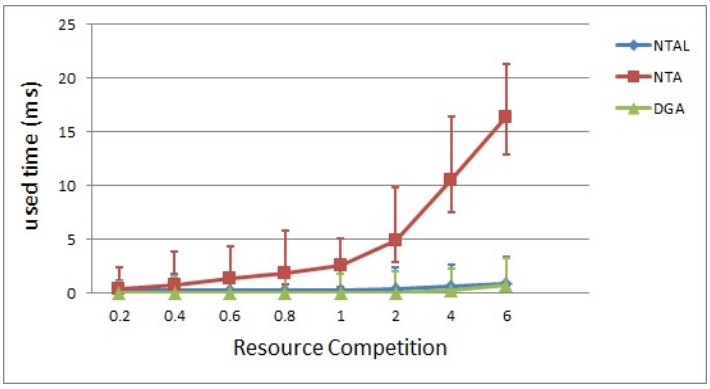


Figure 3.7: Used Time Based on Resource Competitions

It can be seen from Figure 3.7 that the times used by NTAL, NTA and DGA get longer with the increase of $\psi(r)$. This is because in both NTAL and NTA, when $\psi(r)$ gets higher, it takes longer for a consumer to find suitable providers. For DGA, high resource competition indicates the longer task queues of each provider, which results in the longer waiting time for tasks to be selected by the providers. It also can be seen from Figure 3.7 that the time used by NTAL is shorter than the time used by NTA when $\psi(r)$ varies between 0.2 and 6. This can be explained by the negotiation strategy. In NTA, only the consumer takes the time into account in a negotiation process, while both the provider and the consumer consider the time in a negotiation process in NTAL, as shown by Equation (3.4) and Equation (3.5). Consequently, the negotiation process reaches success quicker in NTAL than in NTA, which causes the longer time for task allocation in NTA than that in NTAL.

It can also be seen from Figure 3.7 that when $\psi(r)$ is higher than 1, the time used by NTAL is longer than the time used by DGA. This is because when $\psi(r)$ is high, consumers in NTAL need longer time to find providers for tasks, which makes the task allocation process longer, compared with the situation when $\psi(r)$ is low.

Figure 3.8 shows N_{roun} in both NTAL and NTA.

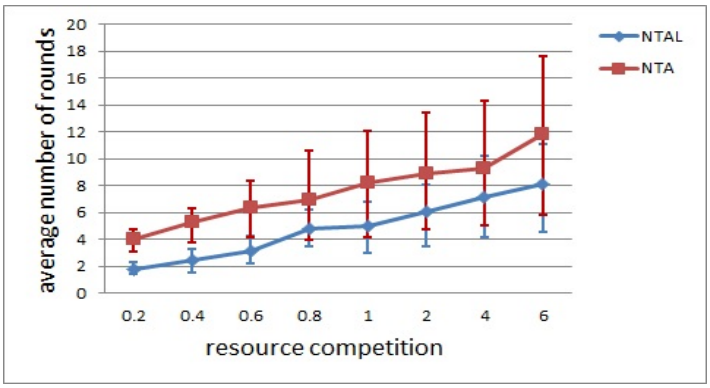


Figure 3.8: Average Number of Rounds Based on Various Resource Competitions

It can be seen from Figure 3.8 that N_{roun} of NTAL is smaller than that of NTA, and the reason for this is the same as that shown in Figure 3.4. It also can be seen that N_{roun} increases when $\psi(r)$ becomes higher in both NTAL and NTA, and this can be explained by the negotiation strategy of the provider in both NTAL and NTA. Because the provider in NTAL has no global view, the provider estimates the resource competition according to the total number of request messages (*Reqexecute*) that it receives, i.e., req_{tp}. The provider in NTAL assumes that the more request messages it receives, the higher $\psi(r)$ is. As can be seen from Equation (3.5) when the other parameters are fixed, the larger req_{tp} is, the higher $op_p(t, t_s)$ is. In NTA, the provider gives fewer concessions when the resource competition becomes higher. Thus, the higher $\psi(r)$ is, the longer it takes before the provider provides a counter offer that the consumer can accept in both NTAL and NTA, which means that the higher $\psi(r)$ is, the larger N_{roun} is. Moreover, it can be seen that the maximum value of N_{roun} of NTAL is below 8, because each round contains two messages, thus, the maximum number of messages in a negotiation process in NTAL is below 16, and the messaging overhead can be ignored. It is reasonable that the higher N_{roun} of NTA causes stronger fluctuations, compared with both NTAL and DGA.

Experimental Results and Analysis of Scenario 3

Figure 3.9 presents the success rates of task allocation in NTAL, NTA and DGA.

It can be seen from Figure 3.9 that the success rates of NTAL, NTA and DGA decrease with the increase of the average number of required resource types of each task, i.e., N_{ave}. The reason is that with the increase of N_{ave}, the providers that can

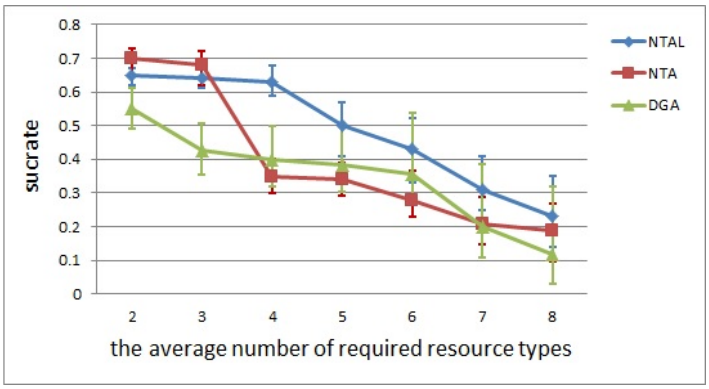


Figure 3.9: Success Rates Based on Various Average Numbers of Required Resource Types per Task

individually meet the resource requirement of a task become fewer, and this causes the decrease in the success rates of NTAL, NTA and DGA. It also can be seen from Figure 3.9 that the success rate of NTAL is higher than that of NTA when N_{ave} is higher than 3. The reason for this is that when N_{ave} is high, it is hard for the consumer to find a provider that can individually meet the resource requirement of the task even though the consumer has a global view of the environment. Thus, compared with the local view of the consumer in NTAL, the global view of the consumer in NTA is no longer advantageous. In this situation, the success rate depends primarily on the negotiation strategy. Because both the consumer and the provider in NTAL take the time into account in a negotiation process, as can be seen from Equation (3.4) and Equation (3.5), while in NTA, only the consumer considers the time in a negotiation process, the negotiation process reaches success more quickly in NTAL than in NTA. Additionally, tasks have strict time constraints, which means that the speed of the negotiation process plays an important role in the success rate, and hence the success rate of NTAL is higher than that of NTA when N_{ave} is over 3.

Apparently, with the increase of N_{ave}, it gets harder for consumers in NTAL, NTA and DGA to find providers that can meet the resource requirement, and this causes the fluctuations in the success rates of the three methods to become greater. The global view of the consumer in NTA can provide a wider range of choice of providers for the consumer, and this results in the steadier success rate, compared with those of both NTAL and DGA, as can be seen from Figure 3.9.

Figure 3.10 demonstrates the total profits of agents in NTAL, NTA and DGA.

It can be seen from Figure 3.10 that when N_{ave} varies between 2 and 8, both the ratio between the total profit of agents in NTA and the total profit of agents in NTAL,

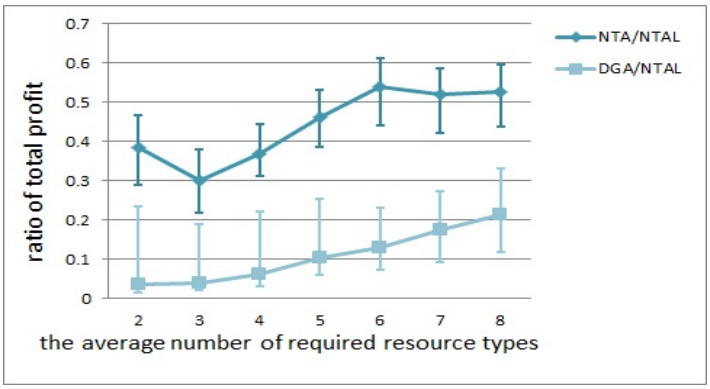


Figure 3.10: Total Profits based on Various Average Numbers of Required Resource Types per Task

and the ratio between the total profit of agents in DGA and the total profit of agents in NTAL are lower than 1. This is because in all of the three methods, if a task fails to be allocated, the profit that the consumer, i.e., the task's owner, gains from this task is 0. Thus the total profit of the agents involved in task allocation is closely related to the success rate of task allocation. As can be seen from Figure 3.9, the corresponding success rates of NTAL are higher than those of both NTA and DGA when N_{ave} varies between 2 and 8. Therefore, the total profits of agents in NTAL are higher than those in both NTA and DGA. Because it gets harder for consumers in NTAL, NTA and DGA to find the providers that can individually meet the resource requirement of a task with the increase of N_{ave}, the total profits of agents in NTAL, NTA, and DGA decrease. The decrease in the total profits of all of the three methods results in the fluctuation of the ratio between the total profit of agents in NTA and the total profit of agents in NTAL remaining steady. For the same reason, the fluctuation of the ratio between the total profit of agents in DGA and the total profit of agents in NTAL also remains steady, as can be seen from Figure 3.10.

Figure 3.11 shows the total time used by the three methods, based on different average numbers of resource types required by each agent. Figure 3.11 is the time usage based on the various average numbers of required resource types per task. As can be seen from Figure 3.11, the time used for task allocation in NTAL, NTA and DGA becomes longer with the increase of N_{ave}. This is reasonable because with the increase of N_{ave}, it takes longer for consumers to find suitable providers that can individually meet the resource requirement of the tasks. It also can be seen from Figure 3.11 that the times used by both NTAL and DGA are shorter than that in NTA. This is because DGA does not need the negotiation process when allocating tasks, and only a greedy

algorithm is employed in DGA. In NTAL, both the provider and the consumer consider the time factor in the negotiation process, while only the consumer does so in NTA. Thus, the negotiation process reaches success quicker in NTAL than in NTA, which results in the time used for task allocation in NTAL being shorter than the time used for task allocation in NTA. Compared with the times used in both NTAL and DGA, the longer time used in NTA causes the greater fluctuation in the time used.

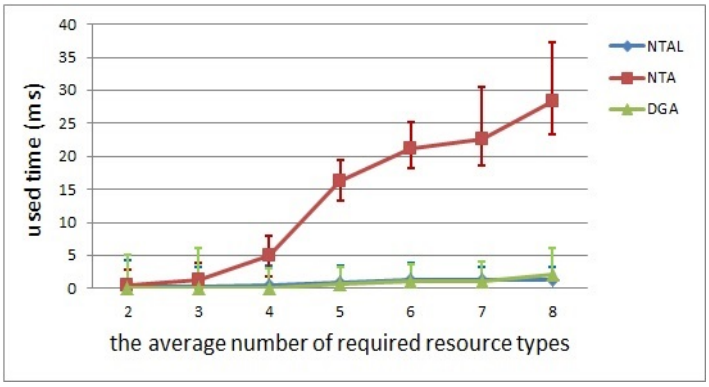


Figure 3.11: Used Time Based on Various Average Numbers of Required Resource Types per Task

Figure 3.12 demonstrates the average numbers of round per negotiation thread of both NTAL and NTA.

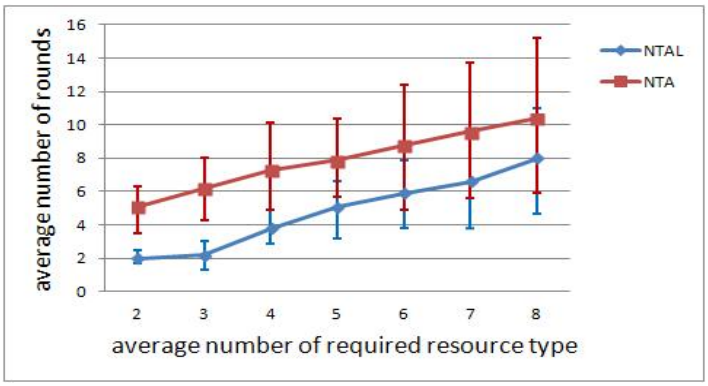


Figure 3.12: Average Number of Rounds Based on Various Average Numbers of Required Resource Types per Task

From Figure 3.12, it can be seen that N_{roun} of NTAL is smaller than that of NTA, and this can be explained by the negotiation strategies of both NTAL and NTA. Because both the consumer and the provider take the time into account in a negotiation process in NTAL, while only the consumer does so in NTA, the negotiation process reaches success quicker in NTAL than in NTA. It also can be seen from Figure 3.12 that N_{roun} becomes larger when N_{ave} increases. This is because with the increase of N_{ave}, the providers that can individually meet the resource requirement of the task become fewer, so the increase of N_{ave} is equivalent to the increase of $\psi(r)$. Consequently, when N_{ave} is higher, there are more rounds in a negotiation process before the negotiation process succeeds or is terminated. Because N_{roun} of NTAL is below 8, and each round contains two messages, the maximum number of messages in a negotiation process is below 16 in NTAL. Consequently, the messaging overhead in NTAL can be ignored. It is the increase of N_{ave} that results in the greater fluctuations of N_{roun} of both NTAL and NTA, as can be seen in Figure 3.12.

3.3.2 Experiment 2

Since each agent tries to maximize its own profit in the proposed method, individual profit is an important criterion of the experiment. Therefore, the individual profit distribution of agents is tested in Experiment 2. Because it is impossible to show the profits of all agents individually, the profit distribution over the several profit ranges is presented. In this experiment, the impact of the task deadlines on the individual profit distribution is in particular examined.

3.3.2.1 Experimental Settings

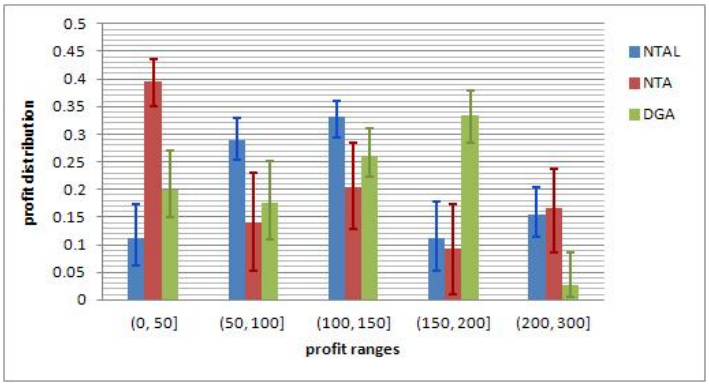
The parameter settings for Experiment 2 are listed in Table 3.4.

Table 3.4: Parameter Settings for Experiment 2

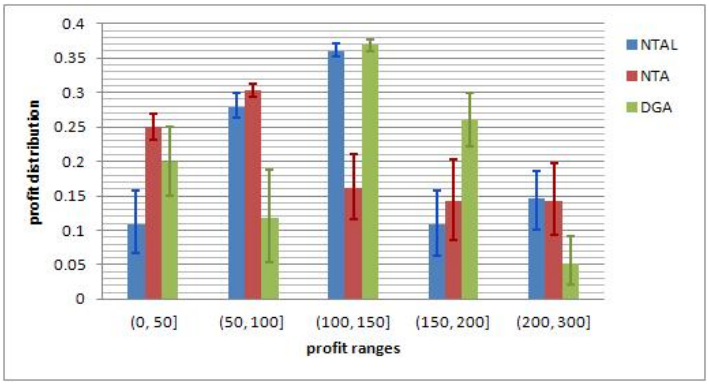
Variables	meanings	values
N_{ra}	number of resource types per agent	[0,10]
$c(R)$	cost to complete a task	[100,150]
$\overline{r}$	maximum reward of a task	[250,500]
$\overline{r}/c(R)$	the ratio between $\overline{r}$ and cost of a task	[1.7, 5]
N_{τ}	number of tasks	[0,100]
N_p	number of resource providers	[0,100]
$\psi(r)$	resource competition	1
N_{ave}	the average required resource types by an agent	5

3.3.2.2 Results of Experiment 2

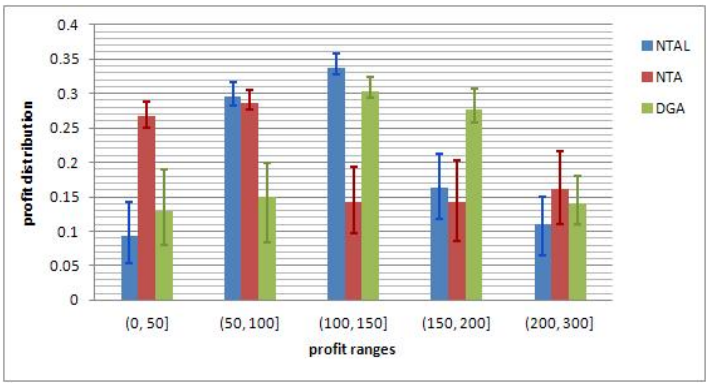
Figure 3.13 shows the performances of the three methods in terms of individual profit distribution based on different task deadlines. The X-axis of Figure 3.13 (a), (b), (c), and (d) represents the profit ranges and the Y-axis is the percentage of the profits that fall into the corresponding profit ranges.



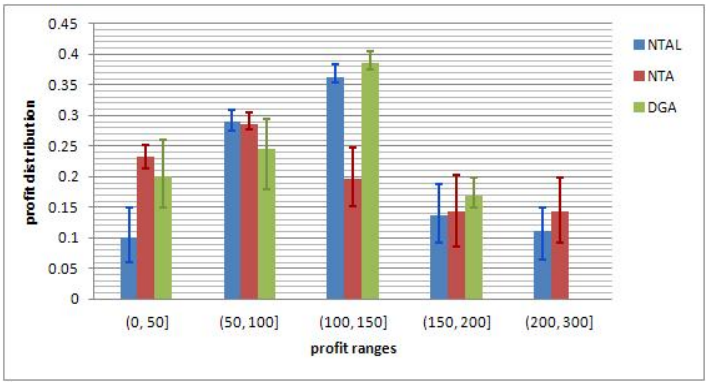
(a) Profit Distribution when Deadlines of Tasks vary from 100 *ms* to 200 *ms*



(b) Profit Distribution when Deadlines of Tasks vary from 200 *ms* to 300 *ms*



(c) Profit Distribution when Deadlines of Tasks vary from 300 *ms* to 400 *ms*



(d) Profit Distribution when Deadlines of Tasks vary from 400 *ms* to 500 *ms*

Figure 3.13: Profit Distribution based on Various Deadlines of Tasks

From Figure 3.13 (a), (b), (c), and (d), it can be seen that compared with NTA, more individual profits of NTAL fall into the middle ranges, i.e., the ranges of [50, 100], [100, 150], [150, 200]. This demonstrates that in NTAL, the profits of agents are distributed more evenly. However, compared with NTAL, more profits of NTA fall into the range of [0, 50] and the range of [200, 300]. Because the range of [0, 50] and the range of [200, 300] are extreme ranges, the gaps in agents' profits in NTA are bigger than those in NTAL. This could easily cause the agents whose profits fall into the range of [0, 50] to lose confidence and interest in remaining in the grid environments. The profit distributions of both NTAL and NTA are reasonable because from Equation (3.4) and Equation (3.5), it can be seen that in NTAL, both the provider and the consumer give concessions due to the time progress during the negotiation process, while in NTA, the counter offer price of the provider is not affected by the time progress. Thus, compared with the providers in NTAL, the providers in NTA give fewer concessions. Consequently, the consumers in NTA give more concessions than the consumers in NTAL. Thus, the profit distribution of NTAL is more even than that of NTA.

It can also be seen from Figure 3.13 (a), (b), (c), and (d) that the profit distribution of DGA is even. This is because DGA does not need negotiation. Rather, it only adopts a greedy algorithm when choosing the task to execute. Thus, compared with NTA, DGA is less sensitive to time constraints of tasks, and this results in the profit distribution of DGA also being even.

In addition to the deadlines, the individual profit distribution based on different resource competitions and different average numbers of required resource types is also examined. The results of such testes and the corresponding reasons are similar to those of Figure 3.13.

3.3.3 Discussion

With the appearance of new features and the consequent challenges of grid/cloud environments, the research about task allocation in such environments should take the new challenges, such as the requirement of decentralisation of task allocations and the openness and dynamism of the environments, into account. Through the experimental comparison between NTAL and NTA, it can be seen that compared with NTA in which all the information of resource providers are known by the consumers, NTAL works as well as NTA or even better in some situations, with each agent only having a local view of the environment. This results from the more efficient negotiation strategy

of NTAL which considered a wider issue space. The comparison between NTAL and DGA demonstrated that the conventional greedy algorithms cannot handle well the new challenges of task allocation in current grid/cloud environments any more.

3.4 Summary

This chapter proposed a decentralised negotiation-based task allocation method for single task allocation. The experimental evaluation demonstrated that the proposed method outperformed two well-known benchmarks in terms of success rate, profit distribution, and the speed of task allocation in open and dynamic grid/cloud environments. The main contributions of the proposed method include (i) the method is based only on local views of agents, which can make the method more applicable in open, dynamic environments, and (ii) the proposed method allows both providers and consumers to freely enter and leave grid environments, so it has wide applicability. The contributions of this method can achieve **Objective 1** of this thesis, set in Chapter 1.

Chapter 4

An Auction-based Method for Group Task Allocation in An Open Grid Environment

This chapter addresses the problem of the task allocation that consists of a group of independent subtasks. The required resource of the task, i.e., the required resources of the group of subtasks, distribute on multiple administratively or locally independent resource providers. Supply chain formation, which is a widely studied research problem in recent years, is one of the applications of group task allocation. For example, someone wants to assemble a computer through obtaining all the required hardware components in an e-market. In the e-market, there are multiple providers each of which can provide one or more parts including CPU, memory, displayer, keyboard, et. al. In the dynamic e-market, resource providers have to collaborate to form a supply chain in order to finish the task of a computer assembly, with the objective of minimising the total cost or maximising the total performance of the assembled computer. If obtaining the CPU is viewed as a subtask of the computer assembly, and obtaining the keyboard is also viewed as a subtask and so on, the computer assembly is a group task allocation problem. To solve this kind of task allocation, a combinatorial reverse auction-based group task allocation approach is proposed, which devises an indicator for all the combinations of bidders (providers) that can meet the resource requirement of the task.

The outline of this chapter is organised as follows. Problem description and the related definitions are presented in Section 4.1, and the proposed task allocation method is in detail introduced in Section 4.2. In Section 4.3, the method is experimentally evaluated, and a brief discussion is given in Section 4.4. This chapter is summarised in Section 4.5.

4.1 Problem Description

In this section, the characteristics of the group task allocation is introduced, then the main components of the problem are formally defined.

The group task allocation addressed in this chapter has six important characteristics.

1. A type of resource may be provided by multiple providers and required by more than one task simultaneously.

2. All of the required resource types of a task are distributed on multiple administratively independent organizations, such as universities or companies.

3. Both resource providers and consumers are self-interested and try to maximise their own profits.

4. Old tasks expire and new tasks appear unpredictably over time.

   Similar to tasks, resources are dynamic as well, being withdrawn or becoming available unpredictably.

5. Both the consumer and the provider only have a local view.

6. Allocation of a task is under an overall time constraint.

   An overall time constraint means that for the allocation of any individual part of the task, there is no time constraint, but for the allocation of the whole task, there is a time constraint (i.e., the latest start execution time of the task, which will be defined later).

Definition 4.1 (Task): A *task*, denoted as τ_k, is defined by a 6-tuple $(R_k, Sub_k, t_{kgen}, d_k, t_{kl}, \overline{r}_k)$, where R_k is the set of the required resources of τ_k, $Sub_k = \{sub_1, sub_2, ..., sub_n\}$ is the set of subtasks of τ_k (n is the number of subtasks of τ_k), t_{kgen} is the generation time of τ_k, d_k is the deadline of τ_k, t_{kl} is the latest start execution time of τ_k, and $\overline{r}_k$ represents the maximal reward that can be obtained by τ_k's owner if τ_k is finished successfully.

The allocation of a task is considered successful if and only if the task is successfully allocated before its latest start execution time. The execution of a task will be terminated when the deadline comes, no matter whether the task has been finished or

not. The task's owner, i.e., the consumer, can obtain profits from the task only if the task is finished before its deadline.

In this chapter, each entity in a grid environment is still modeled as an agent which can make decisions autonomously. When an agent needs other agents' resources to execute its tasks, it is a consumer. When an agent provides its resources to other agents, it is a provider. In this chapter, an agent can be a consumer or a provider, or even both. The **definition of an agent** in this chapter is the same with that in Chapter 3 (i.e., **Definition 3.1**). An agent judges whether its neighbours are still active in the environment through a heartbeat message, of which the definition is also the same with **the definition of the heartbeat message** in Chapter 3 (i.e., **Definition 3.5**).

An agent keeps sending a heartbeat message to its neighbours once each time unit to inform its neighbours that it is still active in the environment. If agent a_i has not received any heartbeat message from its neighbour, a_j, in the previous time unit, it considers that a_j has left the environment and thus abandons the neighbourhood with a_j.

Besides the heartbeat message, there are two other types of messages passed between two neighbouring agents: the request message and the reply message for executing tasks. It is still assumed that a_i and a_j are neighbours. The request message sent from a_i to a_j, denoted by $ReqExecute_{ij}$, means that a_i requests a_j to execute a_i's task. The **definition of the request message**, $ReqExecute_{ij}$, in this chapter is the same with that (i.e., **Definition 3.6**) in Chapter 3, whereas the definition of the reply message sent from a_j to a_i, i.e., $RepExecute_{ji}$, is different from that in Chapter 3, due to the different types of tasks to be allocated in Chapters 3 and 4. In this chapter, the reply message is redefined by Definition 4.2.

Definition 4.2 (Reply Message): $RepExecute_{ji}$ is defined by a 6-tuple $(ID_j, ID_i, t_{fin}, \tau_k, Set_r, pri_j)$, where ID_j and ID_i represent the message sender's ID and the recipient's ID, respectively, t_{fin} is the time when a_j can finish τ_k, Set_r is the resource set that a_j can provide to τ_k, and pri_j is the bidding price asked by a_j as the payment for Set_r.

4.2 The Combinatorial Auction-based Task Allocation Method

In this section, the reasons for choosing first-price sealed-bid combinatorial auction as the basis of ICAA is introduced in Section 4.2.1, then the task allocation process is described in Section 4.2.2. The candidate group formation algorithm and the indicator design are introduced in detail in Sections 4.2.3 and 4.2.4, respectively.

4.2.1 The Reasons for Choosing First-price Sealed-bid Combinatorial Auction as Basis

The required resources of a task in ICAA are distributed on multiple providers, and combinatorial auction is suitable to assemble the required resources. Conventionally, there is a public auctioneer to run auctions for all the consumers in a combinatorial auction but the public auctioneer is a central controller and there are both computational and economic disadvantages of this. From the perspective of computation, it becomes computationally intractable for the central controller when more providers and consumers join the auction. From the perspective of economy, it is difficult to find an auctioneer that can be trusted by self-interested bidders in market-based environments. Against this, decentralization is implemented through assigning the consumer to play the role of the auctioneer for itself, and the consumer obtains bids through its own neighbourhood. In order to be robust against the dynamism and openness of the environment, consumers in ICAA take the dynamism and openness of the environment into consideration by encoding them into the indicator (which will be formulated in 4.4) of each candidate group.

First-Price Sealed-Bid Combinatorial Auction (FPSBCA) is adopted as the basis of ICAA but with some modification from the original FPSBCA, the standard for choosing the winning bidder is the bidding price, whereas in ICAA, this standard is replaced by a comprehensive one (formulated by Equations (4.1) and (4.2)) which takes more factors into account in addition to the bidding price.

There are three reasons for this research to adopt first-price sealed-bid combinatorial auction (FPSBA) as the basis. First, the winner bidder in FPSBCA can verify that the payment it gains is indeed the bid that it made. In contrast, in a second price sealed-bid auction, the winner has to trust the center to fairly compute the payment based on other bidders' bids, which it cannot directly observe. Second, FPSBCA is

still efficient, even if it is not incentive compatible. Third, each bidder in FPSBCA bids only once, and this can save time, which is important for task allocation under time constraints.

4.2.2 The Task Allocation Process

Suppose that a_i is a consumer, when a_i has a task τ_k to be finished, it sends a request message (see **Definition 3.6**) to all of its neighbours. The agent who receives the request message, a_j, checks whether it has some of the required resources. If so, it sends a reply message (see **Definition 4.2**) to a_i, and the reply message is viewed as the bid made by a_j. If not, and if HL in the request message is not 0, a_j transmits the request message to its own neighbours. Otherwise, a_j abandons the received message. Assume that the transmission of a request message by an intermediate agent is safe for the message, and the reason for this is that in real life applications, it is reasonable to assume that a provider is willing to help transmit the message if it does not have any of the required resources. In contrast, if a provider who received the request message has some required resources, it will not transmit the request message to any other providers, in order to reduce the number of its potential competitors. This is why it is stipulated that a provider transmits the request message only when it does not have any of the required resources. If a request message is transmitted, it is re-assembled by the intermediate agent who will change some information in the request message before transmitting the message to its own neighbours. In detail, the ID of the old recipient in the request message is replaced with the ID of its new recipient, and HL is reduced by 1. The transmission will not end until HL becomes 0. If a recipient of the request message has some or all of the required resources of τ_k, but is not the immediate neighbour of a_i, it has to construct a neighbourhood relationship with a_i before sending a reply message straight to a_i. The reason for this is the competitiveness in the environment. The reply message of a provider may be secretly modified by other competitor providers. Consequently, the provider will send a reply message straight to the consumer to avoid the transmission of the reply message.

Due to the transmission of the request message, it is possible that consumer a_i cannot receive any reply message for a short time after its request message has been sent. Thus, it is stipulated that a_i will not start group formation until a predefined time point before the latest start time of its task. If a_i has not received any reply message when the latest start time of its task arrives, it will give up the task. Otherwise,

once a_i starts group formation, it will ignore any later arriving reply messages. Group formation, which will be described in detail in Section 4.2.3, is the process where a_i forms candidate groups after obtaining bids through neighbourhoods. A candidate group is a group of providers which can collaboratively finish the consumer's task. A candidate group is irreducible, which means that the candidate group cannot finish the task without any provider in the group. Given the candidate groups formed, a_i selects the most suitable one.

With the above introduction to the task allocation process in mind, the task allocation in GENI is taken as an example to illustrate the task allocation process. Suppose that in GENI, research group A plans to do a large scale research project, which requires the following resources: a data center to get raw data, high performance CPU clusters to calculate the raw data, and storage to store the calculation results. A medium to be used for data transmission is also required. Due to the large scale of the research project, the research group logs on to GENI to rent the required resources. Suppose that the related resource distribution in GENI is that: Amazon EC2 can provide both the storage and CPU services, HP Cloud can contribute both the transmission and storage services, University A can provide both the data center and CPU services, and University B can provide the data center service. Given the resource distribution, there are multiple combinations of resource groups (combinations), which can finish the research project. These resources may be withdrawn, however, and some new unexpected resources may arrive. The neighbourhood structure of the example is presented in Figure 4.1.

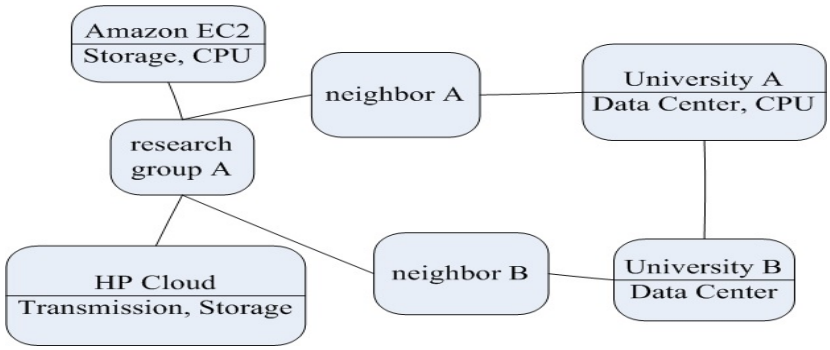


Figure 4.1: Neighbourhood Structure

In Figure 4.1, the immediate neighbouring agents of research group A include Amazon EC2, HP Cloud, and neighbours A and B. The consumer (i.e., research group A) sends a request message to all of its neighbouring agents, and all the recipients

of the request message check their own available resources. According to the resource distribution (it is assumed that neighbours A and B do not have any resources required by the consumer), Amazon EC2 and HP Cloud send reply messages (i.e., bids) to the consumer. Amazon EC2 bids to lease both storage and CPU services, and HP Cloud bids to lease both transmission and storage services to the consumer. Neither neighbour A nor neighbour B has any required resources, thus, neighbour A transmits the request message to its own neighbouring agent, University A. Neighbour B transmits the request message to its own neighbouring agent, University B. After receiving the transmitted request message and checking their own available resources, both Universities A and B first construct neighbourhoods with the consumer, then University A bids to lease both data center and CPU services, and University B bids to lease data center service. All the eventual bids received by the consumer are presented in Figure 4.2.

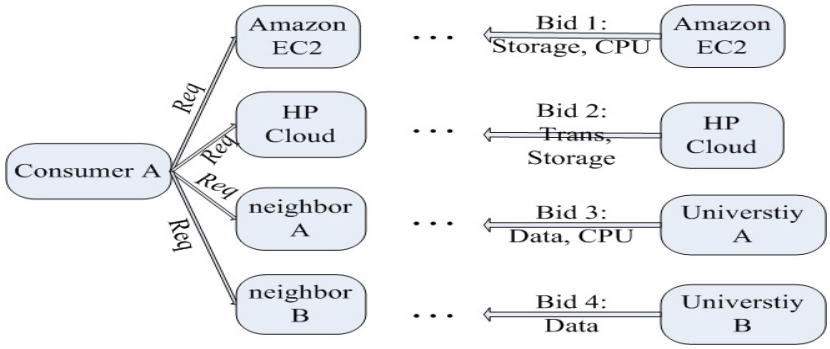


Figure 4.2: Bids Obtained Through Neighbourhood

According to the bids, there are eight candidate groups which are presented in Table 4.1.

Table 4.1: Candidate Groups

	Amazon EC2	HP Cloud	University A	University B
Candidate 1		Trans, Storage	Data, CPU	
Candidate 2		Trans, Storage	CPU	Data
Candidate 3	CPU	Trans, Storage	Data	
Candidate 4	CPU	Trans, Storage		Data
Candidate 5	Storage, CPU	Trans	Data	
Candidate 6	Storage, CPU	Trans		Data
Candidate 7	Storage	Trans	Data, CPU	
Candidate 8	Storage	Trans	CPU	Data

Now it is explained how the consumer selects the most suitable of the eight candidate groups. Generally, provided that consumer a_i obtained in total k candidate groups, which are denoted by $gro_1, gro_2, ..., gro_k$. The indicators of these groups are $indic_1, indic_2, ..., indic_k$ (the indicator design will be introduced in detail in Section 4.2.4), and the prices asked by these groups are $pri_1, pri_2, ..., pri_k$ (asking prices of providers will be studied later). It is stipulated that the winning group is the one, which has the smallest $pri/indic$ ratio. Accordingly, if

$$j = \mathbf{arg}min_i f(i), \tag{4.1}$$

and

$$f(i) = pri_i/indic_i \tag{4.2}$$

where $1 \leq i \leq k$, $1 \leq j \leq k$ $(i, j \in \mathbb{N})$, then gro_j is the winning group. From Equations (4.1) and (4.2), the winning group is chosen from an economic perspective because the asking price pri_i of group gro_i is taken into consideration by the consumer. Indicator $indic_i$ in Equation (4.2) is designed from an economic perspective as well, and the reasons will be analysed when the indicator design is introduced in Section 4.2.4. When there is more than one group that can meet Equation (4.1) and Equation (4.2) simultaneously, a_i picks out the winner randomly. The reason for deciding the winning group using Equation (4.1) and Equation (4.2) will be seen in Subsection 4.2.2.

Due to the competitiveness in the environment and selfishness of bidders, bidders do not always truthfully bid the costs of their resources. Instead, they tend to bid higher prices than the actual costs to gain more profits. Generally, multiple factors are taken into account by bidders. These include the popularity of the resource, the

truthful cost of the resource, and the composition of the bundle of resources that the provider bids for. In detail, the more popular the resource is, the higher the bidding is; the more the resource costs, the higher the bidding is; and the more resources in the bundle of resources that the provider bids to lease, the lower the individual bidding for each of the resources in the bundle.

Based on the analysis above, now the providers' bidding strategy is formally formulated. It is assumed that a_i and a_j are the consumer and the provider, respectively, and a_j bids to lease a bundle of resources to a_i. If r is one of the resources in the bundle, n is the total number of resources in the bundle, and $c(r)$ is the cost of r, n_{req} is the total number of requests for r that a_j received from consumers, then the bidding of a_j for resource r (denoted by $bid(r)$) is formulated by:

$$bid(r) = c(r)(1 + \frac{log_2(n_{req} + 1)}{n}) \tag{4.3}$$

In Equation (4.3), $log_2(n_{req}+1)$ is adopted to weaken the effect of n_{req} to $bid(r)$ which will easily be dominating. In order to make the bidding higher than the truthful cost, $\frac{log_2(n_{req}+1)}{n}$ instead of $\frac{log_2(n_{req})}{n}$ is adopted. The reason for this is if $\frac{log_2(n_{req})}{n}$ is adopted, $bid(r) = c(r)$ when $n_{req} = 1$. It is notable that in Equation (4.3), $n_{req} \geq 1$ in that once the provider calculates the bidding using Equation (4.3), it represents that the provider has received at least one request for resource r.

The winning group will sign a contract with the consumer. The contract between consumer a_i and provider group gro (where gro contains n providers, i.e., $gro = \{a_1, a_2, ..., a_n\}$), denoted as $c(a_i, gro)$, is defined by Definition 4.3.

Definition 4.3 (Contract): $c(a_i, gro)$ is defined by a (5+n)-tuple: (a_i, gro, τ_k, rew, t_{fin}, pri_1, pri_2 ,..., pri_n), where rew is the reward that a_i can gain if τ_k can be finished at time t_{fin}, t_{fin} is the time when τ_k can be completed by gro (t_{fin} will be particularly formulated in Section 4.2.4), and $pri_1, pri_2, ..., pri_n$ are the prices which will be paid by a_i to $a_1, a_2, ..., a_n$, respectively, if τ_k can be finished at time t_{fin}.

In order to make the task allocation process clear, the task allocation process of ICAA is presented from both the perspectives of a consumer and a provider, which are described by Algorithms 4.1 and 4.2, respectively.

In Algorithm 4.1, consumer a_i has a task τ_k to be executed, and has n neighbours which are denoted by $a_1, a_2, ...,$ and a_n. a_i sends a request message for executing τ_k to all of its neighbours (**Lines 1-3**). After obtaining the reply messages from providers, group formation is conducted (**Line 4**). a_i selects the winning group from all the

Algorithm 4.1: Task Allocation Procedure From the Perspective of A Consumer

Assume that a_i has n neighbours from a_1 to a_n;
1 for (j = 1 to n)
2 | a_i sends a request message (see **Definition 3.6**) to a_j to request a_j to execute τ_k;
3 end for
4 group formation;
5 a_i picks out the winning group according to Equation (4.1) and Equation (4.2);
6 if there is more than one group which have the same value of *pri/indic* **then**
7 | a_i picks out the winning group randomly among these groups;
8 end if
9 a_i signs a contract with the winning group.

candidate groups (**Lines 5-8**), and signs a contract with the winning group (**Line 9**).

Algorithm 4.2: Task Allocation Procedure From the Perspective of A Provider

Assume that a_j is a provider and a_i is a consumer;
1 When a_j receives a request message for executing τ_k from a_i,
2 it checks whether $R_j \bigcap R_k = \emptyset$.
3 if $R_j \bigcap R_k \neq \emptyset$ **then**
4 | a_j sends a reply message (see **Definition 4.2**) back to a_i;
5 else if $HL \neq 0$
6 | | a_j transmits the request message to its neighbours; $HL = HL - 1$;
7 | **else** a_j gives the request message up;
8 end if
9 if a_j is selected by a_i
10 | a_j signs a contract with a_i.
11 end if

In Algorithm 4.2, when the provider agent, a_j, receives a request message for executing task τ_k from a consumer a_i, it checks whether it has a partial set or all of the required resources of τ_k (**Lines 1-2**). If yes, a_j sends a reply message to a_i (**Lines 3-4**). If not, and 'HL' in the request message does not equal to 0, a_j transmits the request message to all of its own neighbours (**Lines 5-6**). Otherwise, a_j does nothing to the request message (**Lines 7-8**). If a_j is selected by a_i, a_j signs a contract with a_i (**Lines 9-11**).

4.2.3 The Group Formation

Provided that P is the set of all bidders that bid for the resources required by a_i's task τ_k, $r \in R_k$, and the number of bidders that bid for r is n_p^r, there are four steps for a_i to form all the candidate groups, which is presented in Algorithm 4.3.

Algorithm 4.3: Group Formation (pick out all the candidate groups)

Step 1: Pick out the resource type which has the fewest bidders;
1 Initialize an array $N[|R_k|]$=0
2 for i = 0 to ($|R_k|$-1) **do**
3 | **for** j=0 to ($|P|-1$) **do**
4 | | **if** $R_k[i] \in R_{P[j]}$ **then**
5 | | | N[i]++
6 | | **end if**
7 | **end for**
8 end for
9 $n = N[0]$;
10 for i = 0 to ($|R_k|$-1) **do**
11 | **if** $N[i]<n$ **then**
12 | | $n = N[i], r = R_k[i]$
13 | **end if**
14 end for
Step 2: Insert every bidder that bids for r into a queue P_r;
15for j = 0 to ($|P|$-1) **do**
16 | **if** $r \in R_{P[j]}$ **then**
17 | | insert $P[j]$ into P_r
18 | **end if**
19end for
Step 3: Recursive calling;
20for i = 0 to ($|P_r|$-1) **do**
21 | pick out the group $P_{-r}[i]$ which can meet the resource requirement of $R_k \setminus (R_{P_r[i]} \cap R_k)$
22 | from $P \setminus P_r[i]$ by recursively calling the first three steps of this algorithm,
23 | so that $P_{-r}[i] \cup P_r[i]$ can meet the resource requirement of τ_k and is irreducible
24end for
Step 4: Deal with the eventually obtained candidate groups;
25for i = 0 to ($|C_k|$-1) **do**
26 | **for** j = 0 to ($|C_k|$-1) **do**
27 | | **if** $i \neq j$ and $C_k[j] \subseteq C_k[i]$ **then**
28 | | | $\varnothing \longrightarrow C_k[i]$
29 | | **end if**
30 | **end for**
31end for

Algorithm 4.3 is explained as follows:

Step 1 (Lines 1-14): Pick out resource r, where $r = \textbf{arg}\ min_{r \in R_k}(n_p^r)$.

Step 2 (Lines 15-19): Pick out the bidders set P_r in which every bidder bids for r, that is, if the provider in P_r at index k is denoted by $P_r[k]$, then $\forall k \in (1, ..., |P_r|), r \in R_k$.

Step 3 (Lines 20-24): For each $P_r[k] \in P_r$, pick out the group $P_{-r}[k]$, which can meet the condition of $R_k \setminus (R_k \cap R_{P_r[k]}) \subseteq R_{P_r[k]}$ from $P \setminus P_r[k]$.

If the element of $P_{-r}[k]$ at index j is denoted by $P_{-r}[k][j]$, then $R_k \subseteq (R_{P_r[k]} \cup R_{P_{-r}[k]})$, where $R_{P_{-r}[k]} = \cup_{j=1}^{|P_{-r}[k]|} R_{P_{-r}[k][j]}$ and $\forall h \in (1, ..., |P_{-r}[k]|), R_{P_r[k]} \cup (R_{P_{-r}[k]} \setminus R_{P_{-r}[k][h]}) \subset R_k$. a_i obtains $P_{-r}[k]$ from $P_r \setminus P_r[k]$ through recursively calling Steps 1, 2 and 3. It is done the same to each $P_r[i] \in P_r$ $(i \in 1, ..., |P_r|)$ as what it is done to $P_r[k]$.

Step 4 (Lines 25-31): Suppose that from **Steps 1, 2** and **3**, a_i can obtain n candidate groups for τ_k in total, and C_k is used to denote the set of the candidate groups. The element of C_k at index i $(1 \leq i \leq |C_k|)$ is denoted by $C_k[i]$. What it is done in this step is to delete group $C_k[i]$, if $\exists j \in (1, ..., |C_k|)$ and $j \neq i$, $C_k[j] \subseteq C_k[i]$.

An example is given to demonstrate the group formation process to make it more clear. It is assumed that the resource types required by task τ_k is $R_k = \{1, 2, 3, 5\}$. There are seven bidders which are represented by A, B, C, D, E, F and G, respectively. The resource sets bided by the bidders are {1, 2}, {2, 3}, {1, 2, 5}, {3, 5}, {1, 5}, {3} and {2, 3, 5}, respectively. The candidate group formation process is as follows.

Step 1: Resource type 1 is chosen because it has the fewest bidders, i.e., bidders A, C and E.

Step 2: Because A, C and E can provide resource type 1, they are chosen.

Step 3: Provider A is fixed, then the groups that can meet the resource requirement of τ_k with A include: ABC, ABE, ACF, AEF, AD and AG. In the same way, the groups with C can be obtained: BC, CD, CF and CG. The groups with E include: BE, EG, ADE, AEF, CDE, and CFE.

Step 4: Delete the groups of ABC, ABE, ACF, AED, CDE and CEF, because BC $\subset$ ABC and BE $\subset$ ABE, CF $\subset$ ACF, AD $\subset$ ADE, CD $\subset$ CDE and CF $\subset$ CEF. Then the eventually obtained candidate groups include AEF, AD, AG, BC, CD, CF, CG, BE and EG.

Time Complexity of Group Formation

Time complexity is important to group formation in combinatorial auctions, especially when there are time constraints. Therefore, the time complexity of the coalition formation process (i.e., Algorithm 4.3) is analysed. Firstly, the four steps of Algorithm 4.3 are analysed separately.

Step 1: $\mathcal{O}(|R_k| \times |P|) + \mathcal{O}(|R_k|)$. $|R_k|$ and $|P|$ are denoted by m and n respectively, thus, the complexity of step 1 is $\mathcal{O}(m \times n) + \mathcal{O}(m) \rightarrow \mathcal{O}(m \times n)$.

Step 2: $\mathcal{O}(|P|)$, that is, $\mathcal{O}(n)$.

Step 3: Step 3 is a loop, and the loop body is a recursion. If the time complexity of the loop body is denoted by $T(R_m)$ (m is the number of the required resource types of τ_k), then the time complexity of step 3 is: $|P_r| \times T(R_m)$. $T(R_m)$ is calculated through the recursion equation: $T(R_m) = T(R_{m-1}) + \mathcal{O}(n \times m)$. From the recursion equation, it has: $T(R_m) = \mathcal{O}(m^2 \times n)$. Thus, the time complexity of step 3 is: $|P_r| \times \mathcal{O}(m^2 \times n) \rightarrow \mathcal{O}(|P_r| \times m^2 \times n)$.

Step 4: $\mathcal{O}(|C_k| \times |C_k|) \rightarrow \mathcal{O}((|C_k|^2)$.

According to the above analysis, the time complexity of Algorithm 4.3 denoted by $T(Al_2)$ can be known: $T(Al_2) = \mathcal{O}(m \times n) + \mathcal{O}(n) + \mathcal{O}(|P_r| \times m^2 \times n) + \mathcal{O}(|C_k|^2) \rightarrow \mathcal{O}(|P_r| \times m^2 \times n)$.

4.2.4 The Indicator Design

In this subsection, the factors that are taken into account by the indicator to make ICAA robust to the dynamism and openness of the network environment, are introduced first, then the indicator is formulated accordingly.

4.2.4.1 The Factors of the Indicator Design

In addition to the price asked by a candidate group, the consumer takes four other factors into account when choosing the winning group in ICAA. The four factors are:

(i) the probability of no decommitment in a group, **(ii)** the number of overlapping resource types in a group, **(iii)** the reputation of a group and **(iv)** the reward that can be gained by the consumer from a group.

(i) Probability of no decommitment

In ICAA, a provider is allowed to deviate from an existing group to join another more profitable one, that is, decommitment is allowed. Consequently, a formed group is unstable due to the possible deviations of committed providers. The external offers and opportunities that may provide incentives to providers to deviate from commitments have been studied in other research (e.g., [Sen13]), and it is not the research focus of this chapter. Because a candidate group in ICAA is irreducible, the decomittment of any provider in the group will cause the risk of the failure of the task. As a consequence, the probability of no decommitment of a candidate group should be considered. To do this, a decommitment model is devised to help the consumer predict the probability of no decommitment in a candidate group.

It is assumed that different providers have different probabilities to decommit from a contract, and the decommitments of providers are independent, that is, a provider's decommitment is an independent decision, not affected by other providers in the same group. A consumer anticipates the probability that a provider will not decommit from it according to the trade history with this provider. Suppose that Consumer a_i and Provider a_j have already traded with each other for k_1 ($k_1 \in \mathbb{N}$) times, and a_j decomitted with a_i for k_2 ($k_2 \in \mathbb{N}$) times in total. Normally, if the frequency with which decommitment has happened before is k_2/k_1, then the most likely future probability of decommitment is k_2/k_1 as well. Consequently, the most likely probability of no decommitment of a_j in the future is $(k_1 - k_2)/k_1$. A Beta distribution, $Beta \sim (\alpha, \beta)$, is adopted to formulate the probability distribution of no decommitment. Accordingly, the two shape parameters of the Beta distribution, α and β, are set $k_1 - k_2 + 1$ and $k_2 + 1$, respectively. Therefore, if the distribution of no decommitment probability is denoted by p_{nodec}, then $p_{nodec} \sim Beta(k_1 - k_2 + 1, k_2 + 1)$.

There are two reasons for this research to adopt a Beta distribution to formulate p_{nodec}. First, Beta distribution can be updated as more experience is gained, that is, the values of α and β can be updated with the changing of k_1 and k_2. Second, by moderating over the resulting distribution, the issue of 0 probability can be avoided. The issue of 0 probability must be avoided, and the reason for this will be given later in Equation (4.4).

Now suppose that Consumer a_i has already signed a contract with group gro ($gro = \{a_1, a_2, ..., a_n\}$), the probabilities of these n providers not to decommit from a_i are denoted by $p_{nodec}(1)$, $p_{nodec}(2)$, ..., $p_{nodec}(n)$ ($p_{nodec}(i) \in \mathbb{R}$, $1 \leq i \leq n$), respectively. Due to the independence of these probabilities, according to the multiplication principle, the probability that no provider in gro will decommit from a_i (denoted by $P_{nodec}(gro)$) is:

$$P_{nodec}(gro) = \prod_{j=1}^{n} p_{nodec}(j) \tag{4.4}$$

From Equation (4.4), being equal to 0 of any $p_{nodec}(j)$ will result in that $P_{nodec}(gro)$ is 0, therefore, being equal to 0 of any $p_{nodec}(j)$ have to be avoided.

(ii) Number of overlapping resource types in a group

There might be overlapping resources among the resources provided by all the providers in a group. Suppose that a provider group consists of two providers, denoted by $gro = \{a_1, a_2\}$, the resource types set that a_1 can provide is $set_1 = \{1, 2, 3\}$, and that of a_2 is $set_2 = \{3, 4, 5\}$. Apparently, there is one overlapping resource type, i.e., resource type *'3'*. Generally, provided that there are n providers in gro, denoted by $gro = \{a_1, a_2, ..., a_n\}$, the way the number of the overlapping resource types in gro is calculated as that: the number of the overlapping resource types between a_1 and a_2 is calculated first, then a_1 and a_2 are treated as one entity denoted by a_{12}. The same method is used to calculate the number of overlapping resource types between a_{12} and a_3, and so on until all the providers in gro have been involved into the calculation. The eventually accumulated number of overlapping resource types is treated as the number of overlapping resource types in gro.

If a resource type can be provided by more than one provider in a group, and the resource from one provider will be enough, this resource from other providers will not be needed. In ICAA, a provider is allowed to join more than one group simultaneously to avoid the waste of the extra resources. However, after contributing one part of the resources to one consumer, the extra resources always include only a few resource types. The fewer resource types each provider can averagely provide in a group, the more providers the group includes. From **Step 4** of the group formation algorithm, It is known that normally, the more providers in a group, the higher the decommitment probability of this group is. Consequently, a consumer prefers a provider who can provide more resource types to that provides fewer resource types. Therefore, even though a provider is allowed to join more than one group, its probability of being

selected by other consumers is quite small. According to the 'win-win' rule in economy, the desirable behaviour of a consumer should be reducing overlapping resource types as much as possible when its own profit will not be affected. For this reason, fewer overlapping resources in a group is preferred by a consumer.

(iii) Reputation of a group

The QoS (Quality of Service) of providers may be different, so it is important for consumers to take the potential QoS of providers into account when choosing the winning group. To address this concern, consumers anticipate the potential QoS of providers according to the providers' QoS in the past. A consumer can evaluate the QoS of a provider according to different criteria. For example, if tasks are time-sensitive, the QoS of a provider could be evaluated according to whether the provider finished tasks before the task deadlines in the past services.

Based on the trust model proposed by Yu and Singh in [YS02], a general method is proposed to help consumers formulate providers' reputations in terms of QoS. Assume that Consumer a_m wants to anticipate the reputation of Provider a_n, and a_m has already got k ($k \geq 1$) services delivered by a_n, which are denoted by $S = \{s_1, s_2, ..., s_k\}$. Provided that $r_1, r_2, ..., r_k$ $r_i \in (0, 0.1, 0.2, ..., 1)$ $(1 \leq i \leq k)$ are the respective reputation values of $s_1, s_2, ..., s_k$, a low threshold ω $(0 < \omega < 1)$ and an upper one Ω $(0 < \omega < \Omega < 1)$ are defined to classify the reputation values. In detail, $S(T) = \{s_i | \Omega \leq r_i \leq 1, 1 \leq i \leq k\}$ denotes the support set of trust, and $S(\neg T) = \{s_j | 0 \leq r_j \leq \omega, 1 \leq j \leq k\}$ denotes the support set of distrust. When $k \geq 1$, the reputation of a_n, denoted by r_{a_n}, is formulated by:

$$r_{a_n} = e^{(|S(T)| - |S(\neg T)|)/k} \tag{4.5}$$

From Equation (4.5), it is known that $\frac{1}{e} \leq r_{a_n} \leq e$. The reason to formulate r_{a_n} by an exponential function is to guarantee that the higher $|S(T)| - |S(\neg T)|$ is, the higher r_{a_n} is.

When $k = 0$, that is, a_m has never got service from a_n, r_{a_n} is set $(e + \frac{1}{e})/2$. It is reasonable to do this because $(e + \frac{1}{e})/2$ is the median between $\frac{1}{e}$ and e.

In summary, the reputation of a_n for a_m is formulated as:

$$r_{a_n} = \begin{cases} e^{(|S(T)| - |S(\neg T)|)/k} & (k \geq 1) \\ (e + \frac{1}{e})/2 & (k = 0) \end{cases} \tag{4.6}$$

Now the reputation of a provider group is formulated. Assume that the reputations

of the n providers in $gro = \{a_1, a_2, ..., a_n\}$ are $r_{a_1}, r_{a_2}, ...,$ and r_{a_n}, respectively, if $r_{min} = min\{r_{a_1}, r_{a_2}, ..., r_{a_n}\}$, and $r_{max} = max\{r_{a_1}, r_{a_2}, ..., r_{a_n}\}$, the reputation of gro is formulated as:

$$r_{gro} = \begin{cases} ((\sum_{i=1}^{n} r_{a_i}) - r_{min} - r_{max})/(n-2) & (n \geq 3) \\ (\sum_{i=1}^{n} r_{a_i})/n & (1 \leq n \leq 2) \end{cases} \tag{4.7}$$

The reason to subtract r_{max} and r_{min} when $n \geq 3$, is to avoid the strong impacts of the maximal and minimal reputations on gro.

(iv) Reward gained by a consumer

In market-based environments, the reward that a consumer can gain from its task is one of the critical factors to be considered and thus is encoded into indicator design as well in ICAA. Apparently, when a task is executed by a provider group, the whole task cannot be finished until every provider in the group finishes its own subtasks. Formally, if there are k providers in group gro, and the respective times when the k providers can finish their own subtasks of τ_k are $t_{fin1}, t_{fin2}, ...,$ and t_{fink}, respectively, then the time when τ_k can be finished is: $t_{fin} = max\{t_{fin1}, t_{fin2}, ..., t_{fink}\}$. In ICAA, tasks are time-dependent, which means that the reward that can be gained by a consumer is closely related to the finishing time of its task. Time-dependency is common and normal for both rewards and penalties, which will be particularly defined in Subsection 4.3.2, in the market-based environment.

A piecewise function is adopted to represent the relationship between the finishing time of τ_k and the reward that τ_k's owner can gain. If the finishing time of τ_k is t, without loss of generality, the reward that τ_k's owner can gain, denoted by $rew_k(t)$, is formulated by:

$$rew_k(t) = \begin{cases} \overline{r}_k/(t_1 - t_{kgen}) & t_{kgen} < t \leq t_1 \\ \cdots & \\ \overline{r}_k/(t_{i+1} - t_{kgen}) & t_i < t \leq t_{(i+1)} \\ \cdots & \\ \overline{r}_k/(d_k - t_{kgen}) & t_n < t \leq d_k \\ 0 & d_k < t \end{cases} \tag{4.8}$$

where $\overline{r}_k$ is the maximal reward that task τ_k's owner can gain when τ_k is finished, t_{kgen} and d_k are the generation time and the deadline of τ_k, respectively (see **Definition**

4.1), and $t_1, ..., t_i, ..., t_n$ (t_{kgen}<t_1<...<t_i... <t_n<d_k) are different time points.

Apparently, Equation (4.8) can reflect the time-dependency of tasks in that the later τ_k is finished, the less reward can be gained by τ_k's owner.

4.2.4.2 Formulation of the Indicator

In order to encode the four factors analysed above into the indicator, the simple multi-attribute rating technique (SMART), which is based on a linear additive model, is employed to devise the indicator. Based on SMART, a comprehensive score, which is the summation of the performance scores of all the attributes multiplied by the corresponding weights of the attributes, is used to rate all the candidate providers. For more details about SMART, please refer to [Edw77, VWE$^+$86].

Formally, provided that gro is one of the candidate groups for task τ_k of a_i, a_i assigns equal importance to all the four attributes. For simplicity, the weights of the four attributes are all set to 1. The indicator of gro, denoted by $indic_{gro}$, is formulated by:

$$indic_{gro} = P_{gro} + 1/log_2(No.r + 2) + rew_k(t_{fin})/\overline{r}_k + r_{gro}, \tag{4.9}$$

where $No.r$ is the accumulated number of overlapping resource types in gro, the probability that no provider in gro will decommit from a_i is P_{gro}, the time when gro can finish τ_k is t_{fin}, $rew_k(t_{fin})$ is the reward that a_i can gain if τ_k is finished at time t_{fin}, $\overline{r}_k$ is the maximal reward that a_i can gain when τ_k is finished, and the reputation of gro is r_{gro}.

Because $0 \leq rew(t_{fin})/\overline{r}_k \leq 1$, $1/e \leq r_{gro} \leq e$, $0 \leq P_{gro} \leq 1$, and $0 \leq No.r \leq \infty$, it is apparent that even a very weak fluctuation of $No.r$ can influence the value of $indic_{gro}$ strongly. In order to avoid such a strong impact of $No.r$, the impact of $No.r$ is restrained through introducing the logarithmic function of "log_2" into Equation (4.9). In addition, in order to avoid "$1/log_2 0$" and "$1/log_2 1$" when $No.r$ equals to 0 and 1, respectively, '$1/log_2(No.r + 2)$" instead of "$1/log_2(No.r)$" is adopted.

4.3 Experiments and Analysis

In this section, the performance of ICAA is experimentally tested. The experimental benchmarks are introduced in Subsection 4.3.1, and the experimental criteria are given in Subsection 4.3.2. The experimental settings are explained in Subsection 4.3.3, and the experimental results are presented and analysed in Subsection 4.3.4.

4.3.1 Experimental Benchmarks

Manisterski et al. have proved that in general, no approach that can achieve an optimal solution of task allocation, exists if the agents are self-interested [MDKJ06], and if such approaches could exist, they would inevitably be setting-specific. Due to the absence of optimal approaches, two well-known approaches recently proposed are chosen as benchmarks. One is the multi-resource negotiation-based (MRN) task allocation approach proposed by An et al. [ALS11], and the other one is the Combinatorial Auction-Linear Programming-based approach (CA-LP) proposed by Zaman and Grosu [ZG13].

In MRN, a consumer obtains its required resources through negotiating with providers separately for each of the required resources. Only when all the negotiation threads for all the required resources succeed, can the allocation of the task be considered as successful. CA-LP is a combinatorial auction-based approach and a consumer acquires resources through placing sealed bids for resources to resource providers. After collecting bids, a provider determines the winning consumers by linear programming. In order to increase the chance of obtaining all of the required resources, a consumer is allowed to bid more than once, but is limited to bidding for a partial set of the required resources from only one provider each time. A consumer has two ways to increase the chance of winning each single bid. One is decreasing the bided resources, and the other is increasing the quote for the bided resources. A consumer can benefit from a single bid only when all required resources in the bid are obtained. Therefore, each single bid in CA-LP is all-or-nothing, and thus considered single-minded and not flexible enough [ZG13]. In addition, if the follow-up bids fail, the already successful bids will become useless, and thus this is another disadvantage of CA-LP. In addition, the decision on the winner in CA-LP only takes the bidding price into account, while ignoring other factors (e.g., the task deadline, the dynamism of the environment). This limits the robustness of CA-LP against the time constraints of task allocation and the dynamism of the environments.

4.3.2 Experimental Criteria

One of the main purposes of task allocation approaches in network environments is to successfully allocate as many tasks as possible [8]. In addition, the utility of the agent involved is also an important criterion [2]. Because the agents in ICAA are self-interested, it is the individual utility which agents care about. There are too many

agents and it is impossible to show the utility of every agent. Therefore, the utility distributions of agents is reported. Task allocation speed is also important when there are time constraints. For these reasons, three criteria are experimentally tested: **1)** the success rate of task allocation, **2)** the total time used for task allocation, and **3)** the utility distribution of agents involved in task allocation (the utility of an agent will be formulated next).

Formulation of Utility

(1) The utility gained by Agent a_i from Task τ_k is formulated by:

$$uti_{ik} = (rew_{ik} - cost_{ik})/rew_{ik}, \tag{4.10}$$

where rew_{ik} is the reward that a_i can gain when τ_k is finished, and $cost_{ik}$ is the cost of a_i to finish τ_k. rew_{ik} and $cost_{ik}$ are specified in two cases:

(i) if a_i is a consumer (i.e., the owner of τ_k), $cost_{ik}$ is the price a_i is charged by the provider group which finished τ_k.

(ii) if a_i is a provider in the provider group which finished τ_k, then:

$$cost_{ik} = costres_{ik} + pel_{ik}, \tag{4.11}$$

where $costres_{ik}$ is the cost of a_i's resources to finish τ_k, and pel_{ik} is the penalty (which will be formulated in Subsection 4.3.2) that a_i is charged by other consumers from which a_i has decommitted in order to execute τ_k (if decommitments happened).

(2) The utility that provider group gro gains from τ_k is:

$$uti_{grok} = \Sigma_{i=1}^{n} uti_{ik}, \tag{4.12}$$

where n is the number of providers in gro, and uti_{ik} is the utility gained by Provider a_i from τ_k.

Formulations of Penalty

A provider's decommitment from a contract will incur at least one victim agent that encounters the risk of loss of reward. For example, when Provider a_j decommits from a contract with Consumer a_i, a_i risks not being able to find another provider to replace a_j before the latest start time of its task. In addition, the providers in the same group with a_j (if there are any) encounter the risk of gaining no reward due to the failure of the task caused by the decommitment of a_j. a_i and the other providers in the same provider group are considered to be victim agents. In order to mitigate decommitment, penalties are charged to providers that initiate the decommitments in

ICAA.

Provided that the contract between Consumer a_i and provider group gro is con $(a_i, gro, \tau_k, rew, t_{fin}, pri_1, pri_2, ..., pri_n)$ (see **Definition 4.3** for the definition of contract), a_p and a_q are two of the providers in gro ($1 \leq p \leq n, 1 \leq q \leq n$, and $p \neq q$), if a_p decommits from con, then

(1) The penalty that a_p has to pay to a_i is formulated by:

$$pel_{pi} = \beta \times rew \tag{4.13}$$

(2) The penalty that a_p has to pay to a_q is formulated as:

$$pel_{pq} = \beta \times pri_q \tag{4.14}$$

where $0<\beta<1$.

In order to reflect the time-sensitivity characteristic of tasks, β is formulated as:

$$\beta = (t - t_{kgen})/(t_{kl} - t_{kgen}), \tag{4.15}$$

where t is the time when the decommitment happens, t_{kgen} and t_{kl} are the generation time and latest start execution time of τ_k, respectively.

Intuitively, Equations (4.13), (4.14) and (4.15) represent the fact that the later a provider decommits, the more penalties it must pay to the corresponding victim agents.

4.3.3 Experimental Settings

The most reliable way to evaluate task allocation approaches would be to perform real experimentation. However, three reasons prevent this approach from doing so. First, the research is still in the theoretical phase, focusing on theoretical research and analysis. The proposed approach needs to be improved before being used in real applications. Second, because the task allocation addressed in this chapter is closely related to economy, before the research is proven to be totally mature, it is hard to persuade self-interested real resource providers and consumers to participate in the experiments. Finally, due to the dynamism and openness of the network environment, the experimental results are not repeatable [DG05]. As a consequence, simulations are used to evaluate ICAA using C++. It should be noted that the test bed is not any off-the-shelf product, rather, it is developed by the author.

Because ICAA is devised to work in dynamic and open environments, such an

environment is necessary in the simulation. In addition, each agent is limited to communicating with its neighbours to obtain a local view. In order to level the playing field, it is necessary to study whether it is fair to test the two benchmarks (i.e., MRN and CA-LP introduced in Section 4.3.1) in such an environment. From [ALS11] and [ZG13], both MRN and CA-LP were devised for dynamic environments as well. Additionally, these two approaches do not limit the way the consumer obtains information about the environment. Consequently, the simulation environment designed for ICAA suits both MRN and CA-LP. For this reason, the three approaches are tested in such an environment.

In detail, in the simulation, a controller is employed to generate 100 agents and 300 tasks, which are inactive in advance of running the experiment. After starting to run the experiment, some or all of the 100 agents and 300 tasks will be activated according to specific requirements. Therefore, the sample sizes of agents and tasks are not fixed but decided by specific requirements, and the biggest sample sizes of agents and tasks are 100 and 300, respectively. An agent constructs a neighbourhood with each of the other agents at a predefined probability denoted by P_{con} ($P_{con} \in (0, 1]$). As a consequence, the network could be formed, where each agent has $100 \times P_{con}$ neighbours averagely. In the simulation, P_{con} is set to 0.1 in that it is normal for an agent to have about 10 neighbours. A boolean variable is defined for each agent to represent the agent's states, which include both active and inactive that are represented by 1 and 0, respectively. When the boolean variable is 1, it represents that the agent is in the environment, not in the environment otherwise. In addition, an agent's entering and leaving the environment can be simulated through changing the boolean variable. In detail, when the boolean variable of an agent becomes 0 from 1, it represents that the agent leaves the environment, and it represents the agents enters the environment if the boolean variable becomes 1 from 0. The number of agents in the environment could be controlled through adjusting the probabilities of activating and deactivating agents in each time unit. For example, in order to meet the requirement that 0 to 100 agents are in the environment, 50 agents are set inactive and the other 50 ones are set active in advance. After the start of experiment, the probabilities of each inactive agent's being activated and active agent's being deactivated are both set 0.5. Consequently, the total number of agents in the environment remains 0 to 100, concentrating on $50 - 50 \times 0.5 + 50 \times 0.5 = 50$. Like agent, each task is assigned to a boolean variable as well. When the boolean variable of a task is 1, it represents that the task is in the environment, and not in the environment otherwise. Each task in the environment

will be assigned to a randomly chosen active agent, which consequently becomes the consumer (owner) of the task and is responsible for allocating the task. A consumer cannot leave the environment before successfully allocating its task. With indicators, both the numbers of agents and tasks in the environment could be controlled, and thus the resource competition level (defined in Equation (4.17)) needs to be considered. In addition, the dynamism and openness of the environment could be simulated through activating and deactivating agents and tasks.

The required resource types of a task are complementary, and this means that a task can be accomplished only if all of its required resource types are obtained. Therefore, the number of the required resource types per task plays an important role in the performance of ICAA. In addition, allocating as many tasks as possible under time constraints is one of the objectives of ICAA, and thus, the task deadline is also one of the key factors in the performance of ICAA. Additionally, in ICAA the environment is competitive and the agents are self-interested, therefore the level of resource competition is important to the performance of ICAA as well. Moreover, scalability is also important to a task allocation approach. For the above reasons, the experiment is conducted based on four scenarios:

Scenario 1: Examination of the impact of the average number of required resource types per task

Scenario 2: Examination of the impact of the latest start execution time of tasks

Scenario 3: Examination of the impact of the degree of resource competition

Scenario 4: Examination of scalability

4.3.3.1 *Scenario 1: Examination of the impact of the number of required resource types per task*

In Scenario 1, the impact of the number of required resource types per task on the three criteria, which are introduced at the beginning of Section 4.3.2, is tested. The parameter settings for Scenario 1 are listed in Table 4.2.

Table 4.2: Parameter Settings for Scenario 1

Variables	meanings	values
N_{pro}	The number of provided resource types per provider	[1, 10]
N_t	The number of tasks in the environment	[0, 300]
N_a	The number of agents in the environment	[0, 100]
$flex(t)$	The allocation flexibility, i.e., the available time to allocate a task,	[20, 50] (s)
$\psi(r)$	Resource competition	N_t/N_a

In Table 4.2, the number of resource types provided by a provider (denoted by N_{pro}) is between 1 and 10. The reason for choosing this is that in real world applications, it is common and reasonable for a provider to provide from 1 to 10 types of resources. Because N_{pro} is discrete, a Random Number Generator (RNG) was employed to generate values in [1, 10] for N_{pro}. Therefore, N_{pro} obeys uniform distribution over [1, 10]. 50 of the 100 agents are set active and the other 50 ones are set inactive before the start of experiment, after the start of experiment, and both the probability of each active agent being deactivated and that of each inactive agent being activated are set 0.5. As a consequence, the number of agents in the environment keeps in [0, 100], concentrating on $50 - 50 \times 0.5 + 50 \times 0.5 = 50$. Similarly, 150 of the 300 tasks are set active, and the other 150 are set inactive. After the start of experiment, each active task has a probability of 0.5 to be deactivated, and each inactive task has a probability of 0.5 to be activated. Consequently, the number of tasks in the environment keeps from 0 to 300, concentrating on 150.

The *allocation flexibility*, denoted by $flex$, represents the available time to allocate a task before the task's latest start execution time. If t_{kl} is the latest start execution time of task τ_k, and t is the current time, then the allocation flexibility of τ_k is formulated by:

$$flex(\tau_k) = t_{kl} - t \tag{4.16}$$

Because this scenario is to test the performance with various N_{req}, in order to avoid the strong impact of $flex(t)$ on the experimental results, a modest range [20, 50] (s) is chosen for $flex(t)$. In order to obtain a group of continuous values from 20 to 50 for $flex$, a normal distribution generator $flex(t) \sim (\mu, \sigma^2)$ is employed. The mean number and standard derivation of the normal distribution are set 35 and 5, respectively (i.e., $\mu = 35, \sigma = 5$), and the reason for this is explained as follows. In a normal distribution, $P\{\mu - 3\times\sigma \leq x \leq \mu + 3\times\sigma\} \approx 0.997$ ($P\{m \leq x \leq n\}$ is the probability of x falling into the range of [m, n]. If it is set that $\mu - 3 \times \sigma = 20$, $\mu + 3 \times \sigma = 50$ (i.e., $\mu = 35, \sigma = 5$),

it can guarantee that around 99.7% of the generated values fall into the interval of [20, 50].

$\psi(r)$ is used to denote the resource competition in the environment, and is defined as:

$$\psi(r) = N_t/N_a, \tag{4.17}$$

where N_t and N_a are the numbers of tasks and agents in the network environment, respectively.

From Table 4.2, it can be seen that in order to reflect the dynamism and openness of the network environment, a specific value is not set for any parameter. Instead, each parameter is set a range from which the parameter can take different values randomly at different time points during the course of the experiment.

1, 2, 4, 6, 8 and 10 are assigned to N_{req}, separately, to test the performance. The reason to choose these six values for N_{req} is that in real world applications, it is reasonable for a task to require such numbers of resource types.

4.3.3.2 *Scenario 2: Examination of the impact of allocation flexibility*

In Scenario 2, the impact of allocation flexibility, which is formulated by Equation (4.16) on the three criteria, is tested. The parameter settings for Scenario 2 are listed in Table 4.3.

Table 4.3: Parameter Settings for Scenario 2

Variables	meanings	values
N_{req}	the number of resource types required by each task	[1, 10]
N_{pro}	the number of resource types provided by each provider	[1, 10]
N_t	the number of tasks in the environment	[0, 300]
N_a	the number of agents in the environment	[0, 100]
$\psi(r)$	resource competition	N_t/N_a

The same as Scenario 1, a range of values are assigned to each parameter in this scenario, as shown in Table 4.3. It is common and reasonable for a task to require from 1 to 10 resource types in real world applications, therefore, N_{req} varies over the interval of [1, 10] in the evaluation. A Random Number Generator (RNG) was employed to generate values for both N_{pro} and N_{req}. The generation is the same as that in Scenario 1. Six normal distribution generators were employed to obtain six groups of values for $flex(t)$. The six value ranges are $[1, 10)$, $[10, 20)$, ..., and $[50, 60)$. The

corresponding normal distribution parameters for these groups are $flex(t) \sim (5, 1.67^2)$, $flex(t) \sim (15, 1.67^2)$, $flex(t) \sim (25, 1.67^2)$, $flex(t) \sim (35, 1.67^2)$, $flex(t) \sim (45, 1.67^2)$, and $flex(t) \sim (55, 1.67^2)$, respectively.

4.3.3.3 *Scenario 3: Examination of the impact of resource competition*

In Scenario 3, the impact of resource competition on the three criteria is tested. The parameter settings for Scenario 3 are listed in Table 4.4.

Table 4.4: Parameter Settings for Scenario 3

Variables	meanings	values
N_{req}	the number of resource types required by each task	[1, 10]
N_{pro}	the number of resource types provided by each provider	[1, 10]
$flex(t)$	the allocation flexibilities of tasks	[20, 50] (s)

The same as Scenario 2, a RNG is used to generate values from 1 to 10 for both N_{pro} and N_{req}, and a normal distribution generator, $flex(t) \sim (35, 5^2)$, is adopted to generate values for $flex(t)$ whose range is consequently [20, 50]. The reason to assign [20, 50] to $flex(t)$ is the same as that of Scenario 1. In this scenario, the impact of $\psi(r)$ is tested, and the various values of $\psi(r)$ are obtained through changing both N_a and N_t. All the agents and tasks are set inactive before the start of the experiment. After the start of experiment, agents and tasks are activated at various probabilities in each time unit to obtain various resource competition levels. For example, if agents and tasks are activated at probabilities of 0.75 and 0.25, respectively, the numbers of active agents and tasks in the environment remain around $100 \times 0.75 = 75$ and 300×0.25=75, respectively, and the consequent $\psi(r)$ remains around 1. The probabilities to activate agents and tasks are denoted as P_a and P_t, respectively. The consequent values of $\psi(r)$ are listed in Table 4.5.

Table 4.5: $\psi(r)$ Based on Various P_a and P_t

$\psi(r)$	P_a	P_t
$\frac{0.05\times300}{0.75\times100} = 0.2$	0.75	0.05
$\frac{0.1\times300}{0.75\times100} = 0.4$	0.75	0.1
$\frac{0.15\times300}{0.75\times100} = 0.6$	0.75	0.15
$\frac{0.2\times300}{0.75\times100} = 0.8$	0.75	0.2
$\frac{0.25\times300}{0.75\times100} = 1$	0.75	0.25
$\frac{0.5\times300}{0.75\times100} = 2$	0.75	0.5
$\frac{1\times300}{0.75\times100} = 4$	0.75	1

It should be noted that the values of both P_a and P_t in Table 4.5 are not the only choices, and other values can also work as long as they can meet the requirement of resource competition levels.

4.3.3.4 *Scenario 4: Examination of scalability*

In this scenario, the scalability of ICAA is tested through testing the performances with various numbers of involved agents and tasks. The parameter settings for this scenario are listed in Table 4.6.

Table 4.6: Parameter Settings for Scenario 4

Variables	meanings	values
N_{req}	the number of resource types required by each task	[1, 10]
N_{pro}	the number of resource types provided by each provider	[1, 10]
$flex(t)$	the allocation flexibilities of tasks	[20, 50] (s)
$\psi(r)$	resource competition	2

In Table 4.6, both N_{pro} and N_{req} vary from 1 to 10, and $flex(t)$ is in the interval of [20, 50]. The reasons for such parameter settings are the same as those of Scenario 3. The reason to set $\psi(r)$ to 2 is that 2 is a modest value and thus will not bias the examination of scalability. The same as Scenario 3, all the agents and tasks are set inactive before the start of the experiment, and are activated at the respective probabilities of P_a and P_t after the start of experiment. The settings of P_a, P_t and the consequent N_a, N_t and $\psi(r)$ are listed in Table 4.7.

Table 4.7: N_a and N_t Based on Various P_a and P_t

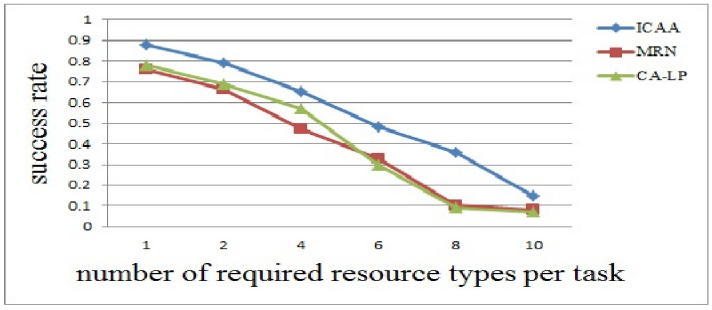
N_a	N_t	$\psi(r)$	P_a	P_t
20	40	$\frac{0.133\times300}{0.20\times100}=2$	0.2	0.133
40	80	$\frac{0.267\times300}{0.40\times100}=2$	0.4	0.267
60	120	$\frac{0.4\times300}{0.60\times100}=2$	0.6	0.4
80	160	$\frac{0.533\times300}{0.80\times100}=2$	0.8	0.533
100	200	$\frac{0.667\times300}{1.0\times100}=2$	1.0	0.667

It is notable that similar to Table 4.5 in Scenario 3, the values of both P_a and P_t in Table 4.7 are not the only choices, other values also work as long as they can meet the requirements of both N_a and N_t.

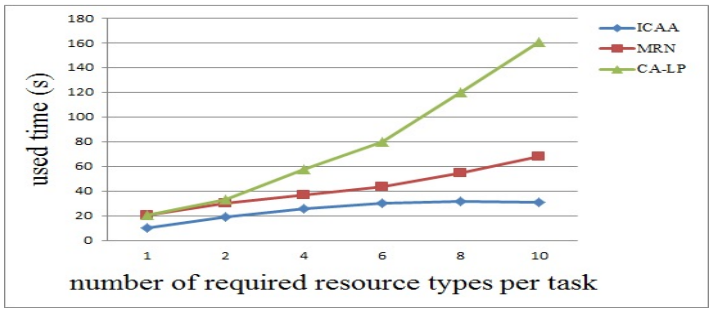
4.3.4 Experimental Results and Analysis

4.3.4.1 *Experimental Results and Analysis of Scenario 1*

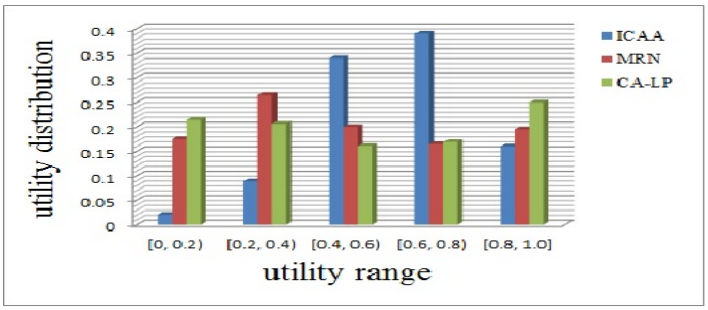
The experimental results of Scenario 1 are shown in Figure 4.3.



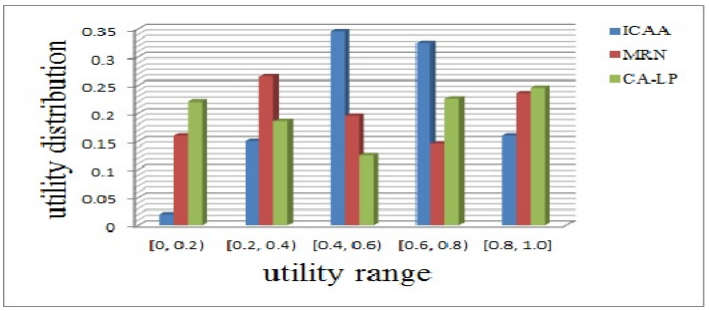
(a) Success Rates based on Various Numbers of Required Resource Types per Task



(b) Time Used based on Various Numbers of Required Resource Types per Task



(c) Utility Distribution When $N_{req} = 4$



(d) Utility Distribution When $N_{req} = 6$

Figure 4.3: Performance based on Various Numbers of Required Resource Types per Task

Impact on success rate of task allocation

Figure 4.3 (a) shows the impact of N_{req} on the success rate of task allocation. From Figure 4.3 (a), it can be seen that when N_{req} varies over the range of [1, 10], ICAA always achieves higher success rates than both MRN and CA-LP. The reason is that the consumer in MRN negotiates with providers for each of the required resource types separately. The separate negotiations always make the consumers only obtain a partial set of the required resources due to the failure of even only one negotiation thread. In addition, the consumer in MRN does not take into account the potential decommitment probabilities of providers when negotiating with the providers. This can result in a high decommitment probability of the commited providers, which results in a high probability of task failure. The all-or-nothing bid of CA-LP makes it hard for the bidders to get all of the required resource types, especially when the resources are scarce. Moreover, the auctioneer in CA-LP does not consider task deadlines when determining the winning bidders, and this further hinders the success of task allocation in CA-LP.

It can also be observed from Figure 4.3 (a) that when N_{req} is lower than 6, the success rate of MRN is lower than that of CA-LP. This is because when N_{req} is quite small, the disadvantage of bidding a bundle of resources from only one provider each time is not very obvious. With the increase of N_{req}, it gets harder for the consumer in CA-LP to successfully obtain the required resources from only one provider. However, even though the decommitment probability of separate negotiation in MRN is high, the consumer still has the chance to find a replacement for the decommited provider when decommitment happens. As a consequence, when N_{req} is lower than 6, the success rate of task allocation of MRN is lower than that of CA-LP, and the result is the opposite when N_{req} is higher than 6.

Impact on time usage of task allocation

Figure 4.3 (b) shows the time used for task allocation based on different values of N_{req}. It can be seen from Figure 4.3 (b) that when N_{req} varies from 1 to 10, the time used for task allocation in ICAA is shorter than those in both MRN and CA-LP. This is because the single-mind of the combinatorial auction of CA-LP makes it take long to successfully obtain all of the required resource types. In MRN, the consumer can negotiate with resource providers for all the required resources simultaneously, and this can save time to obtain all of the required resource types. The consumer in ICAA adopts the combinatorial auction as the basis, but is not limited to bidding from only one provider. Therefore, ICAA has both the advantages of combinatorial auction and

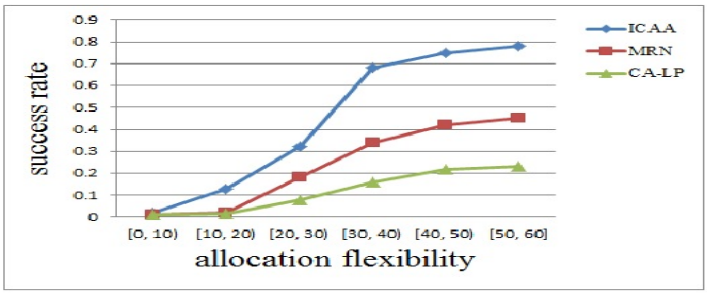
flexibility of MRN. Due to these reasons, the time used in ICAA is shorter than those in both MRN and CA-LP.

Impact on utility distribution of task allocation

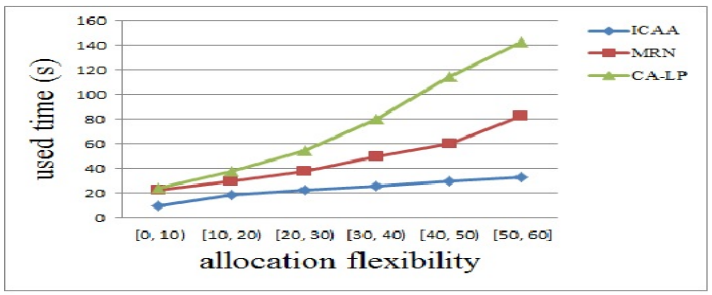
Figure 4.3 (c) and Figure 4.3 (d) present the respective utility distributions when N_{req} are 4 and 6. Before analyzing the results, it is notable that an even utility distribution is important to attract agents to participate in the task allocation. From Figure 4.3 (c), it can be seen that compared with both MRN and CA-LP, more individual utilities of agents in ICAA fall into the middle-ranges (e.g., [0.4, 0.6) and [0.6, 0.8)). This means that the utility distribution of ICAA is more even than those of both MRN and CA-LP. This is because in CA-LP, a consumer is single-minded, which means that the consumer does not accept any resources if it cannot obtain all of its required resources from the provider. Consequently, the all-or-nothing bidding strategy of the consumer in CA-LP results in more utilities falling into both the high-ranges and low-ranges. It is allowed that only a partial set of the resource types of a bidder are chosen in ICAA. Thus, there are less high-range and low-range utilities in ICAA, compared with CA-LP. In addition, besides the bidding price, the consumer in ICAA takes into account the possible decommitment probability of the bidders as well, while the consumer in MRN does not. Moreover, the separate negotiation of MRN can result in high decommitment probability. Consequently, there are more very low-range utilities in MRN, compared with ICAA. For the same reason, both MRN and CA-LP are more sensitive to the values of N_{req} than ICAA, and this can be observed from Figure 4.3 (c) and Figure 4.3 (d), that is, the utility distribution of ICAA does not change as much as those of both MRN and CA-LP, when N_{req} becomes 6 from 4.

4.3.4.2 *Experimental Results and Analysis of Scenario 2*

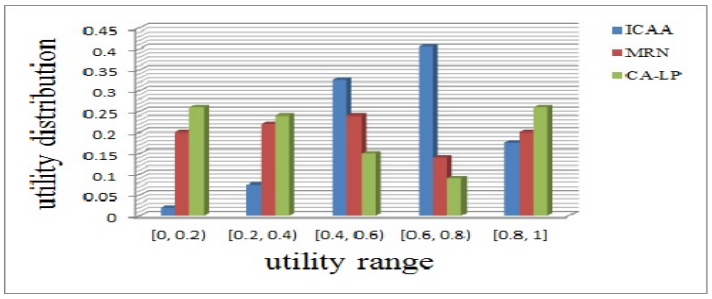
The experimental results of Scenario 2 are demonstrated in Figure 4.4.



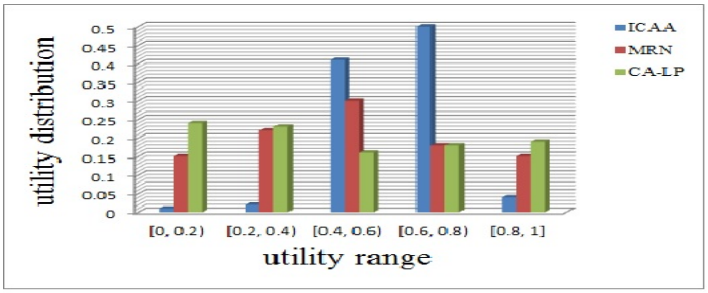
(a) Success Rates based on Various Allocation Flexibilities of Tasks



(b) Time Used based on Various Allocation Flexibilities of Tasks



(c) Utility Distribution when *flex(t)* varies from $30s$ to $40s$



(d) Utility Distribution when *flex (t)* varies from $40s$ to $50s$

Figure 4.4: Performance based on Various Allocation Flexibilities of Tasks

Impact on success rate of task allocation

Figure 4.4 (a) shows the success rate of task allocation based on different allocation flexibilities ($flex(t)$). As can be seen from Figure 4.4 (a), when $flex(t)$ varies from the range of [0, 10] to the range of [50, 60], the success rates of all of ICAA, MRN and CA-LP increase due to the time constraints of task allocation. In addition, the success rate of ICAA is always higher than those of both MRN and CA-LP. This can be explained by the time complexities of the task allocation algorithms which have been analysed in Section 4.2.3. The time complexity of the linear programming-based winner decision of CA-LP is $\mathcal{O}(m \times 2^n)$ where m and n are the number of the required resource types per task and the number of involved providers, respectively. The time complexity of the group formation of ICAA is $\mathcal{O}(|P_r| \times m^2 \times n)$. Compared with the separate negotiations in MRN, the biddings for bundles of resources in ICAA can save the time taken by task allocation. For these reasons, the task allocation in ICAA is faster than those in both CA-LP and MRN.

Impact on time usage of task allocation

Figure 4.4 (b) illustrates the time taken by the task allocation based on different allocation flexibilities. From Figure 4.4 (b), the time taken by task allocation in ICAA is shorter than that in both CA-LP and MRN. Such a comparison between ICAA and CA-LP can be explained from the perspective of time complexity of task allocation algorithms which has been analysed in Figure 4.4 (a). It is hard to analyse the specific time complexity of MRN which depends on many factors such as the negotiation strategy and the negotiation deadlines. Consequently, the comparison between MRN and ICAA is qualitatively analysed. In MRN, the consumer negotiates with providers for each of the required resources separately. Moreover, each of the negotiations comprises at least one round of bargaining. Therefore, compared with the one-shot combinatorial auction in ICAA where bidders bids only once, the time taken in MRN is longer .

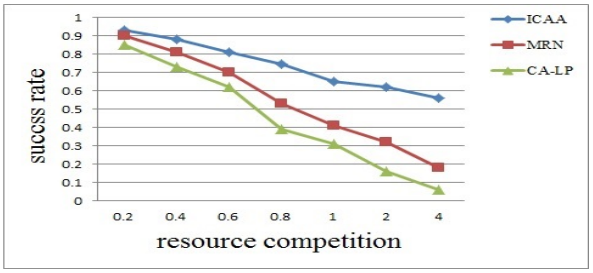
Impact on utility distribution of task allocation

Figure 4.4 (c) and Figure 4.4 (d) present the utility distributions when $flex(t)$ varies over the ranges of [30, 40) and [40, 50), respectively. From Figure 4.4 (c) and Figure 4.4 (d), it can be seen that compared with both MRN and CA-LP, more individual utilities in ICAA fall into the middle-ranges (e.g., [0.4, 0.6) and [0.6, 0.8)). The reason for this is the same as that in Figure 4.3 (c) and Figure 4.3 (d), which has been analysed in Section 4.3.4. It can also be seen that when $flex(t)$ varies over the range of [40, 50), more individual utilities of agents in all of the three approaches fall into the middle ranges, than that when $flex(t)$ varies over the range of [30, 40). This is

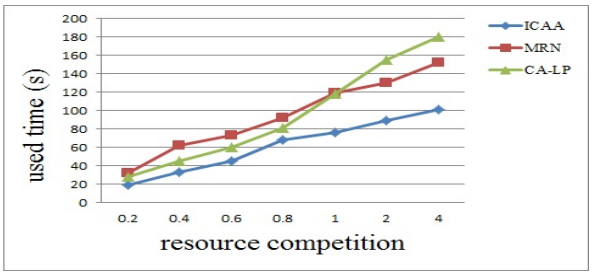
because when $flex(t)$ increases, the consumers in MRN have longer time to negotiate, and those in both ICAA and MRN have longer time to look for replacements for the decommitted providers when decommitments happen. As a consequence, a large $flex(t)$ can decrease the failure probability of task allocation and thus reduces the low-range utilities.

4.3.4.3 *Experimental Results and Analysis of Scenario 3*

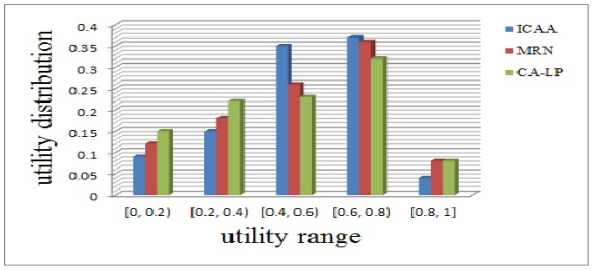
The experimental results of Scenario 3 are shown in Figure 4.5.



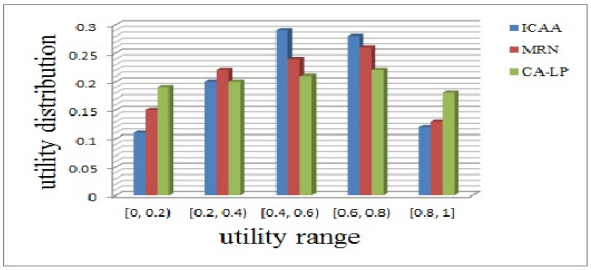
(a) Success Rate based on Various Degrees of Resource Competition



(b) Time Used based on Various Degrees of Resource Competition



(c) Utility Distribution when $\psi(r) = 0.6$



(d) Utility Distribution when $\psi(r) = 2$

Figure 4.5: Performance based on Various Levels of Resource Competition

Impact on success rate of task allocation

From Figure 4.5 (a), when the resource competition ($\psi(r)$) varies over the range of [0.2, 4], ICAA always achieves higher success rates of task allocation than both MRN and CA-LP. In addition, the success rate of ICAA becomes much higher than those of both MRN and CA-LP with the increase of $\psi(r)$. This is because the consumer in CA-LP places all-or-nothing bids for combinations of resources. This is a disadvantage in that it is hard for the consumer to look for the provider which possesses all of the required resource types. In contrast, in ICAA, it is allowed that a partial set of the resource types in a bid are selected by the consumer. The separate negotiation in MRN takes a long time to obtain all of the required resources. This is a disadvantage of MRN due to the time constraints of task allocation. When $\psi(r)$ is low, which means that there are sufficient resources in environment, the above mentioned disadvantages of CA-LP and MRN are not obvious. With the increase of $\psi(r)$, however, such disadvantages become stronger, and consequently hinder the success rate of task allocation more. In addition, providers in ICAA can adjust their bidding prices for a resource according to the competition level (or popularity) of the resource, which can be seen from Equation (4.3). Such a flexible bidding strategy can prevent the success rate of task allocation from increasing/decreasing too sharply when $\psi(r)$ changes. The above are the reasons why the success rates of both MRN and CA-LP become much lower than that of ICAA with the increase of $\psi(r)$.

Impact on time usage of task allocation

As can be seen from Figure 4.5 (b), the time taken by task allocation in ICAA is always shorter than those in both CA-LP and MRN. The reason for this is the same as that in Figure 4.5 (b) which has been analysed in Subsection 4.3.4.

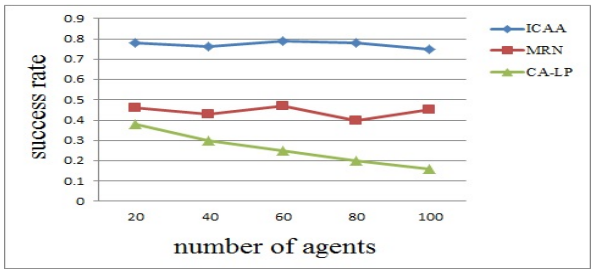
Impact on utility distribution of task allocation

It can be observed from Figure 4.5 (c) and Figure 4.5 (d) that compared with both MRN and CA-LP, more utilities in ICAA fall into the middle ranges [0.4, 0.6) and [0.6, 0.8). The reason for this is the same as that in Figure 4.3 (c) and Figure 4.4 (d). In addition, it is can be seen that when $\psi(r)$ varies from 0.6 to 2, the utility distributions of all of the three approaches become less even, and such a changing is stronger for both CA-LP and MRN than ICAA. The reason for this is that it gets harder for consumers to obtain all the required resource types when resource competition is higher. Moreover, the disadvantages of both CA-LP and MRN which are shown in Figure 4.5 (a) become stronger with the increase of $\psi(r)$. This further prevents the utility distributions from being even. In contrast, the bidding strategy of providers in ICAA (Equation (4.3))

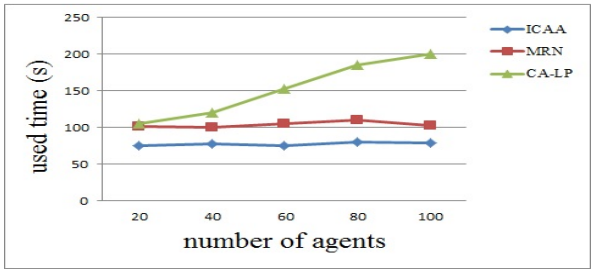
can inhibit the changing of utility distributions when $\psi(r)$ increases.

4.3.4.4 *Experimental Results and Analysis of Scenario 4*

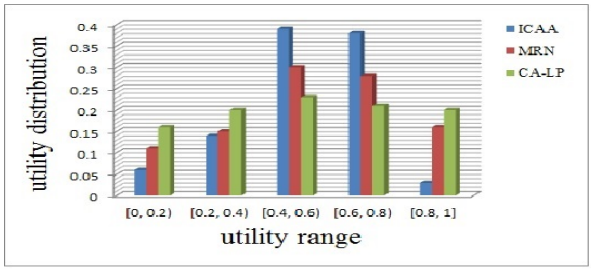
The experimental results of Scenario 4 is shown in Figure 4.6.



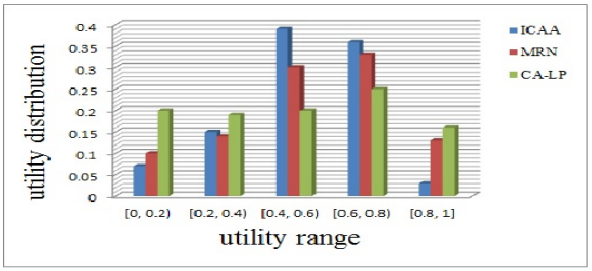
(a) Success Rates based on Various Numbers of Agents



(b) Used Time based on Various Numbers of Agents



(c) Utility Distribution when N_a=60 and N_t=120



(d) Utility Distribution when N_a=100 and N_t=200

Figure 4.6: Performance based on Various Scales of both Tasks and Agents

As can be seen from Figure 4.6, ICAA always achieves better performances in terms of success rate, time taken, and utility distribution than both MRN and CA-LP. The reasons for this are the same as those in Scenario 3. In this scenario, the change in performances when the scales of both agents and tasks vary is concentrated. From Figure 4.6, it can be seen that compared with the performances of both ICAA and MRN, which remain steady, the performance of CA-LP decreases more with the increase of scales. The reason is that in CA-LP, resource providers adopt linear programming to decide the winning bidder (consumer). When there are too many tasks and consumers involved, the linear programming will computationally hinder the performance of task allocation. In ICAA and MRN, however, task allocation is distributed among all the consumers. Consequently, the performances of these two approaches will not be affected as much by the scales of agents and tasks as that of CA-LP.

4.4 Discussion

By adopting combinatorial reverse auction, the providers are allowed to bid for a bundle of the subtasks that they are interested in. Due to this, compared with the multi-resource negotiation where a consumer negotiates with providers for all the required resources separately, the combinatorial reverse auction can decrease the risk of the consumer of obtaining the invalid bundles of resources. In addition, the devised indicator allows the consumer to select the most suitable provider combination according to its preferences which are encoded into the indicator. Compared with the linear programming, which can only ameliorate the NP-hard problem in combinatorial auction, the NP-hard problem can be avoided through the usage of the indicator.

4.5 Summary

This chapter proposed an indicator-based combinatorial auction-based approach for group task allocation in open and dynamic grid network environments. The proposed approach addressed the challenges of decentralization, the dynamism, and the openness of the grid network environments, through devising an indicator to help consumers make the decision as to which provider can be chosen. The experimental results demonstrated that the proposed approach outperformed two well-known approaches for

group task allocation in terms of success rate of task allocation, individual utility distribution of involved agents, the speed of task allocation, and scalability, so as to achieve **Objective 2** of this thesis.

Chapter 5

Two Max-sum Belief Propagation-based Task Allocation Methods

This chapter focuses on the group task allocation with dependency constraints of subtasks in which a task consists of a group of interdependent subtasks with dependency constraints. In recent years, group task allocation with dependency constraints of subtasks in dynamic and competitive environments has been studied much because it can be motivated by various contexts, such as electronic commerce [1], and supply chain formation [KC10, WWY00, PA13, PAVCRA12].

Two max-sum belief propagation-based task allocation methods with different goals for group task allocation with dependency constraints of subtasks are proposed in this chapter. This chapter is organised as follows. Problem description is given in Section 5.1. Section 5.2 introduces the method, which studies the group task allocation with constraints from the perspective of strategic quoting of resource providers, and Section 5.3 describes the method, which studies the group task allocation with constraints from the perspective of computational simplification of belief propagation.

5.1 Problem Description

Assume that the task to be allocated is $\mathcal{T} = \{t_1, t_2, ..., t_m\}$ ($t_1, t_2, ..., t_m$ are the subtasks of $\mathcal{T}$), only when all the subtasks are successfully allocated, can the allocation of $\mathcal{T}$ be considered to succeed. The provider agents, each of which can execute at least one subtask of $\mathcal{T}$, are called alternatives, and there may be multiple alternatives for each subtask. Task allocation here is to select a provider for each of the subtasks from the corresponding alternatives to make them collaboratively finish the task, aiming at maximizing some pre-defined objective function. In other words, the solution of the problem in this chapter is a configuration of providers that can optimise the

[1] www.ebay.com

task allocation according to some predefined criterion. It is notable that maybe some alternatives cannot collaborate with each other due to some reasons (e.g., geography or traffic reasons) that are beyond the study of this thesis. Due to the dynamism and openness of the environment, the sets of both alternatives and tasks change constantly. An example represented in Figure 5.1 illustrates the task allocation problem in this chapter.

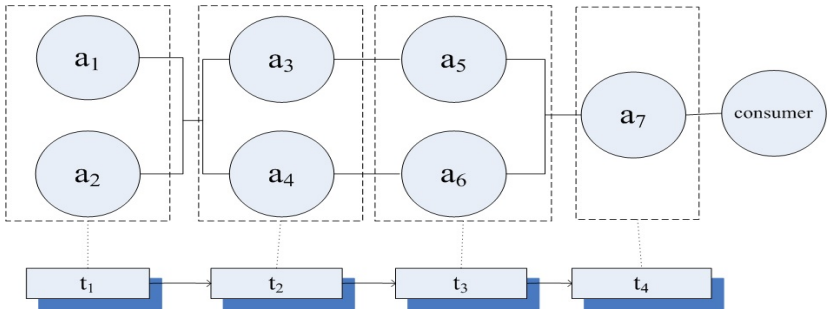


Figure 5.1: Subtasks and Alternatives

Figure 5.1 demonstrates a network of the resource providers and subtasks to be allocated. In Figure 5.1, the task to be allocated is $\mathcal{T} = \{t_1, t_2, t_3, t_4\}$, and the dependency constraint of the subtasks is $t_1 \rightarrow t_2 \rightarrow t_3 \rightarrow t_4$. The set of alternatives for all the subtasks is $\mathcal{A} = \{a_1, a_2, a_3, a_4, a_5, a_6, a_7\}$, and the respective alternative sets for t_1, t_2, t_3, and t_4 are $A_1 = \{a_1, a_2\}$, $A_2 = \{a_3, a_4\}$, $A_3 = \{a_5, a_6\}$, and $A_4 = \{a_7\}$. As shown in Figure 5.1, either agent a_1 or a_2 can execute subtask t_1. If a_1 is selected to execute t_1, it can pass over the task to either a_3 or a_4 to finish t_2 after finishing t_1. If a_3 is selected to execute t_2, it can pass over the task only to a_5 to finish t_3 after finishing t_2. Regardless of the constraints of both reserve price and deadline, the solutions (i.e., configurations) set is $\mathcal{S} = \{\{a_1, a_3, a_5, a_7\}, \{a_1, a_4, a_6, a_7\}, \{a_2, a_3, a_5, a_7\}, \{a_2, a_4, a_6, a_7\}\}$. The purpose of the task allocation is to find the configuration S^* to maximise some objective function in the dynamic environments where both the alternatives and tasks keep changing over time.

Since the type of tasks in this chapter is different from those in Chapters 3 and 4, in this chapter, tasks, agents, and messages between agents are redefined as follows.

Definition 5.1 (Task): A task τ_k is a 3-tuple: (ID_k, dl_k, Sub_k), where ID_k and dl_k are the identifier and deadline of τ_k, respectively, and $Sub_k = \{sub_1, sub_2, ..., sub_n\}$ is the set of subtasks of τ_k (n is the number of subtasks of τ_k). The subtasks of a task have dependency constraints, and this means that the subtasks have to be executed in a determined sequence. If two subtasks are adjacent in the execution sequence, it is

said that they have direct dependency.

Definition 5.2 (Agent): An agent a_i is defined as a 4-tuple: (ID_i, $Neighset_i$, $Taskset_i, Resset_i$), where ID_i is the identifier of a_i, $Neighset_i = \{nei_1, nei_2, ..., nei_n\}$ is the set of neighbouring agents of a_i (n is the number of neighbouring agents of a_i), $Taskset_i = \{task_1, task_2, ..., task_m\}$ is the task set of a_i (m is the number of tasks of a_i), and $Resset_i = \{res_1, res_2, ..., res_p\}$ is the resource set of a_i (p is the number of resource types of a_i). If $Taskset_i \neq \varnothing$, a_i is a consumer, and if $Resset_i \neq \varnothing$, a_i is a provider. a_i is both a provider and consumer when $Taskset_i \neq \varnothing$ and $Resset_i \neq \varnothing$.

Agents communicate with their neighbouring agents through sending messages which include both request messages and reply messages. The **request message** is defined the same as that of Chapter 3 (i.e., **Definition 3.6**). Due to the different tasks to be allocated in this chapter comparing with tasks in Chapter 3, the reply message in this chapter is different from that of Chapter 3 and is redefined as follows.

Definition 5.3 (Reply Message): The message to reply the request message (Req_{ij}, ID_i, ID_j, τ_k, HL), denoted as Rep_{ji}, is a 4-tuple: (Rep_{ji}, ID_j, ID_i, $Subset_k$), where ID_j and ID_i are the identifiers of the messages's sender and recipient, respectively, and $Subset_k \in Sub_k$ (see **Definition 5.1** for Sub_k) is the set of subtasks that a_j is capable of executing.

In the grid/cloud environments, each agent only has a local view of the environments. Consumers seek for resource providers through their neighbourhoods. Provided that agent a_i has a task τ_k ($\tau_k \in Taskset_i$) to allocate, a_i sends a request message ($Req_{ij}, ID_i, ID_j, \tau_k, HL$) to its neighbouring agent a_j to ask a_j to execute τ_k. After receiving the request message, a_j checks whether it is capable of executing any subtask of τ_k according to $Resset_j$. If so, a reply message ($Rep_{ji}, ID_j, ID_i, Subset_k$) is sent by a_j to a_i; if not and $HL \neq 0$, the request message is recomposed into ($Req_{jx}, ID_i, ID_j, ID_x, \tau_k, HL - 1$) and transmitted by a_j to a_j's neighbouring agent a_x. If a_x is capable of executing some subtask(s) of τ_k and $a_x \notin Neighset_i$, a_x constructs a neighbourhood with a_i according to the identifier of a_i in the request message ($Req_{jx}, ID_i, ID_j, ID_x, \tau_k,\ HL - 1$). This is the reason to keep the identifier of the initial sender (i.e., a_i) when the request message is recomposed. After constructing the neighbourhood with a_i, a reply message ($Rep_{xi}, ID_x, ID_i, Subset_k$) is sent by a_x to a_i. Every time when a_i receives a reply message, it floods the information in the reply message, i.e., which provider can execute which subtask(s), through the whole neighbourhood, and the information is stored by all the agents.

5.2 A Strategic Max-Sum Algorithm-based Method for Group Task Allocation with Constraints in Dynamic Networks

In this section, the group task allocation with constraints is studied from the perspective of the strategic quoting of providers (bidders). In particular, a strategic max-sum algorithm-based method in which the max-sum belief propagation runs on a Markov Random Field (MRF) is proposed. In the remainder of this Section, the strategic max-sum algorithm-based method is in detail described in Subsection 5.2.1, and the proposed method is experimentally evaluated in Subsection 5.2.2.

5.2.1 The Strategic Max-sum LBP for Task Allocation

5.2.1.1 Conversion of the task allocation problem to a Markov Random Field graph

Suppose that a consumer has a task τ_k to allocate, and the dependency constraint of the subtasks of τ_k is $sub_1 \rightarrow sub_2 \rightarrow sub_3 \rightarrow sub_4$. Figure 5.2 presents the real network environment, including which provider can execute which subtask, the cost of a provider to execute the related subtask, and the reward that the consumer can gain from the subtask.

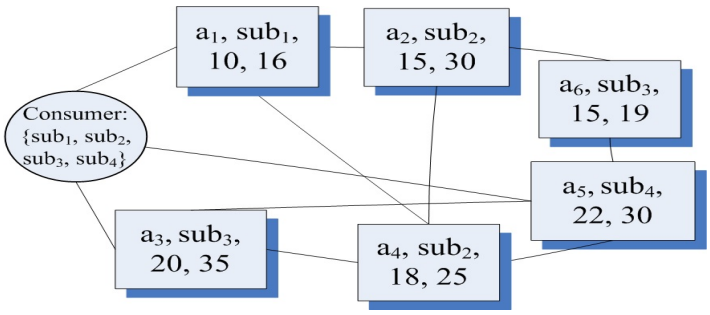


Figure 5.2: Network Environment

In Figure 5.2, a_1 is capable of executing subtask sub_1 at the cost of 10, and the reward that the consumer can gain from sub_1 is 16 when sub_1 is finished. In SLBP, in order to operate max-sum belief propagation, the task allocation problem is encoded into a MRF graph. A MRF model (i.e., undirected graphical model) is a set of random variables having a Markov property described by an undirected graph [KS+80]. In

brief, given an undirected graph G = (V, E), the set of variables can form a MRF with respect to G if they can satisfy the following Markov properties.

1: Pairwise Markov property: given all other variables, any two non-adjacent variables are conditionally independent.

2: Local Markov property: given its neighbours, a variable is conditionally independent of all other variables.

3: Global Markov property: given a separate subset, any two subsets of variables are conditionally independent.

Figure 5.2 is converted into a MRF graph (represented by Figure 5.3) through removing subtasks but retaining the undirected edges between providers.

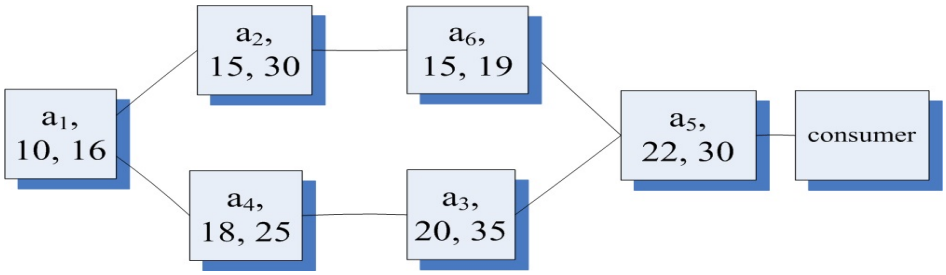


Figure 5.3: The MRF Graph

5.2.1.2 State and Utility

To operate belief propagation, states of providers and the utilities of states are introduced next, which are inspired by those in the Loopy Belief Propagation-based supply chain method (LBP) proposed by Winsper and Chli in [WC13].

State of Provider

In the belief propagation in MRF, there are two possible states of a provider: *active* and *inactive*. If the belief propagation result (which will be in detail introduced later) of a provider is *inactive*, it means that the provider fails in winning the subtask it aims at. The state of *active* of provider a_m further includes which providers a_m takes the task from and passes it over to, respectively.

For example, in Figure 5.3, the possible states of a_2 include *inactive* and taking over the task from a_1 then passing it over to a_6. a_1 and a_2 in Figure 5.3 are taken as

Table 5.1: States of a_1 and a_2

states of a_1	states of a_2
s_{11}: inactive s_{12}: passing the task over to a_2 s_{13}: passing the task over to a_4	s_{21}: inactive s_{22}: taking over the task from a_1 and passing it over to a_6

examples, and the possible states of them are listed in Table 5.1 where s_{mn} denotes the n^{th} state of agent a_m.

Utility of State

There are two types of utilities for a state of a provider, which are 1) a unary utility, and 2) a pairwise utility.

(1) Unary utility: The unary utility of state s_{mn}, denoted by $f(s_{mn})$, represents the reward that can be gained as a whole when a_m is designated the n^{th} state. It is stipulated that the unary utility of state *inactive* is 0. Otherwise, the unary utility is calculated by subtracting the cost of the provider to execute the related subtask from the reward gained by the consumer from the subtask.

a_1 and a_2 are taken as examples, and the unary utilities of all the possible states of them are listed in Table 5.2.

Table 5.2: Unary Utility

unary utilities of states of a_1;	unary utilities of states of a_2
$f(s_{11})$: 0 $f(s_{12})$: 16-10=6 $f(s_{13})$: 16-10=6	$f(s_{21})$: 0 $f(s_{22})$: 30-15=15

(2) Pairwise utility: The pairwise utility between two states which belong to two adjacent providers in MRF represents the compatibility of the two states. Two states are considered to be incompatible when they represent incompatible information.

For example, s_{13} and s_{22} represent incompatible information. The reason is that from Table 5.1, s_{13} represents that a_1 passes over the task to a_4, whereas s_{22} represents that a_2 takes over the task from a_1. According to the stipulation that a subtask can be executed by only one provider, s_{13} and s_{22} are incompatible. It is stipulated that if two states are compatible, the pairwise utility between them is 0, and $-\infty$ otherwise. The pairwise utilities between all the states of a_1 and a_2 are listed in Table 5.3.

Table 5.3: Pairwise Utility

$a_1 \longleftrightarrow a_2$
$g(s_{11}, s_{21})$: 0; $g(s_{11}, s_{22})$: $-\infty$
$g(s_{12}, s_{21})$: $-\infty$; $g(s_{12}, s_{22})$: 0
$g(s_{13}, s_{21})$: 0; $g(s_{13}, s_{22})$: $-\infty$

5.2.1.3 Belief Updating and Message Passing Process

Belief Updating

If a_i and a_j are adjacent, the belief of a_i about a state of a_j is the utility that a_i thinks the consumer can gain as a whole if a_j was designated the state, given the messages received from all of the adjacent providers of a_i except a_j.

A belief consists of two parts: the unary utility of the state and the sum of belief values contained within all the messages received from the provider's adjacent providers. Max-sum LBP starts by initializing beliefs about all states to zero, and the provider updates the belief of its state after receiving messages (which will be defined later) from all of its adjacent providers. Formally, provided that adj_u is the set of adjacent providers of provider a_u, and $bel(s_{us_1})$ is the belief about state s_1 of provider a_u, then $bel(s_{us_1})$ is updated by a_u through:

$$bel(s_{us_1}) = f(s_{us_1}) + \sum_{w \in adj_u} u_{w \to u}(s_{us_1}), \tag{5.1}$$

where $f(s_{us_1})$ is the unary utility of state s_{us_1}, and $u_{w \to u}(s_{us_1})$ (which will be introduced next) is the belief of a_w about state s_{us_1} contained within the message passed from provider a_w to a_u.

Message Passing Process

First of all, it has to be made clear that in this thesis, an iteration of belief propagation means once message passing. A round of belief propagation consists of at least one iteration, and ends when max-sum LBP converges. In each iteration, providers send messages to their adjacent providers to help them update their beliefs about states. A message is in fact a vector that contains n (n is the number of states of the message's recipient) elements each of which represents the message sender's belief about the related state of the message's recipient. Formally, if $m_{u \to v}$ denotes the message passed from provider a_u to provider a_v, then the belief of a_u about the state

s_{vs_1} contained in $m_{u\to v}$ is calculated by:

$$u_{u\to v}(s_{vs_1}) = max_{s_{ux}}(bel(s_{ux}) + g(s_{ux}, s_{vs_1}) + \sum_{w\in adj_u\setminus v} m_{w\to u}(s_{ux})) \tag{5.2}$$

From Equation (5.2), there are three parts in the calculation of $u_{u\to v}(s_{vs_1})$: a_u's belief about its own state s_{ux}, the pairwise utility $g(s_{ux}, s_{vs_1})$, and the sum of beliefs about state s_{ux} contained within the messages passed from all of a_u's adjacent providers except a_v in the previous iteration. Equation (5.2) is calculated for each state of a_v and thus calculated for p times if a_v has p states. The calculation results are writen into $m_{u\to v}$, then $m_{u\to v}$, i.e., the vector, is fulfilled and passed to a_v.

5.2.1.4 Strategic Quoting of Providers

In SLBP, in order to win the subtask(s) that they are interested in, providers strategically update their quotes in each iteration, which is up to both the belief propagation results in the previous iteration and the current situations of the environments. The basic quoting principle is introduced next, then the specific factors considered by the quoting decision are presented, and finally the quotes of providers are formulated.

Basic quoting principle

If the belief propagation result of a provider in the previous iteration is *inactive*, the provider tends to decrease its quote in the follow-up iteration. Otherwise, the provider tends to risks in increasing its quote in the follow-up iteration to gain more rewards. Therefore, the basic principle for a provider to update its quote is to search the best suitable quote in a lower range if its state in the previous iteration is *inactive* and in a higher range otherwise. With this in mind, this method was inspired by and thus adopted binary search algorithm as the basic quote principle with some alterations. The initial upper and lower boundaries of the binary searching range for a provider are the reward that the related consumer can gain from the subtask and the reserved price of the provider (i.e., the cost of the provider to finish the subtask), respectively. Provided that the cost of provider a_m to finish subtask τ_k and the reward that can be gained by the related consumer are c_m and rew_k, respectively, according to the basic quote principle introduced above, the calculation of the fundamental quote of a_m in the $(n+1)^{th}$ iteration, $fun_{a_m}(n+1)$, is presented in Algorithm 5.1. It is notable that $fun_{a_m}(n+1)$ is not the final quote for the $(n+1)^{th}$ iteration, rather, the final quote further takes other factors which will be studied next into account.

Algorithm 5.1: Fundamental Value of Quote

$low = c_m$, $high = rew_k$; $fun_{a_m}(1) = (low + high)/2$;
while (low < high)
  if a_m failed in the n^{th} iteration
    $high = fun_{a_m}(n)$;
  else
    $low = fun_{a_m}(n)$;
  end if
  $fun_{a_m}(n+1) = (high + low)/2$;
end while

Factors taken into quoting decision

There are three more factors should be taken into account when a provider makes quote decision. Firstly, due to the time constraint of task allocation, time should be considered by providers when they make quote decisions. Secondly, in a competitive environment, resource competition is also an important factor that a self-interested provider tends to consider in order to win the subtask that it aims at. Thirdly, it is apparent that a provider will not execute any subtask if the payment is lower than its cost to execute the subtask. Next, the above three factors are encoded into the quote formulation.

Quote formulation

Before formulating the quote, the characteristics of the quote from the perspective of applications in real world is analysed first. First, in real world applications, the closer to the task deadline, the more concessions the provider will make, and this is called time-dependent concession-making. The widely used time-dependent concession-making [ALIZ10][FSJ98][KZYL13][ALS08] suits to the problem, and thus is adopted in the proposed method. According to time-dependent concession-making, the later the time t is, the lower the quote of provider will be. Second, the lower level the resource competition is in, the lower the quote should be. Nevertheless, the provider in the proposed method does not have a global view, thus has no idea about the specific resource competition value in the market. To solve this problem, the provider estimates the resource competition according to the number of request messages about the related resource that it totally received (denoted by n_{req}), until the time it makes the quote decision. It is assumed that a higher n_{req} represents a higher level of resource competition, and consequently, the higher n_{req} is, the higher the quote should be. Third, the quote should meet its upper and lower boundaries. Suppose that c_m and rew_k are the cost of Provider a_m to execute and the reward that the related consumer

can gain from subtask τ_k, respectively, and $pri_m(n)$ is the quote of a_m in the n^{th} iteration. If a_m fails in the n^{th} iteration, the lower and upper boundaries of $pri_m(n+1)$ are c_m and $fun_m(n+1)$, respectively. If a_m wins in the n^{th} iteration, the lower and upper boundaries of $pri_m(n+1)$ are $fun_m(n+1)$ and rew_m, respectively. The problem when the boundary values should be obtained will be studied. Apparently, the lower boundary of $pri_m(n+1)$ should be obtained when the environment is the toughest to a_m. When a_m only receives one request message, and meanwhile the task deadline is coming soon, the situation is the toughest to a_m. Therefore, when $t = dl$ and $n_{req} = 1$, the environment is the toughest for the provider, and $pri_m(n+1)$ should be the lower boundary. In contrast, when $t = t_g$, the environment is the most favorable, and $pri_m(n+1)$ should be the upper boundary. With respect to the above analysis, $pri_m(n+1)$ is separately formulated for the two situations: a_m fails in the n^{th} iteration and a_m wins in the n^{th} iteration.

If a_m fails in the n^{th} iteration, $pri_m(n+1)$ is formulated as:

$$pri_m(n+1) = fun_m(n+1) - (fun_m(n+1) - c_m)\frac{t - t_g}{dl - t_g}\frac{1}{log_2(n_{req}+1)} \tag{5.3}$$

From Equation (5.3), when $t = dl$ and $n_{req} = 1$, the situation is the toughest and $pri_m(n+1) = c_m$. When $t = t_g$, $pri_m(n+1)$ is the highest and $pri_m(n+1) = fun_m(n+1)$.

If a_m wins in the n^{th} iteration, $pri_m(n+1)$ is formulated as:

$$pri_m(n+1) = rew_m - (rew_m - fun_m(n+1))\frac{t - t_g}{dl - t_g}\frac{1}{log_2(n_{req}+1)} \tag{5.4}$$

From Equation (5.4), when $t = dl$ and $n_{req} = 1$, the situation is the toughest and thus $pri_m(n+1)$ is the lowest, that is, $pri_m(n+1) = fun_m(n+1)$. When $t = t_g$, $pri_m(n+1)$ is the highest and equals to rew_m.

It is notable that the situation $t = t_g$ is an ideal situation that rarely happens because t_g is the generation time of the task. In addition, the situation is considered to be the toughest when $n_{req} = 1$, but not when $n_{req} = 0$. This is because once a_m calculates $pri_m(n+1)$, it means that it already received at least one request message. Otherwise, it does not make the quote decision making. Consequently, once Equation (5.3) or (5.4) is calculated, the lowest value of n_{req} is 1. Logarithm function is employed in Equations (5.3) and (5.4) to weaken the strong impact of n_{req} to $pri_m(n+1)$. Additionally, in order to avoid the situation '$log_2 1$' when $n_{req} = 1$, '$log_2(n_{req}+1)$' instead of '$log_2(n_{req})$' is employed in both Equations (5.3) and (5.4).

5.2.1.5 Task Allocation

When deadline of the task to be allocated comes, there are three possible situations.

- Before the coming of deadline of the task, belief propagation already converged. In addition, after the convergence, no selected provider decommites and no new providers comes before the task's deadline. In this situation, the subtasks are allocated to the providers of which the convergence results of belief propagation are not *inactive*.
- Belief Propagation already converged. However, the consumer has to re-operate belief propagation due to the decommitment(s) of some selected provider(s). The re-operation of belief propagation has not converged till the coming of belief propagation. In this situation, task allocation is considered failed.
- Belief Propagation already converged. However, in order to gain a higher utility, the consumer re-operates belief propagation due to the coming of some new provider(s). The re-operation has not converged till the coming(s) of the deadline of the task. In this situation, task allocation is considered failed as well.

5.2.2 Experimental Evaluation

An experiment is conducted to evaluate the performance of the proposed method in terms of testing the quote element of providers, robustness to dynamism, and the scalability.

5.2.2.1 Benchmarks

The work Max-sum Loopy Belief Propagation-based task allocation (MLBP for short) proposed in [WC13] is the foundation of SLBP and thus is selected as one of the benchmarks. It is notable that because MLBP is designed for static environments, for fairness reason, the performance comparison between MLBP and SLBP is studied in static environments. The robustness of SLBP to dynamism is tested through being compared with a well-known task allocation method proposed by An etc, in [ALS11]. Their method adopted negotiation as basis and devised heuristic-based buyer agents (HBA) for task allocation. Like SLBP, HBA is devised to work in dynamic and competitive environments. The reason to choose HBA as a benchmark is that HBA is one

of the well-known negotiation-based methods in recent years, and negotiation has been popular and widely used for task allocation. Through the comparison between SLBP versus HBA, both the advantages and disadvantages of belief propagation, compared with negotiation, can be tested.

5.2.2.2 Evaluation Measures

(1) Efficiency

According to the definition of messages passed between provider agents in Equation (5.2), the goal of belief propagation is to maximise the reward gained by a consumer from its task allocation. The reward, therefore, is one of the measures that should be tested. The possible values of a reward are analysed as follows. It is known that the belief propagation result of a provider may be *active* or *inactive*. Provided that a_i is the consumer, Task τ_k is the task to be allocated which consists of n subtasks, m is the number of providers whose belief propagation results are *active*, and $rew(a_i, \tau_k)$ is the reward gained by a_i from the allocation of τ_k. There are three possible results between m and n: $m > n$, $m < n$ and $m = n$. If $m > n$, it means there must be at least one subtask, which has been assigned to more than one provider; if $m < n$, it means there must be at least one subtask, which has not been assigned to any provider. It is stipulated that $rew(a_i, \tau_k) = -\infty$ when $m > n$ and $m < n$. If $m = n$ and meanwhile, every subtask has been assigned to a provider, task allocation is considered to be successful, and $rew(a_i, \tau_k)$ is defined as:

$$rew(a_i, \tau_k) = r_{sub_1} + r_{sub_n} + \sum_{j=2}^{n-1}(r_{sub_j} + g(sub_{j-1}, sub_j) + g(sub_j, sub_{j+1})) - \sum_{j=1}^{m} pri_j, \quad (5.5)$$

where $g(sub_{j-1}, sub_j)$ is the pairwise utility between the two providers whose belief propagation results are *active* and the relevant subtasks are sub_{j-1} and sub_j, respectively, $g(sub_j, sub_{j+1})$ is the pairwise utility between the two providers whose belief propagation results are active and the relevant subtasks are sub_j and sub_{j+1}, respectively, and pri_j is the payment that a_i pays to the relevant provider for sub_j.

For simplicity, efficiency $effi(a_i, \tau_k)$ is defined by normalizing $rew\ (a_i, \tau_k)$:

$$effi(a_i, \tau_k) = rew(a_i, \tau_k) / \sum_{j=1}^{n} pri_j, \quad (5.6)$$

where pri_j is the payment that a_i pays to the relevant provider for sub_j.

It is notable that when $effi(a_i, \tau_k)$ is tested, only the data which are not $-\infty$ are used. The reason is that even only one $-\infty$ will make the average efficiency $-\infty$, and consequently makes the test meaningless. However, it will bias the evaluation results if only the non-negative data are chosen. To solve this problem, the success rate of task allocation which will be defined next is tested as well.

(2) Success Rate

The success rate of task allocation denoted by r_{suc} is the ratio of the number of tasks from which the efficiencies obtained by the relevant consumers are positive (denoted by n_{suc}) to the total number of tasks involved (denoted by n_{tot}).

$$r_{suc} = n_{suc}/n_{tot} \tag{5.7}$$

The other reason to test the success rate is that efficiency and success rate may conflict with each other, that is, higher success rate could be obtained by decreasing efficiency and vice versa. Consequently, success rate cannot be ignored when efficiency is tested.

(3) Number of Passed Messages before the First Time of Convergence of Belief Propagation

There are two reasons to test the number of passed messages before the first time of convergence of belief propagation. One reason is that the number of passed messages heavily affects the required bandwidth (if bandwidth is needed) for the communication between agents. The other reason is that due to the time constraint, speed is important to task allocation. However, different communication media used for task allocation can result in the time needed by task allocation being very different, thus it is unfair to test the used time. Instead, to test the number of passed messages is quite fair.

5.2.2.3 Evaluation Settings

In SLBP, the employed basic technology, belief propagation, maybe cannot converge in cyclic graphs. In addition, whether belief propagation converges or not plays an important role to the results of task allocation. Consequently, performance examination in both acyclic and cyclic networks are needed. Because belief propagation is an algorithm of optimization inference through local message passing, networks of too large scales are neither instructive nor necessary for evaluation. Considering the above analysis, four networks for evaluation presented in Figures 5.4 - 5.7 are constructed: **1)** a simple tree-structured network, **2)** a middle scale cyclic network with one loop, **3)** a large scale cyclic network with more alternative providers for each subtask, **4)**

and a large scale cyclic network with more tiers (subtasks). It is notable that Figures 5.4 - 5.7 are the schematic graphs indicating the relationships between subtasks and providers.

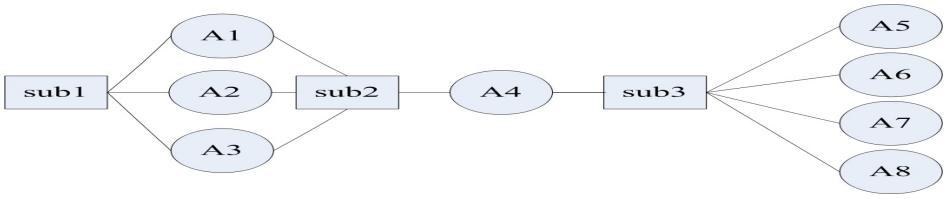


Figure 5.4: NET A: Small Scale Tree-Structured Network

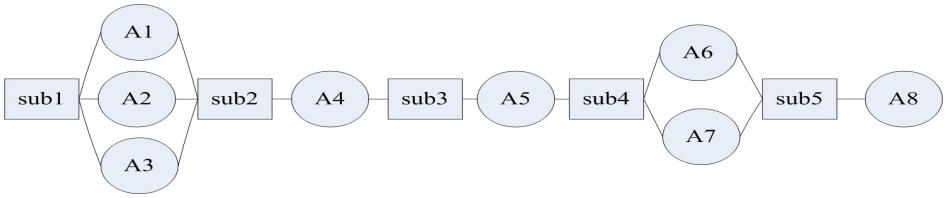


Figure 5.5: NET B: Middle Scale Network with One Loop

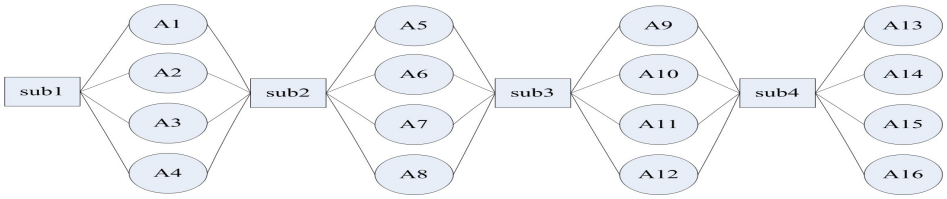


Figure 5.6: NET C: Large Scale Loopy Network with More Alternatives for Each Subtask

There are three scenarios are conducted from different perspectives, which are:

Scenario 1: To Test the Quote Element of Providers

Scenario 2: To Test the Robustness to Dynamism

Scenario 3: To Test the Scalability

Settings for Scenario 1

According to the introduction to quote strategy, the main contribution of the quote strategy is taking both resource competition and task deadline into account for quote decision. Therefore, the quote element is tested from the perspectives of both resource competition and the time constraint.

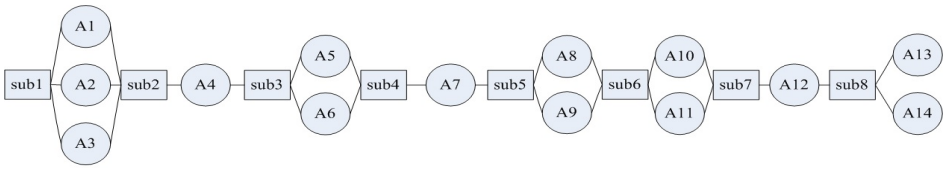


Figure 5.7: NET D: Large Scale Loopy Network with More Tiers (Subtasks)

The provider in SLBP does not have a global view and thus no idea about the specific resource competition level. To solve this problem, the number of request messages received by a provider (denoted by N_{req}) is used to estimate the resource competition level. The more request messages a provider receives, the higher the provider considers the resource competition to be. The resource competition denoted by $\psi(r)$ is classified into three levels based on the number of messages received by a provider, presented in Table 5.4.

Table 5.4: Classification of Resource Competition Level

N_{req}	Resource Competition Level
$[1, 5]$	Low
$(5, 10]$	Middle
$(10, 20]$	High

Different urgency level criteria are defined for the four networks, presented in Table 5.5.

Table 5.5: Classification of Urgency Level

	NET A (*time unit*)	NET B (*time unit*)	NET C (*time unit*)	NET D (*time unit*)
Low urg	1000	2500	30000	7500
Middle urg	750	1500	20000	5000
High urg	500	1000	10000	2500

In Table 5.5, when a task's deadline is longer than 1000 time units, it is considered to be in low urgency in NET A, while only when the task's deadline is longer than 30000 time units, it is considered to be in low urgency in NET C. The reason to define different urgency level criteria for the four networks is the different structures of the networks. Structure is one of the key factors that affect the speed of the convergence of belief propagation. Consequently, it is unfair to define a same urgency level criterion

for different structures. In real life applications, deadlines may be quite long. However, the evaluation is just a simulation by Java but not real allocations, consequently, every 10 *seconds* is viewed as a time unit.

Setting for Scenario 2

In Scenario 2, both SLBP and HBA are implemented by JAVA in the four networks. A controller is employed to generate provider(s) for each subtask, and the providers are set *active* or *inactive* before the run of the experiment. After the experiment starts to run, inactive providers are activated and active ones are deactivated at predefined probabilities in each iteration. The entering and leaving of agents, i.e., the dynamism of the environment, is simulated through the activation and deactivation of providers. Different dynamism levels (denoted by *dyn*) are obtained through changing the probabilities of activation and deactivation, and are classified into four levels based on the probability intervals, presented in Table 5.6.

Table 5.6: Classification of Dynamism Level

Probability Intervals	Dynamism Levels
$(0, 0.2]$	Low
$(0.2, 0.5]$	Middle
$(0.5, 0.8]$	High
$(0.8, 1]$	Very High

Settings of Scenario 3

In Scenario 3, the scalability of SLBP is tested with MLBP to be the benchmark from the viewpoints of both horizontal (i.e., the average number of alternative providers for each subtask, denoted by N_{alt}) and vertical (i.e., the number of subtasks, denoted by N_{tier}). The various N_{alt} are obtained through varying the number of backup providers for each subtask while remaining the probability of backup providers' joining unchanged. For example, if the number of backup providers for each subtask and the probability of joining of each backup provider per iteration are set 10 and 0.1, respectively, then N_{alt} in NET A is $(3 + 1 + 4) \div 3 + 10 \times 0.1 = 3.67$. In this simulation, the probability of joining of providers is fixed at 0.1, the backup providers for each provider and the according N_{alt} are presented in Table 5.7.

Table 5.7: Parameter Settings for Scenario 3

Number of Backup Providers for Each Subtask	N_{alt} Net A	Net B	Net C	Net D
10	3.67	2.6	5	2.75
20	4.67	3.6	6	3.75
40	6.67	5.6	8	5.75
60	8.67	7.6	10	7.75
80	10.67	9.6	12	9.75
100	12.67	11.6	14	11.75

5.2.2.4 Experimental Results and Analysis

Results and Analysis of Scenario 1

The evaluation results based on various resource competition levels are listed in Table 5.8.

Table 5.8: Performance with Various Resource Competition Levels

	Average Efficiency		Success Rate		Average Total Passed Messages	
	SLBP	MLBP	SLBP	MLBP	SLBP	MLBP
Net A (middle *urg*, static)						
low $\psi(r)$:	0.90	0.91	1.00	1.00	92	86
middle $\psi(r)$:	0.88	0.80	1.00	1.00	88	83
high $\psi(r)$:	0.81	0.62	1.00	1.00	89	62
Net B (middle *urg*, static)						
low $\psi(r)$:	0.82	0.81	0.98	0.99	282	192
middle $\psi(r)$:	0.78	0.65	0.99	0.95	248	183
high $\psi(r)$:	0.72	0.40	1.00	0.98	239	189
Net C (middle *urg*, static)						
low $\psi(r)$:	0.72	0.60	0.84	0.80	3901	2109
middle $\psi(r)$:	0.63	0.58	0.72	0.62	3821	2023
high $\psi(r)$:	0.61	0.44	0.61	0.49	3019	1809
Net D: (middle *urg*, static)						
low $\psi(r)$:	0.80	0.79	0.92	0.88	896	396
middle $\psi(r)$:	0.72	0.70	0.86	0.72	853	323
high $\psi(r)$:	0.68	0.58	0.79	0.56	782	258

From Table 5.8, it can be seen that both the average success rate and efficiency of MLBP decrease more sharply than those of SLBP with the increase of $\psi(r)$. In addition, when $\psi(r)$ is in a high level, both the success rate and average efficiency of SLBP are higher than those of MLBP. This is because in SLBP, providers take the resource competition into account when doing the quote decision (see Equations (5.3) and (5.4)). For example, when the resource competition is in a high level, the provider will decrease its quote accordingly. Consequently, this can weaken the impact of the increase of $\psi(r)$ on the chance of the provider in winning the subtask that it aims

at, and thus increase both the success rate and average efficiency. As can be seen from Table 5.8, more messages are passed in SLBP than in MLBP before the belief propagation converged for the first time. The reason is that in SLBP, a provider may update its quota once per iteration of belief propagation, and this prolongs the process before the belief propagation converges in SLBP. Now the evaluation results are studied from the perspective of network structure. As can be seen from Table 5.8, in the four networks, NET C captured the lowest success rate, the lowest average efficiency, and the most passed messages. The reason for this is the strong connectedness of NET C, that is, the average number of edges per agent in NET C is more than those in the other three networks. The strong connectedness can result in the belief propagation taking long time to converge, or even prevent the belief propagation from converging.

Table 5.9 presents the performances of both SLBP and MLBP based on various urgency levels.

Table 5.9: Performance with Various Urgency Levels

	Average Efficiency		Success Rate		Average Total Passed Messages	
	SLBP	MLBP	SLBP	MLBP	SLBP	MLBP
Net A (middle $\psi(r)$, static)						
low urg:	0.96	0.93	0.99	0.93	102	83
middle urg:	0.89	0.91	0.98	0.92	119	89
high urg:	0.83	0.92	0.92	0.93	108	92
Net B (middle $\psi(r)$, static)						
low urg:	0.96	0.94	0.93	0.90	269	201
middle urg:	0.89	0.89	0.89	0.91	271	199
high urg:	0.84	0.88	0.90	0.90	249	188
Net C (middle $\psi(r)$, static)						
low urg:	0.75	0.69	0.78	0.74	3916	2210
middle urg:	0.66	0.66	0.72	0.72	3836	2218
high urg:	0.63	0.68	0.70	0.71	3125	1993
Net D (middle $\psi(r)$, static)						
low urg:	0.82	0.78	0.89	0.82	901	405
middle urg:	0.76	0.79	0.83	0.81	828	383
high urg:	0.72	0.78	0.77	0.79	762	391

As can be seen from Table 5.9, when the urgency level varies from low to high, neither the average success rate nor the efficiency of MLBP changes too much. The reason for this is that quotes of providers in MLBP are fixed through the whole belief propagation, therefore, task allocation succeeds and is terminated when the belief propagation converges for the first time. Consequently, the varying of urgency level does not make much difference to MLBP. Different from MLBP, providers in SLBP update their quotes during the belief propagation process from time to time, and this results

in belief propagation taking longer to converge in SLBP than in MLBP. Therefore, SLBP is more sensitive to urgency level than MLBP. However, even though the quote updating in SLBP delays the convergence of belief propagation, the time-dependent concession of providers (see Equations (5.3) and (5.4)) prevents both the success rate and efficiency from becoming too low. Therefore, when urgency is in a high level, both the average success rate and efficiency of SLBP are almost as high as those of MLBP.

Results and Analysis of Scenario 2

In SLBP, when a new provider enters the environment, it may take part in belief propagation in the subsequent iteration, and this results in the asynchronous message passing. Even though belief propagation has not been proved in theory to converge when participants pass messages asynchronously, large amount of experiments have shown that belief propagation works in asynchronous environments almost as well as in synchronous ones [CP02], and this is also proved by the results presented in Table 5.10.

Table 5.10: Performance with Various Dynamism Levels

	Average Efficiency		Success Rate		Passed Messages	
	SLBP	HBA	SLBP	HBA	SLBP	HBA
Net A (middle $\psi(r)$, middle urg)						
low dyn:	0.982	0.980	0.958	0.90	96	81
middle dyn:	0.883	0.82	0.861	0.80	133	99
high dyn:	0.79	0.63	0.820	0.62	158	109
Net B (middle $\psi(r)$, middle urg)						
low dyn:	0.93	0.89	0.88	0.81	293	198
middle dyn:	0.86	0.68	0.86	0.72	352	209
high dyn:	0.85	0.49	0.81	0.46	379	233
Net C (middle $\psi(r)$, middle urg)						
low dyn:	0.74	0.93	0.76	0.86	3935	2213
middle dyn:	0.65	0.70	0.63	0.73	3866	2392
high dyn:	0.60	0.51	0.62	0.50	4293	2381
Net D (middle $\psi(r)$, middle urg)						
low dyn:	0.83	0.76	0.89	0.66	878	381
middle dyn:	0.78	0.44	0.83	0.63	899	402
high dyn:	0.70	0.38	0.78	0.33	932	438

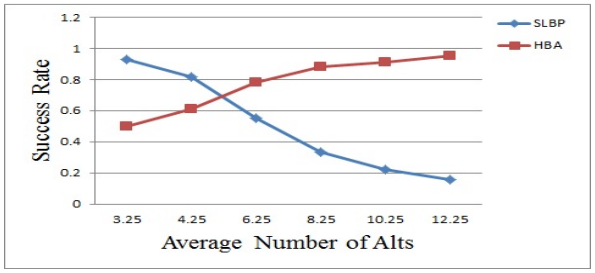
As can be seen from Table 5.10, both the success rate and average efficiency of SLBP do not decrease sharply with the environment becoming more dynamic. Unlike SLBP, when the dynamism level increases, both the average success rate and efficiency of HBA decrease sharply. The reason is that the consumer in HBA handles all the negotiation threads in parallel, when the environment becomes more dynamic, the negotiation threads become more computationally intractable. It can also be seen from Table 5.10 that in NET C, HBA outperformes SLBP in both the success rate

and efficiency when dynamism is in a low level. This is because the belief propagation in SLBP is affected heavily by loops in NET C, while the negotiation in HBA is not. From the above analysis, compared with HBA, SLBP works better in more dynamic environments with fewer loops. From Table 5.10, it can also be seen that compared with the belief propagation in SLBP, the negotiation in HBA needs more messages for task allocation. The passed messages in SLBP are distributed on all the involved providers almost averagely while all of the messages in HBA are sent or received by the consumer, and this can easily cause communication overload to the consumer in HBA.

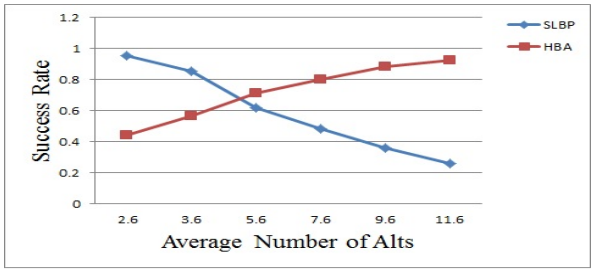
Results and Analysis of Scenario 3

The testing results of Scenario 3 are shown from Figure 5.8 to Figure 5.11.

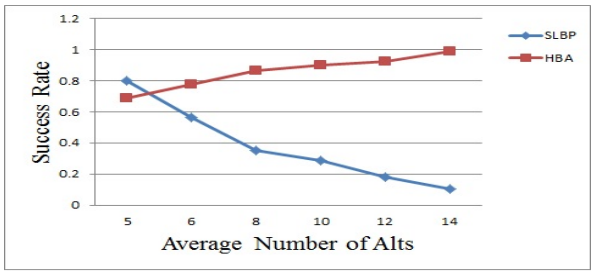
The success rates and average efficiencies based on various average number of alternatives for each subtask are demonstrated in Figures 5.8 and 5.9, respectively.



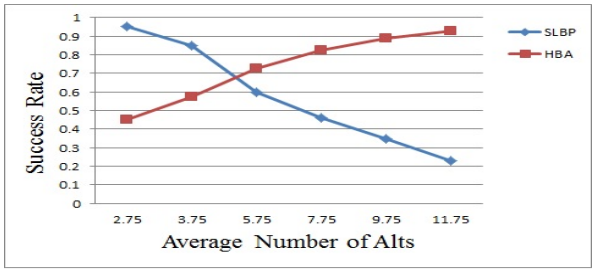
(a) Success Rates based on Average Numbers of Alternatives per Subtask in Net A



(b) Success Rates based on Average Numbers of Alternatives per Subtask in Net B

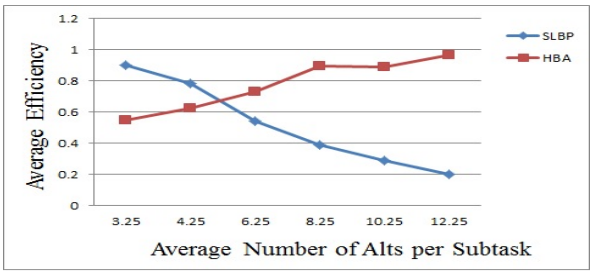


(c) Success Rates based on Average Numbers of Alternatives per Subtask in Net C

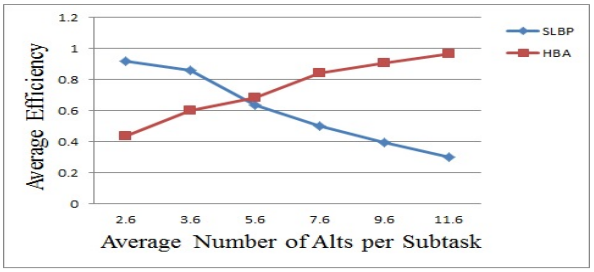


(d) Success Rates based on Average Numbers of Alternatives per Subtask in Net D

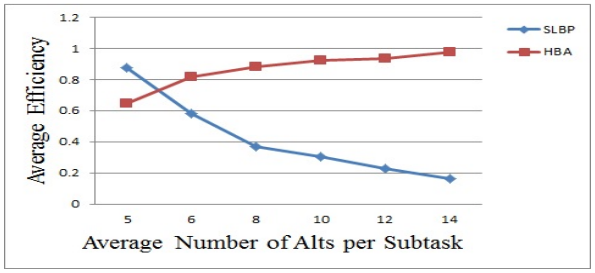
Figure 5.8: Success Rates based on Average Numbers of Alternatives per Subtask



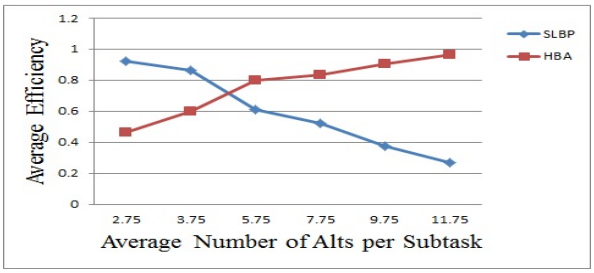
(a) Average Efficiencies based on Average Numbers of Alternatives per Subtask in Net A



(b) Average Efficiencies based on Average Numbers of Alternatives per Subtask in Net B



(c) Average Efficiencies based on Average Numbers of Alternatives per Subtask in Net C

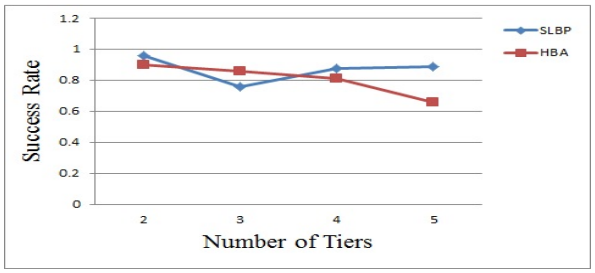


(d) Average Efficiencies based on Average Numbers of Alternatives per Subtask in Net D

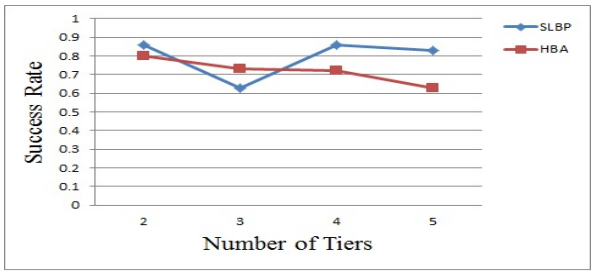
Figure 5.9: Average Efficiencies based on Average Numbers of Alternatives per Subtask

As can be seen from Figures 5.8 and 5.9, with the increase of N_{alt}, both the success rate and average efficiency of SLBP decrease in all of the four networks. This is because networks have more loops when there are more alternative providers for each subtask, and the performance of loopy belief propagation depends in a big part on the number of loops in a network. In contrast, both the success rate and average efficiency of HBA increase with the increase of N_{alt}. This is because more alternative providers for each subtask means a more favorable market for the consumer, and thus a higher chance to successfully allocate a task.

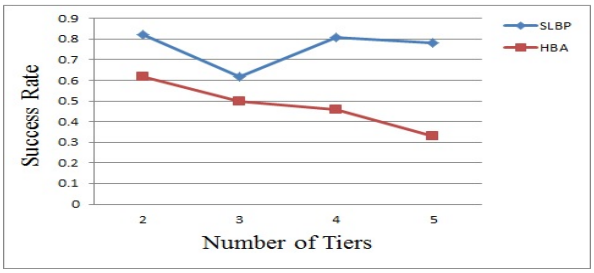
The success rates and average efficiencies based on various average number of tiers are demonstrated in Figures 5.10 and 5.11, respectively.



(a) Success Rates based on Numbers of Tiers in Net A with Low Dynamism Level

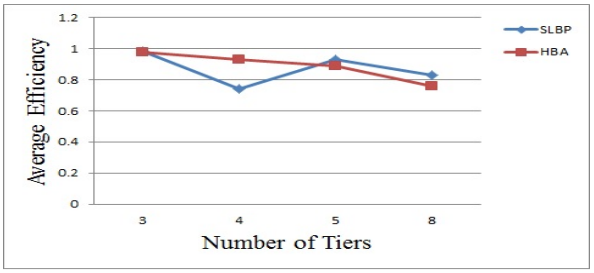


(b) Success Rates based on Numbers of Tiers in Net B with Middle Dynamism Level

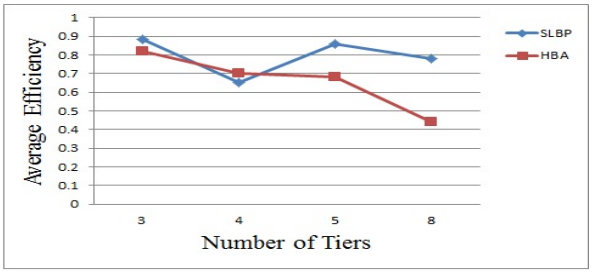


(c) Success Rates based on Numbers of Tiers in Net C with High Dynamism Level

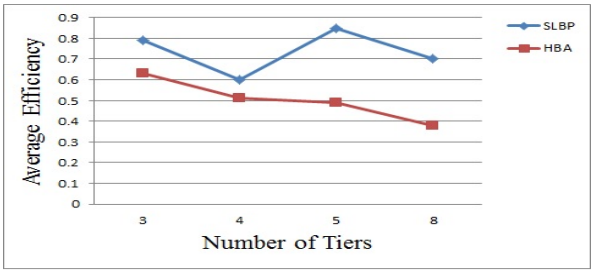
Figure 5.10: Success Rates based on Different Dynamism Levels



(a) Average Efficiencies based on Numbers of Tiers in Net A with Low Dynamism Level



(b) Average Efficiencies based on Numbers of Tiers in Net B with Middle Dynamism Level



(c) Average Efficiencies based on Numbers of Tiers in Net C with High Dynamism Level

Figure 5.11: Average Efficiencies based on Numbers of Tiers

From Figures 5.10 and 5.11, it can be seen that both the success rate and average efficiency of SLBP keep steady when N_{tier} varies from 3 (NET A) to 8 (NET D) with the exception of NET B. NET B is an exception, which is not due to its number of tiers, but is caused by the too many loops in it. This result demonstrates that the number of tiers does not affect the performance of belief propagation as much as the number of loops does. Additionally, belief propagation works in networks with more tiers almost as well as in the ones with less tiers. In contrast, both the success rate and average efficiency of HBA decreases dramatically with the increase of N_{tier}. This could be explained from the separate negotiations in HBA. In HBA, the failure of negotiation for even only one subtask can result in all the already obtained resources becoming invalid. Therefore, more subtasks can result in a higher probability of the failure of task allocation. In summary, SLBP outperforms HBA in networks with large vertical scales, but does not work as well as HBA in networks with large horizontal scales.

5.3 A Max-sum Belief Propagation-based Method for Task Allocation in Open and Dynamic Environments

In this Section, the group task allocation with dependency constraints is studied from a different perspective with the method proposed in Section 5.2. In particular, the task allocation is studied from the perspective of the computational simplification of the belief propagation in this section. To do this, a pruning-decomposition loopy belief propagation-based task allocation method (PD-LBP) is accordingly proposed. PD-LBP is in detail introduced in Subsection 5.3.1, and experimentally evaluated in Subsection 5.3.2.

5.3.1 PD-LBP

PD-LBP is based on the loopy belief propagation-based (LBP) supply chain formation method [WC13] proposed by Whisper and Chli in the state and belief definitions. PD-LBP improves LBP mainly from two points. First, PD-LBP simplifies LBP through two simplification phases (i.e., pruning and decomposition), aiming at improving the

performances in terms of quick convergence of belief propagation, mitigating the communication requirement, and accelerating the online response to and resilience from unpredicted changing when the environment is highly dynamic. Second, unlike LBP where a consumer only considers the quotes of alternatives, both the reserve price and deadline are considered.

5.3.1.1 State and Belief of Providers

Figure 5.12 is a MRF graph used as an example. In Figure 5.12, the task to be allocated consists of two subtasks, t_1 and t_2. p_1 is the alternative for subtask t_1, p_2 and p_3 are the alternatives for subtask t_2, and c_1 is the consumer. The states of an alternative is defined the same way with that in Subsection 5.2.1. Accordingly, the states of p_2 include 1) *inactive*, and 2) taking over the task from p_1 and passing the finished task to the consumer.

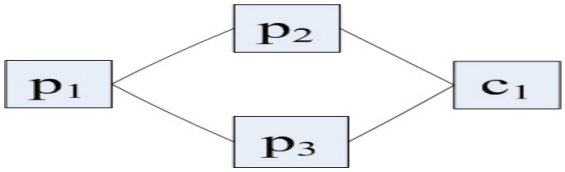


Figure 5.12: Makov Random Field Form

p_1 and p_2 are taken as examples and all the possible states of them are listed in Table 5.11 where s_m^n stands for the nth state of alternative a_m.

Table 5.11: States of p_1 and p_2

states of p_1	states of p_2
s_1^1: inactive	s_2^1: inactive
s_1^2: executing t_1 and then passing over the task to p_2 to execute t_2	s_2^2: taking over the task from p_1 to execute t_2 and passing over the finished task to c_1
s_1^3: executing t_1 and then passing over the task to p_3 to execute t_2	

There are two types of utilities in PD-LBP: the **unary utility** (which will be formulated later) of a state and the **pairwise utility** between two states of two adjacent agents. The pairwise utility in this subsection is defined as the same way as that in Subsection 5.2.1. p_1 and p_2 are taken as examples and the the pairwise utilities between their states are listed in Table 5.12.

Table 5.12: Pairwise utility between States of p_1 and p_2

$p_1 \longleftrightarrow p_2$
$g(s_1^1, s_2^1) = 0$
$g(s_1^1, s_2^2) = -\infty$
$g(s_1^2, s_2^1) = -\infty$
$g(s_1^2, s_2^2) = 0$
$g(s_1^3, s_2^1) = 0$
$g(s_1^3, s_2^2) = -\infty$

Unlike the method in Section 5.2 where the consumer considers only one attribute (i.e., the quote) of the service of an alternative, in PD-LBP, there are two attributes of the delivered service of an alternative taken into account by the consumer, i.e., the quote and the time needed by the alternative to finish the related subtask. Against this, the simple multi-attribute rating technique (SMART) methodology [Edw77, VWE+86] is employed to formulate the unary utility, and there are three main steps for this.

Step 1: To identify the decision maker, alternatives, and attributes;

Apparently, the decision maker and alternatives are the consumer and provider agents, respectively. The attributes include both the quote and the execution time.

Step 2: To scale and develop weights for each attribute;

It is assumed that a consumer has expected ranges for the quote and the needed execution time, denoted by $[q_{low}, q_{high}]$ and $[t_{low}, t_{high}]$, respectively, for each individual subtask. This assumption is reasonable and feasible in real life applications in that the consumer always has explicit and fixed restrains for neither the quote nor the execution time for each individual subtask, but only has explicit and fixed restrains for the total cost and the total execution time for the whole task. $[q_{low}, q_{high}]$ and $[t_{low}, t_{high}]$ are used as the respective scales of quote and execution time. The consumer assigns weight values to the quote and execution time, according to its preference which is beyond the research in this thesis. w_1 and w_2, subject to $w_1 + w_2 = 1$, are used to denote the weights assigned to the quote and execution time, respectively.

Step 3: To score each alternative on each attribute separately first, then value the alternatives accordingly and synthetically.

Suppose that q_i and exe_i are the quotes and execution time of alternative a_i, respectively, the score of a_i on q_i is formulated as:

$$s_{q_i} = \frac{q_{high} - q_i}{q_{high} - q_{low}} \tag{5.8}$$

The score of a_i on exe_i is formulated as:

$$s_{exe_i} = \frac{t_{high} - exe_i}{t_{high} - t_{low}} \tag{5.9}$$

An additive value function, which is widely used, is adopted to score alternative a_i on both q_i and exe_i synthetically by:

$$scor(a_i) = w_1 \times s_{q_i} + w_2 \times s_{exe_i} \tag{5.10}$$

$scor(a_i)$ is used as the unary utility of the states of a_i except the state of *inactive* of which the unary utility is set zero.

After defining the states and corresponding beliefs for belief propagation, the simplification phases of PD-LBP will be in detail introduced next, before introducing the belief updating and message passing later.

5.3.1.2 Computational Simplification

Imagine that if the network formed by all the alternatives and consumer can be separated into more than one independent part, and thus belief propagation can be locally operated in these parts in parallel but not sequentially in the whole network, then the convergence of belief propagation (i.e., the optimization solution) could probably be obtained quicker. In addition, when changing in the environments takes place frequently, the caused large amount of re-computations and message passing may prevent belief propagation from converging. Even after the convergence of belief propagation, the consumer may still want to re-operate the belief propagation before the task deadline whenever a new provider (alternative) comes to check whether a higher utility can be obtained due to the arrival of the new alternative, or has to re-operate the belief propagation due to the defaults of some committed alternatives. Based on all of the above inspirations, two phases, pruning and decomposition, are designed to mitigate the problem. In the pruning phase, given both the reserve price and task deadline, the

alternatives (if any) that will never maximise the whole utility regardless of the configurations of all other alternatives which are pruned in advance of belief propagation. After the pruning, the network is decomposed into independent parts (if possible), and this is the decomposition phase. The two phases are in detail studied separately.

Phase 1: Pruning

The principles of pruning come from both the reserve price and deadline constraints. If no matter what the configuration of all the other alternatives is, the summation of the quote (execution time) of an alternative for the subtask that this alternative is interested in and the quotes (execution time) for all the other subtasks is higher (larger) than the reserve price (task deadline), apparently this alternative will never be selected and thus should be pruned in advance of and to simplify belief propagation. Such an alternative is called a dominated alternative. It is assumed that the consumer has all the alternatives' information (i.e., quote and execution time). This assumption is feasible and practical in that it is common for the alternatives to register in the consumer when they arrive at the environment. It is notable that the consumer has the information about all the alternatives does not mean that PD-LBP is centralised, because belief propagation is decentralised and does not need the control of the consumer. Therefore, it is reasonable for us to designate the consumer to do the pruning and decomposition which will be introduced later. Formally, assume that the quote and execution time of alternative $a_i \in A_i$ (A_i is the alternative set of subtask t_i) are q_i and exe_i, respectively, if at least one of equations (5.11) and (5.12) is met, a_i is dominated and will be pruned.

$$q_i + \sum_{t_k \in \mathcal{T} \setminus t_i} min\{q_k | a_k \in A_k\} > p_{res}, \tag{5.11}$$

or

$$exe_i + t + \sum_{t_k \in \mathcal{T} \setminus t_i} min\{exe_k | a_k \in A_k\} > dl, \tag{5.12}$$

where $\mathcal{T}$ is the subtask set, t is the time when Equation (5.11) is calculated, A_k is the alternative set of subtask t_k, and p_{res} and dl are the reserve price and the deadline of $\mathcal{T}$, respectively.

It is recognised that in the dynamic environments a dominated alternative may become non-dominated due to the arrivals of some new alternatives. In order to reactivate such alternatives, a consumer stores a list of all the dominated alternatives, and checks whether the dominated alternatives become non-dominated whenever a

new alternative comes. If a dominated alternative becomes non-dominated, the consumer informs this alternative and the alternative's intermediate neighbours to make the alternative involved into the belief propagation in the next iteration.

Phase 2: Decomposition

Decomposition in belief propagation has been paid more and more attention recently [KKL11]. Before in detail introducing the decomposition phase in PD-LBP, some concepts have to be made clear. If subtask t_{i+1} cannot start to be executed until subtask t_i is finished, then t_i is called the predecessor of t_{i+1}, and t_{i+1} is the successor of t_i. For example, in Figure 5.12, t_2 is the predecessor of t_3, and t_3 is the successor of t_2. It is possible that all the alternatives of the predecessor subtask can collaborate with all the alternatives of the successor one. This means that the alternative selection for the predecessor subtask does not affect that for the successor one. According to the Markov properties presented in Subsection 5.2.1, non-adjacent nodes in MRF are conditionally independent. Consequently, if the alternative selections for two adjacent subtasks are independent, the MRF (i.e., the corresponding network formed by alternatives and the consumer) can be decomposed between these two adjacent subtasks. To find such adjacent subtasks, a dependency weight between two adjacent subtasks is defined. Assume that t_i and t_{i+1} are adjacent, when the dependency weight between them, $w_{dep}(i, i+1)$, is zero, the network could be separated between them, and the connection between them is called a separation link.

Formally, if A_i and A_{i+1} are the alternative sets of t_i and t_{i+1}, respectively, the dependency weight between t_i and t_{i+1}, i.e., $w_{dep}(i, i+1)$, is zero if the following condition is met: $\forall(a_m \in A_i, s_m \in S_m \setminus inactive, a_k \in A_{i+1}), \exists s_k \in S_k, g(s_m, s_k) = 0$ where S_m and S_k are the state sets of a_m and a_k, respectively.

Assume that the whole network is decomposed into two parts: the left part and the right one. Upon the decomposition, a problem rises: there may be more than one chain formed for the subtasks in the left part. Against this, an agent is added to the end of the left part to play the role of the consumer to guarantee that only one chain is formed. For example, if Figure 5.1 is decomposed into two parts from the connection between t_2 and t_3, then a new agent will be added into the end of the left part, as shown in Figure 5.13.

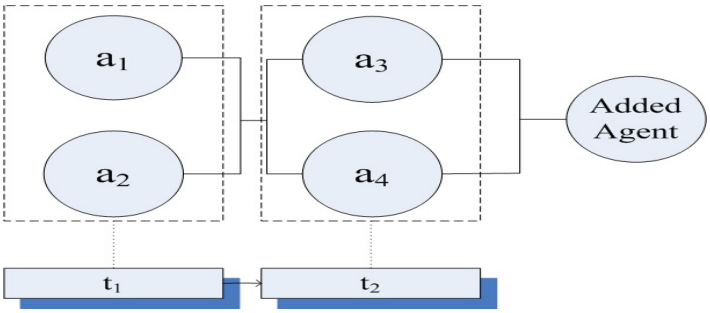


Figure 5.13: The Left Part with an Added Agent

As it was analysed earlier, after the convergence of belief propagation, the committed alternatives may default. Additionally, new alternatives keep coming. Against this, how to deal with the dynamism from the perspectives of both the leaving and arrival of alternatives is not trivial and thus is the problem which is studied in the next subsection.

5.3.1.3 Dealing with the Dynamism

Suppose that t_{i-1} is the predecessor of t_i, and t_i is the predecessor of t_{i+1}. In addition, both $w_{dep}(t_{i-1}, t_i)$ and $w_{dep}(t_i, t_{i+1})$ are zero. Next, how to deal with the leaving and arrival of alternatives are studied separately.

(1) Dealing with the Leaving of an Alternative

In fact, the leaving of an alternative of t_i affects neither $w_{dep}(t_{i-1}, t_i)$ nor $w_{dep}(t_i, t_{i+1})$. Formally:

Theorem 1: After the alternative $a_m \in A_i$ (A_i is the alternative set of t_i) leaving, both $w_{dep}(t_{i-1}, t_i)$ and $w_{dep}(t_i, t_{i+1})$ are still zero. The following is the proof about this.

Proof: Because $w_{dep}(t_{i-1}, t_i) = 0$, it has $\forall(a_n \in A_{i-1}, s_n \in S_n \setminus inactive, a_k \in A_i)$, $\exists s_k \in S_k, g(s_n, s_k) = 0$ (S_n and S_k are the state sets of a_n and a_k, respectively). Consequently, when $a_m \in A_i$ leaves, $\forall(a_n \in A_{i-1}, s_n \in S_n \setminus inactive, a_k \in A_i \setminus a_m)$, it still holds that $\exists s_k \in S_k$, $g(s_n, s_k) = 0$. Therefore, $w_{dep}(t_{i-1}, t_i)$ is still zero.

Similarly, because $w_{dep}(t_i, t_{i+1}) = 0$, then it is known that $\forall(a_n \in A_i, s_n \in S_n \setminus inactive, a_k \in A_{i+1}), \exists s_k \in S_k, g(s_n, s_k)$
$= 0$. Consequently, when $a_m \in A_i$ leaves, $\forall(a_n \in A_i \setminus a_m, s_n \in S_n \setminus inactive, a_k \in A_{i+1})$, it still holds that $\exists s_k \in S_k, g(s_n, s_k) = 0$. Therefore, $w_{dep}(t_i, t_{i+1}) = 0$ still holds.

(2) Dealing with the Arrival of a New Alternative

When a new agent $a_p \in A_i$ comes, if $\forall(s_p \in S_p \setminus inactive, a_k$
$\in A_{i+1}), \exists s_k \in S_k, g(s_p, s_k) = 0$, then it has $\forall(a_n \in A_i \cup \{a_p\}, s_n \in S_n \setminus inactive, a_k \in$

$A_{i+1}), \exists s_k \in S_k, g(s_n, s_k) = 0$. In this situation, $w_{dep}(t_i, t_{i+1})$ is still zero. Otherwise, t_i and t_{i+1} are not independent any more due to the arrival of a_p. As a consequence, the two parts, which contain t_i and t_{i+1} respectively, will be merged into one. Upon the merging, the agent which was added into the end of the left part (i.e., the part that contains t_i) when the two parts were separated is removed.

If $\forall(a_m \in A_{i-1}, s_m \in S_m \setminus inactive), \exists s_p \in S_p, g(s_m, s_p) = 0$, then it has $\forall(a_m \in A_{i-1}, s_m \in S_m \setminus inactive, a_k \in A_i \cup \{a_p\}), \exists s_k \in S_k, g(s_m, s_k) = 0$. In this situation, the separation between t_{i-1} and t_i still holds. Otherwise, the separation becomes invalid, and the two corresponding parts will be merged into one. In addition, the agent added to the end of the left part is removed as well.

5.3.1.4 Belief Updating and Message Passing

After the pruning and decomposition, belief propagation is in parallel operated on the pruned and decomposed parts of the network. The beliefs about all the states are initialised to 0. After receiving messages from all of its adjacent agents, agent a_u updates its belief about its state s_u^i, denoted as $bel_u(s_u^i)$, by:

$$bel_u(s_u^i) = uti(s_u^i) + \sum_{a_v \in N_u} m_{v \to u}(s_u^i), \tag{5.13}$$

where $uti(s_u^i)$ is the unary utility of s_u^i, N_u is the neighbour (i.e., adjacent agent) set of a_u, and $m_{v \to u}(s_u^i)$ is the belief of a_v about s_u^i contained within the message passed from a_v to a_u which will be defined next.

The message passed from a_v to a_u, denoted by $m_{v \to u}$, contains a vector that consists of the beliefs of a_v about all the states of a_u. The belief of a_v about the state s_u^i of a_u contained in $m_{v \to u}$ is calculated by:

$$m_{v \to u}(s_u^i) = max_{s_v^j}(uti(s_v^j) + g(s_v^j, s_u^i) + \sum_{a_p \in N_v \setminus a_u} m_{p \to v}(s_v^j)), \tag{5.14}$$

where $uti(s_v^j)$ is the unary utility of the state s_v^j, $g(s_v^j, s_u^i)$ is the pairwise utility between s_v^j and s_u^i, N_v is the neighbour set of a_v, and $m_{p \to v}(s_v^j)$ is the belief of a_p about the state s_v^j contained within the message passed from a_p to a_v.

The belief propagation converges when the belief values of all the agents about all their states remain the same with those in the previous iteration.

5.3.1.5 Task Allocation

Upon the convergence of belief propagation, task allocation is performed according to the convergence results. If belief propagation has not converged even for the first time till the task deadline arrives, task allocation fails. 'the first time' is emphasised since the situation could also be that the consumer re-operate belief propagation after belief propagation already converged to try to obtain a higher utility, due to the new arrivals of alternatives. If the results of the re-operation of belief propagation is not better than the already obtained supply chain, or the re-operation does not converge when task deadline arrives, the consumer adopts the already obtained chain obtained in the previous belief propagation. In this situation, task allocation is still considered successful. Upon the convergence of belief propagation, the alternatives, whose states are not *inactive*, form a supply chain. As has been analysed earlier, belief propagation may be in parallel operated on multiple independent parts caused by the decomposition. In this situation, the integrated supply chain is obtained through merging the sub-chains obtained in all the independent parts. When belief propagation is re-operated due to some changing in the environment, the scope where the re-operation should take place has been analysed earlier. It is notable that belief propagation may not be re-operated when some changing takes place. For example, the newly coming alternative will be pruned in the pruning phase before the start of belief propagation.

In order to make the whole procedure of task allocation clear, a task allocation procedure algorithm is given in Algorithm 5.2.

Algorithm 5.2: Task Allocation Procedure

It is assumed that the task to be allocated is $\mathcal{T} = \{t_1, t_2, ..., t_m\}$, the alternatives set is $\mathcal{A} = \{a_1, a_2, ..., a_n\}$, and A_i is the alternative set of t_i.

Pruning
for (i=1...n)
  if (a_i meets *Equation (5.11)* or *(5.12)*)
    prune a_i;
Decomposition
for (i=1...m-1)
  if ($w_{dep}(t_i, t_{i+1}) = 0$)
    decompose the network from the connection between t_i and t_{i+1};
Belief propagation is in parallel operated in the decomposed parts
Dealing with dynamism after the convergence of belief propagation
when(an alternative $a_i \in A_i$ or $a_{i+1} \in A_{i+1}$ leaves)
  belief propagation is locally re-operated in the part that concludes t_i or t_{i+1};
when(a new alternative $a_i^1 \in A_i$ or $a_{i+1}^1 \in A_{i+1}$ comes)
  to check whether a_i^1 or a_{i+1}^1 can be pruned
    if yes,
      prune a_i^1 or a_{i+1}^1;
    if no,
      to check whether the separation link between t_i and t_{i+1} is still valid;
        if yes,
          only the part where the changing happens needs to re-operate belief propagation;
        if no,
          the corresponding two parts are combined into one part, and belief propagation needs to be re-operated in the newly formed part;
Task allocation according to belief propagation result
Allocating subtasks upon and according to the convergence results of belief propagation.

In Algorithm 5.2, first of all, the alternatives that will never maximise the task allocation are pruned (**Lines 1-3**), then the pruned network is decomposed into more than one part from the separation links (if possible) (**Lines 4-7**). Assume that the connection between subtasks t_i and t_{i+1} is a separation link, and thus the network is separated between t_i and t_{i+1}. Subsequently, max-sum belief propagation is in parallel operated in the separated parts. When an alternative for t_i/t_{i+1} leaves, belief propagation only needs to be re-operated in the part that contains t_i/t_{i+1} (**Lines 8-10**). When a new alternative for t_i or t_{i+1} comes, it is checked whether the new alternative can be pruned (**Lines 11-14**). If the new alternative cannot be pruned, it needs to be checked whether the separation link between t_i and t_{i+1} is still valid due to the arrival of the new alternative (**Lines 15-17**). If the new alternative cannot be pruned and meanwhile, the separation link between t_i and t_{i+1} is still valid, belief propagation only needs to be re-operated in the part which the new alternative join (**Lines 18-20**); if the new alternative cannot be pruned and meanwhile, the separation link between t_i and t_{i+1} becomes invalid, the two parts that contain t_i and t_{i+1}, respectively, should be combined into one part, where belief propagation needs to be re-operated (**Lines 21-24**). After the belief propagation in all the parts converging, subtasks are allocated according to the results of the belief propagation (**Lines 25-26**).

5.3.2 Evaluation and Analysis

In this subsection, PD-LBP is experimentally evaluated through being compared with two similar max-sum belief propagation-based methods, LBP and PD-LBP.

5.3.2.1 Benchmarks

LBP, which is the method that PD-LBP is based on and tries to improve, is selected as one of the evaluation benchmarks. Penya-Alba *et al.* proposed a Reduced Binary Loopy Belief Propagation based method (RB-LBP) in [PAVCRA12]. Through encoding the TDN model into a binary factor graph, RB-LBP has been experimentally proved to outperform LBP in terms of the communication, computation, and memory requirements. Being an extension of LBP as well, it is necessary to compare PD-LBP with RB-LBP when PD-LBP is evaluated.

5.3.2.2 Evaluation Criteria and Settings

With the three goals of PD-LBP (i.e., **1)** the quick convergence of belief propagation, **2)** the quick response to and resilience from changing in dynamic environments, and **3)** small communication requirement) in mind, the evaluation criteria are correspondingly determined. Because it is hard to test the time used by belief propagation, the passed time from the arrival of a task to the successful allocation of the task is tested instead. This is reasonable in that in the evaluation, the allocation to a task starts as soon as the task arrives and is considered successful as soon as the belief propagation converges. The number of passed messages (N_{mes}) is tested to evaluate the communication requirement. In order to test the performance in terms of response to and resilience from changing, success rates of task allocation in the environments with various dynamism levels are examined.

To obtain satisfactory performance in the face of dynamism of the environment is an important motivation of PD-LBP, as a consequence, the evaluation (which is implemented using JAVA program) should be carried out with various dynamism levels (denoted as *dyn*) of the environments. It should be noted that the test bed is developed by the author but not any off-the-shelf product. In the program, 20 tasks and 40 resource providers are generated and stored in two different arrays. An indicator, a binary variable, is assigned to each task and each provider to simulate the state of the task and the provider. In particular, when the indicator variable of a provider/task is 1, it represents that the provider/task is in the environment; when the indicator variable is 0, it represents that the provider/task is not in the environment. The indicator variables of all the providers and tasks are initialised by 1 at the start of the evaluation. The dynamism and openness of the environment, i.e., the leaving and entering of tasks and providers, are simulated through changing the values of the indicators of the tasks and providers. In particular, it represents that a provider/task enters the environment when the indicator of the provider/task becomes 1 from 0, and leaves the environment when the indicator becomes 0 from 1. The value of *dyn* is the summation of the number of both the leaving and entering of all the providers and tasks per time unit (i.e., 100 *seconds*). For simplicity reason, the dynamism is classified into five levels listed in Table 5.13.

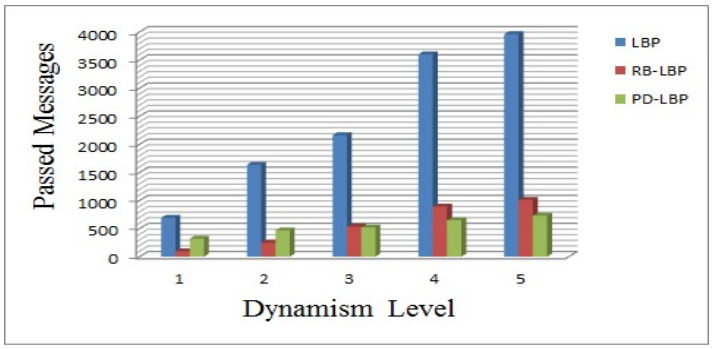
Table 5.13: Definition of Dynamism Level

Range	Dynamism Level
$(0, 2]$	1
$(2, 4]$	2
$(4, 6]$	3
$(6, 8]$	4
$(8, 10]$	5

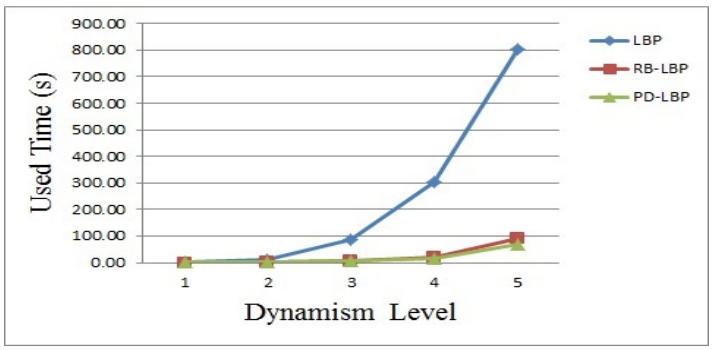
Since if and only if all the subtasks of a task are successfully allocated, the allocation of the task is considered as successful, the number of subtasks that a task consists of is an important factor that affects the evaluation results. Therefore, the evaluation will also be carried out based on various average number of subtasks per task. Moreover, due to the selfishness of agents in the environment and the competitiveness of resources, the number of alternative providers (referred as alternatives in the remainder of this chapter) for each subtask plays an important role and thus cannot be ignored to test the performance of PD-LBP. In addition, to examine the performances of PD-LBP based on various numbers of subtasks and alternatives per subtask is also to test the decomposition and pruning.

5.3.2.3 Experimental Results and Analysis

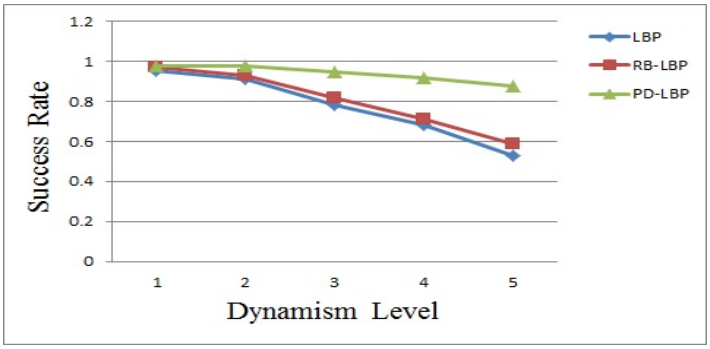
Experimental results are listed from Figure 5.14 to Figure 5.16.



(a) Passed Messages based on Dynamism Levels

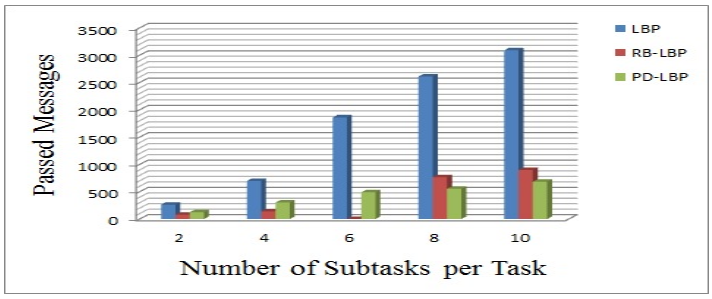


(b) Time Used based on Dynamism Levels

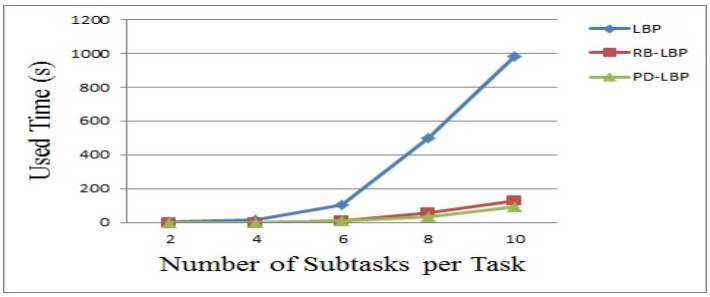


(c) Success rates based on Dynamism Levels

Figure 5.14: Performance Based on Dynamism Levels

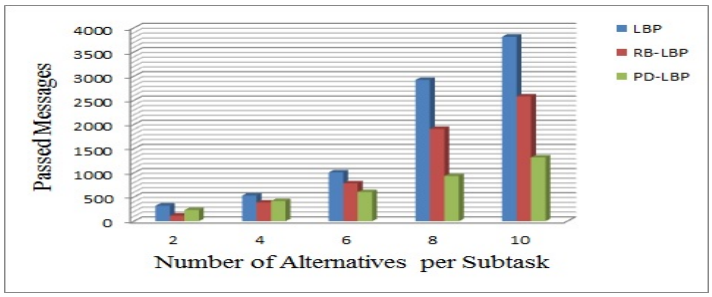


(a) Passed Messages based on Average Numbers of Subtasks per Task

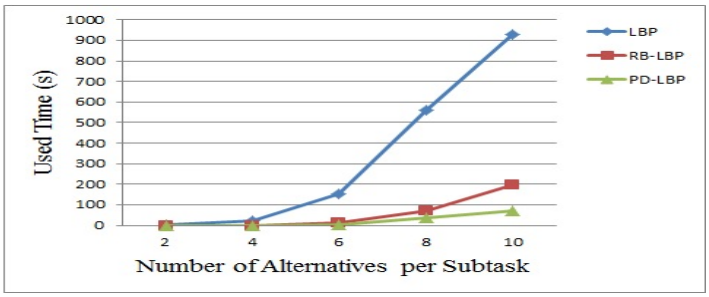


(b) Used Time based on Average Numbers of Subtasks per Task

Figure 5.15: Performance Based on Average Numbers of Subtasks per Task



(a) Passed Messages based on Average Numbers of Alternatives per Subtask



(b) Time Used based on Average Numbers of Alternatives per Subtask

Figure 5.16: Performance Based on Average Numbers of Alternatives per Subtask

From Figure 5.14 (a), Figure 5.15 (a), and Figure 5.16 (a), it can be seen that the passed messages of both RB-LBP and PD-LBP are fewer than that of LBP. The general reason for this is that RB-LBP mitigated the communication requirement of LBP through encoding the TDN model into a binary factor which added some factor nodes to make all the nodes binary (i.e., only has two states). The advantage of simplification (i.e., the pruning and decomposition) of PD-LBP becomes more obvious with the increase of the dynamism level. In more detail, when a changing takes place, the already converged belief propagation needs to be re-operated through the whole network in both LBP and RB-LBP. In opposite, PD-LBP narrows the scope that needs to re-operate the belief propagation through decomposition. This results in that the passed messages of PD-LBP does not increase as dramatically as those of LBP and RB-LBP when the number of subtasks per task increases, as can be seen from Figure 5.15 (a). As it is known, the number of cycles plays an important role in the quantity of message passing before the convergence of belief propagation. As a consequence, when there are more alternatives (i.e., more cycles) for each subtask, the advantage of pruning becomes stronger and caused the fewest passed messages in PD-LBP, as shown in Figure 5.16 (a). Due to the reduction of messages to be calculated and passed, the convergence of belief propagation is consequently accelerated. This can explain the shortest used time of PD-LBP when being compared with both LBP and RB-LBP, as shown in Figure 5.14 (b), Figure 5.15 (b), and Figure 5.16 (b).

The resilience from unpredicted changing is tested through examining the success rates of task allocation based on various dynamism levels of the environments. As can be seen from Figure 5.14 (c), the success rate of PD-LBP is always higher than those of LBP and RB-LBP. This owes to the quick convergence of belief propagation of PD-LBP, which is caused by pruning and decomposition, when belief propagation is frequently re-operated due to the frequent changing. Besides, the higher the dynamism level is, the more PD-LBP outperformed LBP and RB-LBP.

5.3.3 Discussion

Through the experimental evaluations to the two max-sum belief propagation-based task allocation methods, SLBP and PD-LBP proposed in this chapter, belief propagation was proved to be a powerful tool for task allocation with dependency constraints in open and dynamic environments. The reasons for this include **1)** the distribution of the belief propagation does not require any central controller, which can

meet the requirement of scalability of the optimisation, **2)** the concurrency of message passing can speed up the obtainment of the solution, and **3)** belief propagation can deal well with the dynamism including the coming and leaving of agents. Due to these reasons, when a task to be allocated requires too many resources, belief propagation-based methods can mitigate **1)** the expensive or even intractable computation requirement for the consumer to coordinate and adjust all the negotiation threads in multi-resource negotiations, and **2)** the NP-hard problem confronted by the combinatorial auction-based methods. Additionally, through devising the pruning and decomposition of belief propagation in PD-LBP, the performance of belief propagation was further improved in terms of both the speed to obtain the solution and the requirement of message passing, as can be seen in the evaluation in Subsection 5.3.2.

5.4 Summary

Two max-sum belief propagation-based task allocation methods, SLBP and PD-LBP, were proposed with different goals in this chapter, to achieve the **Objective 3** of this thesis.

The most distinguishing contribution of SLBP is the strategic quoting of resource providers during the belief propagation. The evaluation results demonstrated that SLBP achieved good performances in terms of success rate and efficiency of task allocation. Aiming at mitigating the communication requirement, accelerating the convergence of belief propagation, and improving the online response to and resilience from the unpredicted and constant changes in the environments, PD-LBP was also experimentally proved to be successful, as can be seen from the evaluation results.

Chapter 6

Conclusion

As grid/cloud environments have been developed into open and dynamic environments where both resource consumers and resource providers can enter and leave freely, the complexity of task allocation in such environments increases significantly due to the constant and unpredictable changes in the environments. Normally, the participants including both the providers and consumers in grid/cloud environments are administratively independent, with incomplete information about other participants. Consequently, task allocation in such environments have become a challenging problem from two main points. Firstly, task allocation methods/approaches must be able to handle the dynamism and openness of the environments where both the sets of resource providers and consumers can be changed unpredictably and constantly over time. Secondly, both the high scalability of grid/cloud environments and the local views of the participants request the decentralisation of task allocations in such environments.

6.1 Contributions of This Thesis

Focusing on task allocations in grid/cloud environments, this thesis mainly made the following contributions.

1. **A Negotiation-based Method for Single Task Allocation with Time Constraints in Grid/Cloud Environments**

   A negotiation-based method for the allocation of single tasks was proposed. In this method, both resource providers and consumers have only local views (incomplete information) about the environments. In order to maximise their own profits in the changing environments, negotiation parties (i.e., resource providers and consumers) strategically and dynamically make their offers (count-offers) by

taking multiple issues, such as the resource competition and the task deadline, into account. This distinguished the proposed method from most of the existing related task allocation methods. The evaluation results demonstrated that the proposed method outperformed two recent and well-known methods with various resource competition degrees, urgencies of task allocations, and numbers of required resources per task.

2. **An Auction-based Approach for Group Task Allocation in Grid/Cloud Environments**

   Aiming at allocating a task, which consists of a group of independent subtasks, a combinatorial auction-based approach was proposed. In the allocation of a group of independent subtasks, a consumer has to select a group providers which can collaboratively finish its task. Normally, selecting such a group of providers is a challenging problem, especially in open and dynamic environments. Against this, an indicator was devised to help the consumer to select the most suitable group. The important issues that the consumer is concerned with including the quoting prices, the finishing time of executing tasks, and the reputations of providers. These issues are encoded into the indicator. Evaluations were conducted to test the performance of the proposed approach. The evaluation results revealed that the proposed approach achieved desirable performances in terms of the success rate of task allocation, the individual utility distribution of the participants, the speed of the obtainment of the solution, and the scalability of the optimisation of task allocation.

3. **A Strategic Max-Sum Belief Propagation-based Method for Group Task Allocation with Constraints in Grid/Cloud Environments**

   Focusing on the allocation of a task which consists of more than one interdependent subtask with dependency constraints, a strategic max-sum belief propagation-based task allocation method (SLBP) was proposed. The resource providers in SLBP can strategically update their quoting prices during the belief propagation, according to the continuous changes in the environments. This is important in grid/cloud environments because normally, resource providers do not want to quote their truthful prices due to the selfish feature of them and the competitiveness of the environments, rather, providers intend to quote prices strategically to gain as many profits as possible. Through being compared with another max-sum

belief propagation-based task allocation method, in which the quoting prices of providers are fixed, in open and dynamic environments, SLBP were proved to achieve better performance with the strategic quoting element.

4. **A Belief Propagation-based Method for Group Task Allocation with Constraints in Grid/Cloud Environments**

   Another max-sum belief propagation-based method for group task allocation with dependency constraints was developed with a different objective from SLBP. This method focuses on the simplification of the belief propagation through decomposing and pruning the grid/cloud networks. The pruning phase could reduce the search space of solutions through pruning the providers, which will never optimise the task allocation. Therefore, the pruning phase could narrow the scope where belief propagation (message passing) was operated. The decomposition phase addresses decomposing the grid/cloud network into independent parts where belief propagation can be operated in parallel and thus converge quickly. The contributions of this method, which were proved by the experimental evaluation, include three points. First, the proposed method can mitigate both the communication and computation requirements for task allocation in the open and dynamic environments. Second, it can accelerate the convergence of belief propagation, i.e., to accelerate the obtainment of the solution. This is important to the task allocation under time constraints. Third, due to the characteristic of distribution of belief propagation, the performance of the online response to and resilience from the unpredicted changes of the environments can be improved.

6.2 Future Work

The future work includes the following three directions.

1. **Negotiation-based Task Allocation for Single Tasks in Grid/Cloud Environments**

   In the negotiation-based task allocation method for single tasks proposed in this thesis, the negotiation parties took multiple important issues into account when doing the offer (count-offer) decisions, such as the resource competition, the task deadline, and the reserve prices of the negotiation opponents. However, the negotiation parties are self-interested and thus they might not reveal such information

or their bottom lines to their negotiation opponents. Consequently, to learn to predict the information of opponents can help to improve the negotiation results. In the future work, learning mechanisms (e.g., Q-learning) will be utilised into the offer (count-offer) decisions of negotiation parties.

2. **Auction-based Task Allocation for a Group of Independent Tasks in Grid/Cloud Environments**

   In the allocation of a group of independent tasks in this thesis, a resource from one provider was considered comparable as an identical resource to that from another provider in terms of the utility provided by that resource. In reality, however, the utility provided by each resource may depend on the composition of resources selected. Therefore, utilities related to resource composition will be studied in more detail in the future work.

3. **Belief Propagation-based Task Allocation for a Group of Interdependent Tasks with Dependency Constraints in Grid/Cloud Environments**

   The two proposed max-sum belief propagation-based approaches solve the interdependent task allocation with dependency constraints with different objectives and from different perspectives. In fact, there is a large space to improve such a type of task allocation by taking the two objectives simultaneously. However, this will make the problem solving more complicated if providers are allowed to freely update their quotes and meanwhile the belief propagation has to be simplified. The reason for this is when providers can freely update their quotes, the pruning phase, which works through pruning the providers that will never maximise the task allocation in terms of the profit, will become more computationally expensive or even intractable. Consequently, how to solve this problem will be a challenging issue of the future research.

References

[ACSV04] Alvin AuYoung, Brent Chun, Alex Snoeren, and Amin Vahdat. Resource allocation in federated distributed computing infrastructures. In *Proc. of the 1st Workshop on Operating System and Architectural Support for the On-demand IT InfraStructure*, volume 9, 2004.

[AD10] James Atlas and Keith Decker. Coordination for uncertain outcomes using distributed neighbor exchange. In *Proc. of AAMAS2010*, pages 1047–1054, Toronto, Canada, 2010.

[Agh85] Gul A Agha. Actors: A model of concurrent computation in distributed systems. Technical report, DTIC Document, 1985.

[AGL09] Bo An, Nicola Gatti, and Victor Lesser. Bilateral bargaining with one-sided two-type uncertainty. In *Proc. of the International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 403–410, Milano, Italy, 2009.

[AKT05] Syed Ali, Sven Koenig, and Milind Tambe. Preprocessing techniques for accelerating the dcop algorithm adopt. In *Proc. of AAMAS2005*, pages 1041–1048, Utrecht, Netherlands, 2005.

[AL06] Sherief Abdallah and Victor Lesser. Learning the task allocation game. In *Proc. of AAMAS2006*, pages 850–857, Hakodate, Japan, 2006.

[AL10] Bo An and Victor Lesser. Characterizing contract-based multiagent resource allocation in networks. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 40(3):575–586, 2010.

[ALIZ10] Bo An, Victor Lesser, David Irwin, and Michael Zink. Automated negotiation with decommitment for dynamic resource allocation in cloud

computing. In *Proc. of AAMAS2010*, pages 981–988, Toronto, Canada, 2010.

[ALS08] Bo An, Victor Lesser, and Kwang Mong Sim. Decommitment in multi-resource negotiation. In *Proc. of AAMAS2008*, pages 1553–1556, Estoril, Portugal, 2008.

[ALS11] Bo An, Victor Lesser, and Kwang Mong Sim. Strategic agents for multi-resource negotiation. *Autonomous Agents and Multi-Agent Systems*, 23(1):114–153, 2011.

[ALWZ11] Bo An, Victor Lesser, David Westbrook, and Michael Zink. Agent-mediated multi-step optimization for resource allocation in distributed sensor networks. In *Proc. of AAMAS2011*, pages 609–616, Taipei, Taiwan, 2011.

[AM00] Srinivas M Aji and Robert J McEliece. The generalized distributive law. *IEEE Transactions on Information Theory*, 46(2):325–343, 2000.

[BHM97] Birgit Burmeister, Afsaneh Haddadi, and Guido Matylis. Application of multi-agent systems in traffic and transportation. In *Software Engineering. IEE Proceedings*, volume 144, pages 51–60. IET, 1997.

[Buy99] Rajkumar Buyya. High performance cluster computing: Architecture and systems, volume i. *Prentice Hall, Upper SaddleRiver, NJ, USA*, 1:999, 1999.

[Cas95] Cristiano Castelfranchi. Guarantees for autonomy in cognitive agent architecture. In *Intelligent agents*, pages 56–70. Springer, 1995.

[CB02] Madhu Chetty and Rajkumar Buyya. Weaving computational grids: How analogous are they with electrical grids?. *Computing in Science and Engineering*, 4(4):61–71, 2002.

[CBH09] Han-Lim Choi, Luc Brunet, and Jonathan P How. Consensus-based decentralized auctions for robust task allocation. *Robotics, IEEE Transactions on*, 25(4):912–926, 2009.

[CDGM97] Anthony Chavez, Daniel Dreilinger, Rob Guttman, and Pattie Maes. A real-life experiment in creating an agent marketplace. In *Software agents*

and soft computing towards enhancing machine intelligence, pages 160–179. Springer, 1997.

[CG10] Thomas E Carroll and Daniel Grosu. Formation of virtual organizations in grids: a game-theoretic approach. *Concurrency and Computation: Practice and Experience*, 22(14):1972–1989, 2010.

[Che04] Kath Checkland. National service frameworks and uk general practitioners: street-level bureaucrats at work?. *Sociology of health & illness*, 26(7):951–975, 2004.

[CLOa] PANDA CLOUD. www.cloudantivirus.com/.

[CLOb] CLOUDSECURITY. https://cloudsecurityalliance.org/.

[CLZB00] Henri Casanova, Arnaud Legrand, Dmitrii Zagorodnov, and Francine Berman. Heuristics for scheduling parameter sweep applications in grid environments. In *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*, pages 349–363. IEEE, 2000.

[CMKJ09] Archie C Chapman, Rosa Anna Micillo, Ramachandra Kota, and Nicholas R Jennings. Decentralised dynamic task allocation: a practical game-theoretic approach. In *Proc. of AAMAS*, pages 915–922, Budapest, Hungary, 2009.

[CP02] Christopher Crick and Avi Pfeffer. Loopy belief propagation as a basis for communication in sensor networks. In *Proceedings of the Nineteenth conference on Uncertainty in Artificial Intelligence*, pages 159–166, Acapulco, Mexico, 2002. Morgan Kaufmann Publishers Inc.

[CSJ+03] Junwei Cao, Daniel P Spooner, Stephen A Jarvis, Subhash Saini, and Graham R Nudd. Agent-based grid load balancing using performance-driven task scheduling. In *the International Symposium on Parallel and Distributed Processing*, pages 10–pp, Nice, France, 2003.

[DDG] DDGRID. http://www.ddgrid.ac.cn.

[DG05] Anubhav Das and Daniel Grosu. Combinatorial auction-based protocols for resource allocation in grids. In *19th IEEE International Symposium on Parallel and Distributed Processing*, pages 8–pp, Denver, USA, 2005.

[DKM+02] Ewa Deelman, Carl Kesselman, Gaurang Mehta, Leila Meshkat, Laura Pearlman, Kent Blackburn, Phil Ehrens, Albert Lazzarini, Roy Williams, and Scott Koranda. Griphyn and ligo, building a virtual data grid for gravitational wave scientists. In *11th IEEE International Symposium on High Performance Distributed Computing*, pages 225–234, Edinburgh, Scotland, 2002.

[Doo01] Kevin Dooley. *Designing Large-Scale LANs*. O'Reilly Media, Inc, USA, 2001.

[DSCB03] Daniel Paranhos Da Silva, Walfredo Cirne, and Francisco Vilar Brasileiro. Trading cycles for information: Using replication to schedule bag-of-tasks applications on computational grids. In *Euro-Par 2003 Parallel Processing*, pages 169–180. Springer, Klagenfurt, Austria, 2003.

[dWZK07] M. de Weerdt, Y. Zhang, and T. Klos. Distributed task allocation in social networks. In *Proc. of AAMAS*, page 76, 2007.

[Eba] Ebay. http://www.ebay.com.

[EBS] EBS. http://aws.amazon.com/ebs/.

[EC2] EC2. http://aws.amazon.com/ec2/.

[Edw77] Ward Edwards. How to use multiattribute utility measurement for social decisionmaking. *IEEE Transactions on Systems, Man and Cybernetics*, 7(5):326–340, 1977.

[EGI] EGI. http://www.egi.eu/projects/egi-inspire/.

[EXR+11] Neda Edalat, Wendong Xiao, Nirmalya Roy, Sajal K Das, and Mehul Motani. Combinatorial auction-based task allocation in multi-application wireless sensor networks. In *The 9th International Conference on Embedded and Ubiquitous Computing (EUC)*, pages 174–181, Melbourne, Australia, 2011.

[FCC+03] Yun Fu, Jeffrey Chase, Brent Chun, Stephen Schwab, and Amin Vahdat. Sharp: An architecture for secure resource peering. In *ACM SIGOPS Operating Systems Review*, volume 37, pages 133–148, 2003.

[FG97] Stan Franklin and Art Graesser. Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Intelligent agents III agent theories, architectures, and languages, Müller et al. (Eds.)*, pages 21–35. Springer, 1997.

[FGA+98] Richard F Freund, Michael Gherrity, Stephen Ambrosius, Mark Campbell, Mike Halderman, Debra Hensgen, Elaine Keith, Taylor Kidd, Matt Kussow, John D Lima, et al. Scheduling resources in multi-user, heterogeneous, computing environments with smartnet. In *Heterogeneous Computing Workshop*, pages 184–199, Orlando, USA, 1998.

[FJDSB+10] Paulo Roberto Ferreira Jr, Fernando Dos Santos, Ana LC Bazzan, Daniel Epstein, and Samuel J Waskow. Robocup rescue as multiagent task allocation among teams: experiments with task interdependencies. *Autonomous Agents and Multi-Agent Systems*, 20(3):421–443, 2010.

[FK10] Fangwen Fu and Ulas C Kozat. Wireless network virtualization as a sequential auction game. In *Proceedings of INFOCOM2010*, pages 1–9, San Diego, USA, 2010.

[FKT02] Ian Foster, C Kesselman, and S Tuecke. What is the grid?-a three point checklist. gridtoday,(6), july 2002, 2002.

[FM04] J Alexander Fax and Richard M Murray. Information flow and cooperative control of vehicle formations. *Automatic Control, IEEE Transactions on*, 49(9):1465–1476, 2004.

[FMlP96] Klaus Fischer, Jo¨ RG P Mu¨ ller, and Markus Pischel. Cooperative transportation scheduling: an application domain for dai. *Applied Artificial Intelligence*, 10(1):1–34, 1996.

[Fri09] Marc Eduard Frincu. Distributed scheduling policy in service oriented environments. In *Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 11th International Symposium on*, pages 205–212, Timisoara, Romania, 2009. IEEE.

[FRPJ08] Alessandro Farinelli, Alex Rogers, Adrian Petcu, and Nicholas R Jennings. Decentralised coordination of low-power embedded devices using

the max-sum algorithm. In *Proc. of AAMAS2008*, pages 639–646, Estoril, Portugal, 2008.

[FSJ98] Peyman Faratin, Carles Sierra, and Nick R Jennings. Negotiation decision functions for autonomous agents. *Robotics and Autonomous Systems*, 24(3):159–182, 1998.

[FSJ02] Peyman Faratin, Carles Sierra, and Nicholas R Jennings. Using similarity criteria to make issue trade-offs in automated negotiations. *Artificial Intelligence*, 142(2):205–237, 2002.

[GCL04] Aram Galstyan, Karl Czajkowski, and Kristina Lerman. Resource allocation in the grid using reinforcement learning. In *Proc. of AAMAS2004*, pages 1314–1315, NY, USA, 2004.

[GCL05] Aram Galstyan, Karl Czajkowski, and Kristina Lerman. Resource allocation in the grid with learning agents. *Journal of Grid Computing*, 3(1-2):91–100, 2005.

[GEN] GENI. http://www.geni.net/.

[GJDS07] E Gil Jones, M Bernardine Dias, and Anthony Stentz. Learning-enhanced market-based task allocation for oversubscribed domains. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2308–2313, San Diego, USA, 2007.

[GK94] Michael R Genesereth and Steven P Ketchpel. Software agents. *Commun. ACM*, 37(7):48–53, 1994.

[GLO] GLOBUS. http://www.globus.org/demogrid/.

[GM99] Robert H Guttman and Pattie Maes. Agent-mediated integrative negotiation for retail electronic commerce. In *Agent Mediated Electronic Commerce*, pages 70–90. Springer, 1999.

[GVRAC08] Andrea Giovannucci, Meritxell Vinyals, Juan A Rodriguez-Aguilar, and Jesus Cerquides. Computationally-efficient winner determination for mixed multi-unit combinatorial auctions. In *Proc. of AAMAS2008*, pages 1071–1078, Estoril, Portugal, 2008.

[HBH06] Jianwei Huang, Randall A. Berry, and Michael L. Honig. Auction-based spectrum sharing. *Mob. Netw. Appl.*, 11(3):405–418, June 2006.

[HPC] HPCLOUD. http://www.hpcloud.com/.

[ICG+06] David E Irwin, Jeffrey S Chase, Laura E Grit, Aydan R Yumerefendi, David Becker, and Ken Yocum. Sharing networked resources with brokered leases. In *Proc. of the USENIX Technical Conference*, pages 199–212, Boston, USA, 2006.

[JBPM00] Kyungkoo Jun, Ladislau Boloni, Krzysztof Palacz, and Dan C Marinescu. Agent-based resource discovery. In *Heterogeneous Computing Workshop, 2000.(HCW 2000) Proceedings. 9th*, pages 43–52, Cancun, Mexico, 2000. IEEE.

[JGN99] William E Johnston, Dennis Gannon, and Bill Nitzberg. Grids as production computing environments: the engineering aspects of nasa's information power grid. In *the Eighth International Symposium on High Performance Distributed Computing*, pages 197–204, Redondo Beach, CA, 1999.

[JMC+96] Nick R Jennings, EH Mamdani, Jose Manuel Corera, Inaki Laresgoiti, F Perriolat, Paul Skarek, and Laszlo Zsolt Varga. Using archon to develop real-world dai applications. 1. *IEEE Expert*, 11(6):64–70, 1996.

[JSW98] Nicholas R Jennings, Katia Sycara, and Michael Wooldridge. A roadmap of agent research and development. *Autonomous agents and multi-agent systems*, 1(1):7–38, 1998.

[KB+06] Ryszard Kowalczyk, Peter Braun, et al. Towards agent-based coalition formation for service composition. In *Proceedings of the IEEE/WIC/ACM international conference on Intelligent Agent Technology*, pages 73–80, Hong Kong, China, 2006.

[KC10] Hyun Soo Kim and Jae Hyung Cho. Supply chain formation using agent negotiation. *Decision Support Systems*, 49(1):77–90, 2010.

[KKL11] Yoonheui Kim, Michael Krainin, and Victor Lesser. Effective variants of the max-sum algorithm for radar coordination and scheduling. In

Proceedings of the 2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology-Volume 02, pages 357–364, Lyon, France, 2011.

[KN09] Tomas Klos and Bart Nooteboom. Adaptive learning in evolving task allocation networks. In *Proc. of AAMAS2009*, pages 465–472, Budapest, Hungary, 2009.

[KS+80] Ross Kindermann, James Laurie Snell, et al. *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI, 1980.

[KST03] Sarit Kraus, Onn Shehory, and Gilad Taase. Coalition formation with uncertain heterogeneous information. In *Proc. of AAMAS2003*, pages 1–8, Melbourne, Australia, 2003.

[KXK09] Joanna Kołodziej, Fatos Xhafa, and Łukasz Kolanko. Hierarchic genetic scheduler of independent jobs in computational grid environment. *Proc. of 23rd ECMS, Madrid*, pages 9–12, 2009.

[KZYL13] Yan Kong, Minjie Zhang, Dayong Ye, and Xudong Luo. A negotiation method for task allocation with time constraints in open grid environments. In *Proceedings of 6th International Workshop on Agent-based Complex Automated Negotiations*, Saint Paul, Minnesota, 2013.

[LBY03] Jonathan RT Lawton, Randal W Beard, and Brett J Young. A decentralized approach to formation maneuvers. *Robotics and Automation, IEEE Transactions on*, 19(6):933–941, 2003.

[LCLM+01] Lauri Loebel-Carpenter, Lee Lueking, Carmenita Moore, Ruth Pordes, Julie Trumbo, Sinisa Veseli, Igor Terekhov, Matthew Vranicar, Stephen White, and Victoria White. Sam and the particle physics data grid. In *Proceedings of Computing in High-Energy and Nuclear Physics*, Taipei, Taiwan, 2001. Citeseer.

[LFC+03] Oleg Lodygensky, Gilles Fedak, Franck Cappello, Vincent Neri, Miron Livny, and Douglas Thain. Xtremweb & condor: sharing resources between internet connected condor pool. In *Cluster Computing and the*

Grid, 2003. Proceedings. CCGrid 2003. 3rd IEEE/ACM International Symposium on, pages 382–389. IEEE, 2003.

[LJS+03] Xudong Luo, Nicholas R Jennings, Nigel Shadbolt, Ho-fung Leung, and Jimmy Ho-man Lee. A fuzzy constraint based model for bilateral, multi-issue negotiations in semi-competitive environments. *Artificial Intelligence*, 148(1):53–102, 2003.

[LL92] Magnus Ljungberg and Andrew Lucas. The oasis air traffic management system. In *Proceedings of the Second Pacific Rim International Conference on Artificial Intelligence*, Seoul, Koreal, 1992.

[LLC06] Liang-Teh Lee, Chin-Hsiian Liang, and Hung-Yuan Chang. An adaptive task scheduling system for grid computing. In *Proceedings of the Sixth IEEE International Conference on Computer and Information Technology*, pages 57–57, 2006.

[LLS06] Guoming Lai, Cuihong Li, and Katia Sycara. Efficient multi-attribute negotiation with incomplete information. *Group Decision and Negotiation*, 15(5):511–528, 2006.

[LLSG04] Guoming Lai, Cuihong Li, Katia Sycara, and Joseph Giampapa. Literature review on multi-attribute negotiations. *Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Technical Report CMU-RI-TR-04-66*, 2004.

[LRA+05] Kevin Lai, Lars Rasmusson, Eytan Adar, Li Zhang, and Bernardo A Huberman. Tycoon: An implementation of a distributed, market-based resource allocation system. *Multiagent and Grid Systems*, 1(3):169–182, 2005.

[LSL08] Guoming Lai, Katia Sycara, and Cuihong Li. A decentralized model for automated multi-attribute negotiations with incomplete information and general utility functions. *Multiagent and Grid Systems*, 4(1):45–65, 2008.

[M+94] Pattie Maes et al. Agents that reduce work and information overload. *Communications of the ACM*, 37(7):30–40, 1994.

[MAS+99] Muthucumaru Maheswaran, Shoukat Ali, HJ Siegal, Debra Hensgen, and Richard F Freund. Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems. In *Proceedings of the Eighth Heterogeneous Computing Workshop*, pages 30–44, 1999.

[MDKJ06] Efrat Manisterski, Esther David, Sarit Kraus, and Nicholas R. Jennings. Forming efficient agent groups for completing complex tasks. In *Proc. of AAMAS2006*, pages 834–841, Hakodate, Japan, 2006.

[MG14] Lena Mashayekhy and Daniel Grosu. A merge-and-split mechanism for dynamic virtual organization formation in grids. *IEEE Transactions on Parallel and Distributed Systems*, 25(3):540–549, 2014.

[MLT82] Perng-Yi Richard Ma, Edward Y. S. Lee, and Masahiro Tsuchiya. A task allocation model for distributed computing systems. *IEEE Transactions on Computers*, 31(1):41–47, 1982.

[MM87] R Preston McAfee and John McMillan. Auctions and bidding. *Journal of economic literature*, 25(2):699–738, 1987.

[MMB03] Alberto Montresor, Hein Meling, and Özalp Babaoğlu. Messor: Load-balancing through a swarm of autonomous agents. In *Agents and Peer-to-Peer Computing*, pages 125–137. Springer, 2003.

[MSRJ11] Kathryn S Macarthur, Ruben Stranders, Sarvapali D Ramchurn, and Nicholas R Jennings. A distributed anytime algorithm for dynamic task allocation in multi-agent systems. In *Proc. of AAAI*, pages 356–362, San Francisco, USA, 2011.

[MSTY05] Pragnesh Jay Modi, Wei-Min Shen, Milind Tambe, and Makoto Yokoo. Adopt: Asynchronous distributed constraint optimization with quality guarantees. *Artificial Intelligence*, 161(1):149–180, 2005.

[MWJ99] Kevin P Murphy, Yair Weiss, and Michael I Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, pages 467–475, Stockholm, Sweden, 1999. Morgan Kaufmann Publishers Inc.

[NITM02] Ranjit Nair, Takayuki Ito, Milind Tambe, and Stacy Marsella. Task allocation in the robocup rescue simulation domain: A short note. In *RoboCup 2001: Robot Soccer World Cup V*, pages 751–754. Springer, 2002.

[NJ06] Thuc Duong Nguyen and Nicholas R Jennings. Managing commitments in multiple concurrent negotiations. *Electronic Commerce Research and Applications*, 4(4):362–376, 2006.

[NPC+04] Timothy J Norman, Alun Preece, Stuart Chalmers, Nicholas R Jennings, Michael Luck, Viet D Dang, Thuc D Nguyen, Vikas Deora, Jianhua Shao, W Alex Gray, et al. Agent-based formation of virtual organisations. *Knowledge-based systems*, 17(2):103–111, 2004.

[OF08] Brammert Ottens and Boi Faltings. Coordinating agent plans through distributed constraint optimization. In *Proceedings of the ICAPS-08 Workshop on Multiagent Planning*, Sydney, Australia, 2008.

[PA13] Toni Penya-Alba. From supply chain formation to multi-agent coordination. In *Proc. of AAMAS2013*, pages 1447–1448, Saint Paul, Minnesota, 2013. International Foundation for Autonomous Agents and Multiagent Systems.

[PAVCRA12] Toni Penya-Alba, Meritxell Vinyals, Jesús Cerquides, and Juan Antonio Rodriguez-Aguilar. A scalable message-passing algorithm for supply chain formation. In *Proc. of AAAI2012*, pages 1436–1442, Toronto, Canada, 2012.

[PDJ+04] Laura Pearlman, Mike D'Arcy, Erik Johnson, Carl Kesselman, and Pawel Plaszczak. Neesgrid teleoperation control protocol (ntcp). *Report NEESgrid-2004*, 23, 2004.

[Pea88] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann, 1988.

[PFR09] Manisa Pipattanasomporn, Hassan Feroze, and S Rahman. Multi-agent systems in a distributed smart grid: Design and implementation. In *Proceedings of Power Systems Conference and Exposition*, pages 1–8, 2009.

[PKCD07] Sourav Pal, Sumantra R Kundu, Mainak Chatterjee, and Sajal K Das. Combinatorial reverse auction based scheduling in multi-rate wireless systems. *IEEE Transactions on Computers*, 56(10):1329–1341, 2007.

[PL] PLANET-LAB. http://www.planet-lab.org/.

[Rai82] Howard Raiffa. *The art and science of negotiation*. Harvard University Press, 1982.

[RAMN+97] Juan A Rodriguez-Aguilar, Francisco J Martín, Pablo Noriega, Pere Garcia, and Carles Sierra. Towards a test-bed for trading agents in electronic auction markets. *AI Communications*, 11(1):5–19, 1997.

[RBP+91] James Rumbaugh, Michael Blaha, William Premerlani, Frederick Eddy, William E. Lorensen, et al. *Object-oriented modeling and design*, volume 199. Prentice-hall Englewood Cliffs, 1991.

[RCJ09] Alex Rogers, Daniel D Corkill, and Nicholas R Jennings. Agent technologies for sensor networks. *IEEE Intelligent Systems*, 24(2):13–17, 2009.

[RCMA98] Ricardo J Rabelo, Luis M Camarinha-Matos, and Hamideh Afsarmanesh. Multiagent perspectives to agile scheduling. In *Intelligent Systems for Manufacturing*, pages 51–66. Springer, 1998.

[RFMJ10] Sarvapali D Ramchurn, Alessandro Farinelli, Kathryn S Macarthur, and Nicholas R Jennings. Decentralized coordination in robocup rescue. *The Computer Journal*, 53(9):1447–1461, 2010.

[RNI95] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25, 1995.

[RPF+10] Sarvapali D Ramchurn, Maria Polukarov, Alessandro Farinelli, Cuong Truong, and Nicholas R Jennings. Coalition formation with spatial and temporal constraints. In *Proc. of AAMAS2010*, pages 1181–1188, Toronto, Canada, 2010.

[RZS13] Fenghui Ren, Minjie Zhang, and Danny Sutanto. A multi-agent solution to distribution system management by considering distributed generators. *IEEE Transactions on Power Systems*, 28(2):1442–1451, 2013.

[SA09] Kwang Mong Sim and Bo An. Evolving best-response strategies for market-driven agents using aggregative fitness ga. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 39(3):284–298, 2009.

[SC01] Kwang Mong Sim and CY Choi. Foundation of market-driven agents: An adaptation of zeuthen's bargaining model. In *Proceedings of the 2nd International Conference on Intelligent Agent Technology*, pages 405–411, 2001.

[SC03] Kwang Mong Sim and Chung Yu Choi. Agents that react to changing market situations. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 33(2):188–201, 2003.

[SdSAB97] António Lucas Soares, Jorge Pinho de Sousa, Américo Lopes Azevedo, and João Augusto Bastos. Using an informal ontology in the development of a planning and control system-the case of the virtual enterprise. In *Re-engineering for Sustainable Industrial Production*, pages 107–118. Springer, 1997.

[Sen13] Sandip Sen. A comprehensive approach to trust management. In *Proc. of AAMAS2013*, pages 797–800, Saint Paul, Minnesota, 2013.

[SFOT05] Paul Scerri, Alessandro Farinelli, Steven Okamoto, and Milind Tambe. Allocating tasks in extreme teams. In *Proc. of AAMAS2005*, pages 727–734, Utrecht, Netherlands, 2005.

[SFRJ09] Ruben Stranders, Alessandro Farinelli, Alex Rogers, and Nick Jennings. Decentralised coordination of mobile sensors using the max-sum algorithm. In *Proc 21st Int. Joint Conf on AI (IJCAI)*, pages 299–304, Pasadena, USA, 2009.

[Sim06] Kwang Mong Sim. Grid commerce, market-driven g-negotiation, and grid resource management. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 36(6):1381–1394, 2006.

[Sim09] Kwang Mong Sim. Agent-based cloud commerce. In *Proceedings of the IEEE International Conference on Industrial Engineering and Engineering Management*, pages 717–721, Hong Kong, China, 2009.

[Sim10] Kwang Mong Sim. Grid resource negotiation: survey and new directions. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 40(3):245–257, 2010.

[SK98] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[SK05] David Sarne and Sarit Kraus. Solving the auction-based task allocation problem in an open environment. In *Proceedings of the National Conference on Artificial Intelligence*, volume 20, page 164, 2005.

[SK11] Naidila Sadashiv and SM Dilip Kumar. Cluster, grid and cloud computing: a detailed comparison. In *6th International Conference on Computer Science & Education*, pages 477–482, Singapore, 2011.

[SP11] Adrian Schoenig and Maurice Pagnucco. Evaluating sequential single-item auctions for dynamic task allocation. In *AI 2010: Advances in Artificial Intelligence*, pages 506–515. Springer, Adelaide, Australia, 2011.

[SRG+00] Ben Segal, Les Robertson, Fabrizio Gagliardi, Federico Carminati, and G Cern. Grid computing: The european data grid project. In *IEEE Nuclear Science Symposium and Medical Imaging Conference*, volume 1, page 2, 2000.

[TPVS07] Christina Theocharopoulou, Ioannis Partsakoulakis, George A Vouros, and Kostas Stergiou. Overlay networks for task allocation and coordination in dynamic large-scale networks of cooperative agents. In *Proc. of AAMAS2007*, page 55, Hawai'i, USA, 2007.

[VJ00] Nir Vulkan and Nicholas R Jennings. Efficient mechanisms for the supply of services in multi-agent environments. *Decision Support Systems*, 28(1):5–19, 2000.

[VWE+86] Detlof Von Winterfeldt, Ward Edwards, et al. *Decision analysis and behavioral research*, volume 604. Cambridge University Press Cambridge, 1986.

[WC10] Michael Winsper and Maria Chli. Decentralised supply chain formation: a belief propagation-based approach. *Agent-Mediated Electronic Commerce*, page 1, 2010.

[WC13] Michael Winsper and Maria Chli. Decentralized supply chain formation using max-sum loopy belief propagation. *Computational Intelligence*, 29(2):281–309, 2013.

[WJ+95] Michael Wooldridge, Nicholas R Jennings, et al. Intelligent agents: Theory and practice. *Knowledge Engineering Review*, 10(2):115–152, 1995.

[WW03] William E Walsh and Michael P Wellman. Decentralized supply chain formation: A market protocol and competitive equilibrium analysis. *Jouranl of Artificial Intellegence Research*, 19:513–567, 2003.

[WWW98] Peter R Wurman, Michael P Wellman, and William E Walsh. The michigan internet auctionbot: A configurable auction server for human and software agents. In *Proceedings of the Second International Conference on Autonomous Agents*, pages 301–308, Minneapolis, USA, 1998.

[WWY00] William E. Walsh, Michael P. Wellman, and Fredrik Ygge. Combinatorial auctions for supply chain formation. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, EC '00, pages 260–269, New York, NY, USA, 2000.

[WYM12] Yair Weiss, Chen Yanover, and Talya Meltzer. Map estimation, linear programming and belief propagation with convex free energies. *arXiv preprint arXiv:1206.5286*, 2012.

[YL07] Jingan Yang and Zhenghu Luo. Coalition formation mechanism in multiagent systems based on genetic algorithms. *Applied Soft Computing*, 7(2):561–568, 2007.

[YS02] Bin Yu and Munindar P Singh. An evidential model of distributed reputation management. In *Proc. of AAMAS2002*, pages 294–301, Bologna, Italy, 2002.

[YZS11] Dayong Ye, Minjie Zhang, and Danny Sutanto. A hybrid multiagent framework with q-learning for power grid systems restoration. *IEEE Transactions on Power Systems*, 26(4):2434–2441, 2011.

[YZS13] Dayong Ye, Minjie Zhang, and Danny Sutanto. Self-adaptation-based dynamic coalition formation in a distributed agent network: a mechanism and a brief survey. *IEEE Transactions on Parallel and Distributed Systems*, 24(5):1042–1051, 2013.

[ZG13] Sharrukh Zaman and Daniel Grosu. Combinatorial auction-based allocation of virtual machine instances in clouds. *Journal of Parallel and Distributed Computing*, 73(4):495–508, 2013.

[ZK08] Xiaoming Zheng and Sven Koenig. Reaction functions for task allocation to cooperative agents. In *Proc. of AAMAS2008*, pages 559–566, Estoril, Portugal, 2008.

[ZL05] Dayi Zhou and Virginia Lo. Wave scheduler: Scheduling for faster turnaround time in peer-based desktop grid systems. In *Job Scheduling Strategies for Parallel Processing*, pages 194–218. Springer, 2005.