UNIVERSITY OF SOUTHAMPTON

Coalition Formation and Operation in Virtual Organisations

by

Viet Dung Dang

A thesis submitted in partial fulfillment for the degree of Doctor of Philosophy

in the

Faculty of Engineering, Science and Mathematics

School of Electronics and Computer Science

December 2004

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ENGINEERING, SCIENCE AND MATHEMATICS

SCHOOL OF ELECTRONICS AND COMPUTER SCIENCE

Doctor of Philosophy

by Viet Dung Dang

The concept of Virtual Organisations (VOs) or Virtual Enterprises (VEs) is rapidly emerging as an important topic in many areas of computing including e-commerce, grid computing and the semantic Web. One reason for this interest is that VOs provide a means of bringing together a number of autonomous stakeholders in a dynamic fashion in order to address a specific problem or niche. These agents then work together for some period of time and then disband when it is deemed appropriate to do so. There are, however, many technical, social and economic issues associated with this VO lifecycle (i.e. creation, operation, maintenance and dissolution) that need to be addressed before VOs can be considered to be practicable. While previous technical work on VOs has concentrated on providing tools to support different aspects of the VO lifecycle, comparatively little work has focused on the mechanisms for automated VO creation, operation and maintenance. To address this shortcoming, this research aims to study and design mechanisms for the VO creation, operation and maintenance phases. In this thesis, our approach is to use combinatorial auctions and coalition formation mechanisms. In particular, novel algorithms for clearing multi-unit single-item and multi-unit combinatorial auctions have been developed as a means of tackling VO creation and one part of VO maintenance. A novel algorithm for coalition structure generation has also been developed to address VO operation and another part of VO maintenance.

Contents

Acknowledgements vi

1 Introduction 1
- 1.1 Virtual Organisations . . . 2
- 1.2 Research Objectives . . . 5
  - 1.2.1 The Creation Phase . . . 7
  - 1.2.2 The Operation Phase . . . 9
  - 1.2.3 The Maintenance Phase . . . 10
- 1.3 Research Contributions . . . 11
- 1.4 Thesis Structure . . . 12

2 Background 14
- 2.1 Virtual Organisations . . . 14
- 2.2 Partner Selection in Virtual Organisations . . . 19
  - 2.2.1 Auction Clearing with Demand/Supply Function Bids . . . 21
  - 2.2.2 The State of The Art in Clearing Algorithms . . . 24
- 2.3 Task Distribution within Virtual Organisations . . . 26
  - 2.3.1 Coalition Structure Generation . . . 29
  - 2.3.2 The State of The Art in Coalition Formation . . . 30
- 2.4 Summary . . . 31

3 Polynomial Auction Clearing Algorithms 32
- 3.1 Multi-Unit Single-Item Auctions . . . 34
  - 3.1.1 Proof of NP-completeness . . . 34
  - 3.1.2 The Algorithm . . . 41
- 3.2 Multi-Unit Combinatorial Auctions . . . 48
- 3.3 Experimental Evaluation . . . 60
  - 3.3.1 Multi-Unit Single-Item Auctions . . . 60
  - 3.3.2 Multi-Unit Combinatorial Auctions . . . 65
- 3.4 Summary . . . 70

4 Optimal Auction Clearing Algorithms 72
- 4.1 Piece-wise Linear Supply/Demand Curve Bids . . . 73
  - 4.1.1 Multi-Unit Single-Items . . . 74
  - 4.1.2 Multi-Unit Combinatorial Items . . . 80
- 4.2 Monotonic One-Unit-Difference Supply/Demand Functions . . . 84
  - 4.2.1 Multi-Unit Single-Items . . . 84

4.2.2 Multi-Unit Combinatorial Items . . . 90
4.3 Summary . . . 93

5 Coalition Structure Generation Algorithm 94
5.1 The Algorithm . . . 94
5.2 Performance Evaluation . . . 100
5.3 Summary . . . 106

6 Virtual Organisations in Operation 107
6.1 The Creation Phase . . . 108
6.2 The Operation Phase . . . 114
6.3 The Maintenance Phase . . . 119
6.3.1 Adding New Members into The VO . . . 119
6.3.2 Re-organising The Work . . . 124

7 Conclusions 128

List of Figures

1.1 The VO lifecycle. 6

2.1 A demand (supply) curve for multi-unit single-item case. 22
2.2 A demand (supply) function for multi-unit combinatorial case. 23

3.1 The clearing algorithm for the multi-unit single-item case. 41
3.2 The clearing algorithm for the multi-unit combinatorial case. 53
3.3 The experimental result of our algorithm for multi-unit single-item forward auctions (varying the number of bidders). 63
3.4 The experimental result of our algorithm for multi-unit single-item forward auctions (varying the maximum number of segments). 63
3.5 The experimental result of our algorithm for multi-unit single-item reverse auctions (varying the number of bidders). 64
3.6 The experimental result of our algorithm for multi-unit single-item reverse auctions (varying the maximum number of segments). 64
3.7 The experimental result of our algorithm for multi-unit combinatorial forward auctions (varying the number of bidders). 67
3.8 The experimental result of our algorithm for multi-unit combinatorial forward auctions (varying the maximum number of segments). 68
3.9 The experimental result of our algorithm for multi-unit combinatorial forward auctions (varying the number of items). 68
3.10 The experimental result of our algorithm for multi-unit combinatorial reverse auctions (varying the number of bidders). 69
3.11 The experimental result of our algorithm for multi-unit combinatorial reverse auctions (varying the maximum number of segments). 69
3.12 The experimental result of our algorithm for multi-unit combinatorial reverse auctions (varying the number of items). 70

4.1 Clearing algorithm for multi-unit single-item case with piece-wise linear supply function bids. 79
4.2 Clearing algorithm for multi-unit combinatorial case with piece-wise linear supply function bids. 83
4.3 Clearing algorithm for multi-unit single-item case with monotonic one-unit-difference demand/supply function bids. 88
4.4 Clearing algorithm for multi-unit combinatorial case with monotonic one-unit-difference demand/supply function bids. 92

5.1 The coalition structure generation algorithm. 96
5.2 Comparison of the searching paths between our algorithm and Sandholm et al.'s. 97

5.3 Sandholm et al.'s algorithm. 101
5.4 The case $\mathbf{n = 50}$. 103
5.5 The case $\mathbf{n = 100}$. 104
5.6 The case $\mathbf{n = 500}$. 104
5.7 The case $\mathbf{n = 1000}$. 105

6.1 The searching steps of our coalition structure generation in the operation phase of the scenario. 118

Acknowledgements

During the duration of my Ph.D., I have received lots of advice, support and help from many people.

In particular, I would like to express my deepest appreciation and gratitude to my supervisor Prof. Nicholas R. Jennings for his continuously great supervision and support. His dynamism, experience and flexibility have also inspired, encouraged and taught me in numerous ways.

I gratefully thank BT Exact for kindly financing my research. I would particularly thank Dr. Simon Thompson for his advice and his support in my time in BT Exact as well as throughout my Ph.D. Moreover, the research has been done in the scope of the CONOISE project and I would also like to thank all CONOISE's members for their help.

I thank Dr David Parkes for giving me useful comments on my auction clearing paper.

The Intelligent, Agents, Multimedia group community has been extremely supportive of me academically, technically as well as socially. I owe them all. Special thanks to Dr. Stephan Chan, Dr. Alex Rogers, Dr. Royan Ong, Dr. Esther David, Rajdeep Dash, Sarvapali (Gopal) Ramchurn and Steve Munroe for all your help.

I thank my parents and my sister for all the moral support that they have continuously provided me remotely from Vietnam and France. I also thank them for giving me a great environment to grow up in.

Last but certainly not least I thank my girlfriend Vinh Hanh Nguyen for all the help and support she has given me during my Ph.D. time. Vinh made this time joy, more meaningful and well-rounded. She has always been there for me. Thank you.

This thesis is dedicated to my parents, Giang Huong and Hanh Vinh.

Chapter 1

Introduction

An increasing number of computer systems are being developed to operate in open, networked environments such as the Internet and the Grid. Moreover, in many such cases, these systems are populated with independent components that have been developed by different stakeholders, each of which has their own aims and objectives. To achieve these aims and objectives, the components invariably need to interact with one another in flexible ways. In particular, a key form of interaction is when a number of initially distinct components come together to form a temporary alliance (or *virtual organisation*) to achieve a particular objective.

Against this background, this research develops new methods for forming, maintaining, and managing virtual organisations (VOs). Specifically, the independent components are viewed as *autonomous agents* that can act in flexible ways in order to provide and deliver particular services [Jennings and Wooldridge, 1995]. Then, when the need for a new VO is detected, these agents participate in online auctions in order to indicate what contributions they are willing to make (if any) and under what terms and conditions. Here these terms and conditions relate to the other services that the agent is contributing to the VO and the quantity of its contribution of the various services (e.g. if an agent is contributing many different services or a large number of a particular service then the unit price for the service may be less than if it has a smaller role). However, developing auctions that can express such relationships is a challenging task and has required the development of new algorithms to determine which sets of agents and bids should be

selected[1]. Having determined the participants of the VO and their various contributions, specific tasks need to be allocated to the individual participants. Again this is a complex decision making task (because it needs to find an efficient allocation among a very large number of possible allocations) and has required the development of new algorithms for achieving this in an optimal fashion. Finally, in the types of environment in which VOs are most useful, there is likely to be significant degrees of dynamism as new tasks are added and existing ones are removed or modified. To cope with this, the aforementioned algorithms have been developed in such a way that they can also re-configure the VO once it is operational. In short, therefore, this thesis is concerned with developing efficient mechanisms to automate the creation, operation and maintenance of the VO lifecycle.

In more detail, this first chapter gives a brief overview of the virtual organisation research area and sets the basic background for the research developed in this thesis. In particular, section 1.1 introduces the field of virtual organisations and reviews some significant projects that have been undertaken in this area. Building on this, section 1.2 states the specific objectives of this research and section 1.3 details its main contributions. Finally, section 1.4 outlines how the remainder of the thesis is organised.

1.1 Virtual Organisations

The concept of Virtual Organisations (VOs) is rapidly emerging as an important topic of research in many areas of computing. It is becoming so important because ever more open distributed systems are being developed and, in such cases, VOs provide a means for related entities to band together to deliver services that no one single component can provide. In particular, this thesis concentrates on the use of VOs in the context of e-business, although the technologies developed in this work are more widely applicable (see section 1.3 for more details).

In e-commerce, vigourous competition, as well as a fast-changing business environment, forces companies to focus on their core competences, keep a high degree of flexibility,

[1]This kind of algorithm is called an *auction clearing algorithm* or a *winner determination algorithm*.

and collaborate with other companies to enhance their competitive advantages to ensure their very survival [Beer *et al.*, 1990]. Moreover, creating added value for the customer is becoming an increasingly complex process that involves the combination of a great many different types of knowledge that the separate organisations do not necessarily possess. Therefore, many have come to understand that the key to competitive advantage is to transform the way they function [Beer *et al.*, 1990]. In particular, it is recognised that firms should not operate in isolation, but, in fact, their success depends on the relationships with different parties including competitors, complementors (horizontal relationships) and buyers and suppliers (vertical relationships) [Porter, 1980].

Now, in a market characterised by rigorous competition, one way to succeed is to collaborate with these related parties to promote synergies via increasing market power, lessening competition, specialisation and economy of scale (advantages gained from high output production)) [Contractor and Lorange, 1988]. As a consequence of this, the Virtual Organisation (VO) model — viewed as a "temporary consortium or alliance of individual/organisations formed to share costs and skills and exploit fast-changing opportunities" (adapted from [NIIIP, 1998]) — is becoming ever more important [NIIIP, 1998]. By means of an example, consider the situation in which a number of media providers (e.g. news, movies and music providers), mobile operators, and mobile handset manufacturers come together to make a VO to provide advanced customised multimedia services for fourth-generation (4G) mobile phone users [Norman *et al.*, 2003] [Norman *et al.*, 2004]. With this service, a mobile phone user can order a customised combination of movies, news and music and get them sent to his/her mobile phone. This is valuable because for the companies, they can increase their competitiveness by providing this new service, while saving costs via specialisation.

In more detail, the VO model offers several potential superior advantages over the independent organisation and traditional collaborative models (such as mergers, acquisitions and joint ventures).

We will start with the traditional advantages of a collaborative relationship over the independent organisation (as such, they apply both for the VO model and traditional collaborative models). First, it offers opportunities to improve productivity, efficiency

and optimises resources through specialisation [NIIIP, 1998]. While these advantages are typical of a collaborative relationship over an independent organisation, they are better realised in the VO model because the VO members focus on their core competences only [Sieber and Griese, 1998]. In the 4G mobile phone example above, for instance, the mobile operators would be better off not manufacturing the mobile handsets by themselves, but rather they should leave this to the handset manufacturer and concentrate on their core competences of providing mobile network coverage. Secondly, a VO enables its members to provide new services that they themselves cannot deliver. For example, none of the individual companies in the 4G mobile phone example can provide the combined service (customised multimedia service) by itself. Thus, it allows companies to access new markets with competitive solutions [Hardwick and Bolton, 1997]. This also increases the competitiveness of small and medium enterprises (SMEs) because SMEs usually cannot provide a wide range of services as they have very limited resource, compared to large multi-national companies. This is especially true given the current situation in which SMEs are usually subcontracted by a large company, which makes them very dependent on the contracting company [Neubert *et al.*, 2001]. Meanwhile, for customers, it means they can have combined services conveniently with one point of contact — the VO — instead of having to contact different companies for different parts of the combined services.

Beside the aforementioned traditional advantages of a collaborative relationship over the independent organisation, the VO model also offers additional advantages over traditional collaborative models. First, it provides a great deal of agility in that the nature of the VO can be continually adjusted according to the prevailing market context. Thus, it enables a rapid response to changes and opportunities in the dynamic business environment [Sieber and Griese, 1998]. For example, in the 4G mobile phone scenario above, the VO can adjust its structure (add/remove members) to cope with changes in the business environment (for instance, they can add more members into the VO to provide additional service when there is a demand for it). Second, the VO members retain their entrepreneurial independence. For example, in the 4G mobile phone scenario, the media providers, network coverage providers and handset providers are still independent firms after joining the VO. This is different from mergers and acquisitions,

in which it is usually the case that the joining firms no longer exist as independent firms. For example, after the HP - Compaq merger, Compaq no longer exists. Entrepreneurial independence is desirable for a firm's owners and/or managers, because they can keep the control of their firm and the entrepreneurial identity is retained. Third, the VO partners should be able to unite quickly without lengthy negotiations and disband without any problems. Traditional collaborative models, for example, mergers and acquisitions as well as joint ventures, require lengthy and very costly procedures (especially the costs for financial adviser firms and legal adviser firms), for alliance creation and dissolution. This is because, for example, in the case of mergers and acquisitions, firms are merged permanently and so require a lot more time and cost in terms of legal and financial advising.

To fully realise these benefits, however, it is necessary to make extensive use of a range of information technologies. Thus there is starting to be considerable research in this area (see chapter 2 for more details). However, although these projects provide a variety of tools to support different aspects of the VO lifecycle, the degree of automation provided is still somewhat rudimentary. In particular, existing systems typically provide the available information to humans who are actually responsible for the decision making. However this is a lengthy and time consuming process that could be made significantly faster and more efficient if it was automated. Thus this research develops various algorithms that will help automate the creation, operation and maintenance of VOs.

1.2 Research Objectives

There are many technical advances that need to be made before the VO lifecycle can be fully automated. These can be organised according to the VO lifecycle, which is composed of creation, operation, maintenance[2] and dissolution (see figure 1.1).

In more detail, the *creation* phase involves one or more of the entities coming to believe that it might be worthwhile to create a VO. This entity then contacts a number of

[2]The maintenance phase is also called the "modification phase" in [Camarinha-Matos and Afsarmanesh, 1999].

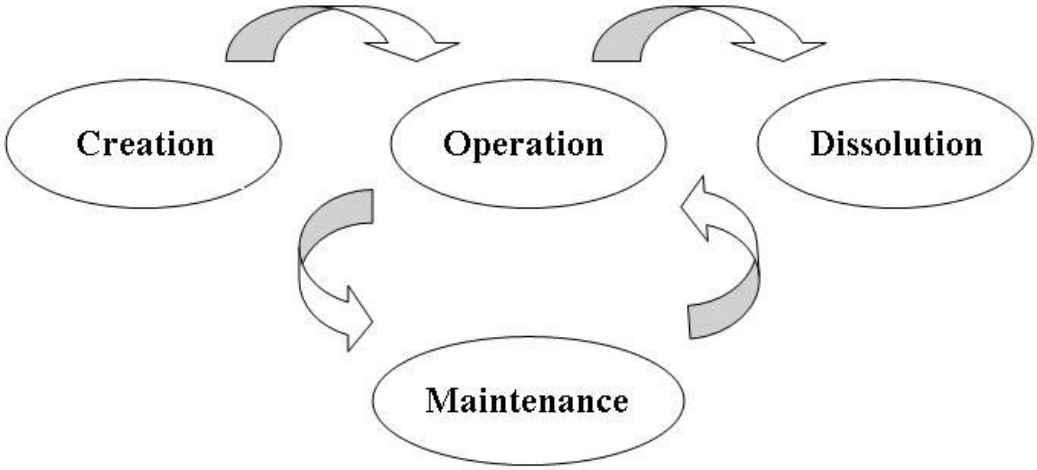


FIGURE 1.1: The VO lifecycle.

potential participants to determine whether they would be willing to join the VO, and, if it is successful, this will establish a group that are willing to work together in the context of the VO. The *operation* phase occurs once the VO has been created and is concerned with the way tasks are decomposed and distributed between the participants. The *maintenance* phase occurs when the VO structure (the members and the agreed task distribution) need to be changed. This can happen for a number of reasons including the failure in carrying out the contract of a VO member, or the bankruptcy of a VO member, or some change in the business environment. Finally, the *dissolution* phase occurs when the VO is disbanded because it is no longer deemed effective. This may happen because the combined service that the VO provides is obsolete, no longer needed, or because the VO is no longer making profit.

In the remainder of this section, the main research issues associated with each phase are detailed and the particular focus of this thesis is given. The thesis does not deal with the dissolution phase because the scope for automation in this phase is somewhat limited. In particular, this phase mostly concerns legal aspects or a technical analysis of the VO's performance and VO members' performance.

Underpinning all the phases, however, is the view that the distinct entities in the system are represented as software agents [Jennings and Wooldridge, 1995]. Here a software agent can be viewed as an autonomous software entity that is capable of acting flexibly

in a changing environment [Jennings and Wooldridge, 1995]. These agents are capable of providing one or more services in the VO and in so doing each agent is assumed to be interested in maximising their individual gain (when it wants to join the VO) or the VO gain (when it is a member of the VO). Software agents were chosen as the basic representation because they are well-suited for environments that are open, changeable, complex, with decentralised control [Jennings and Wooldridge, 1995] — which is the type of environment that VOs operate on.

1.2.1 The Creation Phase

In the creation phase, the main research issues are identification of needs, enterprise capability representation, partner search and partner selection mechanisms (see section 2.1 for more details). This research will focus on partner selection mechanism issue because it is arguably the most important step in this phase.

Partner selection occurs once the VO initiator has identified the task that needs to be solved, as well as the skills and capacities needed from the prospective members of the VO. In this context, the key issue is to determine what mechanism should be used to select the best partners for the VO. This is arguably the most critical step in the creation phase because choosing the appropriate partners is central to the success of the VO, while making the wrong choice can lead to a poorly performing VO. There are several requirements that need to be met by this process:

- The most suitable set of partners from those that are available should be selected. In this context, most suitable means the ones with the lowest cost of providing the services. The *cost* here does not only mean the monetary value of the services but may be a combined rating value, calculated from monetary value and other attributes of the services offered by the partners (e.g. delivery time) so that the cost can be considered more accurately and thoroughly. For example, in the 4G mobile phone example, the cost of the network coverage service provided by a mobile operator does not mean monetary value but may be calculated by combining this value with the quality rating of the network coverage.

- The selection should occur within a computationally reasonable time frame so that the market niche can be exploited as it becomes available. For example, the VO in the 4G mobile phone example needs to be setup quickly before other competitors recognise the niche.

- The potential partners should be able to vary their potential involvement in the VO. This is because this flexibility will allow more potential partners to join the selection process and so should lead to a better formed VO. Thus, for example, a partner may be willing to complete services more cheaply if it has a high degree of involvement in the VO (because the intrinsic costs can be depreciated over many instances). In contrast, if an agent has a comparatively small involvement then the unit cost may be much higher. For example, in the 4G mobile phone example, media providers should be able to vary the number of services, as well as the quantity of each service, that they will potentially provide for the VO.

There have been several approaches to this problem (e.g. using auctions or utilising mobile agents). However, they either do not provide a sufficient degree of automation or they do not give the VO's potential partners enough flexibility to vary their potential involvement in the VO (see subsection 2.1.1 for more details). Thus, given the open and competitive nature of the environment, we believe this creation process is best achieved using some form of marketplace structure, in particular, using some form of *auction* (an auction is a market institution with an explicit set of rules determining resource allocation and prices on the basis of bids from the market participants [McAfee and McMillan, 1987]). Markets are chosen because they provide a highly effective structure for allocating resources in situations in which there are many self-interested and autonomous stakeholders.

There are, however, many different types of auction (see [Wurman, 2001] for a classification) but in this work it was decided to adopt a *combinatorial auction* approach. A combinatorial auction is a sophisticated type of auction where multiple units of multiple (potentially inter-related) items are traded simultaneously (if there is only a single unit of each type of item, the auctions are called *single-unit combinatorial auctions*, whereas

if there are multiple units of each type of item, the auctions are called *multi-unit combinatorial auctions* [Sandholm *et al.*, 2002]). This particular type of auction is suitable for this problem because it provides the potential partners with a high degree of flexibility in expressing their requirements. Thus, in the 4G mobile scenario, for example, media providers can vary the number of services, as well as the quantity of each service that they intend to provide for the VO, in their bids. For instance, a media provider can make a bid of providing 10 movies/month and 5 songs/day for a total price of 50. No other type of auction allows such flexibility.

However, the main disadvantages of combinatorial auctions stem from the lack of computationally tractable **clearing algorithms**, that is, algorithms for determining the prices, quantities and trading partners as a function of the bids made.[3] Without such algorithms, combinatorial auctions are not really practicable because the time it takes to determine the winners is exponential [Sandholm *et al.*, 2002]. This means there may be unacceptable delays for auctions that have only a medium number of participants or items. Thus, a large portion of the research is devoted to developing tractable clearing algorithms for combinatorial auctions that determine the set of winners in a sub-optimal way (e.g. the solution is within a finite bound of the optimal) [4]. However, for auctions those have a small number of participants and a small number of items, it is possible to determine the optimal set of winners even with exponential algorithms.[5] Thus, the thesis also develops optimal clearing algorithms for combinatorial auctions that apply for wide classes of bidding functions.

1.2.2 The Operation Phase

In the operation phase, the main research issues are task distribution mechanism and partner collaboration support tools(see section 2.1 for more details). This research will focus on task distribution mechanism because although it is one of the deciding factors for the success of the VO, it has been largely neglected in the literature.

[3]The clearing problem in auctions is also called the *winner determination problem* [Sandholm *et al.*, 2002] or the *bid evaluation problem* [Eso *et al.*, 2001].

[4]It has been shown that it is impossible for a polynomial (e.g. tractable) algorithm to determine the optimal set of winners, unless $P = NP$ [Sandholm and Suri, 2001].

[5]Appropriate figures for these are around 20 - 30 for the number of participants and 10 - 20 for the number of items.

In particular, the focus is on what mechanism can be used to automate the distribution of tasks in order to cope with the unexpected conditions of the environment and the flexible nature of VOs. For example, in the 4G mobile scenario, once the VO has been formed, a mechanism is needed to automate the distribution of mobile phone users' requests for personalised media services. Generally speaking, this problem has been largely neglected in the literature; most of the work related to this phase has concentrated on the IT infrastructure needed for coordinated resource sharing and problem solving between the VO members (again, see chapter 2 for more details). In this thesis the approach is to use techniques from **coalition formation**, a branch of multi-agent systems research [Sandholm *et al.*, 1999] based on game theory [Rapoport and Kahan, 1984] that provides solutions to partition a set of agents into various subsets in order to maximise some criteria of efficiency and/or stability. This approach is chosen because it provides a provably optimal way to distribute tasks to sub-groups of the VO members.

While there has been an extensive amount of work in coalition formation (see section 2.3 for more details), one of the main problems that hinders the wide spread adoption of this technology is the computational complexity of *coalition structure generation*. That is, once a group of agents has been identified, how can it be partitioned into sub-groups in order to maximise the social payoff for the group? This problem has been shown to be NP-hard and even finding a sub-optimal solution requires searching an exponential number of solutions [Sandholm *et al.*, 1999]. Thus, this research concentrates on developing more efficient coalition structure generation algorithms.

1.2.3 The Maintenance Phase

In the maintenance phase, the main research issues are how to add new members into the VO and how to distribute or redistribute the necessary tasks between the members of the new structure. In particular, in a dynamic environment, there are two main situations that may arise:

- The situation changes, but the members remain unchanged. This means that the work distribution between VO members needs to be reorganised. Again, this

research uses coalition formation algorithms to partition a VO's members into various subsets working on various activities to seek maximal efficiency.

- Some members fail or withdraw from the VO: in such cases, the VO will have to find the substitutes. In order to do this, the research applies similar mechanisms to those that are used in the creation phase. The main difference is, in this case, only some members of the VO need to be substituted. For instance, in the 4G mobile phone example, when a mobile network operator withdraws from the VO, we use the auction mechanism to select the additional operators to replace this operator.

1.3 Research Contributions

The research described in this thesis makes significant contributions to the state of the art in the areas of auction clearing algorithms and coalition structure generation.

In more detail, the contributions to auction clearing algorithms are as follows:

- Novel polynomial clearing algorithms were developed for multi-unit single-item and multi-unit combinatorial forward and reverse auctions with demand/supply function bidding that satisfies discount and free disposal properties [Dang and Jennings, 2002] [Dang and Jennings, 2004b]. No previous polynomial algorithms exist for this broad class of auctions. And although multi-unit single-item auctions are not our main target case, our algorithms for this setting still represents a contribution in its own right. While Sandholm and Suri's algorithms target the same environment as this, they are only applicable in the specific case where the supply curves are linear [Sandholm and Suri, 2001]. In contrast, our result is applicable to the more general case; that is, discount, free disposal supply curves.

  Moreover, the algorithms are shown to produce a solution that is within a finite bound of the optimal. Finally, our empirical results show for realistic settings, their solutions are within a much smaller bound (than the proved theoretical bound) of the optimal.

- Novel optimal clearing algorithms were developed for multi-unit single-item and multi-unit combinatorial forward and reverse auctions with demand/supply function bidding [Dang and Jennings, 2003]. This was carried out for two sets of **bidding functions**[6]: piece-wise linear and monotonic one-unit-difference.[7] This set of algorithms is necessarily not polynomial, but is guaranteed to produce the optimal allocation.[8] Again no previous optimal algorithms existed for this class of auctions.

The contributions to the area of coalition structure generation are as follows:

- A novel anytime algorithm for coalition structure generation was developed that can produce solutions within a finite bound from the optimal [Dang and Jennings, 2004a]. This algorithm is anytime — it can be interrupted at any time, and it establishes a monotonically improving bound from the optimal. Most previous work in this area cannot give such guarantees for its solutions (see section 2.3 for more details). The only other algorithm that can establish a worst-case bound from the optimal is [Sandholm *et al.*, 1999] and our algorithm was shown to be significantly faster. For example, with bound 3, our algorithm is more than 10^7 times faster for $n = 50$, more than 10^{23} times faster for $n = 100$, more than 10^{171} times faster for $n = 500$, and more than 10^{379} times faster for $n = 1000$.

1.4 Thesis Structure

The remainder of the thesis is organised as follows.

The next chapter gives a more in depth analysis of existing approaches to virtual organisations. Specifically, first, it explains the problem of partner selection in VOs. In this context, it discusses the use of auctions to solve this problem, and explains the need for

[6] Bidding functions are the functions that specify the relation between the quantity of the items and the price in the bids

[7] *piece-wise linear* bidding functions are those in which the demand/supply curves for each individual commodity are composed of many linear segments, while *monotonic one-unit-difference functions* are those in which the function indicating the price for adding one more single unit into a package is monotonic (non-increasing or non-decreasing).

[8] It will be shown that an algorithm that produces the optimal allocation cannot be polynomial unless $P = NP$ (see chapter 3 and 4 for more details).

clearing algorithms for combinatorial auctions with demand/supply function bids. Our auction setting is then described in details that will be solved in chapter 3 and 4. It also gives an extensive review of existing work in auction clearing. Second, it explains the problem of task distribution within VOs, formalises the problem of coalition structure generation that will be solved in chapter 5, and gives a literature review of the area.

Chapter 3 presents polynomial clearing algorithms that are applicable for a broad class of bidding function that satisfies discount and free disposal properties. First, it presents the algorithms for multi-unit single-item and multi-unit combinatorial cases. The algorithms are proved to generate solution that is within a finite bound of the optimal. Second, it presents our benchmark test and the empirical results reveal that in realistic settings, the bound of the optimal can be even smaller than that in the theoretical worst-case analysis.

In contrast to chapter 3, chapter 4 presents optimal clearing algorithms that are guaranteed to produce the optimal allocation, but which are not polynomial. Two sets of algorithm are presented for two broad classes of bidding functions: piece-wise linear and one-unit-difference.

Chapter 5 presents our novel coalition structure generation that can be disrupted anytime and is guaranteed to produce solutions that are within a finite bound of the optimal (the longer we run the algorithm, the smaller the bound is). It is then benchmarked against the only other algorithm by [Sandholm *et al.*, 1999] that is also guaranteed to produce solutions that are within a finite bound of the optimal. The benchmark results shows that our algorithm to be considerably faster than its alternative.

Finally, chapter 6 concludes and presents future work.

Chapter 2

Background

This chapter gives an introduction and a literature review on the area of virtual organisations. In particular, this review focuses on the main problems that we are tackling in this thesis; namely partner selection and task distribution in VOs. Specifically, section 2.1 outlines the main research issues in VOs in general and gives a detailed analysis of some of the main existing projects on VOs. Section 2.2 then focuses on the problem of partner selection in VOs. It discusses, in detail, the use of auctions to solve this problem and explains the need for clearing algorithms for combinatorial auctions with demand/supply function bids. The state of the art of this area is then analysed and the shortcomings against our requirements are identified. Section 2.3 follows by focusing on the problem of task distribution within VOs, and formalises the problem of coalition structure generation that will be solved in chapter 5. Here, again, the existing literature in this area is reviewed and the shortcomings with respect to our requirements are identified.

2.1 Virtual Organisations

As noted in chapter 1, virtual organisations (VOs) are becoming an ever more important research area because they offer a number of potential advantages over the independent organisation and traditional collaborative models. However, there are still a number of key research challenges that need to be overcome in order to make the VO vision a

practical reality. The challenges, discussed below, can be organised according to the VO lifecycle (as per figure 1.1).

- *Creation phase*: in which one or more of the entities comes to believe that it might be worthwhile to create a VO. This agent contacts a number of potential participants to determine whether they would be willing to join the VO, and, if it is successful, this will establish a group of agents that are willing to work together in the context of the VO. The main issues in the creation phase are:
  - Identification of needs: the VO initiator needs to be able to identify the task the VO will be doing, decompose the task, then identify the skills and capacities needed from members of the planned VO.
  - Enterprise capability representation: agent capabilities (descriptions of the services that each agent provides) need to be well-defined using some rich representation and standardised to support inter-operability that facilitates the activities of searching for partners. Specifically, partners need to be searchable based on multiple attributes such as the services that they provide, their geography of operation, and the quality of service they provide. This is necessary as the VO initiator may not only look for companies that provide specific services, but may also need to know the quality of the services that they provide, as well as their geography of operation, to ensure the success of the future VO.
  - Partner selection mechanism: the mechanism to select from among the candidates those that will be the most appropriate ones to actually form the VO. In this context, the most appropriate could be the ones who can provide specific services with the lowest costs, the highest quality, or the greatest reliability.
- *Operation phase*: in which the tasks that need to be carried out are determined and it is decided which members of the VO will be responsible for which of these various tasks. Here the key issues are:

  - Task distribution mechanism: the means by which the flow of incoming tasks are assigned to the members of the VO in order to maximise the efficiency of the overall collective.
  - VO partners collaboration support tools: the tools that facilitate the collaborative activities between the members of the VO, for example, secure data exchange and data sharing, group planning and scheduling, and other VO management tools.

- *Maintenance phase*: in which the VO structure (in terms of its members or the task distribution) needs to be changed because of member failure or changes in the environment. The main issues in this phase are how to add new members into the VO and how to distribute or redistribute the necessary tasks between the members of the new structure. The former case is similar to the partner selection mechanism part of the creation phase (but not identical to it, as here we don't want to build the VO from scratch, but rather build on what is already there), while the latter case is similar to the task distribution mechanism part of the operation phase.

- *Dissolution phase*: in which the VO is disbanded because it is no longer effective. Here the main issues relate to how the VO's performance is measured and assessed, the mechanism that is put in place to enable the collective to disband and to absolve itself of any remaining commitments.

While a number of projects have now started in this area, each tends to deal with a specific aspect of the VO lifecycle. Moreover, in many of the existing projects, the degree of automation of the VO lifecycle is limited.

For example, the NIIIP project (National Industrial Information Infrastructure Protocols) [NIIIP, 1998] and the PRODNET II project [Camarinha-Matos and Afsarmanesh, 1999] are concerned with the development of IT and cooperation platforms for VOs. The former was developed by the NIIIP Consortium [NIIIP, 1998] which is a U.S. Industry/Government initiative to develop a software technology that will enable Virtual Enterprise Computing. Specifically, it exploits core technologies defined by: Internet and related communications facilities and services; the Object Management Group (OMG)[1]

[1]Object Management Group, http://www.omg.org.

and related object technology; and The Standard for the Exchange of Product Model Data (STEP) [2] and related information modelling technologies, and develops additional technology to integrate these technologies for work and knowledge management of VEs. The three main areas that it concerns are: communication (based on Internet and object technology by OMG), data and information exchange (based on STEP) and knowledge and task management (based on work by the Workflow Management Coalition (WfMC)). As such, it lacks any significant degree of automation in any of the key phases of the VO lifecycle.

PRODNET II, on the other hand, aims to develop an open and highly flexible support infrastructure for virtual enterprises that is particularly suited to the needs of small and medium enterprises (SMEs). PRODNET II's basic platform facilitates the exchange of commercial data (EDIFACT), the exchange of technical product data (STEP), order status monitoring, quality related information exchange and information management supporting administrative information about the virtual enterprise. It also incorporates a coordination module that handles all cooperation related events (execution of a local work flow), a component that allows the definition and parametrization of the virtual enterprise and the behaviour of each particular enterprise and a component that manages incompletely and imprecisely specified orders (along their life cycle). Within this project, the main supported phases in the VO lifecycle are creation and operation. In the former case, the project utilises private supplier lists or some public directories to search for potential partners. In the latter case, the project develops coordination and workflow management tools that help the VO members to cooperate effectively. Thus, it also lacks a significant degree of automation in partner selection and task distribution of the VO.

Other projects in this area tend to address particular aspects in a specific phase of the lifecycle. For example, AVE (Agents in Virtual Enterprises) [Fischer *et al.*, 1996] concentrates on using agents in the formation of a VO (creation phase). In particular, the project uses auctions mechanism to form the VO. However, it uses only simple auction mechanisms (such as English, Dutch, first-price sealed bid and second-price sealed-bid

[2]The Standard for the Exchange of Product Model Data (STEP), ISO 10303, http://www.nist.gov.

auctions) and so cannot allow the potential partners to vary their potential involvement in the VO, and thus, may not be able to select the most suitable set of partners either.

MASSYVE (Multiagent Manufacturing Agile Scheduling Systems for Virtual Enterprises) [Rabelo *et al.*, 1998] focuses on agile scheduling (the operation phase). In particular, it addresses task distribution by using software agents that utilise the Contract Net Protocol [3] to assign tasks among agents. Specifically, the procedure is to announce a task (an enterprise activity) through the MAS network and then make the agents exchange information about it with other agents until one of them is selected to perform the task. However, this approach only deals with distributing one task at a time. Thus it is likely to be slow when multiple such tasks need to be assigned. Moreover, such sequential allocation may lead to sub-optimal outcomes because it ignores the inter-dependencies that may almost invariably exist between tasks.

Following a related approach, [Rocha and Oliveira, 1999] develop a system in which a market agent (VO broker) sends invitations to the potential partners corresponding to each of the VO's sub-tasks. Interested enterprise agents then formulate bids according to their own capabilities and send bids back to the VO broker. The market agent then uses a multi-criteria function to evaluate bids, and uses constraint satisfaction techniques to resolve any incompatibility between them. However, bids are made for each of the sub-tasks, thus this approach cannot take into account any relationship/interdependence that may exist between the sub-tasks. In [Daviddrajuh and Deng, 2000], mobile agents are sent by the VO creator to collect data from potential partners. Then an assessment on the potential partners about their suitability to the prospective VO is made, based on the collected data. However, there is a clear question made about whether potential partners will expose their true private information to the VO creator's mobile agents.

The VEGA project [Stephens, 1999] develops an information infrastructure to support the technical and business operations of VOs using groupware tools and distributed architectures (the operation phase). Specifically, the VEGA platform supports people in information sharing (data exchange, distributed user access, distributed database,

[3] FIPA Contract Net Interaction Protocol Specification: http://www.fipa.org/specs/fipa00029/.

concurrent user access) and managing group activity. However, it does not provide an automated task distribution mechanism.

As can be seen, these projects typically provide tools to support different aspects of the VO lifecycle. However, they rarely provide efficient mechanisms to automate the various phases. This is particularly true when it comes to partner selection mechanisms in the creation phase, task distribution in the operation phase and adding/removing partners and task redistribution in the maintenance phase. Given this, this thesis seeks to start addressing this shortcoming by developing efficient mechanisms for partner selection and task distribution in VOs. The next two sections will detail our approach in solving these problems. Specifically, section 2.2 focuses on the problem of partner selection, while section 2.3 concentrates on the problem of task distribution.

2.2 Partner Selection in Virtual Organisations

As discussed earlier, it was decided that this research will use combinatorial auctions to tackle the partner selection problem (the reasons and rationale for this choice are given in subsection 1.2.1). Specifically, this means that an agent, after detecting a market opportunity (niche), will determine the capabilities or services that need to be present in order to deliver the functionality of the new virtual organisation. This agent will then send out requests for proposals to all interested parties, who will reply with bids indicating the services and associated capacities they are willing to offer. The initiating agent (acting as the auctioneer) will then use the clearing algorithms to determine the best set of agents, services and capacities to constitute the new virtual organisation. Here best can be most efficient, lowest cost, or best quality. A similar method can then also be used in the maintenance phase for adding new members to the VO. The difference is that in this case the requests for proposals only concern the additional services and/or capacities needed. For more detail, see chapter 6 where the techniques developed in this thesis for partner selection are applied into a concrete scenario.

However, existing clearing algorithms cannot be taken off-the-shelf for this problem because they only consider *atomic proposition* bids (that is, bids are either accepted

in their entirety or rejected) (see subsection 2.2.2 for more details). This, in turn, has the disadvantage of limiting the choice, and hence the potential profit, available to the auctioneer. For example, consider the case where there are only two bids: x_1 units of one good at price p_1 and x_2 units at price p_2, and the auctioneer wants to trade less than $x_1 + x_2$ units of the good. In this case, the auctioneer has no choice other than selecting one or other of the two bids. This may prevent the auctioneer from maximising its payoff. For example, the auctioneer may find it more beneficial to accept both bids partially; that is, trade y_1 ($y_1 < x_1$) units with bidder 1 at price $\frac{y_1}{x_1} \cdot p_1$ and trade y_2 ($y_2 < x_2$) units with bidder 2 at price $\frac{y_2}{x_2} \cdot p_2$.

Moreover, if the bids are expressed in terms of the correlation between the quantity of items and the price (rather than the simple linear extrapolation above[4]), there will be even more choice for the auctioneer, and, consequently, even more chance of maximising its payoff. When viewed from the bidder's perspective, the atomic nature of bids and the inability to explicitly relate price and quantity means that opportunities for trade are lost because the auctioneer may not want the entire package being offered, even though elements of it may be acceptable. Although nearly all the aforementioned work permits XOR (exclusive-or) bids[5], and, in theory, the correlation function between the quantity and the price may be expressed using XOR atomic proposition bids to specify points; in practice, it is nearly impossible as the number of points on the graph of the function could be exponential. For example, let us suppose a bidder wants to trade 1000 units with unit price 10 if the quantity is less than 100, and with unit price 9 if the quantity is in the range between 100 and 1000. With XOR bidding, the bid has to be expressed as XOR of 1000 atomic proposition bids, in which each atomic bid is a pair of quantity and price for every quantity from 1 to 1000. This is clearly inefficient.

To overcome the aforementioned shortcomings associated with atomic propositions, Sandholm and Suri consider the case in which agents can submit bids that correspond

[4] In many cases, linear extrapolation does not work because bidders may value bundles of items non-linearly.

[5] An XOR bid is one in which a bidder submit an arbitrary number of atomic proposition bids with the condition that it is willing to obtain at most one of these bids [Sandholm, 1999]. For an overview on atomic-related bidding languages see [Nisan, 2000].

to a demand or supply curve depending on whether it is an auction or a reverse auction[6] respectively [Sandholm and Suri, 2001]. Thus, bids are expressed in terms of a curve which correlates the quantity with the price of an item. For example, an agent may express the bid as $q = 2 * p + 1$, which means that the agent is willing to trade up to $q = 2 * p + 1$ units if the unit price equals p.[7] Unfortunately, their work is limited to multi-unit single-item auctions[8] and does not deal with the combinatorial case. This means their algorithm cannot explicitly cope with any interdependencies that may exist between the purchasing of multiple items.

In the next two chapters, we develop clearing algorithms for both forward and reverse auctions that remove the shortcomings associated with the atomic proposition nature of previous combinatorial clearing algorithms and the non-combinatorial nature of Sandholm and Suri's demand/supply curve functions. Specifically, we consider multi-unit single-item and multi-unit combinatorial forward auctions and reverse auctions in which bids contain an agent's demand/supply function. This is necessary when applying auctions to VO creation as the expressiveness of auctions with demand/supply function bids and the potential benefit they bring make them highly suitable for VO efficient formation. To this end, the next subsection formalises the problem of clearing auctions with demand/supply function bids, before subsection 2.2.2 analyses the state of the art in this area in more detail.

2.2.1 Auction Clearing with Demand/Supply Function Bids

This subsection formalises the problem of clearing in multi-unit combinatorial forward (reverse) auctions. Assume there are m items (goods/services): $1, 2, ..., m$ and n bidders $a_1, a_2, ..., a_n$. The auctioneer has a supply (demand) $(q_1, q_2, ..., q_m)$, in which q_j is the quantity of item j that the auctioneer is willing to sell (buy).[9] Let u_i^j be the maximum

[6] In an auction (forward auction), there is one seller and multiple buyers; while in a reverse auction, there is one buyer and multiple sellers.

[7] Their price function calculates the quantity from the unit price. However, in our work, the price function will calculate the unit price from the quantity, because we find the later more natural.

[8] By *single item* we mean that there is only one type of good/service for trading in the auction.

[9] In the remainder of this thesis, when describing forward and reverse auctions, the first word deals with the forward case and the word in brackets applies to the reverse case. But since both cases are not simply the inverse of one another in general way, sometimes we need to show things separately for the forward and for the reverse case.

quantity of item j that a_i is able or willing to buy (sell) (if a_i is not willing to buy (sell) an item j, then $u_i^j = 0$). Let $\mathbb{N}$ be the set of natural numbers and $\mathbb{Q}^*$ be the set of non-negative rational numbers.

The *demand (supply) function* is the price function of the items that each bidder is willing to buy (sell). The demand (supply) function of bidder i is:

$$P_i : (\mathbb{N} \cap [1, u_i^1]) \times (\mathbb{N} \cap [1, u_i^2]) \times ... \times (\mathbb{N} \cap [1, u_i^m]) \rightarrow \mathbb{Q}^*$$

where $P_i(r_1, r_2, .., r_m)$ is the price offered by bidder i for the package of items $(r_1, r_2, ..., r_m)$ and r_j is the quantity of item j, $r_j \in \mathbb{N}$, $0 \leq r_j \leq u_i^j$, $\forall\ 1 \leq j \leq m$. For example, suppose that $m = 3$, then $P_1(1, 3, 2)$ will be the price agent 1 offers for a package which is composed of 1 unit of item 1, 3 units of item 2 and 2 units of item 3 altogether. In the single-item case, the graph of a demand (supply) function will be a curve (figure 2.1), while in the combinatorial case, it will be a surface (figure 2.2).

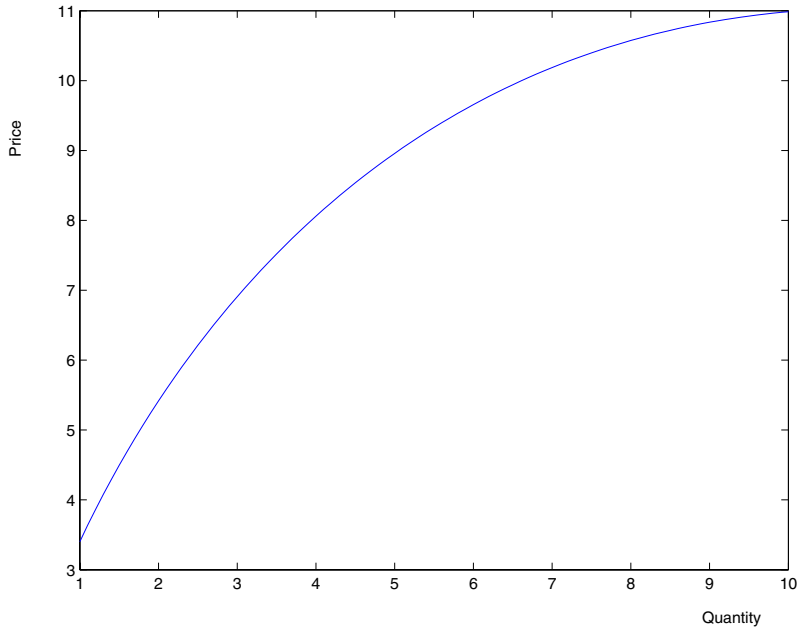


FIGURE 2.1: A demand (supply) curve for multi-unit single-item case.

Having determined the demand function, we now consider the *supply allocation* which is the amount that the auctioneer trades with each bidder.

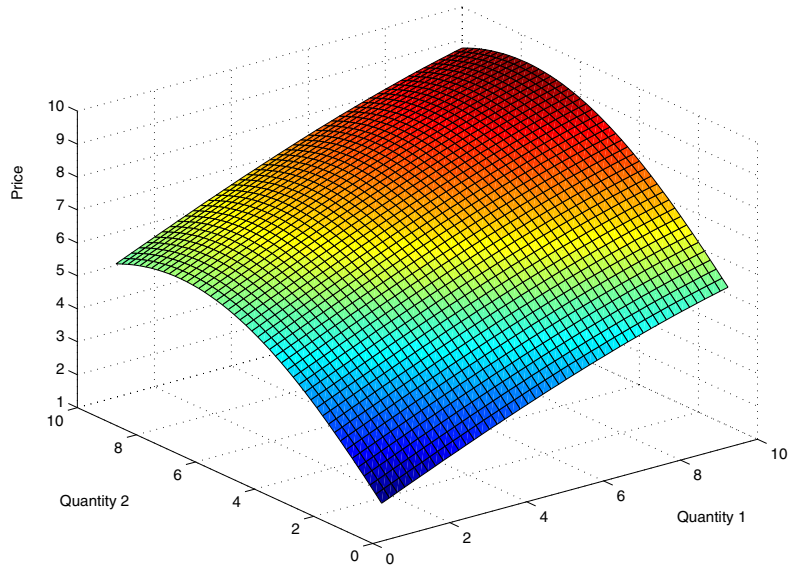


FIGURE 2.2: A demand (supply) function for multi-unit combinatorial case.

Definition 2.1. A supply allocation is a tuple $\langle r_i^j \rangle, 1 \leq i \leq n, 1 \leq j \leq m$ such that the auctioneer sells (buys) r_i^j units of item j to (from) each agent a_i.[10]

Given the definitions of the demand (supply) function and the supply allocation, the problem of forward (reverse) auction clearing is then to find a supply allocation $\langle \alpha_i^j \rangle, 1 \leq i \leq n, 1 \leq j \leq m$ that:

- Satisfies the supply (demand) constraint

$$\sum_{i=1}^{n} \alpha_i^j \leq q_j, \forall 1 \leq j \leq m \text{ (forward case)} \tag{2.1}$$

$$\sum_{i=1}^{n} \alpha_i^j \geq q_j, \forall 1 \leq j \leq m \text{ (reverse case)} \tag{2.2}$$

  That is, the quantity of each item that the auctioneer sells (buys) to (from) all bidders is not bigger (less) than the auctioneer's supply (demand) for that item.

[10] Because the auctioneer sells (buys) items at the price that the bidders offer, it may well be the case that the auctioneer will sell (buy) the same package from two different bidders at different prices. That is, the auctions have *discriminatory pricing* [Sandholm and Suri, 2001].

- Optimise the auctioneer's total revenue:

$$\sum_{i=1}^{n} P_i(\alpha_i^1, \alpha_i^2, ..., \alpha_i^m) \text{ is maximal (minimal).} \tag{2.3}$$

  That is, the total price of all the units of all the items supplied by the auctioneer (bidders) should be as big (small) as possible.

However, the auction clearing problem has been shown to be NP-complete, even for the simplified case of single-items with piecewise linear supply curves [Sandholm and Suri, 2001] [11]. Thus, it is impossible to find a polynomial algorithm whose solution is guaranteed to be the optimal allocation, unless P = NP.

2.2.2 The State of The Art in Clearing Algorithms

As stated in subsection 2.2.1, most of the previous work on clearing algorithms for combinatorial auctions has been based on atomic proposition auctions. In particular, Sandholm et al. have categorised and analysed the complexity of various kinds of atomic proposition types (e.g. auctions, reverse auctions, and exchanges, which are then categorised as single or multiple units, with or without free disposal) [Sandholm *et al.*, 2002]. In this line of work, they showed that clearing combinatorial atomic proposition auctions is NP-complete, even for the case of single-units. Thus, heuristic methods are typically used to tackle this problem.

In more detail, Nisan used Linear Programming to investigate the single-unit combinatorial case [Nisan, 2000]. He showed that Linear Programming can produce the optimal solution in a reasonable time in some specific cases (e.g. linear order bids[12], mutual

[11] The proof of NP-completeness in [Sandholm and Suri, 2001] is not directly applicable to our specific auction setting in chapter 3 (because the demand/supply function in their proof is not compatible with this setting). Thus we will present a proof of NP-completeness for our specific setting in chapter 3. Their proof, however, can be applied to our auction settings in chapter 4 (because in this case, the demand/supply function in their proof is compatible with our setting).

[12] This means the set of items can be linearly ordered $G = \{g_1...g_n\}$ such that all bids S are for a consecutive sub-range $S = \{g_k...g_l\}$ of items [Rothkopf *et al.*, 1998].

exclusion bids[13] and substructure bids.[14]) He then suggested using greedy and Branch-and-Bound algorithms based on Linear Programming for the other cases. His greedy algorithm is computationally efficient (polynomial complexity), but cannot guarantee to produce the optimal solution, while his Branch-and-Bound algorithm provably produces the optimal solution, but cannot guarantee to run in polynomial time.

Other researchers such as Gonen and Lehmann and Leyton Brown et al. have further investigated the use of Branch-and-Bound techniques to solve the clearing problem [Gonen and Lehmann, 2000] [Leyton-Brown *et al.*, 2000]. Both of their algorithms build up a partial allocation one bid at a time, while using a depth-first search with backtracking to cover the whole search space. Although these Branch-and-Bound algorithms cannot guarantee to produce the optimal solution in polynomial time, they presented various methods to speed up the algorithms. In more detail, Leyton Brown et al. designed a function for computing upper bounds for the optimal outcome, tailored specifically to the multi-unit combinatorial auction problem. Dynamic programming techniques, techniques for pre-processing and caching, and heuristics for determining search orderings are also used to further improve their algorithm. Gonen and Lehmann discussed various methods of bounding from above (linear programming, projections, average price consideration) and for choosing the most promising bid (ordering the bids by price, average unit price, normalised average unit price) [Gonen and Lehmann, 2000].

However, as stated above, all of the above work has been based on atomic proposition auctions. So by removing this restriction, our algorithms produce more efficient allocations.

In contrast to the above, however, [Sandholm and Suri, 2001] considered multi-unit single-item auctions with bids in the form of supply/demand curves. By limiting these curves to a specific type (linear and piecewise linear curves[15]), they were able to analyse

[13] This means the bid is presented in the OR-of-XORs language, where each atomic bid is for a singleton set [Nisan, 2000].

[14] This means the auction is a "sum" of other auctions for which Linear Programming can produce optimal solutions.

[15] Their concepts of linear and piecewise linear curves are different from ours, as they consider the unit price function, not the total price function. Thus, when they speak of a linear unit price function, this means a quadratic total price function.

the complexity and suggest an algorithm for clearing.[16] However, as discussed in section 2.2, this work does not deal with the multi-unit combinatorial case.

Other researchers such as Davenport et al. and Eso et al. have further considered multi-unit combinatorial reverse auctions with supply curves [Davenport and Kalagnanam, 2001] [Eso *et al.*, 2001]. They showed that in the case where the supply curves are piecewise linear, the clearing problem can be modeled as a Linear Program and solved using Linear Programming techniques. However, in this work, bidders submit separate supply curves for different items, and it is assumed that the price of a package of items is equal to the sum of all the prices of the separate items.[17] This means that these auctions are not truly *combinatorial* in nature as the correlation between items is ignored.

2.3 Task Distribution within Virtual Organisations

In this thesis, the problem of task distribution within virtual organisations is tackled using coalition formation techniques developed in the field of multi-agent systems (see section 2.3.2 for a review of the state of the art in this area). Now, in this context, the coalition formation process can be viewed as being composed of three main activities [Sandholm *et al.*, 1999]:

1. *Coalition structure generation*: forming coalitions of agents such that those within a coalition coordinate their activities, but those in different coalitions do not. This primarily involves partitioning the set of all agents in the system into exhaustive and disjoint coalitions.[18] Such a partition is called a **coalition structure**. For example, in a multi-agent system composed of three agents $\{a_1, a_2, a_3\}$, there exist seven possible coalitions:

   $\{a_1\}$

   $\{a_2\}$

   $\{a_3\}$

   $\{a_1, a_2\}$

[16] They provided an algorithm for the linear case only, not for the piecewise linear case.
[17] This property is called *additive separability* in [Eso *et al.*, 2001].
[18] Some research also considers non-disjoint coalitions (see subsection 2.3.2 for details).

$\{a_1, a_3\}$

$\{a_2, a_3\}$

$\{a_1, a_2, a_3\}$

and five possible coalition structures:

$\{\{a_1, a_2, a_3\}\}$

$\{\{a_1\}, \{a_2, a_3\}\}$

$\{\{a_2\}, \{a_3, a_1\}\}$

$\{\{a_3\}, \{a_1, a_2\}\}$ $\{\{a_1\}, \{a_2\}, \{a_3\}\}$.

2. *Optimising the value of each coalition*: pooling the resources and tasks of the agents in a given coalition to maximise the coalition value. For example, given the coalition structure $\{\{a_1\}, \{a_2, a_3\}\}$, each of the two coalitions $\{a_1\}$ and $\{a_2, a_3\}$ will try to optimise its value.

3. *Payoff distribution*: dividing each coalition's value among its members. For example, if the coalition $\{a_2, a_3\}$ produces a payoff of X then this value needs to be divided between a_2 and a_3 according to some scheme (e.g. equality or stability).

Although these activities are distinct and, in a sense, conceptually sequential, it is also clear that they interact. For example, in a competitive environment, the coalition that an agent wants to join depends on the payoff that it is likely to receive (activities 1 and 3). However, in cooperative environments, where the agents work together to maximise the social welfare, payoff distribution is less important, and coalition structure generation that maximises the social welfare is the dominant concern. In the context of this work, the VO operation phase can be considered a cooperative environment because once the members of the VO are established, their aim is to work together to maximise the payoff of the VO as a whole. Thus, the focus of this thesis is in developing new algorithms for coalition structure generation.

Classically, game theoretic work on coalition formation is mainly concerned with coalition structure generation and payoff distribution [Rapoport and Kahan, 1984]. However, it is static in nature, and while it addresses the question of which coalition structure should form, it does not address the question of how to generate the coalition structure that maximises the social welfare.

Much work on coalition formation considers super-additive environments (meaning any two disjoint coalitions are better off by merging together) [Ketchpel, 1994] [Rapoport and Kahan, 1984] [Shehory and Kraus, 1995] [Zlotkin and Rosenschein, 1994]. In such cases, coalition structure generation is trivial because all agents are better off by forming the grand coalition (i.e. the coalition that contains all the agents). Moreover, it has even been argued that almost all environments are super-additive because, at worst, the agents in the composite coalition can use solutions as if they are in separate coalitions [Rapoport and Kahan, 1984]. However, this assumption is not valid for many real-world problems, including those that drive our work, because of the cost of forming coalitions and the cost of coordination between members in the same coalition.

In non-super-additive environments, coalition structure generation is a major concern (because of the exponential size of the set of all possible coalition structures). In such cases, the desirable goal is usually to maximise the social welfare. However, it has been shown that this problem is NP-hard and, moreover, even finding a sub-optimal solution requires searching an exponential number of solutions [Sandholm *et al.*, 1999]. To tackle this problem, several researchers have proposed algorithms for coalition structure generation. However, most of the existing algorithms cannot establish a worst-case bound from the optimal. This is clearly undesirable because it means the solutions they generate can be arbitrarily bad. To overcome this drawback, Sandholm et al. developed an anytime algorithm that can establish a worst-case bound (until our algorithm it was the only one that could do this) [Sandholm *et al.*, 1999]. However, as their algorithm's computational complexity is exponential, it is desirable to see if its complexity can be reduced in order to make it usable in practical applications. Specifically, reducing its complexity is essential if it is to be applicable in the VO operation phase.

Against this background, chapter 5 will develop a novel coalition structure generation algorithm that is shown to be significantly faster than its alternative in our benchmark tests (for an example of the algorithm's application in VO task distribution, see chapter 6 where the algorithm is applied into a concrete scenario). To this end, the next subsection formalises the problem of coalition structure generation.

2.3.1 Coalition Structure Generation

This subsection formalises the problem of coalition structure generation. Let A be the set of agents, and n be the number of agents in A (i.e., $|A| = n$). As is common practice in the literature (e.g. [Ketchpel, 1994] [Rapoport and Kahan, 1984] [Sandholm *et al.*, 1999] [Shehory and Kraus, 1996]), we consider coalition formation in *characteristic function games (CFGs)*. In such settings, there is a value $v(S)$ for each and every subset S of A, known as *the value of coalition* S, which is the utility that members of S can jointly attain. Fundamentally, this means each coalition's value is independent of the actions of agents that are not members of the coalition. Although, in general, the value of a coalition may depend on non-members' actions, CFGs can be applied in many real-world multi-agent problems [Sandholm *et al.*, 1999].

As in [Sandholm *et al.*, 1999], we assume that every coalition's value is non-negative:

$$v(S) \geq 0, \forall S \subseteq A \tag{2.4}$$

This assumption is not very restrictive, because if there exist some negative coalitional values, and if all coalitional values are bound from below (i.e., they are not infinitely negative), they can always be normalised by subtracting from each of them a value $\min_{S \subseteq A} v(S)$.

A coalition structure CS is a partition of A into disjoint, exhaustive coalitions. That is, each agent belongs to exactly one coalition. The *value* of a coalition structure, $V(CS)$, is expressed in terms of its social welfare. That is:

$$V(CS) = \sum_{S \in CS} v(S) \tag{2.5}$$

Also, we define the *size* of a coalition structure as the number of coalitions that it contains and L as the set of all coalition structures.

Given the above terms, the problem of coalition structure generation is then to find a coalition structure CS^* that maximises the social welfare. That is:

$$CS^* = argmax_{cs \in L} V(CS) \tag{2.6}$$

However, the problem of coalition structure generation is computationally complex. Sandholm et al. [Sandholm *et al.*, 1999] showed that the number of coalition structures (i.e. $|L|$) is exponential, specifically, $O(n^n)$ and $\omega(n^{n/2})$, and that the problem is NP-hard. Moreover, they showed that for any algorithm to establish any bound from the optimal, it must search at least 2^{n-1} coalition structures.

2.3.2 The State of The Art in Coalition Formation

As mentioned above, most of the existing work in coalition formation in game theory [Rapoport and Kahan, 1984] has focused on coalition structure generation and payoff distribution. In this context, many solutions have been proposed based on different stability concepts (e.g. the core, the Shapley value, the kernel, the stable set, and the bargaining set). *Transfer schemes* have also been developed to transfer non-stable payoff distributions to stable ones (while keeping the coalition structure unchanged) (e.g. transfer schemes have been developed for the bargaining set and the kernel).[19]

Recently, however, researchers in multi-agent systems have paid more attention to the problem of coalition structure generation. As mentioned above, [Sandholm *et al.*, 1999] developed an anytime algorithm that guarantees to produce solutions within a finite bound from the optimal. However, as we will demonstrate in chapter 5, this algorithm is significantly slower than ours. On the other hand, [Shehory and Kraus, 1998] consider a somewhat broader environment, where the coalitions can overlap. In this work, however, they reduce the complexity of the problem by limiting the size of the coalitions. They then develop a greedy algorithm that guarantees to produce a solution that is within a bound from the best solution possible given the limit on the number of agents. However,

[19] For a comprehensive review on stability concepts and transfer schemes in game theory, see [Rapoport and Kahan, 1984].

this best solution can be arbitrarily far from the actual optimal solution (without the limit on the size of the coalitions).

Some other researchers address both coalition structure generation and payoff distribution in competitive environments. Specifically, [Ketchpel, 1994] presents a coalition formation method with cubic running time in the number of agents, but his method can neither guarantee a bound from the optimal nor stability. Shehory and Kraus's protocol guarantees that if the agents follow it, a certain stability (kernel-stability) is met [Shehory and Kraus, 1996]. In the same paper, they also present an alternative protocol that offers a weaker form of stability with polynomial running time. However, in both cases, no bound from the optimal is guaranteed.

More recent research in coalition formation area has also begun to pay attention to dynamic environments, where agents may enter or leave the coalition formation process and many uncertainties are present (e.g. the coalition value is not fixed, but it is context-based [Klusch and Gerber, 2002]). However, to date, no algorithm with bound guarantees has been developed for this environment.

2.4 Summary

This chapter has outlined the background for the problems that will be addressed in the remainder of this thesis. Specifically, it gives a literature review on the general area of virtual organisations, as well as formalising the problems of auction clearing and coalition structure generation. It also highlights the need to develop new algorithms for both of these problems so that the ensuing solutions can be made applicable to partner selection and task distribution within VOs. Against this background, the next three chapters will present our algorithms for these problems. Specifically, chapter 3 presents our novel polynomial algorithms for clearing multi-unit combinatorial auctions, while chapter 4 presents our optimal algorithms for clearing multi-unit combinatorial auctions. Finally, chapter 5 details our novel coalition structure generation algorithm. Chapter 6 then follows to draw the algorithms developed in these three chapters together by demonstrating how they can be applied in a VO lifecycle in a scenario.

Chapter 3

Polynomial Auction Clearing Algorithms

This chapter develops polynomial algorithms for clearing multi-unit single-item and multi-unit combinatorial forward and reverse auctions [1]. Specifically, we consider settings where bidders submit their bids in the form of a demand/supply function and the auctions have sub-additive pricing with free disposal. The algorithms are based on a greedy strategy and they are shown to be of polynomial complexity. Furthermore, the solutions they generate are shown to be within a finite bound of the optimal.

In more detail, we consider settings where the price function satisfies two properties:

- Discount: $\forall\ 0 \leq r_j, s_j \leq u_i^j$,

$$\begin{aligned} & P_i(r_1 + s_1, r_2 + s_2, ..., r_m + s_m) \\ & \leq P_i(r_1, r_2, ..., r_m) + P_i(s_1, s_2, ..., s_m) \end{aligned} \tag{3.1}$$

  That is, the price of any combination of two packages altogether is always cheaper than or equal to the price of these two bundles separately. For example, buying 10 units of item 1 and 12 units of item 2 is always cheaper than buying 5 units of

[1] Forward and reverse auctions are not simply the converse of each other in this context; they have different properties and so require different proofs.

item 1 and 6 units of item 2 twice. In game-theoretic terms, this property is also called *sub-additive* [Tennenholtz, 2000].

- Free Disposal: if $\forall\ j : 0\ \leq r_j \leq s_j \leq u_i^j$, then:

$$P_i(r_1, r_2, ..., r_m) \leq P_i(s_1, s_2, ..., s_m) \tag{3.2}$$

That is, if one package has no fewer units of each item than another package, the former is not less expensive than the latter. For example, 10 units of item 1 and 20 units of item 2 is always cheaper than 15 units of item 1 and 25 units of item 2.

The above assumptions are needed for the subsequent analysis of our algorithms and, moreover, we believe they are applicable to a wide range of applications. The free disposal property is a standard assumption that is adopted in most of the aforementioned work on auction clearing (section 2.2). The sub-additivity assumption is less frequently used but, we believe, is still reasonable in many situations. In particular, in our VO scenario, this would mean, for example, the price of a package of movies and news provided by a media provider is cheaper or equal to the total price of the two equivalent packages being provided separately by the same media provider. Thus, their adoption does not significantly limit the scope of our results.[2]

To this end, section 3.1 presents our algorithm for the single-item case (i.e. where $m = 1$), including the proof of NP-completeness of the clearing problem. Then we will deal with the combinatorial case as a generalisation in section 3.2. Section 3.3 will present our benchmark test of the algorithms provided. This empirical results show that for the cases we considered, the bounds from the optimal of our algorithms' solutions are considerably smaller than the theoretical results proved in sections 3.1 and 3.2.

[2]There are domains where these assumptions do not hold, for example, in nuclear electric industry, where it is costly to dump nuclear waste away.

3.1 Multi-Unit Single-Item Auctions

This section will first present the proof of NP-completeness of the problem of clearing forward and reverse multi-unit single-item auction with the free disposal and the discount properties (subsection 3.1.1). Then it will present our clearing algorithms and analyse their properties.

3.1.1 Proof of NP-completeness

Using the notation of the previous section, the multi-unit single-item forward (reverse) auction case can be formulated as follows: Let n be the number of bidders. Let q be the supply (demand) of the auctioneer and u_i be the maximum quantity of the item that a_i is willing to buy (sell). The demand (supply) function (in the single-item case it can be drawn as a curve, so we can call it the demand (supply) curve) is the price function of the item:

$$P_i : \mathbb{N} \cap [1, u_i] \rightarrow \mathbb{Q}^*$$

where $\mathbb{N}$ and $\mathbb{Q}^*$ are the sets of natural numbers and non-negative rational numbers, respectively, and $P_i(r)$ is the price bidder i offers for r units altogether.

For mathematical convenience, in this subsection we will use the unit price function instead of the price function. The unit price function for each bidder i is:

$$p_i : \mathbb{N} \cap [1, u_i] \rightarrow \mathbb{Q}^*$$

where $p_i(r)$ is the unit price bidder i offers for r units altogether. That is,

$$p_i(r) = \frac{P_i(r)}{r}$$

As before, we consider settings where the demand (supply) curve satisfies the following properties:

- Discount (the more units that are sold, the less the unit price is):

$$p_i(r) \geq p_i(s), \forall\ 0 \leq r \leq s \leq u_i \tag{3.3}$$

- Free Disposal (the more units of the item that are sold, the more the total price is):

$$r \cdot p_i(r) \leq s \cdot p_i(s), \forall\ 0 \leq r \leq s \leq u_i \tag{3.4}$$

The clearing problem is then one of finding a supply allocation $\langle \alpha_i \rangle, 1 \leq i \leq n$, i.e., the auctioneer will sell (buy) α_i units to (from) agent a_i, such that:

- The quantities the auctioneer sells (buys) to the bidders satisfy the supply (demand) constraint:

$$\sum_{i=1}^{n} \alpha_i \leq q \text{ (forward case)} \tag{3.5}$$

$$\sum_{i=1}^{n} \alpha_i \geq q \text{ (reverse case)} \tag{3.6}$$

- Optimise the auctioneer's total revenue:

$$\sum_{i=1}^{n} P_i(\alpha_i) \text{ is maximal (minimal).} \tag{3.7}$$

This is an optimization problem. Thus, in order to analyse the complexity according to NP-completeness, we need to convert it into a decision problem. The decision problem is, given a set of bids and the auctioneer's supply (demand), is there a supply allocation that will give the auctioneer a revenue exceeding a certain value?

First of all, we show that the clearing problem is NP-complete.

Theorem 3.1. *Consider a multi-unit single-item auction with free disposal and discount bidding functions. Then the problem of clearing the auction is NP-complete, even for the simple case when the bidding function is composed of linear segments (i.e. piece-wise linear).*

Proof. First of all, it is trivial that the problem is in NP, because given a supply allocation, we can calculate the revenue it gives the auctioneer in polynomial time. Thus we just need to show that the problem is NP-hard which we do by reducing it to the knapsack problem (as in [Sandholm and Suri, 2001]).

[Forward auction case]

As in [Sandholm and Suri, 2001], we reduce the **knapsack** problem [Martello and Toth, 1990] to our auction problem.

Let $\{(s_1, v_1), (s_2, v_2), ..., (s_n, v_n), c\}$ be an instance of the knapsack problem, that is, c is the knapsack capacity, s_i and v_i are the size and the value of item i, respectively. We then create an instance of the multi-unit single-item auction with free disposal and discount bidding functions as follows.

Let the supply of the auctioneer be: $q = n + c$.

Let the maximum quantity that bidder i is willing to trade be: $u_i = s_i + 1$.

Let K be a number such that K satisfies the two following inequations:

$$K > \max_{1 \leq i,j \leq n, i \neq j} v_i / v_j \tag{3.8}$$

$$K > \max_{1 \leq i \leq n} s_i \tag{3.9}$$

Then, let the price function of bidder i be:

$$\begin{cases} P_i(r) &= K * v_i, \forall 1 \leq r \leq s_i \\ P_i(u_i) &= (K+1) * v_i \end{cases}$$

With the above definitions, we can show that these price functions satisfy both the discount and the free disposal property:

- Discount: As we have $p_i(r) = K * v_i / r$ for all $1 \leq r \leq s_i$, we just need to show that $p_i(s_i) \geq p_i(s_i + 1)$.

We have:

$$\begin{aligned}
&p_i(s_i+1) = (K+1) * v_i/(s_i+1) \\
\Rightarrow &\frac{p_i(s_i+1)}{p_i(s_i)} = \frac{(K+1) * v_i/(s_i+1)}{K * v_i/s_i} \\
\Rightarrow &\frac{p_i(s_i+1)}{p_i(s_i)} = \frac{Ks_i + s_i}{Ks_i + K} \\
\Rightarrow &\frac{p_i(s_i+1)}{p_i(s_i)} < 1 \text{ (because of (3.8))} \\
\Rightarrow &p_i(s_i+1) < p_i(s_i)
\end{aligned}$$

- Free Disposal: it is trivial that $P_i(r)$ satisfies the free disposal property.

The goal of the clearing algorithm is then to find a supply allocation $\langle \alpha_i \rangle$ such that:

- The quantity the auctioneer sells to the bidders satisfy the supply constraint:

$$\sum_{i=1}^{n} \alpha_i \leq q = n + c \tag{3.10}$$

- Optimise the auctioneer's total revenue:

$$\sum_{i=1}^{n} P_i(\alpha_i) \text{ is maximal.} \tag{3.11}$$

This leads to the following lemma.

Lemma 3.2. *Suppose $\langle \alpha_i \rangle$ is the optimal allocation. Then $\alpha_i > 0$, for all $1 \leq i \leq n$.*

Proof. We prove by contradiction. Suppose there exists k such that $\alpha_k = 0$.

Consider the two possible cases.

- Case 1: $\sum_{i=1}^{n} \alpha_i = q$:

  As $q > n$, there must exist l such that $\alpha_l > 1$.

  Now consider the following supply allocation $\{\langle \beta i \rangle\}_{i=1}^{n}$ such that:

$$\begin{cases} \beta_i &= \alpha_i, \forall 1 \leq i \leq n, i \neq k, i \neq l \\ \beta_k &= 1 \\ \beta_l &= \alpha_l - 1 \end{cases}$$

With the above definition, as $\sum_{i=1}^{n} \beta_i = \sum_{i=1}^{n} \alpha_i$, and $\beta_i \leq u_i$, $\forall 1 \leq i \leq n$, we can see that $\{\langle \beta i \rangle\}_{i=1}^{n}$ is a valid allocation.

We have:

$$\begin{aligned} &\sum_{i=1}^{n} P_i(\beta_i) \\ &= \sum_{i=1}^{n} P_i(\alpha_i) + P_k(1) + P_l(\alpha_l - 1) - P_l(\alpha_l) \\ &= \sum_{i=1}^{n} P_i(\alpha_i) + K * v_k + K * v_l - P_l(\alpha_l) \\ &\geq \sum_{i=1}^{n} P_i(\alpha_i) + K * v_k + K * v_l - P_l(u_l) \\ &= \sum_{i=1}^{n} P_i(\alpha_i) + K * v_k + K * v_l - (K+1) * v_l \end{aligned}$$

$$\begin{aligned} &= \sum_{i=1}^{n} P_i(\alpha_i) + K * v_k - v_l \\ &> \sum_{i=1}^{n} P_i(\alpha_i) \text{ (because of inequation (3.9))} \end{aligned}$$

This leads to contradiction, as $\langle \alpha_i \rangle$ is the optimal allocation.

- Case 2: $\sum_{i=1}^{n} \alpha_i < q$:

  Let us consider the following supply allocation $\{\langle \beta i \rangle\}_{i=1}^{n}$ such that:

$$\begin{cases} \beta_i &= \alpha_i, \forall 1 \leq i \leq n, i \neq k \\ \beta_k &= 1 \end{cases}$$

With the above definition, as $\sum_{i=1}^{n} \beta_i = \sum_{i=1}^{n} \alpha_i + 1 \leq q$, and $\beta_i \leq u_i$, $\forall 1 \leq i \leq n$, we can see that $\{\langle \beta i \rangle\}_{i=1}^{n}$ is a valid allocation.

We have:

$$\begin{aligned}
&\sum_{i=1}^{n} P_i(\beta_i) \\
&= \sum_{i=1}^{n} P_i(\alpha_i) + P_k(1) \\
&= \sum_{i=1}^{n} P_i(\alpha_i) + K * v_k \\
&> \sum_{i=1}^{n} P_i(\alpha_i)
\end{aligned}$$

This also leads to contradiction, as $\langle \alpha_i \rangle$ is the optimal allocation.

Thus, as both cases lead to contradiction, we have $\alpha_i > 0$, for all $1 \leq i \leq n$.

□

From the above lemma, we can see that finding a supply allocation $\langle \alpha_i \rangle$ (such that $\langle \alpha_i \rangle$ satisfies the supply constraint and optimises the auctioneer's revenue) is equivalent to finding a tuple $\langle \alpha'_i \rangle$ ($\alpha'_i = \alpha_i - 1$) such that:

- $\sum_{i=1}^{n} \alpha'_i \leq q - n = c$, and
- $\sum_{i=1}^{n} P'_i(\alpha'_i)$ is maximal, where $P'_i(r) = P_i(r+1) - P_i(1)$ (it is because, as $\sum_{i=1}^{n} P'_i(\alpha'_i) = \sum_{i=1}^{n} P_i(\alpha_i) - \sum_{i=1}^{n} P_i(1)$, we have $\sum_{i=1}^{n} P_i(\alpha_i)$ is maximised if and only if $\sum_{i=1}^{n} P'_i(\alpha'_i)$ is maximised).

Also, as $P'_i(r) = P_i(r) - P_i(1)$, this means $P'_i(r) = 0$, $\forall 1 \leq r \leq s_i - 1$ and $P'_i(s_i) = v_i$. Thus, finding the optimal allocation $\langle \alpha_i \rangle$ is equivalent to optimising the instance $\{(s_1, v_1), (s_2, v_2), ..., (s_n, v_n), c\}$ of the knapsack problem. As the latter is NP-complete, so is the problem of finding the optimal allocation of our auction.

[Reverse auction case]

Again, as in [Sandholm and Suri, 2001], we reduce the knapsack problem to our reverse auction problem.

Let $\{(s_1, v_1), (s_2, v_2), ..., (s_n, v_n), c\}$ be an instance of the knapsack problem. We then create an instance of the multi-unit single-item reverse auction with free disposal and discount bidding functions as follows.

Let the maximum quantity that bidder i is willing to trade be $u_i = s_i$.

Let T be the total number of units in all the bids, that is, $T = \sum_{i=1}^{n} s_i$.

Let the demand of the auctioneer be $q = T - c$.

Let the price function of bidder i be $P_i(r) = v_i$, $\forall 1 \leq r \leq s_i$.

It is trivial that these price functions satisfy both the free disposal and the discount property.

The goal of the clearing algorithm is then to find a supply allocation $\langle \alpha_i \rangle$ such that:

- The quantity the auctioneer sells to the bidders satisfy the demand constraint:

$$\sum_{i=1}^{n} \alpha_i \geq q = T - c \tag{3.12}$$

- Optimise the auctioneer's total revenue:

$$\sum_{i=1}^{n} P_i(\alpha_i) \text{ is minimal.} \tag{3.13}$$

Considering this reverse auction, we can see that finding a supply allocation $\langle \alpha_i \rangle$ (such that $\langle \alpha_i \rangle$ satisfies the demand constraint and optimises the auctioneer's revenue) is equivalent to finding a tuple $\langle \alpha'_i \rangle$ ($\alpha'_i = s_i - \alpha_i$) such that:

- $\sum_{i=1}^{n} \alpha'_i \leq T - q = c$, and
- $\sum_{i=1}^{n} P'_i(\alpha'_i)$ is maximal, where $P'_i(r) = P_i(s_i) - P_i(s_i - r)$ (it is because, as $\sum_{i=1}^{n} P'_i(\alpha'_i) = \sum_{i=1}^{n} P_i(s_i) - \sum_{i=1}^{n} P_i(\alpha_i)$, we have $\sum_{i=1}^{n} P_i(\alpha_i)$ is minimised if and only if $\sum_{i=1}^{n} P'_i(\alpha'_i)$ is maximised).

Also, as $P'_i(r) = P_i(s_i) - P_i(s_i - r)$, this means $P'_i(r) = 0$, $\forall 1 \leq r \leq s_i - 1$ and $P'_i(s_i) = v_i$. Thus, finding the optimal allocation $\langle \alpha_i \rangle$ is equivalent to optimising the

Algorithm 1. Repeat the following steps:

- For all i such that $u_i > q$, set $u_i = q$.

  That is, we truncate the demand (supply) function to consider only quantities that are not bigger than the supply (demand). This is because, for the forward auction case, the auctioneer cannot sell more quantity than his supply; for the reverse auction case, in order to minimise the total price, the auctioneer does not need to buy more units than its demand, since the price functions satisfy the free disposal property (inequation (3.4)).

- At each step, find the bidder a_k such that $p_i(u_k)$ is maximal (minimal), then sell (buy) u_k units to (from) a_k.

  That is, we consider all the biggest packages offered by the bidders, then choose the package that offers the biggest (smallest) unit price.

- Repeat the steps above for the set of bidders $A \setminus a_k$ and $q_{new} = q - u_k$.

FIGURE 3.1: The clearing algorithm for the multi-unit single-item case.

instance $\{(s_1, v_1), (s_2, v_2), ..., (s_n, v_n), c\}$ of the knapsack problem. As the latter is NP-complete, so is the problem of finding the optimal allocation of our auction. □

3.1.2 The Algorithm

We are now in a position to express our algorithm for solving this problem. Like [Sandholm *et al.*, 2002] we adopt a greedy approach for solving this problem. Our algorithm is presented in figure 3.1.

We can now analyse this algorithm to assess its properties.

Theorem 3.3. *In the reverse auction case, if there is a solution, this algorithm will find it.*[3] *That is, if the total of the supplies of the bidders is larger than the auctioneer's demand, this algorithm will produce an allocation. Also, the total units of the solution will be exactly equal to the auctioneer's demand.*

[3] In the forward case, it is trivial that there is always a solution: when the auctioneer accepts no bid, or any single bid.

Proof. In each step, the algorithm for the reverse case selects exactly one agent from the set of bidders. And if its supply is less than the auctioneer's remaining demand, the algorithm takes all its supply. Otherwise it takes the quantity that is equal to the remaining demand. So, if the algorithm does not terminate beforehand, it will eventually select all the bidders and take all the supplies. Thus, if the total of the supplies of the bidders is larger than the auctioneer's demand, the algorithm will produce an allocation.

Moreover, in each step, the algorithm takes at most all the remaining demand, thus the solution it produces will have the total units being equal to the auctioneer's demand. □

Theorem 3.4. *The complexity of the algorithm is* $O(n^2)$.

Proof. At each step, it requires $O(n)$ to find the biggest (smallest) element of the set $\{p_1(u_1),$
$p_2(u_2), ..., p_n(u_n)\}$. So each step has $O(n)$ complexity. As there are at most n steps, it is clear that the complexity of the algorithm is $O(n^2)$ □

Theorem 3.5. *The solution generated from the algorithm is within a bound* $b = n$ *from the optimal. That is, let* $P_n(O)$ *be the optimal total price and* $P_n(S)$ *be the total price of the solution of the algorithm. Then:*

$$\frac{P_n(O)}{P_n(S)} \leq n \text{ (forward auction case)} \tag{3.14}$$

$$\frac{P_n(S)}{P_n(O)} \leq n \text{ (reverse auction case)} \tag{3.15}$$

Proof. [*Forward auction case*]

Let $\langle r_1, r_2, .., r_n \rangle$ be the solution of the algorithm; that is, the auctioneer sells r_i units to agent a_i. Then the total price of the solution will be: $P_n(S) = \sum_{i=1}^{n} P_i(r_i)$. Thus, we have to prove that:

$$n \cdot \sum_{i=1}^{n} P_i(r_i) \geq P_n(O) \tag{3.16}$$

Or, equivalently, for all other supply allocations $\langle t_1, t_2, .., t_n \rangle$ that satisfy the supply constraint, their total price will be no more than n times the total price of the solution of our algorithm. That is, $\forall\ t_1, t_2, .., t_n$ such that: $0 \leq t_i \leq u_i$, $\forall 1 \leq i \leq n$ and $\sum_{i=1}^{n} t_i \leq q$, then:

$$n \cdot \sum_{i=1}^{n} P_i(r_i) \geq \sum_{i=1}^{n} P_i(t_i)$$

First, we can assume the maximal quantity that each bidder wants to buy is less than the supply of the auctioneer, that is, $u_i \leq q$, $\forall 1 \leq i \leq n$.

Let s $(s \leq n)$ be the number of rounds of the algorithm, or the number of the agents that were selected in the algorithm.

Without loss of generality, suppose agent a_i, $1 \leq i \leq s$ is selected in round i of the algorithm. Thus:

- In round 1, agent 1 is selected. Because the maximal quantity this agent wants to buy is less than the supply of the auctioneer, the auctioneer will sell the agent the maximal quantity it wants. That is, $r_1 = u_1$.

  Also, the unit price of the biggest package of agent 1 will be the biggest one in all the biggest packages offered by the bidders:

  $$p_1(r_1) = \max_{i=1}^{n} p_i(u_i)$$

- In round k $(2 \leq k \leq s)$, agent k is selected. Because the remaining supply of the auctioneer is $(q - \sum_{i=1}^{k-1} r_i)$, the auctioneer will sell the agent the quantity $r_k = min(q - \sum_{i=1}^{k-1} r_i, u_k)$.

  Also, the unit price of the biggest package of agent k will be the biggest one in all the biggest packages offered by the remaining bidders:

  $$p_k(r_k) = \max_{j=k}^{n} p_j(r_j) \tag{3.17}$$

- The total quantity that the auctioneer sells to all the bidders will be equal to its supply (by Theorem 3.4):

$$\sum_{i=1}^{s} r_i = q$$

From that we have the following lemma:

Lemma 3.6. *For all* $1 \leq k \leq n$*:*

$$\sum_{v=1}^{k} r_v p_v(r_v) \geq t_k p_k(t_k)$$

Proof. Consider the three possible cases:

- For the case $k = 1$, we have : $r_1 p_1(r_1) = u_1 p_1(u_1) \geq t_1 p_1(t_1)$.

- For the case $2 \leq k \leq s$, we have:

  For all $1 \leq v \leq k$: $p_v(r_v) = max_{j=v}^{n} p_j(r_j)$ (by (3.17))

$$\Rightarrow p_v(r_v) \geq p_k(r_k) = p_k(min(q - \sum_{i=1}^{k-1} r_i, u_k)) \tag{3.18}$$

  But $min(q - \sum_{i=1}^{k-1} r_i, u_k) \leq u_k$ and p_k satisfies the discount property in (3.3), thus:

$$p_k(min(q - \sum_{i=1}^{k-1} r_i, u_k)) \geq p_k(u_k) \tag{3.19}$$

  From (3.18) and (3.19) we have: $p_v(r_v) \geq p_k(u_k)$, $\forall 1 \leq v \leq k$.

  Thus: $\sum_{v=1}^{k} r_v p_v(r_v) \geq \sum_{v=1}^{k} r_v p_k(r_k) = (\sum_{v=1}^{k} r_v) p_k(u_k)$.

  But $r_k = min(q - \sum_{i=1}^{k-1} r_i, u_k)$

  $\Rightarrow \sum_{v=1}^{k} r_v = min(q, u_k + \sum_{i=1}^{k-1} r_i) \geq u_k$ (as $q \geq u_k$).

  Thus: $\sum_{v=1}^{k} r_v p_v(r_v) \geq u_k p_k(u_k)$.

  But $u_k \geq t_k$, and P_k satisfies the free disposal property in (3.4), so $P_k(u_k) \geq P_k(t_k)$ or $u_k p_k(u_k) \geq t_k p_k(t_k)$.

So we have $\sum_{v=1}^{k} r_v p_v(r_v) \geq t_k p_k(t_k)$.

- For the case $k > s$:

  For all $1 \leq v \leq s$: $p_v(r_v) = max_{j=v}^{n} p_j(r_j)$ (by (3.17))

$$\Rightarrow p_v(r_v) \geq p_k(r_k) = p_k(min(q - \sum_{i=1}^{k-1} r_i, u_k)) \tag{3.20}$$

  But $min(q - \sum_{i=1}^{k-1} r_i, u_k) \leq u_k$ and p_k satisfies the discount property in (3.3), thus:

$$p_k(min(q - \sum_{i=1}^{k-1} r_i, u_k)) \geq p_k(u_k) \tag{3.21}$$

  From (3.20) and (3.21) we have: $p_v(r_v) \geq p_k(u_k)$, $\forall 1 \leq v \leq s$.

  Thus: $\sum_{v=1}^{k} r_v p_v(r_v) \geq \sum_{v=1}^{s} r_v p_v(r_v)$

  $\geq (\sum_{v=1}^{s} r_v) p_k(u_k) = q \cdot p_k(u_k) \geq u_k \cdot p_k(u_k) \geq t_k \cdot p_k(t_k)$

So we have: $\sum_{v=1}^{k} r_v p_v(r_v) \geq t_k p_k(t_k)$, $\forall 1 \leq k \leq n$. □

From lemma 3.6 we have:

$$\sum_{k=1}^{n} \sum_{v=1}^{k} r_v p_v(r_v) \geq \sum_{k=1}^{n} t_k p_k(t_k)$$
$$\Rightarrow \sum_{k=1}^{n} (n + 1 - k) r_k p_k(r_k) \geq \sum_{k=1}^{n} t_k p_k(t_k)$$
$$\Rightarrow n \cdot \sum_{k=1}^{n} r_k p_k(r_k) \geq \sum_{k=1}^{n} t_k p_k(t_k)$$
$$(\text{As } n + 1 - k \leq n, \forall 1 \leq k \leq n)$$

[*Reverse auction case*]

We prove by induction on the number of bidders n.

Base case ($n = 1$):

In the case where $n = 1$ the solution is optimal (because we have only one bid) so it is clear that the proof is correct with $n = 1$.

Inductive step:

Suppose that (3.15) is true for n, we will prove that (3.15) is also true for $n+1$. That is, let $(r_1, r_2, ..., r_{n+1})$ be the supply allocation that the algorithm generates. Then we have to prove that:

$$\sum_{i=1}^{n+1} r_i \cdot p_i(r_i) \leq (n+1) \cdot P_{n+1}(O)$$

Or equivalently, for all other supply allocations $(t_1, t_2, .., t_{n+1})$ that satisfy the demand, their total price is greater than $\frac{1}{n+1}$ times the total price of the supply allocation produced by the algorithm. That is, $\forall\ t_1, t_2, .., t_{n+1}$ such that: $0 \leq t_i \leq u_i$, $\forall 1 \leq i \leq n+1$ and $\sum_{i=1}^{n+1} t_i \geq q$, then:

$$\sum_{i=1}^{n+1} r_i \cdot p_i(r_i) \leq (n+1) \cdot (\sum_{i=1}^{n+1} t_i \cdot p_i(t_i))$$

Proof of inductive step

Without loss of generality, assume that agent a_{n+1} provides the smallest unit price. That is, $p(u_{n+1}) = min_{i=1}^{n+1} p(u_i)$. This means that agent a_{n+1} is selected in the first step of the algorithm and:

$$r_{n+1} = u_{n+1} \tag{3.22}$$

Because supply allocation $\{t_i\}$ satisfies the demand (as in (3.6)), the total quantity that the auctioneer buys from all bidders is not less than the auctioneer's demand:

$$\sum_{i=1}^{n+1} t_i \geq q \tag{3.23}$$

But supply allocation $\{r_i\}$ supplies exactly the demand quantity (by Theorem 1)

$$\Rightarrow \sum_{i=1}^{n+1} r_i = q$$
$$\Rightarrow \sum_{i=1}^{n+1} t_i \geq \sum_{i=1}^{n+1} r_i \text{ (by (3.23))}$$
$$\Rightarrow \sum_{i=1}^{n} t_i \geq \sum_{i=1}^{n} r_i \text{ (as } t_{n+1} \leq u_{n+1} = r_{n+1}, \text{ from (3.22))}$$

Moreover, by inductive hypothesis, (3.15) is true for n agents.

$$\Rightarrow \sum_{i=1}^{n} r_i \cdot p_i(r_i) \leq n \cdot \sum_{i=1}^{n} t_i \cdot p_i(t_i)$$

$$\Rightarrow \sum_{i=1}^{n} r_i \cdot p_i(r_i) \leq n \cdot \sum_{i=1}^{n+1} t_i \cdot p_i(t_i) \tag{3.24}$$

(as $t_{n+1} \cdot p_{n+1}(t_{n+1}) \geq 0$)

Also:

$$u_{n+1} \cdot p_{n+1}(u_{n+1}) \leq \sum_{i=1}^{n+1} t_i \cdot p_{n+1}(u_{n+1})$$

(as $u_{n+1} \leq q \leq \sum_{i=1}^{n+1} t_i$, from (3.23))

But because $p_{n+1}(u_{n+1})$ is the smallest unit price.

$$\Rightarrow u_{n+1} \cdot p_{n+1}(u_{n+1}) \leq \sum_{i=1}^{n+1} t_i \cdot p_i(t_i)$$

or

$$r_{n+1} \cdot p_{n+1}(r_{n+1}) \leq \sum_{i=1}^{n+1} t_i \cdot p_i(t_i) \tag{3.25}$$

From (3.24) and (3.25), we have:

$$\sum_{i=1}^{n+1} r_i \cdot p_i(r_i) \leq (n+1) \cdot (\sum_{i=1}^{n+1} t_i \cdot p_i(t_i))$$

The completion of the inductive step completes our proof.

□

Although multi-unit single-item auctions are not our main target case, this algorithm still represents a contribution in its own right. While Sandholm and Suri's algorithms target the same environment as this, they are only applicable in the specific case where the supply curves are linear [Sandholm and Suri, 2001]. In contrast, our result is applicable to the more general case; that is, sub-additive, free disposal supply curves.

Having dealt with the multi-unit single-item case, the next section generalises the algorithm to the multi-unit combinatorial case.

3.2 Multi-Unit Combinatorial Auctions

To deal with the multi-unit combinatorial case, we need to add one more assumption about the price functions of the items. This is that there exists a number $K > 1$ such that for any price function from any bidder, K units of any item will be more expensive than 1 unit of any other item:

$\forall\ 1 \leq i, j \leq m,\ i \neq j,\ d \in \mathbb{N}$:

$$P_i(r_1, .., r_i + d, .., r_j, .., r_m) \leq P_i(r_1, .., r_i, .., r_j + Kd, .., r_m) \tag{3.26}$$

That is, for any package, if we substitute d units of any item in this package by $K \cdot d$ units of any other item, then the price of the new package will be more expensive or equal to the price of the old package. For example, in the case $K = 3, m = 2, r_1 = 2, r_2 = 4, d = 1$, we have: for any price function, the price of 3 units of item 1 and 4 units of item 2 will be less than or equal to the price of 2 units of item 1 and 7 (i.e. $4 + 3$) units of items 2. We believe this is a realistic assumption because in a competitive market the unit price of any item is always likely to be within a finite range; that is, it cannot be arbitrarily high or low.[4] This is especially true in the VO creation context, as the VO's potential partners compete against each other to join the VO and so they would not bid with a too high price (otherwise they would fail in the reverse auction). On the other hand, they cannot bid with a too low price, because of the manufacturing cost of a product/service.

From this, a number of lemmas follow:

Lemma 3.7. *For any package of items, if we replace d units of any item with d units of any other item, then the total price of the new package of items is not bigger than K times the total price of the old package:*

[4] There are however domains where this assumption does not hold, for example, when the price equals or is less than zero. This can occur when, for example, a firm gives free promotion to advertise a new product or to expand its market.

$\forall\ 1 \leq i, j \leq m,\ i \neq j,\ d \in \mathbb{N}$:

$$P_i(r_1, .., r_i + d, .., r_j, .., r_m) \leq K \cdot P_i(r_1, .., r_i, .., r_j + d, .., r_m) \tag{3.27}$$

Proof. We have:

$$\begin{aligned} & P_i(r_1, .., r_i + d, .., r_j, .., r_m) \\ & \leq P_i(r_1, .., r_i, .., r_j + Kd, .., r_m) \text{ (by (3.26))} \end{aligned}$$

But $K > 1 \Rightarrow Kr \geq r$ for all r. Also P_i satisfies the free disposal property (in (3.2)).

$$\begin{aligned} & \Rightarrow P_i(r_1, .., r_i + d, .., r_j, .., r_m) \\ & \leq P_i(Kr_1, .., Kr_i, .., Kr_j + Kd, .., Kr_m) \\ & \leq P_i(r_1, .., r_i, .., r_j + d, .., r_m) + P_i((K-1)r_1, \\ & .., (K-1)r_i, .., (K-1)(r_j + d), .., (K-1)r_m) \\ & \leq ... \\ & \leq K \cdot P_i(r_1, .., r_i, .., r_j + d, .., r_m) \end{aligned}$$

(by the discount property in (3.1)) □

Lemma 3.8. *For any two packages, if the total number of units of the first package is not bigger than the total number of units of the second package, then the total price of the first package is not bigger than K^{m-1} times the total price of the second package:*

$\forall 1 \leq i \leq n,\ \forall r_1, r_2, ..., r_m, s_1, s_2, ..., s_m$ *such that*

$$\sum_{j=1}^{m} r_j \leq \sum_{j=1}^{m} s_j$$

Then:

$$P_i(r_1, r_2, ..., r_m) \leq K^{m-1} P_i(s_1, s_2, ..., s_m) \tag{3.28}$$

Proof. Let $d_j = (s_j - r_j)$

$$\Rightarrow \sum_{j=1}^{m} d_j = \sum_{j=1}^{m} s_j - \sum_{j=1}^{m} r_j \geq 0 \tag{3.29}$$

Now there are 2 cases:

- Case 1: $d_i \geq 0$, $\forall 1 \leq i \leq m$. Then we have:

  $$P_i(r_1, r_2, ..., r_m) \leq P_i(s_1, s_2, ..., s_m)$$

  (because P_i satisfies the free disposal property in (3.2))

  $$\Rightarrow P_i(r_1, r_2, ..., r_m) \leq K^{m-1} P_i(s_1, s_2, ..., s_m)$$

- Case 2: There exists a $d_k < 0$.

  Without loss of generality, suppose that $d_m < 0$. Then we have:

  $$P_i(r_1, r_2, ..., r_{m-1}, r_m) \leq K \cdot P_i(r_1 - d_m, r_2, ..., r_{m-1}, s_m)$$

  (by lemma 3.7)

  Let $r_1^{(2)} = r_1 - d_m$, $r_i^{(2)} = r_i$, $\forall\, 2 \leq i \leq m-1$.

  $$\Rightarrow P_i(r_1, r_2, ..., r_{m-1}, r_m) \leq K \cdot P_i(r_1^{(2)}, r_2^{(2)}, ..., r_{m-1}^{(2)}, s_m)$$

  Also:

  $$\sum_{j=1}^{m-1} r_j^{(2)} = \sum_{j=1}^{m-1} r_j - d_m$$
  $$\Rightarrow \sum_{j=1}^{m-1} r_j^{(2)} \leq \sum_{j=1}^{m-1} s_j \text{ (by (3.29))}$$

Repeating the whole step above, it will take at most $m-1$ steps to terminate. Thus, after at most $m-1$ steps, we will have:

$$P_i(r_1, r_2, ..., r_m) \leq K^{m-1} P_i(s_1, s_2, ..., s_m)$$

□

Lemma 3.9. *For any two packages, if the total number of units of the first package is not bigger than the total number of units of the second package, then the average unit price of the first package is not smaller than* $\frac{1}{2K^{m-1}}$ *times the average unit price of the second package:*

$\forall 1 \leq i \leq n$, $r_1, r_2, ..., r_m, s_1, s_2, ..., s_m$ *such that* $\sum_{j=1}^{m} r_j \leq \sum_{j=1}^{m} s_j$, *then:*

$$2K^{m-1} \cdot \frac{P_i(r_1, r_2, ..., r_m)}{r_1 + r_2 + ... + r_m} \geq \frac{P_i(s_1, s_2, ..., s_m)}{s_1 + s_2 + ... + s_m} \tag{3.30}$$

Proof. Let $k = [\frac{\sum_{j=1}^{m} s_j}{\sum_{j=1}^{m} r_j}]$, that is, k is the integral part of $\frac{\sum_{j=1}^{m} s_j}{\sum_{j=1}^{m} r_j}$.

$$\Rightarrow k \leq \frac{\sum_{j=1}^{m} s_j}{\sum_{j=1}^{m} r_j} < k+1 \tag{3.31}$$

$$\Rightarrow (k+1) \sum_{j=1}^{m} r_j > \sum_{j=1}^{m} s_j$$

$$\Rightarrow K^{m-1} P_i((k+1)r_1, (k+1)r_2, ..., (k+1)r_m) \geq P_i(s_1, s_2, ..., s_m)$$

(by lemma 3.8)

$$\Rightarrow K^{m-1}(k+1) P_i(r_1, r_2, ..., r_m) \geq P_i(s_1, s_2, ..., s_m) \tag{3.32}$$

(by the discount property in (3.1))

Also:

$$\sum_{j=1}^{m} r_j \leq \sum_{j=1}^{m} s_j \Rightarrow \frac{\sum_{j=1}^{m} s_j}{\sum_{j=1}^{m} r_j} \geq 1 \Rightarrow k \geq 1$$

$$\Rightarrow k+1 \leq 2k \leq 2 \cdot \frac{\sum_{j=1}^{m} s_j}{\sum_{j=1}^{m} r_j} \text{ (from (3.31))}$$

$$\Rightarrow 2K^{m-1} \cdot \frac{\sum_{j=1}^{m} s_j}{\sum_{j=1}^{m} r_j} P_i(r_1, r_2, ..., r_m) \geq P_i(s_1, s_2, ..., s_m)$$

(by inequation (3.33))

$$\Rightarrow 2K^{m-1} \cdot \frac{P_i(r_1, r_2, ..., r_m)}{r_1 + r_2 + ... + r_m} \geq \frac{P_i(s_1, s_2, ..., s_m)}{s_1 + s_2 + ... + s_m}$$

□

With these lemmas in place, we can now proceed with the presentation of the generalisation of the single-item algorithm in section 3.1. The algorithm for the general case is presented in figure 3.2.

We can now analyse this algorithm to assess its properties.

Theorem 3.10. *If there is a solution, then this algorithm will find it. That is, if the total of the demands (supplies) of the bidders is larger than the auctioneer's supply (demand), this algorithm will produce an allocation. Also, the total units of the solution will be exactly equal to the auctioneer's supply (demand).*

Proof. In each step, the algorithm for the forward (reverse) case selects exactly one agent from the set of bidders. And if its demand (supply) is less than the auctioneer's remaining supply (demand), the algorithm takes all its demand (supply). Otherwise it takes the quantity that is equal to the remaining supply (demand). So, if the algorithm does not terminate beforehand, it will eventually select all the bidders and take all the demands (supplies). Thus, if the total of the demands (supplies) of the bidders is larger than the auctioneer's supply (demand), the algorithm will produce an allocation.

Moreover, in each step, the algorithm takes at most all the remaining supply (demand). Thus the solution it produces will have the total units being equal to the auctioneer's supply (demand). □

Algorithm 2. At each round $k \geq 1$ repeat the following steps:

- For all i, j, set $u_i^j(k) = \min(u_i^j, q_j(k-1))$.

  $(q_j(0) = q_j)$

  That is, we truncate the demand (supply) function to consider only quantities that are not bigger than the supply (demand). This is because, for the forward auction case, the auctioneer cannot sell more quantity than his supply; for the reverse auction case, in order to minimise the total price, the auctioneer does not need to buy more units than its demand, since the price functions satisfy the free disposal property (inequation (3.2)).

- Find the bidder a_{h_k} such that:

$$\frac{P_{h_k}(u_{h_k}^1(k), u_{h_k}^2(k), ..., u_{h_k}^m(k))}{u_{h_k}^1(k) + u_{h_k}^2(k) + ... + u_{h_k}^m(k)} \text{ is maximal (minimal),}$$

  then select a_{h_k} to provide all its units $(u_{h_k}^1(k), u_{h_k}^2(k), ..., u_{h_k}^m(k))$.

  That is, we consider all the biggest packages offered by the bidders, then choose the package that offers the biggest (smallest) average unit price.

  Note that this is not necessarily the package that offers the biggest (smallest) average in all packages, because a smaller package may have a bigger (smaller) average unit price.

- Repeat the steps with the new set of bidders $A \setminus a_{h_k}$ and new supply (demand) $q_j(k) = q_j(k-1) - u_{h_k}^j(k)$.

FIGURE 3.2: The clearing algorithm for the multi-unit combinatorial case.

Theorem 3.11. *The complexity of the algorithm is* $O(n^2)$

Proof. At each step, it requires $O(n)$ to find the biggest (smallest) element of the set $\{\frac{P_k(u_k^1,u_k^2,...,u_k^m)}{u_k^1+u_k^2+...+u_k^m}\}_{i=1}^n$. So each step has $O(n)$ complexity. As there are at most n steps, the complexity of the algorithm is $O(n^2)$. □

Theorem 3.12. *The solution generated from the algorithm is within a bound* $b = 2n \cdot K^{m-1}$ *from the optimal. That is, let* $P_n(O)$ *be the optimal total price and* $P_n(S)$ *be the total price of the solution of the algorithm. Then:*

$$\frac{P_n(O)}{P_n(S)} \leq 2n \cdot K^{m-1} \text{ (forward auction case)} \tag{3.33}$$

$$\frac{P_n(S)}{P_n(O)} \leq 2n \cdot K^{m-1} \text{ (reverse auction case)} \tag{3.34}$$

Proof. [Forward auction case]

Let $\{r_i^j\}$, $1 \leq i \leq n, 1 \leq j \leq m$ be the solution of the algorithm; that is, the auctioneer sells r_i^j units of item j to agent a_i. We have to prove that:

$$2nK^{m-1} \cdot \sum_{i=1}^{n} P_i(r_i^1, .., r_i^m) \geq P_n(O) \tag{3.35}$$

Or, equivalently, for all other supply allocations $\{t_i^j\}$ that satisfy the supply constraint, their total price will be no more than $2nK^{m-1}$ the total price of the solution of the algorithm. That is, $\forall\, t_i^j$ such that: $0 \leq t_i^j \leq u_i^j$, $\forall 1 \leq i \leq n, 1 \leq j \leq m$ and $\sum_{i=1}^n t_i^j \leq q_j$, then:

$$2nK^{m-1} \cdot \sum_{i=1}^{n} P_i(r_i^1, .., r_i^m) \geq \sum_{i=1}^{n} P_i(t_i^1, .., t_i^m)$$

First, we can assume the maximal quantity of each item that each bidder wants to buy is less than the auctioneer's supply for that item, that is, $\forall 1 \leq i \leq n$, $1 \leq j \leq m$: $u_i^j \leq q_j$. So $u_i^j(1) = u_i^j$.

Let s ($s \leq n$) be the number of rounds of the algorithm, or the number of agents that were selected in the algorithm.

Without loss of generality, suppose agent a_k, $1 \leq k \leq s$ is selected in round k of the algorithm. Thus we have:

$$h_k = k, \forall 1 \leq k \leq s \tag{3.36}$$

$$r_k^j = u_k^j(k) \,, \forall 1 \leq k \leq s \tag{3.37}$$

$$\frac{P_k(r_k^1, .., r_k^m)}{\sum_{j=1}^m r_k^j} = \max_{i=k}^{n} \frac{P_i(u_i^1(k), .., u_i^m(k))}{\sum_{j=1}^m u_i^j(k)} \,, \forall 1 \leq k \leq s \tag{3.38}$$

Also, the total quantity of each item that the auctioneer sells to all the bidders will be equal to its supply for that item (by Theorem 3.10). That is:

$$\sum_{i=1}^{s} r_i^j = q_j \tag{3.39}$$

From the algorithm, we can see that : $u_i^j(k) = \min(u_i^j, q_j - \sum_{v=1}^{k-1} u_{h_v}^j(v))$. Thus, by equation (3.36):

$$u_i^j(k) = \min(u_i^j, q_j - \sum_{v=1}^{k-1} u_v^j(v)) \tag{3.40}$$

From that we have the following lemma:

Lemma 3.13. *For all* $1 \leq k \leq n$*:*

$$2K^{m-1} \cdot \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq P_k(t_k^1, .., t_k^m)$$

Proof. There are two cases:

- Case 1, $1 \leq k \leq s$:

For all $1 \leq v \leq k$ we have:

$$P_v(r_v^1, .., r_v^m) \geq (\sum_{j=1}^{m} r_v^j) \frac{P_k(u_k^1(k), .., u_k^m(k))}{\sum_{j=1}^{m} u_k^j(k)} \text{ (from (3.38))}$$

But $\sum_{j=1}^{m} u_k^j(k) \leq \sum_{j=1}^{m} u_k^j$

$$\Rightarrow 2K^{m-1} \cdot \frac{P_k(u_k^1(k), .., u_k^m(k))}{\sum_{j=1}^{m} u_k^j(k)} \geq \frac{P_k(u_k^1, .., u_k^m)}{\sum_{j=1}^{m} u_k^j}$$

Thus:

$$2K^{m-1} \cdot P_v(r_v^1, .., r_v^m) \geq (\sum_{j=1}^{m} r_v^j) \frac{P_k(u_k^1, .., u_k^m)}{\sum_{j=1}^{m} u_k^j}$$

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq (\sum_{v=1}^{k} \sum_{j=1}^{m} r_v^j) \frac{P_k(u_k^1, .., u_k^m)}{\sum_{j=1}^{m} u_k^j}$$

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq (\sum_{j=1}^{m} \sum_{v=1}^{k} r_v^j) \frac{P_k(u_k^1, .., u_k^m)}{\sum_{j=1}^{m} u_k^j}$$

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq (\sum_{j=1}^{m} \sum_{v=1}^{k} u_v^j(v)) \frac{P_k(u_k^1, .., u_k^m)}{\sum_{j=1}^{m} u_k^j}$$

$$u_k^j(k) = \min(u_k^j, q_j - \sum_{v=1}^{k-1} u_v^j(v)) \text{ (from (3.40))}$$

$$\Rightarrow \sum_{v=1}^{k} u_v^j(v) = \min(u_k^j + \sum_{v=1}^{k-1} u_v^j(v), q_j)$$

But $u_k^j + \sum_{v=1}^{k-1} u_v^j(v) \geq u_k^j$ and $q_j \geq u_k^j$ thus:

$$\sum_{v=1}^{k} u_v^j(v) \geq u_k^j$$

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq (\sum_{j=1}^{m} u_k^j) \frac{P_k(u_k^1, .., u_k^m)}{\sum_{j=1}^{m} u_k^j}$$

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq P_k(u_k^1, .., u_k^m)$$

- Case 2, $k \geq s$: By proving similarly to case 1, we have:

$$2K^{m-1} \cdot \sum_{v=1}^{s} P_v(r_v^1, .., r_v^m) \geq (\sum_{j=1}^{m} \sum_{v=1}^{s} r_v^j) \frac{P_k(u_k^1, .., u_k^m)}{\sum_{j=1}^{m} u_k^j}$$

But $\sum_{v=1}^{s} r_v^j = u_j$

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{s} P_v(r_v^1, .., r_v^m) \geq (\sum_{j=1}^{m} u_k^j) \frac{P_k(u_k^1, .., u_k^m)}{\sum_{j=1}^{m} u_k^j}$$

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{s} P_v(r_v^1, .., r_v^m) \geq P_k(u_k^1, .., u_k^m)$$

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{s} P_v(r_v^1, .., r_v^m) \geq P_k(t_k^1, .., t_k^m)$$

(by the free disposal property in (3.2))

$$\Rightarrow 2K^{m-1} \cdot \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq P_k(t_k^1, .., t_k^m)$$

So we have: $2K^{m-1} \cdot \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq P_k(t_k^1, .., t_k^m)$, $\forall 1 \leq k \leq n$.

□

From lemma 3.13 we have:

$$2K^{m-1} \cdot \sum_{k=1}^{n} \sum_{v=1}^{k} P_v(r_v^1, .., r_v^m) \geq \sum_{k=1}^{n} P_k(t_k^1, .., t_k^m)$$

$$\Rightarrow 2K^{m-1} \cdot \sum_{k=1}^{n} (n+1-k) P_k(r_k^1, .., r_k^m) \geq \sum_{k=1}^{n} P_k(t_k^1, .., t_k^m)$$

$$\Rightarrow 2nK^{m-1} \cdot \sum_{k=1}^{n} P_k(r_k^1, .., r_k^m) \geq \sum_{k=1}^{n} P_k(t_k^1, .., t_k^m)$$

(as $n+1-k \leq n$, $\forall 1 \leq k \leq n$)

[Reverse auction case]

We prove by induction of the number of bidders n.

Base case ($n = 1$):

In the case where $n = 1$ the solution is optimal (because there is only one bid to choose from), so it is clear that the proof is correct with $n = 1$.

Inductive step:

Suppose that (3.34) is true for n, we will prove that (3.34) is also true for $n+1$. That is, let $\{r_i^j\}, 1 \leq i \leq n+1, 1 \leq j \leq m$ be the supply allocation that the algorithm generates. Then we have to prove that:

$$\sum_{i=1}^{n+1} P_i(r_i^1, r_i^2, ..., r_i^m) \leq 2n \cdot K^{m-1} P_{n+1}(O)$$

Or equivalently, for every other supply allocation $\{t_i^j\}$ that satisfies the auctioneer's demand, the total price of $\{r_i^j\}$ is not bigger than $2n \cdot K^{m-1}$ times the total price of $\{t_i^j\}$:

$$\sum_{i=1}^{n+1} P_i(r_i^1, r_i^2, ..., r_i^m) \leq 2(n+1) \cdot K^{m-1} \sum_{i=1}^{n+1} P_i(t_i^1, t_i^2, ..., t_i^m)$$

Proof of inductive step

Without loss of generality, assume that agent a_{n+1} provides the lowest average price in all the biggest packages:

$$\frac{P_{n+1}(u_{n+1}^1, u_{n+1}^2, ..., u_{n+1}^m)}{u_{n+1}^1 + u_{n+1}^2 + ... + u_{n+1}^m} = \min_{i=1}^{n+1} \frac{P_i(u_i^1, u_i^2, ..., u_i^m)}{u_i^1 + u_i^2 + ... + u_i^m} \tag{3.41}$$

This means a_{n+1} is selected in the first step of the algorithm and:

$$r_{n+1}^j = u_{n+1}^j, \text{ for all } 1 \leq j \leq m \tag{3.42}$$

For all $1 \leq j \leq m$, because supply allocation $\{t_i^j\}$ satisfies the auctioneer's demand:

$$\Rightarrow \sum_{i=1}^{n+1} t_i^j \geq q_j \tag{3.43}$$

(by inequation (2.2))

But supply allocation $\{r_i^j\}$ supplies exactly the demand quantity (by Theorem 3.10).

$$\Rightarrow \sum_{i=1}^{n+1} r_i^j = q_j$$
$$\Rightarrow \sum_{i=1}^{n+1} t_i^j \geq \sum_{i=1}^{n+1} r_i^j$$
$$\Rightarrow \sum_{i=1}^{n} t_i^j \geq \sum_{i=1}^{n} r_i^j$$

(as $t_{n+1}^j \leq u_{n+1}^j = r_{n+1}^j$ by (3.42))

Moreover, by inductive hypothesis, (3.34) is true for n agents.

$$\Rightarrow \sum_{i=1}^{n} P_i(r_i^1, r_i^2, ..., r_i^m) \leq 2nK^{m-1} \cdot \sum_{i=1}^{n} P_i(t_i^1, t_i^2, ..., t_i^m)$$

$$\Rightarrow \sum_{i=1}^{n} P_i(r_i^1, r_i^2, ..., r_i^m) \leq 2nK^{m-1} \cdot \sum_{i=1}^{n+1} P_i(t_i^1, t_i^2, ..., t_i^m) \tag{3.44}$$

(because $P_{n+1}(t_{n+1}^1, t_{n+1}^2, ..., t_{n+1}^m) \geq 0$)

Also:

$$\begin{aligned} & P_{n+1}(r_{n+1}^1, r_{n+1}^2, ..., r_{n+1}^m) \\ &= P_{n+1}(u_{n+1}^1, u_{n+1}^2, ..., u_{n+1}^m) \text{ (by (3.42))} \\ &= (\sum_{j=1}^{m} u_{n+1}^j) \cdot \frac{P_{n+1}(u_{n+1}^1, u_{n+1}^2, ..., u_{n+1}^m)}{\sum_{j=1}^{m} u_{n+1}^j} \end{aligned}$$

But $u_{n+1}^j \leq q_j$, for all $1 \leq j \leq m$.

$$\begin{aligned} & \Rightarrow P_{n+1}(r_{n+1}^1, r_{n+1}^2, ..., r_{n+1}^m) \\ & \leq (\sum_{j=1}^{m} q_j) \frac{P_{n+1}(u_{n+1}^1, u_{n+1}^2, ..., u_{n+1}^m)}{\sum_{j=1}^{m} u_{n+1}^j} \\ & \leq (\sum_{j=1}^{m} \sum_{i=1}^{n+1} t_i^j) \frac{P_{n+1}(u_{n+1}^1, u_{n+1}^2, ..., u_{n+1}^m)}{\sum_{j=1}^{m} u_{n+1}^j} \text{ (by (3.43))} \end{aligned}$$

$$\leq \sum_{i=1}^{n+1}(\sum_{j=1}^{m} t_i^j \frac{P_i(u_i^1, u_i^2, ..., u_i^m)}{\sum_{j=1}^{m} u_i^j}) \text{ (because of (3.41))}$$

$$\leq \sum_{i=1}^{n+1}(\sum_{j=1}^{m} t_i^j 2K^{m-1} \frac{P_i(t_i^1, t_i^2, ..., t_i^m)}{\sum_{j=1}^{m} t_i^j}) \text{ (by lemma 3.9)}$$

$$\Rightarrow P_{n+1}(r_{n+1}^1, r_{n+1}^2, ..., r_{n+1}^m) \leq 2K^{m-1} \cdot (\sum_{i=1}^{n+1} P_i(t_i^1, t_i^2, ..., t_i^m)) \tag{3.45}$$

From (3.44) and (3.45) we have:

$$\sum_{i=1}^{n+1} P_i(r_i^1, r_i^2, ..., r_i^m) \leq 2(n+1) \cdot K^{m-1} \sum_{i=1}^{n+1} P_i(t_i^1, t_i^2, ..., t_i^m)$$

The completion of the inductive step completes our proof. □

3.3 Experimental Evaluation

To accompany the theoretical analysis that we have done so far, this section outlines the experimental evaluation of our clearing algorithms to see how they perform in reality. This is because the theoretical analysis is in terms of worst-case, however by doing an experimental analysis we can have a clearer idea of average case and how the algorithms perform in practical scenarios. Specifically, we want to assess how much closer to the optimal are the solutions generated by the algorithms compared to the worst-case bound. We will do the evaluation for both the single-item (subsection 3.3.1) and combinatorial cases (subsection 3.3.2).

3.3.1 Multi-Unit Single-Item Auctions

We implement a problem generator that enables us to evaluate the performance of our algorithms for a range of values that are typical for the types of VO problems in which we are interested (see chapter 6 for relevant examples). However, as there is no standard benchmark in this area, we make our problem generator similar to [Eso *et al.*, 2001],

which is the only previous attempt to generate realistic problems for auctions with bidding functions. Below are the parameters used in our problem generator:

- **NumBids** - number of bidders: varied from 10 to 16.
- **MaxNumSegments** - Maximum number of segments of a curve bid: from 2 to 4.

Given the two parameters above, our problem generator consists of the following steps:

- Generating the call for bids: the requested quantity is randomly selected in the interval $[101, 50 * NumBids]$. This range is chosen because:
  - It makes sure the generated problem has solutions: The auctioneer's requested quantity is less than or equal to $50 * NumBids$, and also the maximum quantity that each bidder is willing to trade is at least 50 (see below) so there is always enough demand (supply) from the bidders.
  - It ensures there are at least 2 winners: The auctioneer's requested quantity is at least 101, and the maximum quantity that each bidder is willing to trade is not bigger than 100, so the number of winners will be at least 2. This is preferable because if the auctioneer's requested quantity is small (for example, less than 50) there will be only one winner and the algorithm will be very straightforward — just choosing the bidder who bid the highest (lowest) price for that quantity of items.
- For bidders $i = 1, ..., NumBids$, construct the bid curve for bidder i by doing the following:
  - The maximum quantity that it is willing to trade (MaxQuantity) is randomly selected in the interval $[50, 100]$. This range is chosen for the above reasons.
  - The number of segments of its curve bid (NumSegs) is randomly selected in the interval $[1, MaxNumSegments]$. This range is chosen so that it adds more variety into the experiment.

- For the first segment: start quantity is: $Start_1 = 1$, end quantity (End_1) is randomly selected in the interval $[1, MaxQuantity - NumSegs + 1]$ and unit price UP_1 is randomly selected in the interval $[100, 102]$.
- For segments $j > 1$: start quantity is segment $j-1$'s end quantity $+1$: $Start_j = End_{j-1}+1$, end quantity (End_j) is randomly selected in the interval $[Start_j, MaxQuantity - NumSegs + j]$ (except that $End_j = MaxQuantity$ when $j = NumSegs$) and unit price $UP_j = UP_{j-1} * (1 - DiscountRate)$ where $DiscountRate$ is randomly selected in the interval $[0.01, 0.05]$.

  The last two steps are carried out in these ways to ensure that the generated bidding functions satisfy the free disposal and discount properties, while still adding a degree of variety into the experiment.

We now turn to the results. We ran the algorithm for 40 generated problems and record the bounds from the optimal of the generated solutions. The results are presented in figures 3.3, 3.4 (forward auction case) and figures 3.5, 3.6 (reverse auction case) as box plots[5]. The number of runs (40) guarantees us a 95% confidence interval, if we accept the bound to have a precision of:

- Plus or minus 0.001 for the box plots in figure 3.3 and the first box plot in figure 1.2
- Plus or minus 0.002 for the second and third box plots in figure 3.4
- Plus or minus 0.0001 for the first box plot in figure 3.5
- Plus or minus 0.00002 for the second and third box plots in figure 3.5
- Plus or minus 0.0005 for the fourth box plot in figure 3.5 and the box plots in figure 3.6.

As can be seen, all of the bounds are very close to 1, specifically, they are in the interval $[1, 1.02]$. This is significantly lower than the theoretically proved bound which

[5] A box plot is a graphical representation of a set of one-dimensional data and comprises of a central box (bounded below by the lower hinge, bounded above by the upper hinge, and with a central line showing the median), two protruding lines (whiskers) extending from the central box with lengths no larger than 1.5 times the length of the box, and outliers marked individually data points that lie beyond the whiskers.

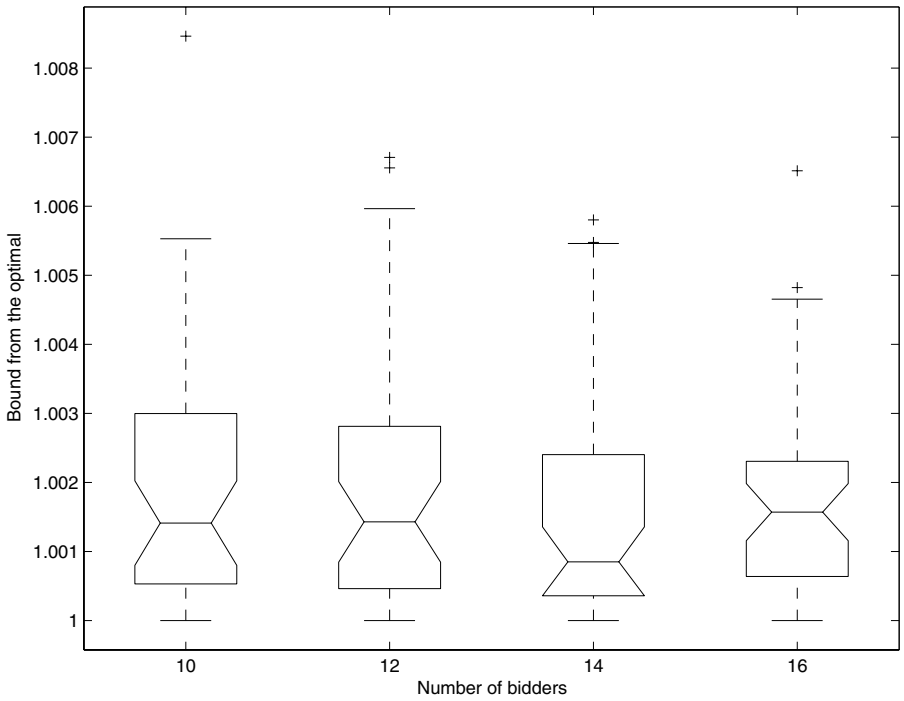


FIGURE 3.3: The experimental result of our algorithm for multi-unit single-item forward auctions (varying the number of bidders).

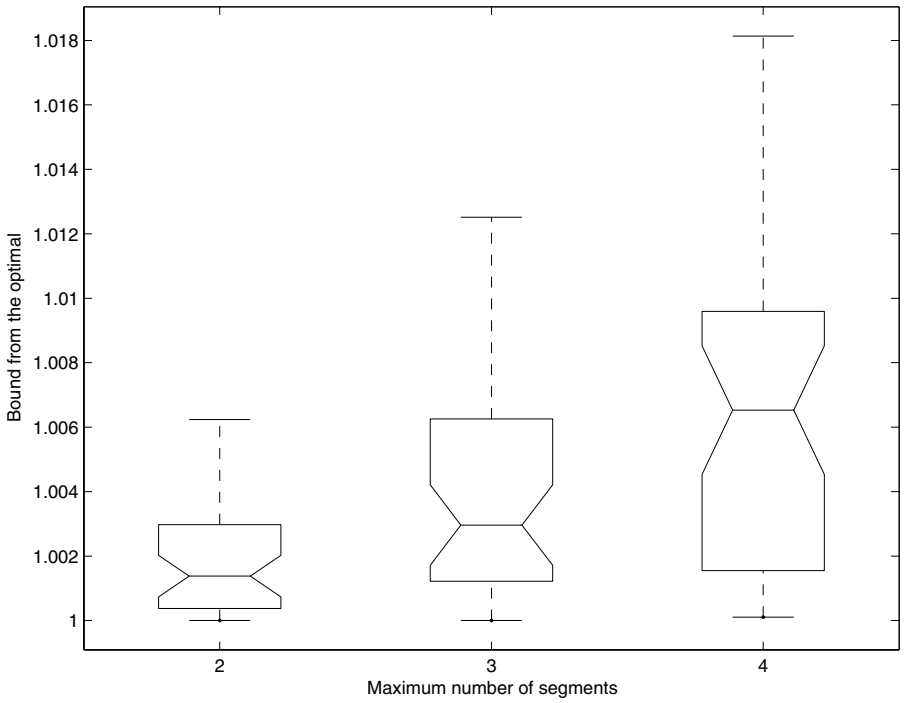


FIGURE 3.4: The experimental result of our algorithm for multi-unit single-item forward auctions (varying the maximum number of segments).

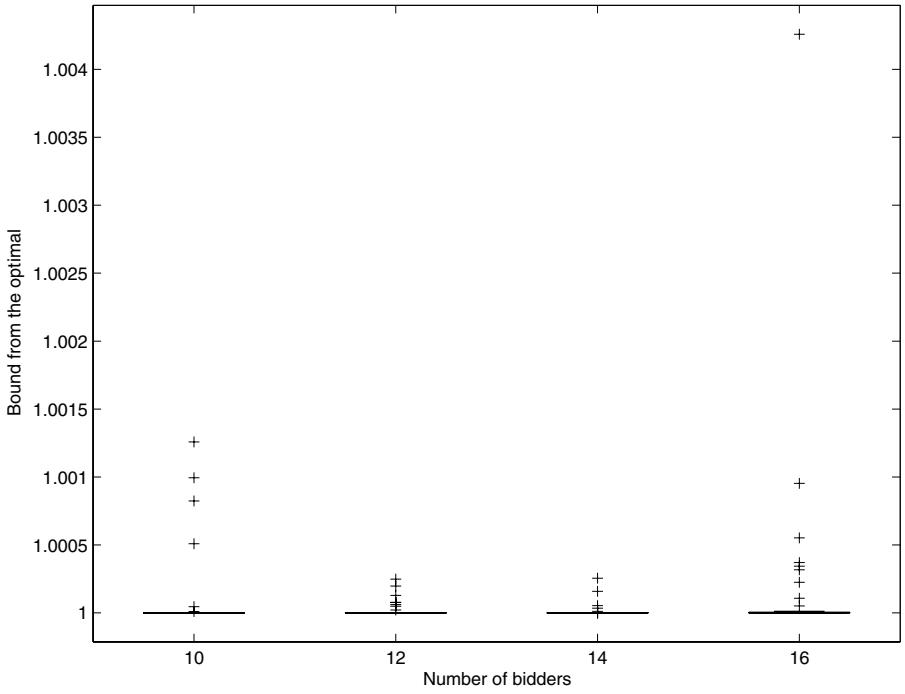


FIGURE 3.5: The experimental result of our algorithm for multi-unit single-item reverse auctions (varying the number of bidders).

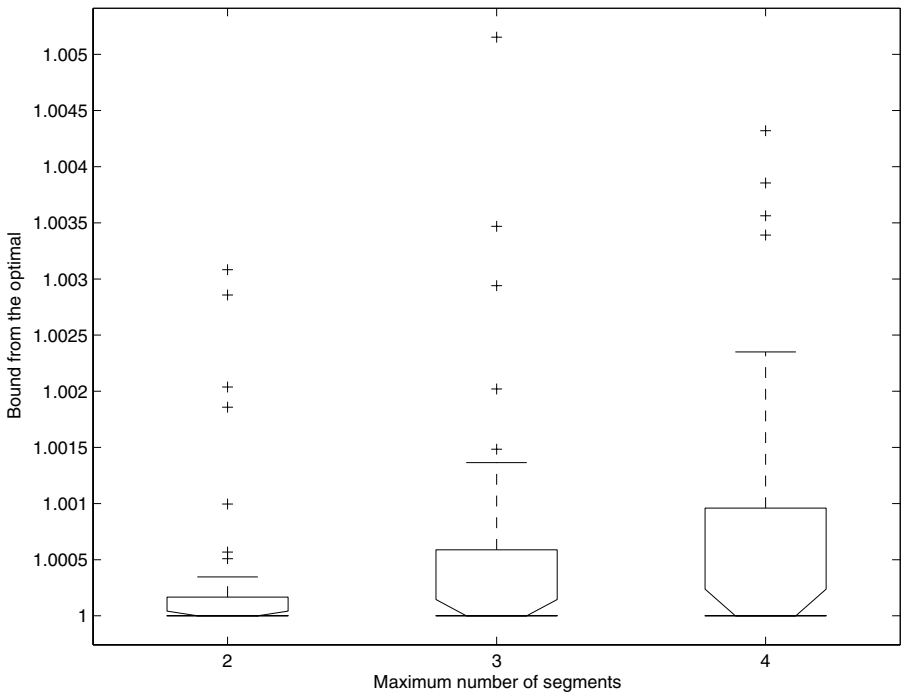


FIGURE 3.6: The experimental result of our algorithm for multi-unit single-item reverse auctions (varying the maximum number of segments).

is $NumBids$ (i.e., 10, 12, 14 or 16 in this experiment). This suggests that in many practical cases, our algorithm performs significantly better than the theoretical proved worst-case analysis.

Moreover, by performing ANOVA tests, we can make a number of additional observations. Specifically, in the forward case we can determine that there is no significant difference between the different numbers of bidders. However, there is a significant difference between the different maximum numbers of segments. This suggests that the maximum number of segments has a bigger influence on the bound from the optimal of the solutions generated by our algorithms. Turning now to the reverse case, we can determine there is no significant difference between the different numbers of bidders nor between the different maximum numbers of segments. Thus no hypotheses can be made about the effect of these parameters on the values of the bound.

3.3.2 Multi-Unit Combinatorial Auctions

As in the multi-unit single-item case, we implement a problem generator that enables the performance of our algorithms to be evaluated in a range of environments. Below are the parameters used in this problem generator:

- **NumBids** — number of bidders: varied from 5 to 8.
- **NumItems** — number of items: from 2 to 4.
- **MaxNumSegments** — maximum number of segments of a curve bid: from 2 to 4.

Given the above parameters, our problem generator consists of the following steps:

- Generating the call for bids: the requested quantity is randomly selected in the interval $[101, 50 * NumBids]$. Again, this range is chosen for the same reasons given above in the single-item case (except that in this case, we cannot guarantee that the generated problem has solutions, because a bidder may want to trade only one or few items, not all of them).

- For bidders $i = 1, ..., NumBids$: The actual number of items that bidder i wants to trade is chosen randomly from the range $[1, NumItems]$. Now for any item k that bidder i is willing to trade, construct the bid curve for bidder i by doing the following:

  - The maximum quantity that it is willing to trade (MaxQuantity) is randomly selected in the interval $[50, 100]$.
  - The number of segments of its curve bid (NumSegs) is randomly selected in the interval $[1, MaxNumSegments]$.
  - For the first segment: start quantity is: $Start_1^k = 1$, end quantity (End_1^k) is randomly selected in the interval $[1, MaxQuantity - NumSegs + 1]$ and unit price UP_1^k is randomly selected in the interval $[100, 102]$.
  - For segments $j > 1$: start quantity is segment $j-1$'s end quantity $+1$: $Start_j^k = End_{j-1}^k + 1$, end quantity ($End_j^k$) is randomly selected in the interval $[Start_j^k, MaxQuantity - NumSegs + j]$ (except that $End_j^k = MaxQuantity$ when $j = NumSegs$) and unit price $UP_j^k = UP_{j-1}^k * (1 - DiscountRate)$ where $DiscountRate$ is randomly selected in the interval $[0.01, 0.05]$.
  - For all bidders, set the correlation values $w_i(NumSegs_i^1, NumSegs_i^2, ..., NumSegs_i^{NumItems})$ to be a random value in the interval $[MinVal, 1]$ where $MinVal$ is the smallest possible value to satisfy the free disposal assumption.

    Again, the last three steps are carried out in these ways to ensure that the generated bidding functions satisfy the free disposal and discount properties, while still adding a degree of variety into the experiment.

- Check if the generated problem has solutions, if it does not, re-generate it[6].

We now turn to the results. As before, we ran the algorithm for 40 generated problems and record the bounds from the optimal of the generated solutions. The results are presented in figures 3.7, 3.8 and 3.9 (forward auction case) and figures 3.10, 3.11 and 3.12 (reverse auction case). Again, the number of runs (40) guarantees us a 95% confidence

[6] In some reverse auction cases the generated problem does not have any solutions because the total supply from the bidders does not meet the auctioneer's demand

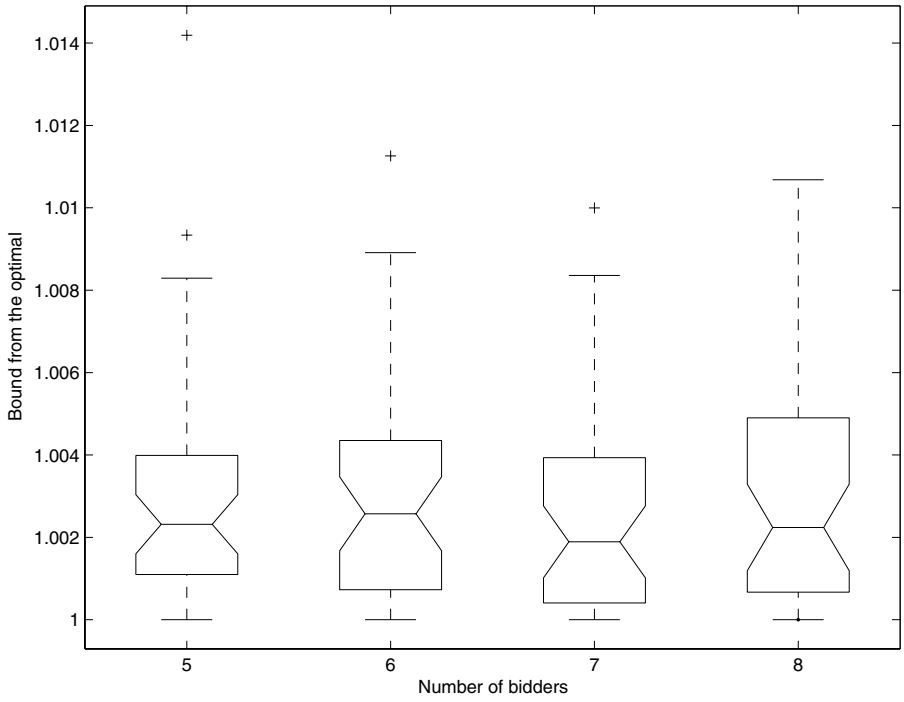


FIGURE 3.7: The experimental result of our algorithm for multi-unit combinatorial forward auctions (varying the number of bidders).

interval, if we accept the bound to have some specific precisions (these are broadly as per subsection 3.3.1).

As can be seen, all of the bounds are very close to 1, specifically, in the interval $[1, 1.025]$. Again this is significantly lower than the theoretically proved bound which is $2 * NumBids * K^{NumItems-1}$ (K is a constant). This suggests that in many cases, our algorithm is likely to perform significantly better than the theoretical proved worst-case analysis.

On the other hand, the ANOVA tests performed on this data show no significant difference between the different tested values of the number of bidders, the maximum number of segments or the number of items, so no hypotheses can be made about the effect of these parameters on the values of the bound.

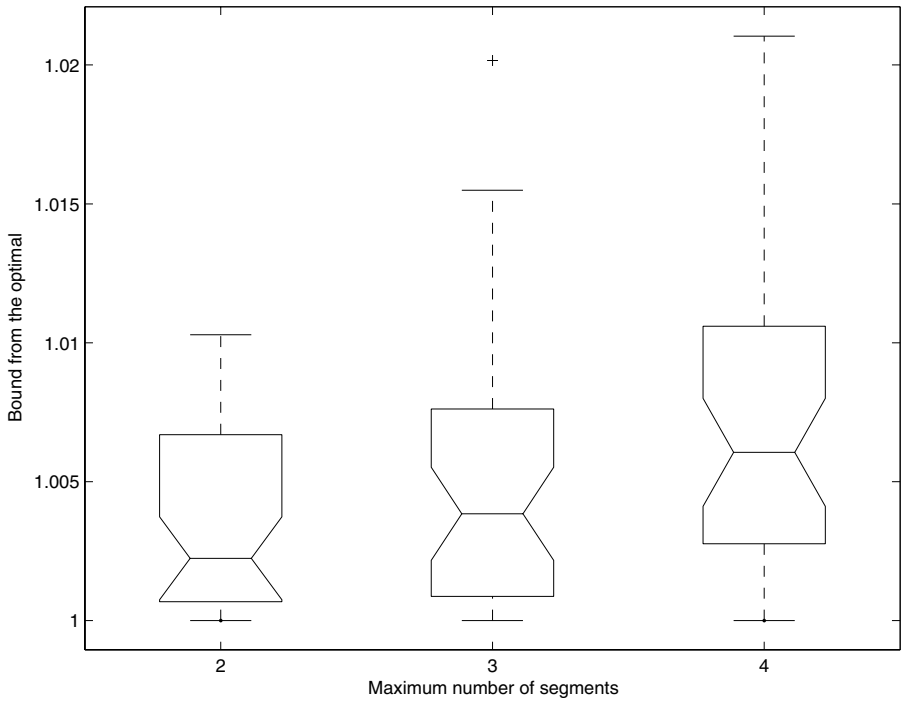


FIGURE 3.8: The experimental result of our algorithm for multi-unit combinatorial forward auctions (varying the maximum number of segments).

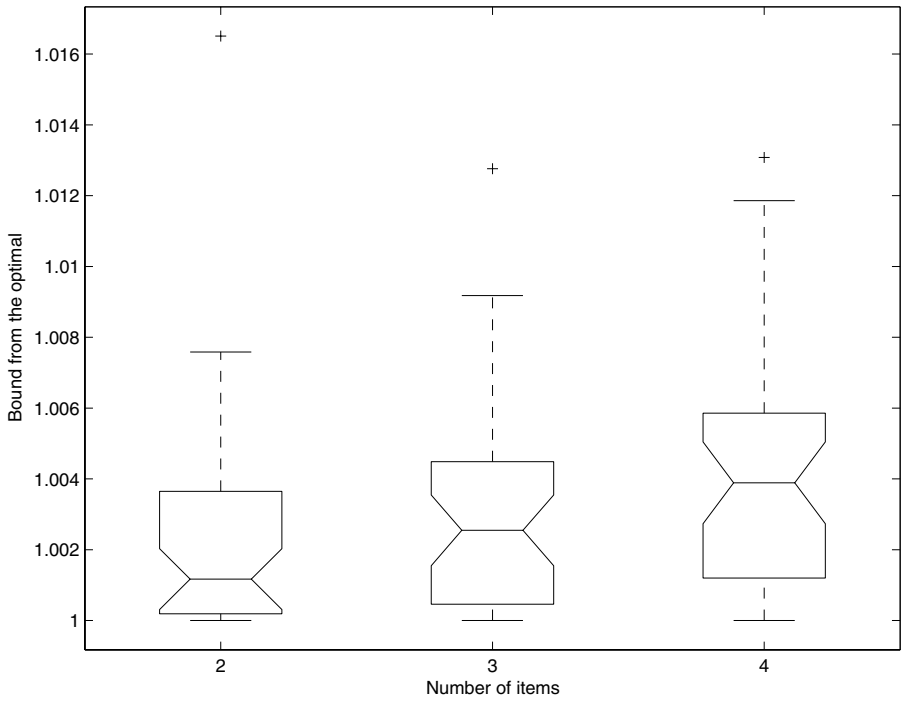


FIGURE 3.9: The experimental result of our algorithm for multi-unit combinatorial forward auctions (varying the number of items).

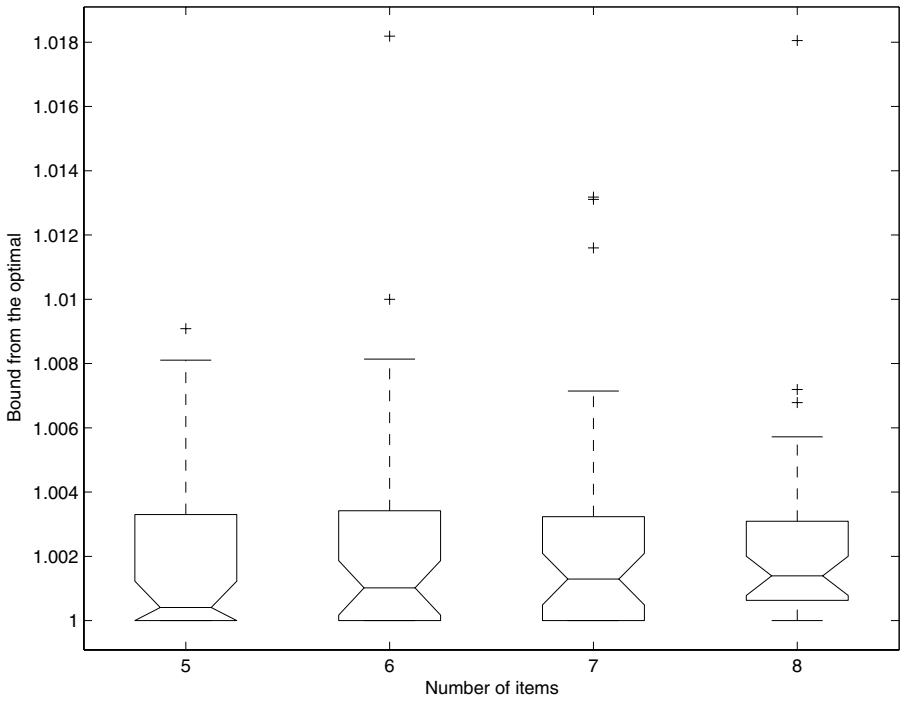


FIGURE 3.10: The experimental result of our algorithm for multi-unit combinatorial reverse auctions (varying the number of bidders).

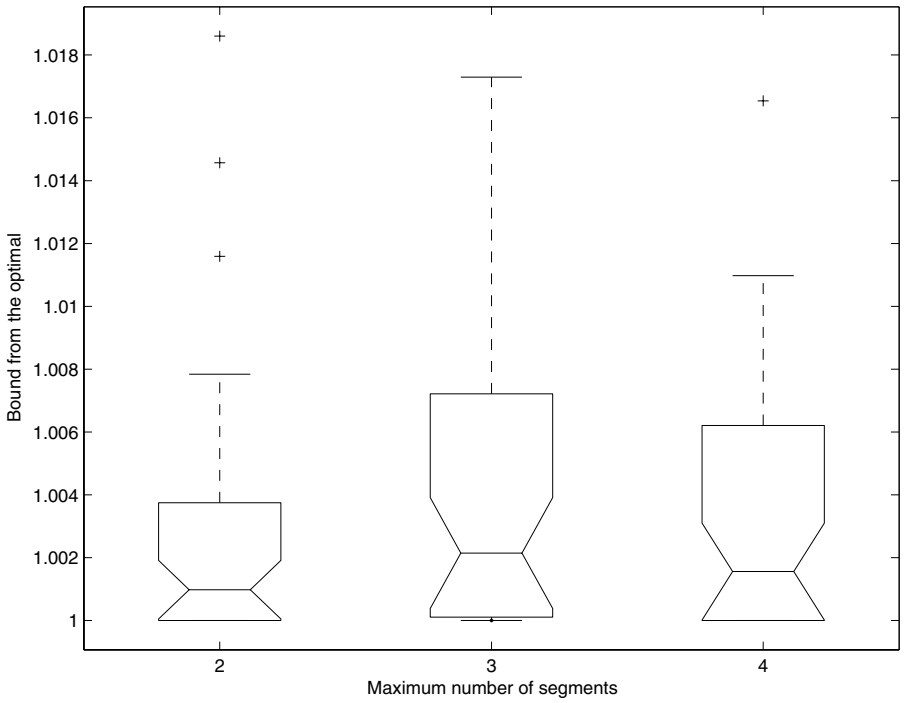


FIGURE 3.11: The experimental result of our algorithm for multi-unit combinatorial reverse auctions (varying the maximum number of segments).

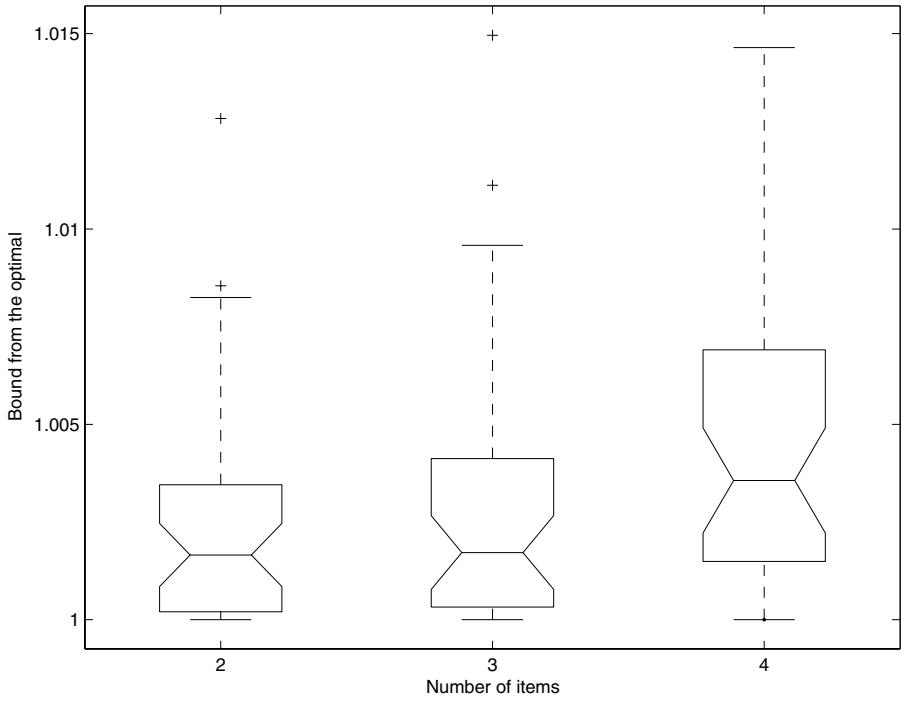


FIGURE 3.12: The experimental result of our algorithm for multi-unit combinatorial reverse auctions (varying the number of items).

3.4 Summary

In this chapter we developed, for the first time, polynomial algorithms for clearing multi-unit combinatorial auctions with demand/supply functions. While previous work has concentrated on single-item auctions with demand/supply curves or combinatorial auctions with atomic propositions, we generalised the problem to multi-unit single-item and multi-unit combinatorial auctions with demand/supply functions. For this very general case, we showed that our algorithms are of polynomial complexity and can generate solutions that are within a bound of the optimal. We then showed our experimental results for both the multi-unit single-item and multi-unit combinatorial cases. For the problems we considered, our algorithms produce solutions that are within a bound from the optimal that is much smaller than the theoretically proved bound. Specifically, all of the bounds are within the range $[1, 1.025]$, while the theoretical bound is n in the multi-unit single-item case, and is $2nK^{m-1}$ in the multi-unit combinatorial case (n is the number of bidders, m is the number of items, and K is a constant). Moreover, we believe the generalisation to multi-unit single-item and multi-unit combinatorial auctions

with demand/supply functions is an important step toward realising the full application potential of combinatorial auctions since it enables us to deal with a maximally flexible and efficient scheme in a computationally tractable manner.

Chapter 4

Optimal Auction Clearing Algorithms

In contrast to chapter 3, this chapter presents optimal clearing algorithms for multi-unit single-item and multi-unit combinatorial forward and reverse auctions with demand/supply functions. Specifically, we consider two classes of demand/supply function. Firstly, those in which the demand/supply curves for each individual commodity are composed of many linear segments (i.e. *piece-wise linear functions*) (section 4.1). Price functions that are composed of linear segments are common in business, thus this is a very appropriate auction settings for the VO context and for e-business in general. Secondly, those in which the function indicating the price for adding one more single unit into a package is monotonic (called *monotonic one-unit-difference functions*) (section 4.2). This second class is not particularly common, however, it is considered because it is amenable to the same solution as piece-wise linear and it covers another portion of the general auction clearing problem space. For each of these cases, we analyse the complexity of our algorithms and prove that they are guaranteed to find the optimal allocation.

As mentioned in section 2.2, it is impossible to find a polynomial algorithm that is guaranteed to find the optimal allocation, unless P = NP. In this section, we concentrate on optimality and so, necessarily, our algorithms are not polynomial.

4.1 Piece-wise Linear Supply/Demand Curve Bids

In this section, we consider the case where:

$$P_i(r_1, r_2, ..., r_m) = \omega_i(t_1, t_2, ..., t_m) \cdot (\sum_{j=1}^{m} P_i^j(r_j))$$

where P_i^j is the price function of agent i for item j, in the form of a piecewise linear curve (i.e. the function's graph is composed of many segments, each of which is linear), t_j is the segment number of P_i^j that r_j belongs to and

$$\omega_i : \{(t_1, t_2, ..., t_m) | t_j \text{ is a segment number of } P_i^j\} \rightarrow \mathbb{Q}^*$$

is the function that expresses correlations between items in the set S.

More precisely, each piece-wise linear function P_i^j is composed of N_i^j linear segments, numbered from 1 to N_i^j. Each individual segment with segment number l, $1 \leq l \leq N_i^j$, is described by a starting quantity $s_{i,l}^j$ and an ending quantity $e_{i,l}^j$, a unit price $\pi_{i,l}^j$ and a fixed price $c_{i,l}^j$, with the meaning that: bidder i wants to trade any r units of item j, $s_{i,l}^j \leq r \leq e_{i,l}^j$ with the price:

$$P = \pi_{i,l}^j \cdot r + c_{i,l}^j$$

Note that the segments are not required to be continuous; that is, $(s_{i,l+1}^j - e_{i,l}^j)$ may not equal 1. Also, for convenience, we call segment number 0 the segment in which the starting quantity, the ending quantity, the unit price and the fixed price are all equal to 0. Thus, the number of segments of P_i^j, including this special segment, will equal $N_i^j + 1$.

The correlation function ω_i has many potential uses in real-life scenarios. For example, suppose bidder i, selling 3 items (1, 2 and 3), wants to express things like "I am willing to sell r_1 units of item 1 and r_2 units of item 2 together with a price p, but not separately". Using our correlation function, this can be expressed by adding segments t_1 and t_2, which contain only r_1 and r_2, to the functions P_i^1 and P_i^2, respectively, then giving $\omega_i(t_1, t_2, t_3)$ a very small value, for every t_3, and giving $P_i^1(r_1)$ and $P_i^2(r_2)$ very big values. This way, the auctioneer will never choose to buy r_1 or r_2 separately.

For convenience in presentation, from this section on, we will use the following terms.

Definition 4.1. A *valid allocation* is a supply allocation that completely satisfies the demand constraint.

Definition 4.2. A supply allocation $\langle t_i^j \rangle$ is *not less profitable than* a supply allocation $\langle r_i^j \rangle$ if the former brings the auctioneer an equal or bigger revenue than the latter. That is:

$$\begin{aligned} P(\langle t_i^j \rangle) \geq P(\langle r_i^j \rangle) \text{ (forward case)} \\ P(\langle t_i^j \rangle) \leq P(\langle r_i^j \rangle) \text{ (reverse case)} \end{aligned} \tag{4.1}$$

According to this definition of profitability, the most profitable valid allocation optimises the auctioneer's total revenue. Thus, this is what our algorithms aim to find. We first consider the multi-unit single-item case (subsection 4.1.1), before moving onto the combinatorial case (subsection 4.1.2).

4.1.1 Multi-Unit Single-Items

Using the notation from the previous section, the single-item case can be re-formulated as follows. Let n be the number of bidders. The auctioneer has a supply (demand) q. Each bidder i submits bids in the form of a piece-wise linear demand (supply) curve: $P_i : N \rightarrow R$, which is composed of N_i linear segments. Each segment l, $0 \leq l \leq N_i$ is described by a starting quantity $s_{i,l}$ and an ending quantity $e_{i,l}$, a unit price $\pi_{i,l}$ and a fixed price $c_{i,l}$.

Definition 4.3. The dominant set D is the set of all allocations $(r_1, r_2, .., r_n)$ such that there exists a k, $1 \leq k \leq n$, such that all $r_{\lambda_1}, ..., r_{\lambda_{k-1}}$ equal the ending quantity of the segments that they belong to, and all $r_{\lambda_{k+1}}, ..., r_{\lambda_n}$ equal the starting quantity of the

segments that they belong to:[1]

$$\begin{cases} r_{\lambda_i} &= e_{\lambda_i,t_{\lambda_i}}, \forall 1 \leq i \leq k-1 \\ r_{\lambda_i} &= s_{\lambda_i,t_{\lambda_i}}, \forall k+1 \leq i \leq n \\ r_{\lambda_k} &= q - \sum_{i=1,i\neq k}^{n} r_{\lambda_i} \end{cases}$$

where:

- t_i is the segment on P_i that r_i belongs to. That is, $s_{i,t_i} \leq r_i \leq e_{i,t_i}$.
- $(\lambda_i)_{i=1}^n$ is any permutation of $(1, 2, ..., n)$ such that $\{\pi_{\lambda_1,t_{\lambda_1}}\}_{i=1}^n$ is sorted decreasingly (increasingly):[2]

$$\pi_{\lambda_1,t_{\lambda_1}} \geq \pi_{\lambda_2,t_{\lambda_2}} \geq ... \geq \pi_{\lambda_n,t_{\lambda_n}} \text{ (forward case)}$$
$$\pi_{\lambda_1,t_{\lambda_1}} \leq \pi_{\lambda_2,t_{\lambda_2}} \leq ... \leq \pi_{\lambda_n,t_{\lambda_n}} \text{ (reverse case)}$$

From this, a number of lemmas follow:

Lemma 4.4. *For every allocation $(r_1, r_2, ..., r_n)$ there exists an allocation in the dominant set D that is not less profitable than it.*

Proof. Let $(r_1, r_2, ..., r_n)$ be an allocation. Let t_i be the segment that r_i belongs to. Suppose $(\lambda_i)_{i=1}^n$ is a permutation of $(1, 2, ..., n)$ such that:

$$\pi_{\lambda_1,t_{\lambda_1}} \geq \pi_{\lambda_2,t_{\lambda_2}} \geq ... \geq \pi_{\lambda_n,t_{\lambda_n}} \text{ (forward case)}$$
$$\pi_{\lambda_1,t_{\lambda_1}} \leq \pi_{\lambda_2,t_{\lambda_2}} \leq ... \leq \pi_{\lambda_n,t_{\lambda_n}} \text{ (reverse case)} \quad (4.2)$$

Step 1: we prove that there exists an allocation $\langle r_i^{(1)} \rangle$, that is not less profitable than $\langle r_i \rangle$, where $r_i^{(1)}$ belongs to segment t_i of P_i, $\forall 1 \leq i \leq n$ and, either $r_{\lambda_1}^{(1)} = e_{\lambda_1,t_{\lambda_1}}$ or:

$$\begin{cases} r_{\lambda_i}^{(1)} &= s_{\lambda_i,t_{\lambda_i}}, \forall 2 \leq i \leq n \\ r_{\lambda_1}^{(1)} &= q - \sum_{i=2}^{n} r_{\lambda_i}^{(1)} \end{cases}$$

[1] There may be many dominant sets D, as there may exist many permutations $(\lambda_i)_{i=1}^n$.

[2] There may exist many such permutations $(\lambda_i)_{i=1}^n$, as there may be many ways to sort the set $\{\pi_{i,t_i}\}_{i=1}^n$.

Let us consider the case where $r_{\lambda_1} < e_{\lambda_1,t_{\lambda_1}}$ and there exists a k, $2 \leq k \leq n$, such that $r_{\lambda_k} > s_{\lambda_k,t_{\lambda_k}}$.

Consider the allocation $(r'_1, r'_2, ..., r'_n)$ where:

$$\begin{cases} r'_{\lambda_1} &= r_{\lambda_1} + 1 \\ r'_{\lambda_k} &= r_{\lambda_k} - 1 \\ r'_{\lambda_i} &= r_{\lambda_i}, \forall 1 \leq i \leq n, i \neq 1, i \neq k \end{cases}$$

Because $r_{\lambda_1} < e_{\lambda_1,t_{\lambda_1}}$ and $r_{\lambda_k} > s_{\lambda_k,t_{\lambda_k}}$, r'_i belongs to segment t_i of P_i, $\forall 1 \leq i \leq n$.

Now let us compare the revenues of two allocations $\langle r_i \rangle_{i=1}^n$ and $\langle r'_i \rangle_{i=1}^n$. We have:

$$\begin{aligned} & P(\langle r_i \rangle) - P(\langle r'_i \rangle) \\ &= \sum_{i=1}^{n} (P_{\lambda_i}(r_{\lambda_i})) - \sum_{i=1}^{n} (P_{\lambda_i}(r'_{\lambda_i})) \\ &= P_{\lambda_1}(r_{\lambda_1}) + P_{\lambda_k}(r_{\lambda_k}) \\ &\quad -(P_{\lambda_1}(r_{\lambda_1} + 1) + P_{\lambda_k}(r_{\lambda_k} - 1)) \\ &= (\pi_{\lambda_1,t_{\lambda_1}} \cdot r_{\lambda_1} + c_{\lambda_1,t_{\lambda_1}}) + (\pi_{\lambda_k,t_{\lambda_k}} \cdot r_{\lambda_k} + c_{\lambda_k,t_{\lambda_k}}) \\ &\quad -(\pi_{\lambda_1,t_{\lambda_1}} \cdot (r_{\lambda_1} + 1) + c_{\lambda_1,t_{\lambda_1}}) \\ &\quad -(\pi_{\lambda_k,t_{\lambda_k}} \cdot (r_{\lambda_k} - 1) + c_{\lambda_k,t_{\lambda_k}}) \\ &= \pi_{\lambda_k,t_{\lambda_k}} - \pi_{\lambda_1,t_{\lambda_1}} \end{aligned}$$

But by inequation (4.2):

$$\pi_{\lambda_k,t_{\lambda_k}} \leq \pi_{\lambda_1,t_{\lambda_1}} \text{ (forward case)}$$

$$\pi_{\lambda_k,t_{\lambda_k}} \geq \pi_{\lambda_1,t_{\lambda_1}} \text{ (reverse case)}$$

Thus:

$$P(\langle r_i \rangle) \leq P(\langle r'_i \rangle) \text{ (forward case)}$$

$$P(\langle r_i \rangle) \geq P(\langle r'_i \rangle) \text{ (reverse case)}$$

This means by taking 1 more unit from bidder λ_1 and taking 1 less unit from bidder λ_k, we will have a new allocation that is not less profitable than the original one.

Repeating the above process, we will always get a new allocation that is not less profitable than the original one. Eventually we get an allocation $\langle r_i^{(1)} \rangle$, that is not less profitable than the original one, where $r_i^{(1)}$ belongs to segment t_i of P_i, $\forall 1 \leq i \leq n$, and either $r_{\lambda_1}^{(1)} = e_{\lambda_1,t_{\lambda_1}}$ or:

$$\begin{cases} r_{\lambda_i}^{(1)} & = & s_{\lambda_i,t_{\lambda_i}}, \forall 2 \leq i \leq n \\ r_{\lambda_1}^{(1)} & = & q - \sum_{i=2}^{n} r_{\lambda_i}^{(1)} \end{cases}$$

Step 2: In the case if $r_{\lambda_1}^{(1)} = e_{\lambda_1,t_{\lambda_1}}$ and $r_{\lambda_1}^{(1)} < q - \sum_{i=2}^{n} s_{\lambda_i,t_{\lambda_i}}$, by repeating the above step, there exists an allocation $\langle r_i^{(2)} \rangle$, that is not less profitable than $\langle r_i \rangle$, where:

- $r_i^{(2)}$ belongs to segment t_i of P_i, $\forall 1 \leq i \leq n$.
- $r_{\lambda_1}^{(2)} = r_{\lambda_1}^{(1)} = e_{\lambda_1,t_{\lambda_1}}$
- Either $r_{\lambda_2}^{(2)} = e_{\lambda_2,t_{\lambda_2}}$ or:

$$\begin{cases} r_{\lambda_i}^{(2)} & = & s_{\lambda_i,t_{\lambda_i}}, \forall 3 \leq i \leq n \\ r_{\lambda_2}^{(2)} & = & q - \sum_{i=1,i\neq 2}^{n} r_{\lambda_i}^{(2)} \end{cases}$$

By repeating the above steps again and again, we will finally stop at some step k, $1 \leq k \leq n$ and get an allocation $\langle r_i^{(k)} \rangle$, that is not less profitable than $\langle r_i \rangle$, where $r_i^{(k)}$ belongs to segment t_i of P_i, $\forall 1 \leq i \leq n$, and:

$$\begin{cases} r_{\lambda_i}^{(k)} & = & e_{\lambda_i,t_{\lambda_i}}, \forall 1 \leq i \leq k-1 \\ r_{\lambda_i}^{(k)} & = & s_{\lambda_i,t_{\lambda_i}}, \forall k+1 \leq i \leq n \\ r_{\lambda_k}^{(k)} & = & q - \sum_{i=1,i\neq k}^{n} r_{\lambda_i}^{(k)} \end{cases}$$

□

The above lemma leads directly to the following corollary:

Corollary 4.5. *The dominant set D must contain an optimal allocation.*

Lemma 4.6. *The number of elements in the set D is not more than $\prod_{i=1}^{n}(N_i+1)$.*

Proof. For each tuple $\langle t_i\rangle_{i=1}^{n}$, in which t_i is a segment on P_i, there exists at most one k,[3] so the number of elements in the set D is not more than the number of such tuples. But the number of tuples $\langle t_i\rangle_{i=1}^{n}$ is $\prod_{i=1}^{n}(N_i+1)$. Thus:

$$|D| \leq \prod_{i=1}^{n}(N_i+1)$$

□

With these lemmas in place, we can now present our algorithm for the single-item case (see figure 4.1). Basically, the algorithm searches through all the allocations of the set D and chooses the most profitable valid one. We can now analyse the algorithm to assess its properties.

Theorem 4.7. *The algorithm is guaranteed to find an optimal allocation.*

Proof. The algorithm searches all the allocations of the dominant set D. Also, by corollary 4.5, the dominant set D contains an optimal allocation. Thus the algorithm is guaranteed to find an optimal allocation. □

Theorem 4.8. *The complexity of the algorithm is $O(n\cdot(K+1)^n)$, where K is the upper bound on the number of segments of P_i.*

Proof. The number of allocations searched by the algorithm is equal to the number of elements of the dominant set. By lemma 4.6, the number of elements of the dominant set is not more than $\prod_{i=1}^{n}(N_i+1) \leq (K+1)^n$. Also, it takes $O(\log n)$ to sort $\{\pi_{i,t_i}\}$ and $O(n)$ to find k, so the complexity of the algorithm is $O(n\cdot(K+1)^n)$. □

Having dealt with the multi-unit single-item case, the next subsection generalises the algorithm to the multi-unit combinatorial case.

[3] There may be more than one k, for example, in the case where $s_{i,t_i} = e_{i,t_i}$ for every i, but in such cases, it does not matter which k is chosen.

Algorithm 3. For every tuple $\langle t_i \rangle_{i=1}^{n}$ such that t_i is a segment on P_i:

- If $\sum_{i=1}^{n} e_{i,t_i} < q$ or $\sum_{i=1}^{n} s_{i,t_i} > q$:
  Continue; // Jump to the next $\langle t_i \rangle$ tuple.
- Sort $\{\pi_{i,t_i}\}$ decreasingly (increasingly).
- For $k = 1$ to n do:
  - If $\sum_{i=1}^{k} e_{i,t_i} + \sum_{i=k+1}^{n} s_{i,t_i} > q$:
    - Set:
      $$\begin{cases} r_i &= e_{i,t_i}, \forall 1 \leq i \leq k-1 \\ r_i &= s_{i,t_i}, \forall k+1 \leq i \leq n \\ r_k &= q - \sum_{i=1, i \neq k}^{n} r_i \end{cases}$$
    - End k for loop.
- Compare $P(\langle r_i \rangle)$ with the price of the best allocation found so far.

FIGURE 4.1: Clearing algorithm for multi-unit single-item case with piece-wise linear supply function bids.

4.1.2 Multi-Unit Combinatorial Items

As before, we define a dominant set that is proved to contain an optimal allocation.

Definition 4.9. The dominant set D is the set of all allocations $\langle r_i^j \rangle$ such that for every $1 \leq j \leq m$, there exists a k_j, $1 \leq k_j \leq n$, such that all $r^j_{\lambda_1^j}, ..., r^j_{\lambda_{k-1}^j}$ equal the ending quantities of the segments that they belong to, and all $r^j_{\lambda_{k+1}^j}, ..., r^j_{\lambda_n^j}$ equal the starting quantities of the segments that they belong to:[4]

$$\begin{cases} r^j_{\lambda_i^j} &= e^j_{\lambda_i^j, t^j_{\lambda_i^j}}, \forall 1 \leq i \leq k-1 \\ r^j_{\lambda_i^j} &= s^j_{\lambda_i^j, t^j_{\lambda_i^j}}, \forall k+1 \leq i \leq n \\ r^j_{\lambda_k^j} &= q_j - \sum_{i=1, i \neq k}^{n} r^j_{\lambda_i^j} \end{cases}$$

where:

- t_i^j is the segment on P_i^j that r_i^j belongs to.
- $(\lambda_i^j)_{i=1}^n$ is any permutation of $(1, 2, ..., n)$ such that $\{\omega_{\lambda_i^j}(\langle t^j_{\lambda_i^j} \rangle) \cdot \pi^j_{\lambda_i^j, t^j_{\lambda_i^j}}\}_{i=1}^n$ is sorted decreasingly (increasingly):

$$\omega_{\lambda_1^j}(\langle t^j_{\lambda_1^j} \rangle) \cdot \pi^j_{\lambda_1^j, t^j_{\lambda_1^j}} \geq \omega_{\lambda_2^j}(\langle t^j_{\lambda_2^j} \rangle) \cdot \pi^j_{\lambda_2^j, t^j_{\lambda_2^j}} \geq ... \geq \omega_{\lambda_n^j}(\langle t^j_{\lambda_n^j} \rangle) \cdot \pi^j_{\lambda_n^j, t^j_{\lambda_n^j}} \quad \text{(forward case)}$$

$$\omega_{\lambda_1^j}(\langle t^j_{\lambda_1^j} \rangle) \cdot \pi^j_{\lambda_1^j, t^j_{\lambda_1^j}} \leq \omega_{\lambda_2^j}(\langle t^j_{\lambda_2^j} \rangle) \cdot \pi^j_{\lambda_2^j, t^j_{\lambda_2^j}} \leq ... \leq \omega_{\lambda_n^j}(\langle t^j_{\lambda_n^j} \rangle) \cdot \pi^j_{\lambda_n^j, t^j_{\lambda_n^j}} \quad \text{(reverse case)}$$

From this, a number of lemmas follow:

Lemma 4.10. *For every allocation $\langle r_i^j \rangle$ there exists an allocation in the dominant set D that is not less profitable than it.*

Proof. Let $\langle r_i^j \rangle$ be an allocation. Let t_i^j be the segment that r_i^j belongs to. Suppose $(\lambda_i^j)_{i=1}^n$ is any permutation of $(1, 2, ..., n)$ such that $\{\omega_{\lambda_i^j}(\langle t^j_{\lambda_i^j} \rangle) \cdot \pi^j_{\lambda_i^j, t^j_{\lambda_i^j}}\}_{i=1}^n$ is sorted

[4] Similar to subsection 4.1.1, there may be many dominant sets D.

decreasingly (increasingly):

$$\begin{aligned}
&\omega_{\lambda_1^j}(\langle t^j_{\lambda_1^j}\rangle)\cdot\pi^j_{\lambda_1^j,t^j_{\lambda_1^j}} \geq \omega_{\lambda_2^j}(\langle t^j_{\lambda_2^j}\rangle)\cdot\pi^j_{\lambda_2^j,t^j_{\lambda_2^j}} \geq ... \geq \omega_{\lambda_n^j}(\langle t^j_{\lambda_n^j}\rangle)\cdot\pi^j_{\lambda_n^j,t^j_{\lambda_n^j}} \quad \text{(forward case)}\\
&\omega_{\lambda_1^j}(\langle t^j_{\lambda_1^j}\rangle)\cdot\pi^j_{\lambda_1^j,t^j_{\lambda_1^j}} \leq \omega_{\lambda_2^j}(\langle t^j_{\lambda_2^j}\rangle)\cdot\pi^j_{\lambda_2^j,t^j_{\lambda_2^j}} \leq ... \leq \omega_{\lambda_n^j}(\langle t^j_{\lambda_n^j}\rangle)\cdot\pi^j_{\lambda_n^j,t^j_{\lambda_n^j}} \quad \text{(reverse case)}
\end{aligned} \tag{4.3}$$

For any $\bar{j}$, $1 \leq \bar{j} \leq m$, by proving in similar manner to lemma 4.4, there exists an allocation $\langle \bar{r}_i^j \rangle$, that is not less profitable than $\langle r_i^j \rangle$, where $\bar{r}_i^j$ belongs to segment t_i^j of P_i^j, $\forall 1 \leq i \leq n$, $\forall 1 \leq j \leq m$ and for some k, $1 \leq k \leq n$:

$$\begin{aligned}
\bar{r}^{\bar{j}}_{\lambda_i^{\bar{j}}} &= e^{\bar{j}}_{\lambda_i^{\bar{j}},t^{\bar{j}}_{\lambda_i^{\bar{j}}}}, \forall 1 \leq i \leq k-1\\
\bar{r}^{\bar{j}}_{\lambda_i^{\bar{j}}} &= s^{\bar{j}}_{\lambda_i^{\bar{j}},t^{\bar{j}}_{\lambda_i^{\bar{j}}}}, \forall k+1 \leq i \leq n\\
\bar{r}^{\bar{j}}_{\lambda_k^{\bar{j}}} &= q_j - \sum_{i=1, i\neq k}^{n} \bar{r}^{\bar{j}}_{\lambda_i^{\bar{j}}}\\
\bar{r}_i^j &= r_i^j, \forall 1 \leq i \leq n, \forall 1 \leq j \leq m, j \neq \bar{j}
\end{aligned}$$

Repeating the above step for every $\bar{j}$ from 1 to m, we complete the proof. □

The above lemma leads directly to the following corollary:

Corollary 4.11. *The dominant set D must contain an optimal allocation.*

Lemma 4.12. *The number of elements in the set D is not more than $\prod_{i=1}^{n}\prod_{j=1}^{m}(N_i^j+1)$.*

Proof. Consider an allocation $\langle r_i^j \rangle$ in D. By lemma 2, for each $\bar{j}$ ranging from 1 to m, the number of possible values of a tuple $\langle r_i^{\bar{j}} \rangle_{i=1}^{n}$ is not more than $\prod_{i=1}^{n}(N_i^{\bar{j}}+1)$. Thus, the number of possible values of $\langle r_i^j \rangle$ is not more than $\prod_{i=1}^{n}\prod_{j=1}^{m}(N_i^j+1)$. □

With these lemmas in place, we can now present our algorithm for the combinatorial case (see figure 4.2), which, as before, searches through all allocations of the dominant

set D and chooses the most profitable valid one. We can now analyse the algorithm to assess its properties.

Theorem 4.13. *The algorithm is guaranteed to find the optimal allocation.*

Proof. The algorithm searches all the allocations of the dominant set D. Also, by corollary 4.11, the dominant set D contains an optimal allocation. Thus the algorithm is guaranteed to find an optimal allocation. □

Theorem 4.14. *The complexity of the algorithm is $O(mn \cdot (K+1)^{mn})$, where K is the upper bound on the number of segments of P_i^j.*

Proof. The number of allocations searched by the algorithm is equal to the number of elements of the dominant set. By lemma 4.12, the number of elements of the dominant set is not more than $\prod_{i=1}^{n}\prod_{j=1}^{m}(N_i^j+1) \leq (K+1)^{mn}$. Also, for each j running from 1 to m, it takes $O(\log n)$ to sort $\{\omega_i(\langle t_i^j \rangle) \cdot \pi^j_{i,t_i^j}\}$ and $O(n)$ to find k, so the complexity of the algorithm is $O(mn \cdot (K+1)^{mn})$.

□

Note that this is a worst-case analysis. In many real-life scenarios, each bidder is likely to provide a subset of the goods/services, not all of them. So if bidder i does not provide an item j, then $N_i^j = 0$, meaning the number $\prod_{i=1}^{n}\prod_{j=1}^{m}(N_i^j+1)$ is much smaller than $(K+1)^{mn}$. For example, given the values suggested in [Eso *et al.*, 2001] (that are claimed to resemble real-life problems in the domain of e-commerce), the number $\prod_{i=1}^{n}\prod_{j=1}^{m}(N_i^j+1)$ normally reduces to $3^{\frac{m(n+4)}{2}}$. While this is certainly not an average case analysis, it provides an indication of the complexity that may be encountered in practice.

Algorithm 4. For every tuple $\langle t_i^j \rangle$, $1 \leq i \leq n$, $1 \leq j \leq m$ such that t_i^j is a segment on P_i^j:

- For every $j = 1$ to m do:
  - If $\sum_{i=1}^{n} e^j_{i,t_i^j} < q_j$ or $\sum_{i=1}^{n} s^j_{i,t_i^j} > q_j$:
    Continue; // Jump to the next $\langle t_i^j \rangle$ tuple.
  - Sort $\{\omega_i(\langle t_i^j \rangle) \cdot \pi^j_{i,t_i^j}\}$ decreasingly (increasingly).
  - For $k = 1$ to n do:
    - If $\sum_{i=1}^{k} e^j_{i,t_i^j} + \sum_{i=k+1}^{n} s^j_{i,t_i^j} > q_j$:
      - Set: $\begin{cases} r_i^j &= e^j_{i,t_i^j}, \forall 1 \leq i \leq k_j - 1 \\ r_i^j &= s^j_{i,t_i^j}, \forall k_j + 1 \leq i \leq n \\ r^j_{k_j} &= q_j - \sum_{i=1, i \neq k_j}^{n} r_i^j \end{cases}$
      - End k for loop.
- Compare $P(\langle r_i^j \rangle)$ with the price of the best allocation found so far.

FIGURE 4.2: Clearing algorithm for multi-unit combinatorial case with piece-wise linear supply function bids.

4.2 Monotonic One-Unit-Difference Supply/Demand Functions

In this section, we apply broadly the same techniques as those used in the previous section to another class of demand/supply functions. As stated before, although this class is not common, it is considered because the same solution as piece-wise linear can be used and it covers another portion of the general auction clearing problem space. Specifically, the case where the one-unit-difference demand/supply functions are monotonic (non-decreasing for the forward case and non-increasing for the reverse case). That is, $\forall 1 \leq i \leq n$, $\forall 1 \leq j \leq m$, $\forall a_1, .., a_{j-1}, a_{j+1}, .., a_m$:

$$Q_i^j(r) = P_i(a_1, .., a_{j-1}, r+1, a_{j+1}, .., a_m) - P_i(a_1, .., a_{j-1}, r, a_{j+1}, .., a_m)$$

is non-decreasing (non-increasing):

$$\begin{aligned} &Q_i^j(r+1) \geq Q_i^j(r), \forall 0 \leq r \leq u_i^j - 1 \text{ (forward case)} \\ &Q_i^j(r+1) \leq Q_i^j(r), \forall 0 \leq r \leq u_i^j - 1 \text{ (reverse case)} \end{aligned} \tag{4.4}$$

In the previous case of piece-wise linear function bids, if we limit each piece-wise linear function to be a single segment only, then the one-unit-difference demand/supply function is constant (which is a special case of monotonicity). Thus the same technique can be used in both the cases. As before, we first consider the multi-unit single-item case (subsection 4.2.1) before moving onto the combinatorial case (subsection 4.2.2).

4.2.1 Multi-Unit Single-Items

In the single-item case, the monotonicity assumption can be re-formulated as follows:

For all $1 \leq i \leq n$, the one-unit difference function

$$Q_i(r) = P_i(r+1) - P_i(r)$$

is non-decreasing (non-increasing). That is:

$$\begin{aligned} &Q_i(r+1) \geq Q_i(r), \forall 0 \leq r \leq u_i - 1 \text{ (forward case)} \\ &Q_i(r+1) \leq Q_i(r), \forall 0 \leq r \leq u_i - 1 \text{ (reverse case)} \end{aligned} \tag{4.5}$$

Again, we define a dominant set that is proved to contain an optimal allocation.

Definition 4.15. The dominant set D is the set of all allocations $(r_1, r_2, ..., r_n)$ such that there exists a k, $1 \leq k \leq n$, and a permutation of $(1, 2, ..., n)$: $\{\lambda_i\}_{i=1}^n$ such that:

$$\begin{cases} r_{\lambda_i} &= u_{\lambda_i}, \forall 1 \leq i \leq k-1 \\ r_{\lambda_k} &= q - \sum_{i=1}^{k-1} u_{\lambda_i} \\ r_{\lambda_i} &= 0, \forall k+1 \leq i \leq n \end{cases}$$

From this, a number of lemmas follow:

Lemma 4.16. *For every allocation $(r_1, r_2, ..., r_n)$ there exists an allocation in the dominant set D which is not less profitable than it.*

Proof. Let $(r_1, r_2, ..., r_n)$ be an allocation.

Step 1: Suppose the one-unit difference function of bidder λ_1 at r_{λ_1} is the biggest (smallest):

$$\begin{aligned} &Q_{\lambda_1}(r_{\lambda_1}) = \max\{Q_i(r_i) | 1 \leq i \leq n, r_i > 0\} \text{ (forward case)} \\ &Q_{\lambda_1}(r_{\lambda_1}) = \min\{Q_i(r_i) | 1 \leq i \leq n, r_i > 0\} \text{ (reverse case)} \end{aligned} \tag{4.6}$$

By proving similarly to the proof in theorem 1, we have: there exists an allocation $\{r_i^{(1)}\}$, which is not less profitable than $\{r_i\}$, where either $r_{\lambda_1}^{(1)} = u_{\lambda_1}$ or $r_{\lambda_1}^{(1)} = q$.

Step 2: In the case if $r_{\lambda_1}^{(1)} = u_{\lambda_1}$ and $r_{\lambda_1}^{(1)} < q$, by repeating the above step, there

exists an allocation $\{r_i^{(2)}\}$, which is not less profitable than $\{r_i\}$, where $r_{\lambda_1}^{(2)} = r_{\lambda_1}^{(1)} = u_{\lambda_1}$ and either $r_{\lambda_2}^{(2)} = u_{\lambda_2}$ or $r_{\lambda_2}^{(2)} = q - u_{\lambda_1}$.

By repeating the above steps again and again, we will finally stop at some step k, $1 \leq k \leq n$, and get an allocation $\{r_i^{(k)}\}$, $1 \leq k \leq n$, which is not less profitable than the original allocation $\{r_i\}$, where $\{\lambda_i\}_{i=1}^n$ is a permutation of $(1, 2, ..., n)$ and:

$$\begin{cases} r_{\lambda_i} &= u_{\lambda_i}, \forall 1 \leq i \leq k-1 \\ r_{\lambda_k} &= q - \sum_{i=1}^{k-1} u_{\lambda_i} \\ r_{\lambda_i} &= 0, \forall k+1 \leq i \leq n \end{cases}$$

□

The above lemma leads directly to the following corollary:

Corollary 4.17. *The dominant set D must contain an optimal allocation.*

Lemma 4.18. *The number of elements in the set D is not more than $n \cdot 2^{n-1}$.*

Proof. For any k, $1 \leq k \leq n$, let D_k be the set of all the tuples $(r_1, r_2, ..., r_n)$ such that there exists a permutation of $(1, 2, ..., n)$: $\{\lambda_i\}_{i=1}^n$ such that:

$$\begin{cases} r_{\lambda_i} &= u_{\lambda_i}, \forall 1 \leq i \leq k-1 \\ r_{\lambda_k} &= q - \sum_{i=1}^{k-1} u_{\lambda_i} \\ r_{\lambda_i} &= 0, \forall k+1 \leq i \leq n \end{cases}$$

We have $D = \bigcap_{k=1}^n D_k$. Thus:

$$|D| \leq \sum_{k=1}^{n} |D_k| \tag{4.7}$$

The number of elements of set D_k is equal to the number of ways to choose a $(k-1)$-element subset $\{\lambda_1, \lambda_2, ..., \lambda_{k-1}\}$ of the set $\{1, 2, ..., n\}$ multiplied by the number of ways to choose one element from the set $\{1, 2, ..., n\} \setminus \{\lambda_1, \lambda_2, ..., \lambda_{k-1}\}$. But the number of ways to choose a $(k-1)$-element subset of the set $\{1, 2, ..., n\}$ is $\frac{n!}{(k-1)!(n-k+1)!}$, and the number of ways to choose one element from the set $\{1, 2, ..., n\} \setminus \{\lambda_1, \lambda_2, ..., \lambda_{k-1}\}$ is

$(n-k+1)$ (as $|\{1,2,...,n\} \setminus \{\lambda_1, \lambda_2, ..., \lambda_{k-1}\}|$
$= n-k+1$). Thus we have:

$$|D_k| = \frac{n!}{(k-1)!(n-k+1)!}(n-k+1)$$
$$\Rightarrow |D_k| = \frac{n!}{(k-1)!(n-k)!} \tag{4.8}$$

From (4.7) and (4.8) we have:

$$|D| \leq \sum_{k=1}^{n} \frac{n!}{(k-1)!(n-k)!}$$
$$= n \cdot \sum_{k=1}^{n} \frac{(n-1)!}{(k-1)!(n-k)!}$$
$$= n \cdot \sum_{k=0}^{n-1} \frac{(n-1)!}{k!(n-1-k)!}$$
$$= n \cdot 2^{n-1}$$

□

With these lemmas in place, we can now present our algorithm for the single-item case. The algorithm is presented in figure 4.3. Again, the algorithm searches through all the allocations of the set D and chooses the most profitable valid one. The algorithm has the following properties.

Theorem 4.19. *The algorithm is guaranteed to find the optimal allocation.*

Proof. The algorithm searches all the allocations of the dominant set D. Also, by corollary 4.17, the dominant set D contains an optimal allocation. Thus the algorithm is guaranteed to find an optimal allocation. □

Theorem 4.20. *The complexity of the algorithm is* $O(n \cdot 2^{n-1})$.

Proof. The number of allocations searched by the algorithm is equal to the number of elements of the dominant set. By lemma 4.18, the number of elements of the dominant set is not more than $n \cdot 2^{n-1}$. Thus, the complexity of the algorithm is $O(n \cdot 2^{n-1})$. □

Algorithm 5. For $k = 1...n$:
For every $(k-1)$-element subset $\{\lambda_1, \lambda_2, ..., \lambda_{k-1}\}$ of the set $\{1, 2, ..., n\}$:
For every λ_k in the set $\{1, 2, ..., n\} \setminus \{\lambda_1, \lambda_2, ..., \lambda_{k-1}\}$:

- Set:
$$\begin{cases} r_{\lambda_i} = u_{\lambda_i}, \forall 1 \leq i \leq k-1 \\ r_{\lambda_k} = q - \sum_{i=1}^{k-1} u_{\lambda_i} \\ r_i = 0, \forall i \in \{1, 2, ..., n\} \setminus \{\lambda_i\}_{i=1}^{k} \end{cases}$$
- If $0 \leq r_{\lambda_k} \leq u_{\lambda_k}$
Compare $P(\{r_{\lambda_i}\})$ with the price of the best allocation found so far.

FIGURE 4.3: Clearing algorithm for multi-unit single-item case with monotonic one-unit-difference demand/supply function bids.

Having dealt with the multi-unit single-item case, the next subsection generalises the algorithm to the multi-unit combinatorial case.

4.2.2 Multi-Unit Combinatorial Items

As previously, we define a dominant set that is proved to contain an optimal allocation.

Definition 4.21. The dominant set D is the set of all allocations $\{r_i^j\}$, $1 \leq i \leq n$, $1 \leq j \leq m$, such that: For any j, $1 \leq j \leq m$, there exists a k_j, $1 \leq k_j \leq n$, and a permutation of $(1, 2, ..., n)$: $\{\lambda_i^j\}_{i=1}^n$ such that:

$$\begin{cases} r^j_{\lambda_i^j} &= u^j_{\lambda_i^j}, \forall 1 \leq i \leq k_j - 1 \\ r^j_{\lambda_k^j} &= q - \sum_{i=1}^{k_j-1} u^j_{\lambda_i^j} \\ r^j_{\lambda_i^j} &= 0, \forall k_j + 1 \leq i \leq n \end{cases}$$

From this, a number of lemmas follow:

Lemma 4.22. *For every allocation $\{r_i^j\}$ there exists an allocation in the dominant set D which is not less profitable than it.*

Proof. For any $\bar{j}$, $1 \leq \bar{j} \leq m$, by proving similarly to the proof of lemma 5, there exists an allocation $\{\bar{r}_i^j\}$, which is not less profitable than $\{r_i^j\}$, where $\{\lambda_i^{\bar{j}}\}_{i=1}^n$ is a permutation of $(1, 2, ..., n)$ and there exists a k, $1 \leq k \leq n$ such that:

$$\begin{aligned} &\bar{r}^{\bar{j}}_{\lambda_i^{\bar{j}}} = u^{\bar{j}}_{\lambda_i^{\bar{j}}}, \forall 1 \leq i \leq k-1 \\ &\bar{r}^{\bar{j}}_{\lambda_k^{\bar{j}}} = q - \sum_{i=1}^{k-1} u^{\bar{j}}_{\lambda_i^{\bar{j}}} \\ &\bar{r}^{\bar{j}}_{\lambda_i^{\bar{j}}} = 0, \forall k+1 \leq i \leq n \\ &\bar{r}_i^j = r_i^j, \forall 1 \leq i \leq n, 1 \leq j \leq m, j \neq \bar{j} \end{aligned}$$

Repeating the above step for every $\bar{j}$ ranging from 1 to m, we complete the proof. □

The above lemma leads directly to the following corollary:

Corollary 4.23. *The dominant set D must contain an optimal allocation.*

Lemma 4.24. *The number of elements in the set D is not more than $n^m \cdot 2^{m(n-1)}$.*

Proof. Consider an allocation $\{r_i^j\}$ in D. By lemma 6, for each $\bar{j}$ ranging from 1 to m, the number of possible values of a tuple $\{r_i^{\bar{j}}\}_{i=1}^n$ is not more than $n \cdot 2^{n-1}$. Thus, the number of possible values of $\{r_i^j\}$ is not more than $(n \cdot 2^{n-1})^m$ or $n^m \cdot 2^{m(n-1)}$. □

With these lemmas in place, we can now present our algorithm for the combinatorial case. As before, the algorithm, presented in figure 4.4, searches through all the allocations of the set D and chooses the most profitable valid one. The algorithm has the following properties.

Theorem 4.25. *The algorithm is guaranteed to find the optimal allocation.*

Proof. The algorithm searches all the allocations of the dominant set D. Also, by corollary 4.23, the dominant set D contains an optimal allocation. Thus the algorithm is guaranteed to find an optimal allocation. □

Theorem 4.26. *The complexity of the algorithm is $O(n^m \cdot 2^{m(n-1)})$.*

Proof. The number of allocations searched by the algorithm is equal to the number of elements of the dominant set. By lemma 4.24, the number of elements of the dominant set is not more than $n^m \cdot 2^{m(n-1)}$. Thus, the complexity of the algorithm is $O(n^m \cdot 2^{m(n-1)})$. □

Algorithm 6. For every tuple $(k_1, k_2, ..., k_m)$ such that $1 \leq k_j \leq n$, $\forall 1 \leq j \leq m$:
For every tuple $\{\lambda_1^1, ..., \lambda_{k_1-1}^1, \lambda_1^2, ..., \lambda_{k_2-1}^2, ..., \lambda_1^m, ..., \lambda_{k_m-1}^m\}$ such that $\{\lambda_1^j, \lambda_2^j, ..., \lambda_{k-1}^j\}$ is a $(k_j - 1)$-element subset of the set $\{1, 2, ..., n\}$, $\forall 1 \leq j \leq m$:
For every tuple $\{\lambda_{k_j}^j\}$ such that $\lambda_{k_j}^j$ is in the set $\{1, 2, ..., n\} \setminus \{\lambda_1^j, \lambda_2^j, ..., \lambda_{k-1}^j\}$, $\forall 1 \leq j \leq m$:

- For $j = 1...m$ set:
$$\begin{cases} r^j_{\lambda_i^j} & = & u^j_{\lambda_i^j}, \forall 1 \leq i \leq k_j - 1 \\ r^j_{\lambda_{k_j}^j} & = & q - \sum_{i=1}^{k_j-1} u^j_{\lambda_i^j} \\ r^j_{\lambda_i^j} & = & 0, \forall k_j + 1 \leq i \leq n \end{cases}$$
- If $0 \leq r^j_{\lambda_{k_j}^j} \leq u^j_{\lambda_{k_j}^j}$, for all $1 \leq j \leq m$
  Compare $P(\{r_{\lambda_i}\})$ with the price of the best allocation found so far.

FIGURE 4.4: Clearing algorithm for multi-unit combinatorial case with monotonic one-unit-difference demand/supply function bids.

4.3 Summary

This chapter presents, for the first time, optimal clearing algorithms for multi-unit single-item and multi-unit combinatorial auctions where bids are expressed through supply/demand functions. Specifically, we consider two classes of supply/demand functions where the demand/supply curves for each individual commodity are piece-wise linear (an important and often considered case) and where the demand/supply curves are monotonic one-unit-difference. This means our algorithms enable us to deal with a more general case than any previous work in this area. Moreover, we believe this degree of expressiveness is important for obtaining the maximum benefit from combinatorial auctions in practical settings.

Chapter 5

Coalition Structure Generation Algorithm

This chapter presents our novel algorithm for coalition structure generation. We will show that it produce solutions that are within a finite bound of the optimal. It will then be compared with an algorithm by [Sandholm *et al.*, 1999] which is the only other non-trivial algorithm (i.e. not including exhaustive search) for coalition structure generation that can also establish a finite bound of the optimal.

5.1 The Algorithm

In this section, we present our algorithm for coalition structure generation and prove that the solution it generates is within a finite bound from the optimal.

To this end, let L_k be the set of all coalition structures with size k. Thus we have:

$$L = \bigcup_{k=1}^{n} L_k$$

The number of coalition structures in L_k is $S(n, k)$, widely known in Mathematics as *the Stirling number of the Second Kind* [Roman, 1984]. The value of $S(n, k)$ can be

computed by the following formula [Roman, 1984]:

$$S(n,k) = \frac{1}{k!}\sum_{i=0}^{k-1}(-1)^i\binom{k}{i}(k-i)^n$$

Definition 5.1. Let $SL(n,k,c)$ be the set of all coalition structures that have exactly k coalitions and at least one coalition whose cardinality is not less than c.

Definition 5.2. Let $SL(n,c)$ be the set of all coalition structures whose cardinality is between 3 and $n-1$ that have at least one coalition whose cardinality is not less than c. That is:[1]

$$SL(n,c) = \bigcup_{k=3}^{n-1} SL(n,k,c)$$

With these definitions in place, we can now express our algorithm for solving the problem (see figure 39). Basically, at first it searches all the coalition structures that have one, two or n coalitions (i.e. all the coalition structures in the sets: L_1, L_2 and L_n) (as Sandholm et al.'s algorithm does). But after that, instead of searching through the sets L_k (for $3 \leq k \leq n-1$) one by one (as Sandholm et al. do), our algorithm only searches some specific subsets of L_k (see figure 5.2 for a diagramatic representation). In particular, it searches the set of all coalition structures that have k coalitions and at least one coalition whose cardinality is not less than $\lceil n(q-1)/q \rceil$ (with q running from $\lfloor \frac{n+1}{4} \rfloor$ down to 2 as in Figure 5.1). Note that we start from $q = \lfloor \frac{n+1}{4} \rfloor$ because Sandholm et al. showed that, after searching L_1, L_2, L_n, the algorithm can establish a bound $b = \lceil n/2 \rceil$ and, later in this chapter, we will show that after searching $SL(n, \lceil n(q-1)/q \rceil)$, our algorithm can establish a bound $b = 2q-1$. Thus, we start from the biggest q such that $2q-1 < \lceil n/2 \rceil$ or $q = \lfloor \frac{n+1}{4} \rfloor$.

The next step is to show that the solution generated by the algorithm is within a bound from the optimal and that the bound is reduced further after each round. Thus ours is an **anytime algorithm**: it can be interrupted at any time and the bound keeps improving with an increase in execution time.[2]

[1] In fact, as $SL(n,k,c) = \emptyset$ for all $n-c+1 < k \leq n-1$, this formula can be rewritten as: $SL(n,c) = \bigcup_{k=3}^{n-c+1} SL(n,k,c)$

[2] If the domain happens to be super-additive, the algorithm finds the optimal coalition structure (grand coalition) immediately.

Algorithm 7. The algorithm proceeds as follows:

- Step 1: Search through the sets L_1, L_2, L_n
- From step 2 onward, search, consequently, through the sets $SL(n, \lceil n(q-1)/q \rceil)$ with q running from $\lfloor \frac{n+1}{4} \rfloor$ down to 2.

  That is, search $SL(n, \lceil n(\lfloor \frac{n+1}{4} \rfloor - 1)/\lfloor \frac{n+1}{4} \rfloor \rceil)$ at step 2, search $SL(n, \lceil n(\lfloor \frac{n+1}{4} \rfloor - 2)/(\lfloor \frac{n+1}{4} \rfloor - 1) \rceil)$ at step 3 and so on.
  Moreover, from step 3 onward, as $SL(n, \lceil nq/(q+1) \rceil) \subseteq SL(n, \lceil n(q-1)/q \rceil)$ (it is easy to see that $SL(n, \lceil n(a-1)/a \rceil) \subseteq SL(n, \lceil n(b-1)/b \rceil)$ for every $a > b$) we only have to search through the set $SL(n, \lceil n(q-1)/q \rceil) \setminus SL(n, \lceil nq/(q+1) \rceil)$ in order to search through the set $SL(n, \lceil n(q-1)/q \rceil)$.

- At each step return the coalition structure with the biggest value (i.e. best social welfare) so far.

FIGURE 5.1: The coalition structure generation algorithm.

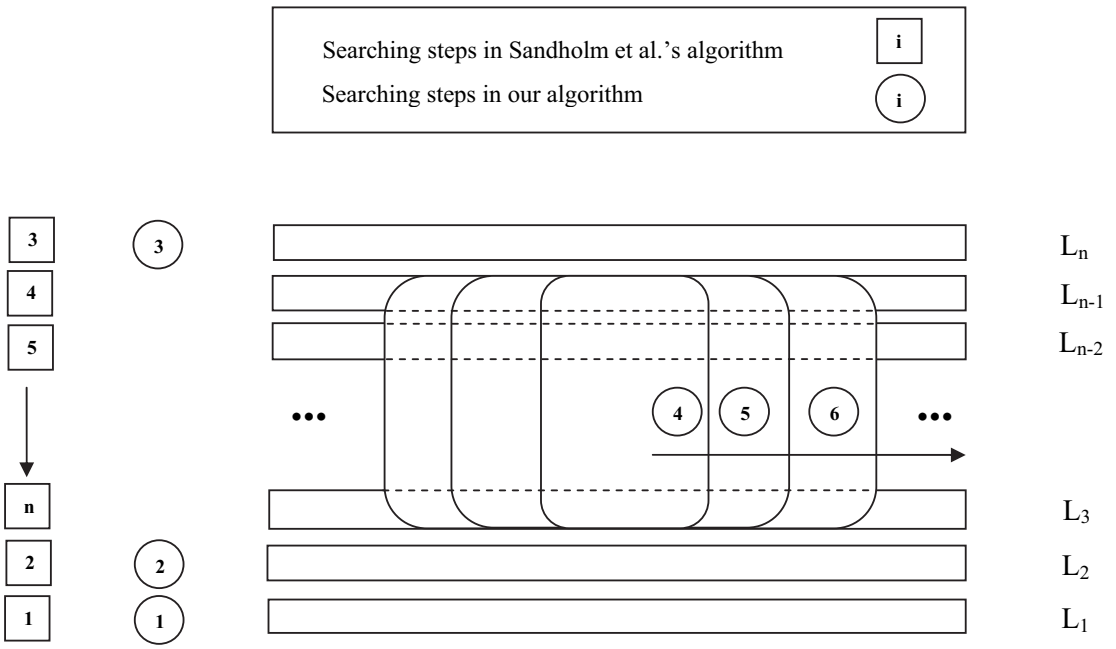


FIGURE 5.2: Comparison of the searching paths between our algorithm and Sandholm et al.'s.

Theorem 5.3. *Immediately after finishing searching* $SL(n, \lceil n(q-1)/q \rceil)$*, the solution generated by our algorithm is within a finite bound* $b = 2q - 1$ *from the optimal.*

Proof. Let CS^a be the coalition structure that our algorithm generates. Let CS^* be an optimal coalition structure. Assume CS^* contains t coalitions $C_1, C_2, ..., C_t$. We have to prove:

$$\frac{V(CS^*)}{V(CS^a)} \leq 2q - 1 \tag{5.1}$$

For the cases where t equals 1, 2 or n, the proof is trivial, as CS^a will also be an optimal coalition structure. Thus $V(CS^a) = V(CS^*)$, so $\frac{V(CS^*)}{V(CS^a)} = 1 \leq 2q - 1$.

Now we only have to prove for the case where $3 \leq t \leq n - 1$. Without loss of generality, we can assume the cardinalities of the sets $C_1, C_2, ..., C_t$ are in decreasing order. That is:

$$|C_1| \geq |C_2| \geq ... \geq |C_t| \tag{5.2}$$

For the convenience of presentation, we assume $C_i = \emptyset$ and $v(C_i) = 0$ for every $i > t$.

First, we will show that for every coalition $C \subseteq A$:

$$v(C) \leq V(CS^a)$$

Considering the coalition structure $CS_0 = \{C, A \setminus C\}$. As $CS_0 \in L_2$, we have:

$$\begin{aligned} & V(CS_0) \leq V(CS^a) \\ & \Rightarrow v(C) + v(A \setminus C) \leq V(CS^a) \\ & \Rightarrow v(C) \leq V(CS^a) \text{ (because of assumption (2.4))} \end{aligned}$$

Thus we have:

$$\begin{aligned} & v(C_i) \leq V(CS^a) \ ,\forall 1 \leq i \leq q-1 \\ & \Rightarrow \sum_{i=1}^{q-1} v(C_i) \leq (q-1) \cdot V(CS^a) \end{aligned} \tag{5.3}$$

Now let us consider the other coalitions of CS^*, namely, $C_q, C_{q+1}, ..., C_t$.

Considering the following coalition structure:

$$CS_1 = \{C_q, C_{2q}, ..., C_{\lfloor \frac{t}{q} \rfloor q}, D\}$$

That is:

$$D = A \setminus \cup_{i=1}^{\lfloor \frac{t}{q} \rfloor} C_{iq} \tag{5.4}$$

Let us analyse the cardinality of coalition D. We have for all $1 \leq i \leq \lfloor \frac{t}{q} \rfloor$:

$$\begin{aligned} & |C_{(i-1)q+1}| \geq |C_{(i-1)q+2}| \geq ... \geq |C_{iq}| \text{(because of (5.2))} \\ & \Rightarrow q|C_{iq}| \leq \sum_{j=1}^{q} |C_{(i-1)q+j}| \\ & \Rightarrow q \sum_{i=1}^{\lfloor \frac{t}{q} \rfloor} |C_{iq}| \leq \sum_{i=1}^{\lfloor \frac{t}{q} \rfloor} \sum_{j=1}^{q} |C_{(i-1)q+j}| \end{aligned}$$

$$\Rightarrow q\sum_{i=1}^{\lfloor \frac{t}{q} \rfloor} |C_{iq}| \leq |C_1| + |C_2| + ... + |C_{\lfloor \frac{t}{q} \rfloor q}|$$
$$\Rightarrow q\sum_{i=1}^{\lfloor \frac{t}{q} \rfloor} |C_{iq}| \leq |C_1| + |C_2| + ... + |C_t| = n$$
$$\Rightarrow \sum_{i=1}^{\lfloor \frac{t}{q} \rfloor} |C_{iq}| \leq n/q$$

As $|D| = n - \sum_{i=1}^{\lfloor \frac{t}{q} \rfloor} |C_{iq}|$ (from (5.4)), we then have:

$$|D| \geq n - n/q$$
$$\Rightarrow |D| \geq n(q-1)/q$$
$$\Rightarrow |D| \geq \lceil n(q-1)/q \rceil$$
$$\Rightarrow CS_1 \in SL(n, \lceil n(q-1)/q \rceil)$$
$$\Rightarrow V(CS_1) \leq V(CS^a)$$
$$\Rightarrow v(C_q) + v(C_{2q}) + ... + v(C_{\lfloor \frac{t}{q} \rfloor q}) + v(D)$$
$$\leq V(CS^a)$$
$$\Rightarrow v(C_q) + v(C_{2q}) + ... + v(C_{\lfloor \frac{t}{q} \rfloor q}) \leq V(CS^a)$$
(as $v(D) \geq 0$, by assumption (2.4))

For all $1 \leq j \leq q-1$, by proving similarly to the above, we have:

$$v(C_{q+j}) + v(C_{2q+j}) + ... + v(C_{\lfloor \frac{t}{q} \rfloor q+j}) \leq V(CS^a)$$

Thus for all $0 \leq j \leq q-1$, we have:

$$\sum_{i=1}^{\lfloor \frac{t}{q} \rfloor} v(C_{iq+j}) \leq V(CS^a)$$
$$\Rightarrow \sum_{j=0}^{q-1}\sum_{i=1}^{\lfloor \frac{t}{q} \rfloor} v(C_{iq+j}) \leq q \cdot V(CS^a)$$

$$\Rightarrow \sum_{i=q}^{\lfloor \frac{t}{q} \rfloor q+q-1} v(C_i) \leq q \cdot V(CS^a) \tag{5.5}$$

Also:

$$\begin{aligned} &\lfloor \frac{t}{q} \rfloor q + q - 1 > (\frac{t}{q} - 1)q + q - 1 \\ &\Rightarrow \lfloor \frac{t}{q} \rfloor q + q - 1 > t - 1 \\ &\Rightarrow \lfloor \frac{t}{q} \rfloor q + q - 1 \geq t \end{aligned} \tag{5.6}$$

Thus, from (5.5) and (5.6), we have:

$$\sum_{i=q}^{t} v(C_i) \leq q \cdot V(CS^a) \tag{5.7}$$

From (5.3) and (5.7) we have:

$$\sum_{i=1}^{t} v(C_i) \leq (2q - 1) \cdot V(CS^a) \tag{5.8}$$

□

From theorem 1, we can also see that the bound decreases after each round, because $2q - 1$ decreases as q decreases. Thus our algorithm is an anytime one.

Having presented our algorithm for coalition structure generation, the next section compares it with Sandholm et al.'s.

5.2 Performance Evaluation

To evaluate the effectiveness of our algorithm we compare it against Sandholm et al.'s [Sandholm *et al.*, 1999] since this is the only other known algorithm with worst-case bounds. In more detail, Sandholm et al.'s algorithm operates as described in figure 5.3. Basically, it first searches all the coalition structures that have 1 or 2 coalitions, then

- Search the bottom two levels of the coalition structure graph (Note that level k of the coalition structure graph in Sandholm et al.'s algorithm corresponds exactly to the set L_k in ours).
- Continue with a breadth-first search from the top of the graph as long as there is time left, or until the entire graph has been searched (this occurs when this breadth-first search completes level 3 of the graph, i.e., depth $n-3$)
- Return the coalition structure that has the highest welfare among those seen so far.

FIGURE 5.3: Sandholm et al.'s algorithm.

it continues to search all the coalition structures that have n, $n-1$, ..., 3 coalitions (in that order). Sandholm et al. then prove that after having completed searching L_k, the solution the algorithm generates is within a bound b', where $b' = \lceil \frac{n}{h} \rceil$ if $n \equiv h-1 (mod\ h)$ and $n \equiv k (mod\ 2)$, or $b' = \lfloor \frac{n}{h} \rfloor$ otherwise ($h = \lfloor \frac{n-k}{2} \rfloor + 2$).

To evaluate the performance of our algorithm, we compare it with Sandholm et al.'s on a worst-case basis. That is, we compare the size of the search space of the two algorithms (i.e. the number of coalition structures each algorithm has to search) in order to establish the same bound from the optimal.

To calculate the number of coalition structures that our algorithm needs to search, we present the following formula.

Lemma 5.4. *For all $n > k \geq 3$ and $c \geq \lceil n/2 \rceil$, the cardinality of $SL(n,k,c)$ can be calculated as follows:*

$$|SL(n,k,c)| = \sum_{i=c}^{n-k+1} S(n-i,k-1) \cdot \frac{n!}{i!(n-i)!} \tag{5.9}$$

Proof. For each i such that $c \leq i \leq n-k+1$, let $T(n,k,i) \subseteq SL(n,k,c)$ be the set of all coalition structures that have exactly k coalitions and at least one coalition whose cardinality equals i. As for every coalition structure CS in $SL(n,k,c)$, any coalition in CS has at most $n-k+1$ agents (because $(k-1)$ other coalitions in CS must contain

at least $(k-1)$ agents), we have:

$$SL(n,k,c) = \bigcup_{i=c}^{n-k+1} T(n,k,i)$$

Now we will show that $T(n,k,i_1) \bigcap T(n,k,i_2) = \emptyset$ for every $i_1 \neq i_2$, $i_1 \geq c$ and $i_2 \geq c$. This can be proved by contradiction. Suppose there exist i'_1 and i'_2 such that $T(n,k,i'_1) \bigcap T(n,k,i'_2) \neq \emptyset$. This means there exists a coalition structure CS' such that: $CS' \in T(n,k,i'_1) \bigcap T(n,k,i'_2)$. Now $CS' \in T(n,k,i'_1)$ means it has at least one coalition whose cardinality equals i'_1, and, similarly, $CS' \in T(n,k,i'_2)$ means it has at least one coalition whose cardinality equals i'_2. Moreover, CS' has at least 3 coalitions (as $k \geq 3$), so the number of agents in CS' will be greater or equal than $i'_1 + i'_2 + 1$. Thus:

$$\begin{aligned}
& n \geq i'_1 + i'_2 + 1 \\
& \Rightarrow n \geq c + c + 1 = 2c + 1 \\
& \Rightarrow n \geq 2\lceil n/2 \rceil + 1 \\
& \Rightarrow n \geq 2n/2 + 1 \\
& \Rightarrow n \geq n + 1
\end{aligned}$$

As we reach contradiction, we must have:

$$T(n,k,i_1) \bigcap T(n,k,i_2) = \emptyset$$

for every $i_1 \neq i_2$, $i_1 \geq c$ and $i_2 \geq c$. Thus we have:

$$\Rightarrow |SL(n,k,c)| = \sum_{i=c}^{n-k+1} |T(n,k,i)| \tag{5.10}$$

Now let us consider a coalition structure $CS' \in T(n,k,i)$. As one of the coalitions in CS' has exactly i agents, $k-1$ other coalitions in CS' must have exactly $n-i$ agents. Also, the number of ways to choose an i-agent set from n agents is $\frac{n!}{i!(n-i)!}$. Thus the number of coalition structures in $T(n,k,i)$ equals the number of coalition structures that have exactly $k-1$ coalitions in a multi-agent system with $n-i$ agents multiplied with

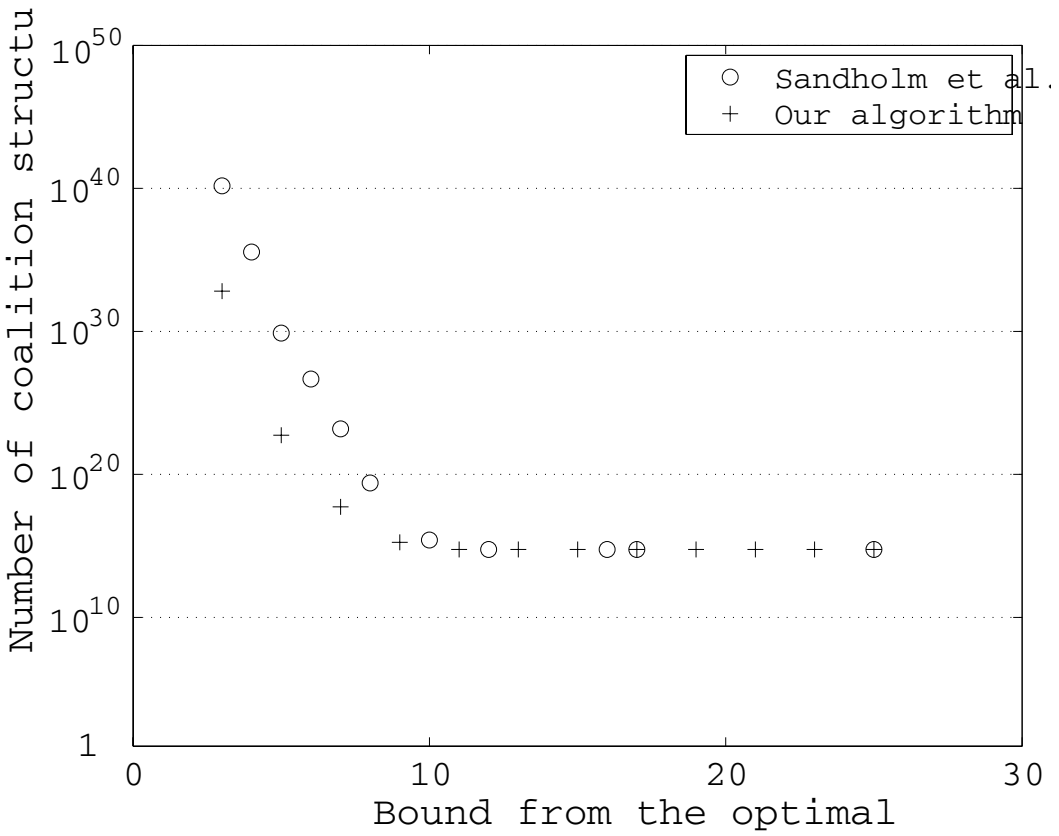


FIGURE 5.4: The case $\mathbf{n = 50}$.

$\frac{n!}{i!(n-i)!}$:

$$|T(n,k,i)| = S(n-i,k-1) \cdot \frac{n!}{i!(n-i)!} \tag{5.11}$$

From (5.10) and (5.11) we have:

$$|SL(n,k,c)| = \sum_{i=c}^{n-k+1} S(n-i,k-1) \cdot \frac{n!}{i!(n-i)!}$$

□

With this in place, we test the algorithms with the number of agents $n = 50$, 100, 500, and 1000.[3] The result of the tests are presented in the following graphs.[4] As we are calculating the number of coalition structures each algorithm has to search in order to

[3]We observe similar patterns with other values of n varying from 50 to 1000.

[4]The big bounds are not shown in the graphs (that is, bounds greater than 30 in the case $n = 100$, greater than 40 in the case $n = 500$ and greater than 50 in the case $n = 1000$), because the results for the two algorithms are nearly the same for these bounds.

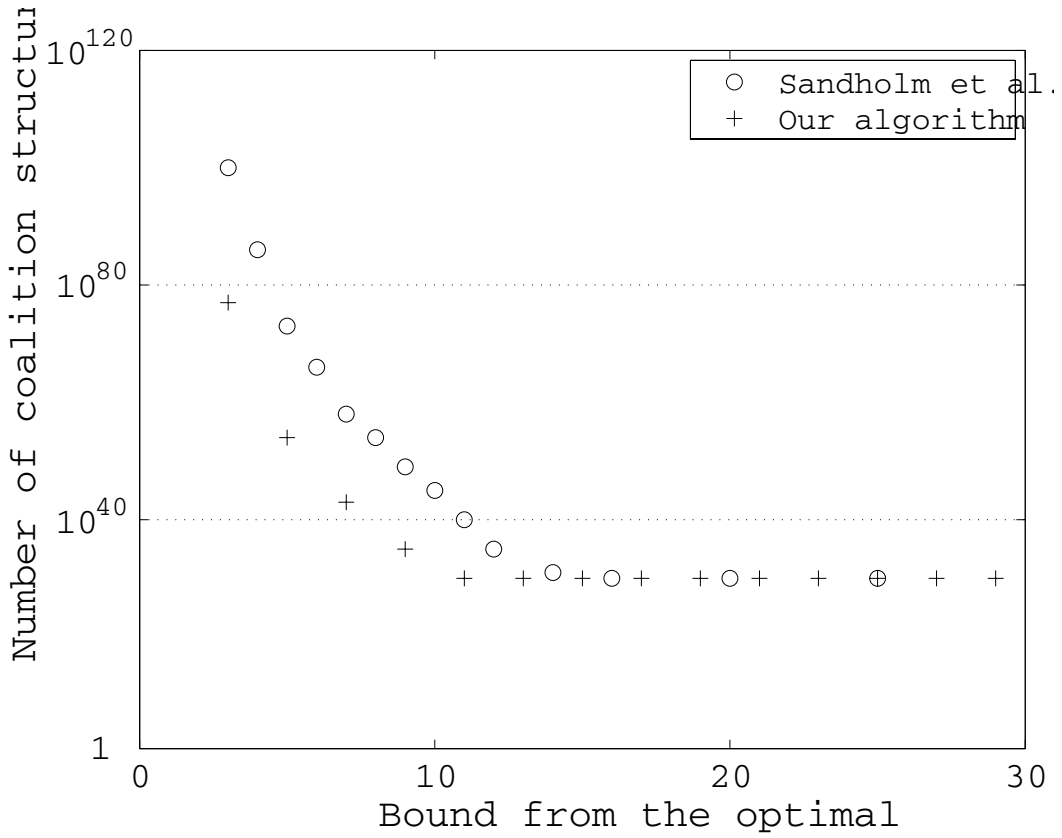


FIGURE 5.5: The case $\mathbf{n = 100}$.

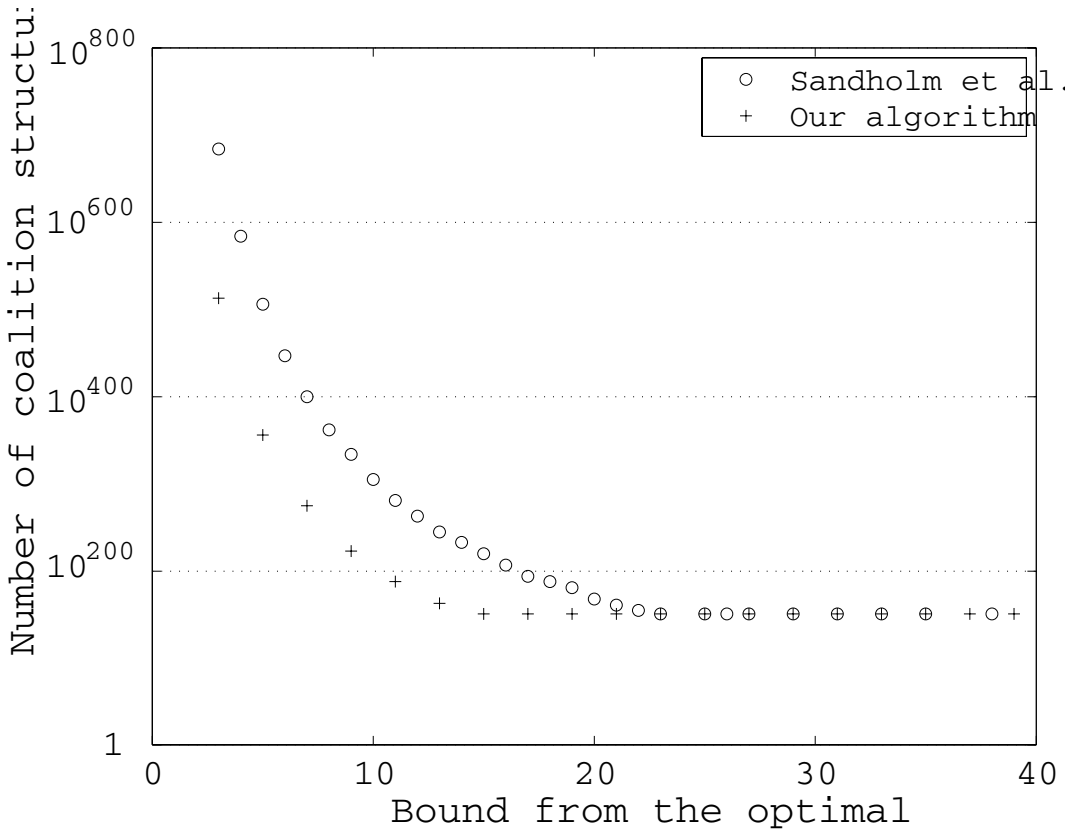


FIGURE 5.6: The case $\mathbf{n = 500}$.

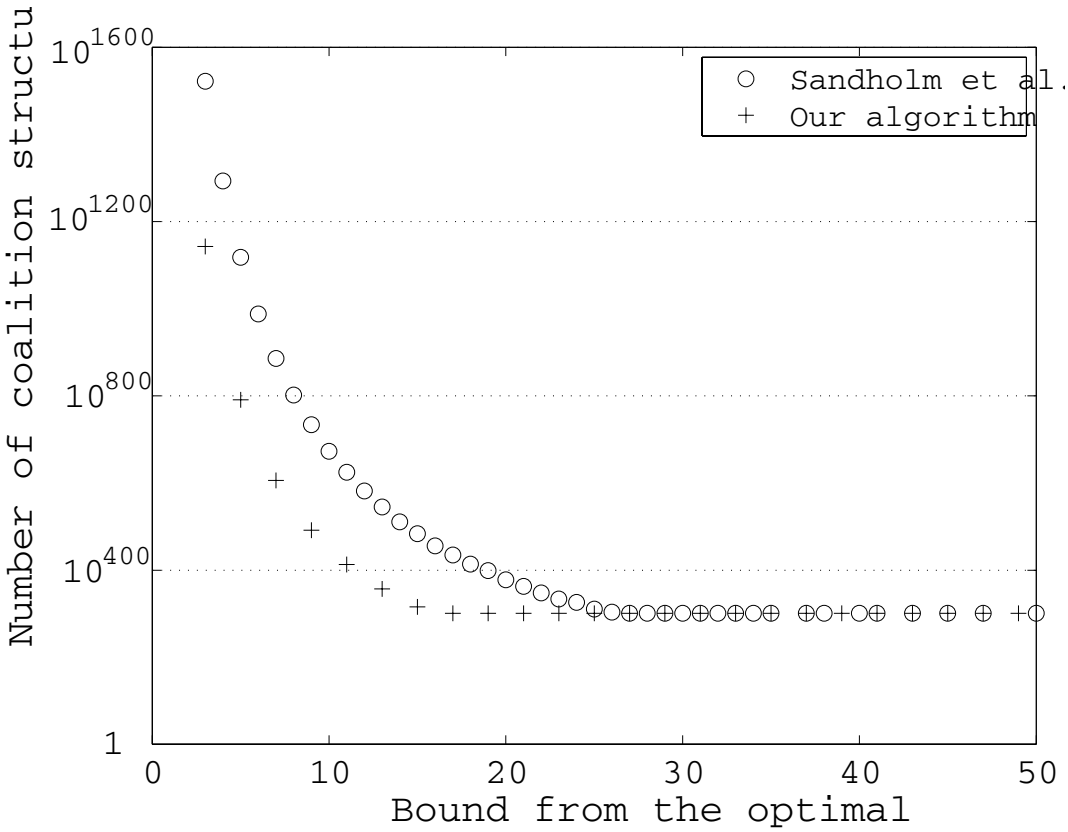


FIGURE 5.7: The case $\mathbf{n = 1000}$.

establish a bound from the optimal, the smaller the number of coalition structures the better.

As we can see from the graphs, for large bounds from the optimal, there is no significant difference between the performance of our algorithm and Sandholm et al.'s: the number of coalition structures that each has to search are similar. However, for small bounds from the optimal, our algorithm is much faster (up to 10^{379} times faster in the graphs shown here). In these cases, the number of coalition structures that our algorithm has to search is much smaller because of our greater selectivity in searching through the subsets of L_k.

Moreover, for small bounds, our algorithm scales very well as n increases. Thus the bigger n is, the more our algorithm outperforms Sandholm et al.'s. For example, with bound 3, our algorithm is more than 10^7 times faster for $n = 50$, more than 10^{23} times faster for $n = 100$, more than 10^{171} times faster for $n = 500$, and more than 10^{379} times

faster for $n = 1000$. Note that these numbers would continue to increase the bigger we made n.

5.3 Summary

In this chapter, we developed an anytime algorithm for coalition structure generation that can produce solutions within a finite bound from the optimal. As it is an anytime algorithm, it can be interrupted at any time and the bound keeps improving with an increase in execution time. We then benchmarked our algorithm against [Sandholm *et al.*, 1999] which is the only other known algorithm for this task that can also establish a worst-case bound from the optimal. This comparison showed our algorithm to be significantly faster; for example, being up to 10^{379} times faster for systems containing 1000 agents for small bounds.

This algorithm addresses one of the key weaknesses in the work on coalition formation to date (namely the complexity of coalition structure generation). With this substantially more efficient algorithm, coalition formation can start to be used in the VO operation phase to partition the VO's members into several subsets working on different activities in order to maximise the VO's payoff.

Chapter 6

Virtual Organisations in Operation

Having detailed the mechanisms for clearing combinatorial auctions and coalition structure generation, this chapter shows how they can be applied to the activities of partner selection and task distribution within VOs. This is done by undertaking a walkthrough of the VO lifecycle on a particular scenario:

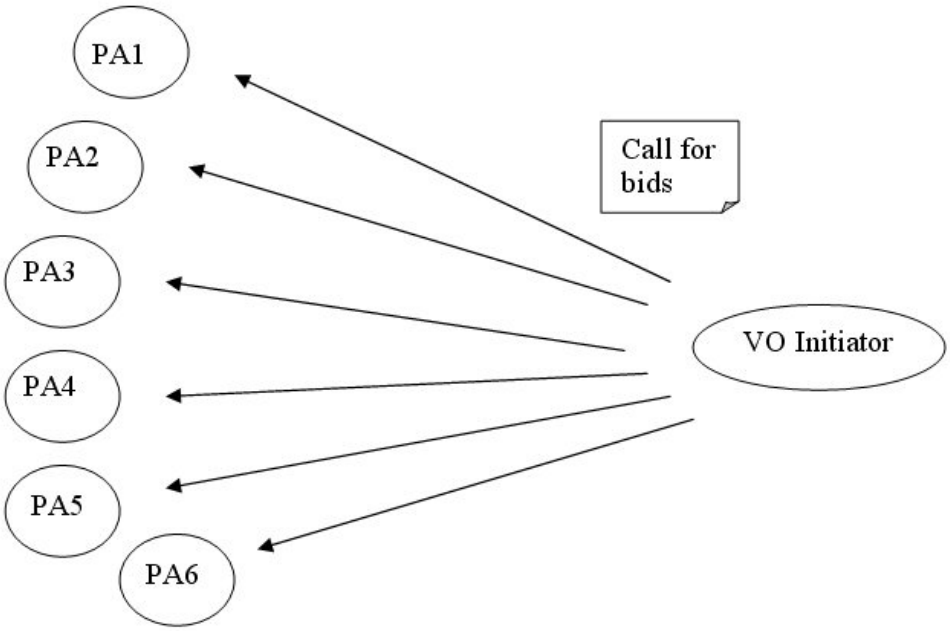
Suppose a festival is going to take place in Edinburgh. Identifying this business opportunity, an enterprising entrepreneur decides to establish a VO to offer festival attendees a one stop shop for their trip. Thus the entrepreneur needs to bring together many hotels, local tour organisers and airlines to meet the needs of the attendees. In so doing, the VO will offer travel packages, which include airline travel, accommodation and tours to various attractions in and around Edinburgh.

Against this background, the chapter presents, in detail, the application of the algorithms that we have developed in previous chapters, to automate partner selection and task distribution in the creation, operation and maintenance phases of the VO. Specifically, section 6.1 will address the creation phase, section 6.2 will deal with the operation phase and section 6.3 will address the maintenance phase.

6.1 The Creation Phase

This section will apply the auction clearing algorithms developed in chapters 3 and 4 to automate partner selection in the creation phase of the VO. Specifically, the creation phase proceeds with the following steps:

Step 1



The VO initiator (the entrepreneur) identifies the market niche and decides to form a VO to exploit this niche; that is, a VO to provide travel packages, which includes airline travel, accommodation and tour to various attractions in and around Edinburgh. The VO initiator then contacts potential VO partners and sends them a call for bids as follows:

Required services and associated quantities:

Air travel:	for 400 persons
Accommodation:	400 rooms
Tour:	for 400 persons

Using our formal notations introduced in previous chapters, the call for bids can be written as (q_1, q_2, q_3) with:

$$\begin{cases} q_1 = 400 \\ q_2 = 400 \\ q_3 = 400 \end{cases}$$

Step 2

The potential VO partners (PA1, PA2, PA3, PA4, PA5, PA6) send the bids back to the VO initiator. The services and the associated quantities offered by the bidders are as follows:

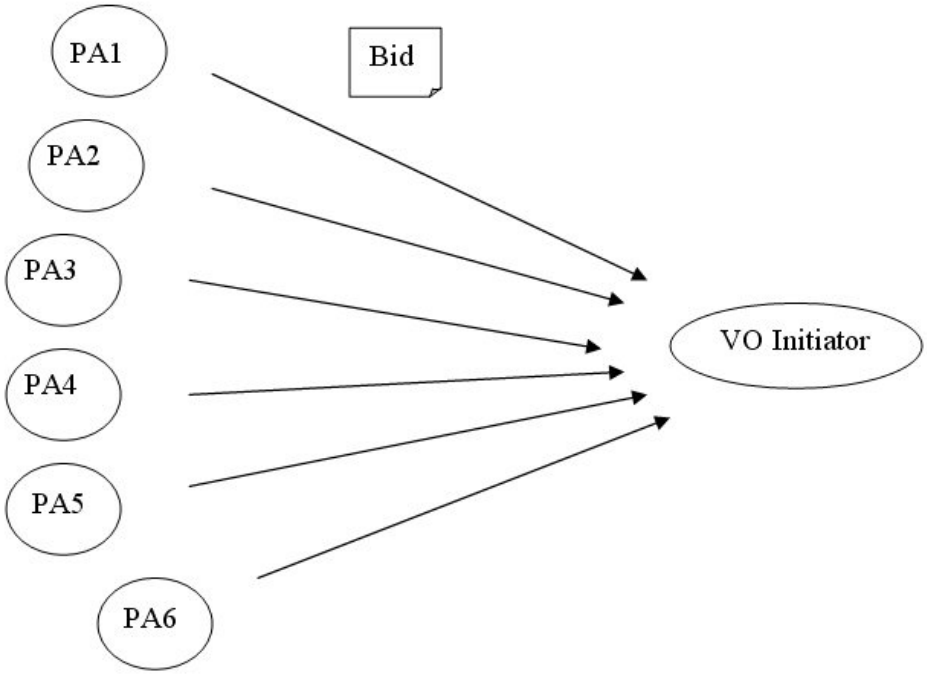
Agent	Air tickets	Accommodation	Local tour
PA1	500	300	
PA2		200	
PA3		300	
PA4			300
PA5	450		
PA6			250

Using our formal notations, this means:

$$\begin{cases} u_1^1 = 500, u_1^2 = 300, u_1^3 = 0 \\ u_2^1 = 0, u_2^2 = 200, u_2^3 = 0 \\ u_3^1 = 0, u_3^2 = 300, u_3^3 = 0 \\ u_4^1 = 0, u_4^2 = 0, u_4^3 = 300 \\ u_5^1 = 450, u_5^2 = 0, u_5^3 = 0 \\ u_6^1 = 0, u_6^2 = 0, u_6^3 = 250 \end{cases}$$

Suppose the details of the bids are as follows:

Bidder PA1 can provide air travel for 500 persons and accommodation for 300 persons. The price function is as follows:



- Air travel: from 1 — 20 persons: the price is 150/person; from 21 — 500 persons: the price is 143/person.

  Using our formal notations, this means:

  $$\begin{cases} P_1^1(r) &= 150 * r, \forall 1 \leq r \leq 20 \\ P_1^1(r) &= 143 * r, \forall 21 \leq r \leq 500 \end{cases}$$

- Accommodation: From 1 — 20 persons: the price is 25/person. From 21 — 300 persons: the price is 23.85/person.

  Using our formal notations, this means:

  $$\begin{cases} P_1^2(r) &= 25 * r, \forall 1 \leq r \leq 20 \\ P_1^2(r) &= 23.85 * r, \forall 21 \leq r \leq 300 \end{cases}$$

- For a package which consists of at least 21 air tickets and at least 21 rooms, the total price of the package is $0.99 * [P(airtickets) + P(accommodation)]$ otherwise the total price is $P(airtickets) + P(accommodation)$.

  Formally, this means:

  $P_1(r_1, r_2) = \omega_1(t_1, t_2)(P_1^1(r_1) + P_1^2(r_2))$ with the correlation function ω_1 calculated as follows:

$$\begin{cases} \omega_1(2,2) & = & 0.99 \\ \omega_1(t_1,t_2) & = & 1, \text{ if } t_1 \neq 2 \text{ or } t_2 \neq 2 \end{cases}$$

Bidder PA2 can provide 200 rooms with the following price function: from 1 — 10 rooms: the price is 24/person; from 11 — 200 rooms: the price is 22.5/person.

Formally, this means:

$$\begin{cases} P_2^2(r) & = & 24 * r, \forall 1 \leq r \leq 10 \\ P_2^2(r) & = & 22.5 * r, \forall 11 \leq r \leq 200 \end{cases}$$

Bidder PA3 can provide 300 rooms with the following price function: from 1 — 20 rooms: the price is 25.5/person; from 21 — 300 rooms: the price is 24.5/person.

Formally:

$$\begin{cases} P_3^2(r) & = & 25.5 * r, \forall 1 \leq r \leq 20 \\ P_3^2(r) & = & 24.5 * r, \forall 21 \leq r \leq 300 \end{cases}$$

Bidder PA4 can provide tour for 300 persons with the following price function: from 1 — 20 persons: the price is 30/person; from 21 — 300 persons: the price is 28.8/person.

Formally:

$$\begin{cases} P_4^3(r) & = & 30 * r, \forall 1 \leq r \leq 20 \\ P_4^3(r) & = & 28.8 * r, \forall 21 \leq r \leq 300 \end{cases}$$

Bidder PA5 can provide air tickets for 450 persons with the following price function: from 1 — 20 persons: the price is 148/person; from 21 — 450 persons: the price is 143.5/person.

Formally:

$$\begin{cases} P_5^1(r) & = & 148 * r, \forall 1 \leq r \leq 20 \\ P_5^1(r) & = & 143.5 * r, \forall 21 \leq r \leq 450 \end{cases}$$

Bidder PA6 can provide tour for 250 persons with the following price function: from 1 — 15 persons: the price is 31/person; from 16 — 250 persons: the price is 30/person.

Formally:

$$\begin{cases} P_6^3(r) &= 31 * r, \forall 1 \leq r \leq 15 \\ P_6^3(r) &= 30 * r, \forall 16 \leq r \leq 250 \end{cases}$$

Step 3: The VO initiator receives the bids, clears the auction using our clearing algorithms and forms the VO.

Using our clearing algorithms, we get the following results:

- Using the polynomial algorithm:
  - At round 1: Consider all the biggest packages offered by the bidders (after truncating them according to the auctioneer's demand) and their average unit price:
    - Bidder PA1: the biggest package offered is (400, 300, 0) (400 air tickets, 300 rooms and) with average unit price 91.94
    - Bidder PA2: (0, 200, 0) with average unit price 22.5
    - Bidder PA3: (0, 300, 0) with average unit price 24.5
    - Bidder PA4: (0, 0, 200) with average unit price 28.8
    - Bidder PA5: (400, 0, 0) with average unit price 143.5
    - Bidder PA6: (0, 0, 250) with average unit price 30

    Thus, PA2 is selected to provide (0, 200, 0).
  - At round 2: The new auctioneer's demand is (400, 200, 400). Again, consider all the biggest packages offered by the bidders (after truncating them according to the auctioneer's demand) and their average unit price:
    - Bidder PA1: (400, 200, 0) with average unit price 103.28
    - Bidder PA3: (0, 200, 0) with average unit price 24.5
    - Bidder PA4: (0, 0, 200) with average unit price 28.8
    - Bidder PA5: (400, 0, 0) with average unit price 143.5
    - Bidder PA6: (0, 0, 250) with average unit price 30

    Thus, PA3 is selected to provide (0, 200, 0).

- At round 3: The new auctioneer's demand is (400, 0, 400). Again, consider all the biggest packages offered by the bidders (after truncating them according to the auctioneer's demand) and their average unit price:
  - Bidder PA1: (400, 0, 0) with average unit price 143
  - Bidder PA4: (0, 0, 200) with average unit price 28.8
  - Bidder PA5: (400, 0, 0) with average unit price 143.5
  - Bidder PA6: (0, 0, 250) with average unit price 30

  Thus, PA4 is selected to provide (0, 0, 200).
- At round 4: The new auctioneer's demand is (400, 0, 200). Again, consider all the biggest packages offered by the bidders (after truncating them according to the auctioneer's demand) and their average unit price:
  - Bidder PA1: (400, 0, 0) with average unit price 143
  - Bidder PA5: (400, 0, 0) with average unit price 143.5
  - Bidder PA6: (0, 0, 200) with average unit price 30

  Thus, PA4 is selected to provide (0, 0, 200).
- At round 5: The new auctioneer's demand is (400, 0, 0). Again, consider all the biggest packages offered by the bidders (after truncating them according to the auctioneer's demand) and their average unit price:
  - Bidder PA1: (400, 0, 0) with average unit price 143
  - Bidder PA5: (400, 0, 0) with average unit price 143.5

  Thus, PA1 is selected to provide (400, 0, 0).

To this end, we have the following winning agents:

PA1 (400 air tickets)

PA2 (200 rooms)

PA3 (200 rooms)

PA4 (tour for 200 persons)

PA6 (tour for 200 persons)

for a total bid price of 78360.

- Using the optimal algorithm: by searching through the dominant set, the winning agents are:

  PA1 (400 air tickets and 200 rooms)

  PA2 (200 rooms)

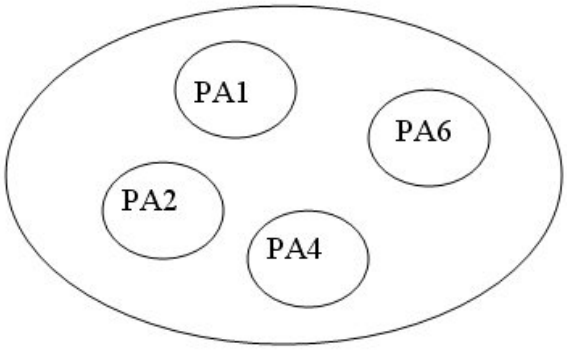
  PA4 (tour for 200 persons)

  PA6 (tour for 200 persons)

  for a total bid price of 78230.

As we can see, the results from the two algorithms are different. It is because the polynomial, while being very fast, cannot guarantee to find the optimal solution.

In this case, we will use the optimal result and so the members of the new VO winning agents are: PA1, PA2, PA4, PA6.



6.2 The Operation Phase

In this section, we will examine the use of coalition structure generation algorithm developed in chapter 5 in VO operation phase. We focus on the third service of the VO: providing tours to various attractions in and around the city. To provide this service, the local tour organiser members of the VO will have to coordinate with the hotel members to pick up visitors from the hotels as well as returning them. We intend to use the coalition structure generation to organise this service in an optimal (that is, cost-minimising) way. In more details:

Suppose there are 5 car centres from where the cars of the local tour organisers start: C1, C2, C3, C4, C5 and there are 2 hotels where the visitors wait to be picked up: H1, H2. The tour drivers then go from one of the car centres to the hotels to pickup the visitors there, take them for a tour then come back.

The question is then to how can we decide which car centres should pickup visitors from which hotels in order to maximise the profit of the VO, that is, to minimise the cost?

To solve this, each car centre as well as each hotel is represented by an autonomous agent. These agents are then coordinated with one another using our coalition structure generation algorithm to maximise the payoff of the system. Specifically, we partition the set of all agents (5 car centres and 2 hotels) into several coalitions such that agents in the same coalition cooperate with one another and agents in different coalitions do not cooperate. For example, suppose we partition the set of agents into the following coalition structure: $\{(C1, C2, H1), (C3, C4, C5, H2)\}$. Then car centre C1, C2 will pickup visitors from hotel H1, while car centres C3, C4, C5 will pickup visitors from hotel H2.

The value of each coalition can be calculated based on the following points:

- If a coalition C contains only car centres or hotels, then $V(C) = 0$, because there is no work done.
- If a coalition C contains C1, C2,..., Cm and H1, H2, , Hn, then $V(C1, C2, , Cm, H1, H2, , Hm)$ is the profit that the VO gets from taking the visitors around.

The aim is then to arrange the car centres and the hotels in order to maximise the profit of the whole VO.

To illustrate how the algorithm works, we can assume the following values of the coalitions:

$V(C1, H1) = 20$ $V(C2, H1) = 32$

$V(C3, H1) = 23$ $V(C4, H1) = 21$

$V(C5, H1) = 20$ $V(C1, H2) = 22$

$V(C2, H2) = 31$ $V(C3, H2) = 33$

$V(C4, H2) = 45$ $V(C5, H2) = 20$

$V(C1, H1, H2) = 30$ $V(C2, H1, H2) = 31$

$V(C3, H1, H2) = 28$ $V(C4, H1, H2) = 29$

$V(C5, H1, H2) = 26$ $V(C1, C2, H1) = 31$

$V(C1, C3, H1) = 32$ $V(C1, C4, H1) = 44$

$V(C2, C3, H1) = 33$ $V(C2, C4, H1) = 34$

$V(C3, C4, H1) = 35$ $V(C1, C5, H1) = 32$

$V(C2, C5, H1) = 41$ $V(C3, C5, H1) = 34$

$V(C4, C5, H1) = 31$ $V(C1, C2, H2) = 32$

$V(C1, C3, H2) = 43$ $V(C1, C4, H2) = 34$

$V(C2, C3, H2) = 45$ $V(C2, C4, H2) = 33$

$V(C3, C4, H2) = 34$ $V(C1, C5, H2) = 33$

$V(C2, C5, H2) = 31$ $V(C3, C5, H2) = 42$

$V(C4, C5, H2) = 34$ $V(C1, C2, H1, H2) = 37$

$V(C1, C3, H1, H2) = 38$ $V(C1, C4, H1, H2) = 37$

$V(C2, C3, H1, H2) = 38$ $V(C2, C4, H1, H2) = 38$

$V(C3, C4, H1, H2) = 37$ $V(C1, C5, H1, H2) = 36$

$V(C2, C5, H1, H2) = 37$ $V(C3, C5, H1, H2) = 36$

$V(C4, C5, H1, H2) = 38$ $V(C1, C2, C3, H1) = 42$

$V(C1, C2, C4, H1) = 31$ $V(C1, C3, C4, H1) = 30$

$V(C2, C3, C4, H1) = 31$ $V(C5, C1, C2, H1) = 30$

$V(C5, C1, C3, H1) = 31$ $V(C5, C1, C4, H1) = 29$

$V(C5, C2, C3, H1) = 32$ $V(C5, C2, C4, H1) = 28$

$V(C5, C3, C4, H1) = 31$ $V(C1, C2, C3, H2) = 31$

$V(C1, C2, C4, H2) = 29$ $V(C1, C3, C4, H2) = 30$

$V(C2, C3, C4, H2) = 31$ $V(C5, C1, C2, H2) = 31$

$V(C5, C1, C3, H2) = 30$ $V(C5, C1, C4, H2) = 29$

$V(C5, C2, C3, H2) = 30$ $V(C5, C2, C4, H2) = 29$

$V(C5, C3, C4, H2) = 31$ $V(C1, C2, C3, H1, H2) = 34$

$V(C1, C2, C4, H1, H2) = 35$ $V(C1, C3, C4, H1, H2) = 33$

$V(C2, C3, C4, H1, H2) = 34$ $V(C5, C1, C2, H1, H2) = 31$

$V(C5, C1, C3, H1, H2) = 31$ $V(C5, C1, C4, H1, H2) = 29$

$V(C5, C2, C3, H1, H2) = 32$ $V(C5, C2, C4, H1, H2) = 28$

$V(C5, C3, C4, H1, H2) = 30$ $V(C1, C2, C3, C4, H1) = 32$

$V(C1, C2, C3, C5, H1) = 31$ $V(C1, C3, C4, C5, H1) = 30$

$V(C1, C2, C4, C5, H1) = 32$ $V(C2, C3, C4, C5, H1) = 31$

$V(C1, C2, C3, C4, H2) = 43$ $V(C1, C2, C3, C5, H2) = 42$

$V(C1, C3, C4, C5, H2) = 41$ $V(C1, C2, C4, C5, H2) = 40$

$V(C2, C3, C4, C5, H2) = 42$ $V(C1, C2, C3, C4, H1, H2) = 46$

$V(C1, C2, C3, C5, H1, H2) = 44$ $V(C1, C3, C4, C5, H1, H2) = 45$

$V(C1, C2, C4, C5, H1, H2) = 44$ $V(C2, C3, C4, C5, H1, H2) = 45$

$V(C1, C2, C3, C4, C5, H1) = 30$ $V(C1, C2, C3, C4, C5, H2) = 31$

$V(C1, C2, C3, C4, C5, H1, H2) = 49$

We can then use the coalition structure generation that we developed in chapter 5 to achieve the sub-optimal solutions depends on the time of calculation permitted. The more time for calculation, the better the solution is.

Here is the execution of our algorithm (see figure 6.1 for a diagramatic presentation) and the result:

- After round 1 (searching layer 1, 2 and 7): the result coalition structure is: $\{(C4, H2), (C1, C2, C3, C5, H1)\}$. The group payoff is: 76.00.

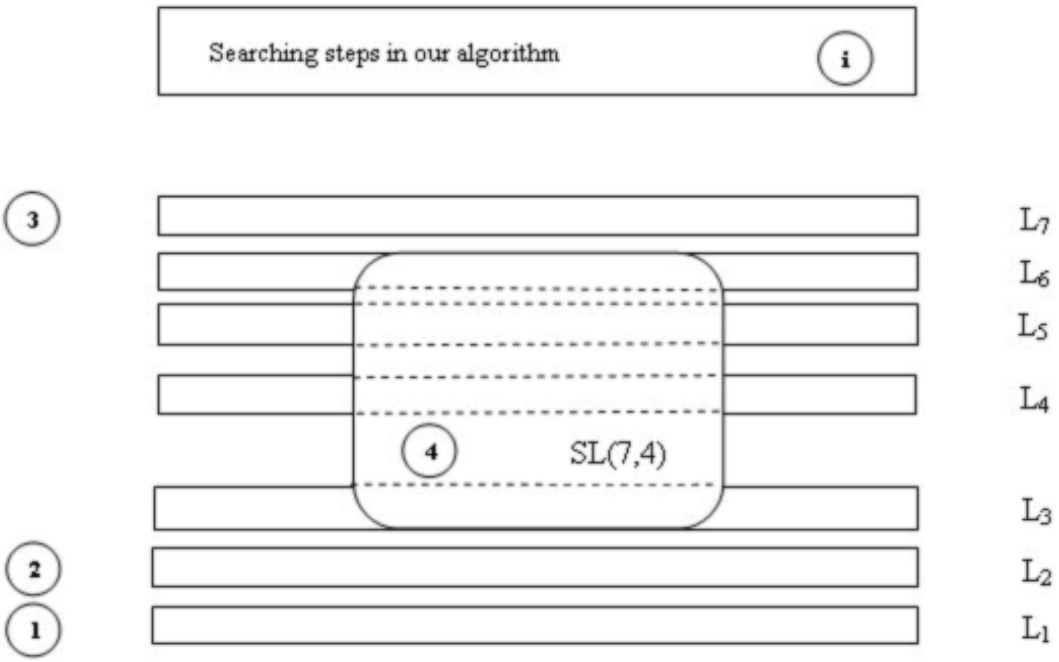
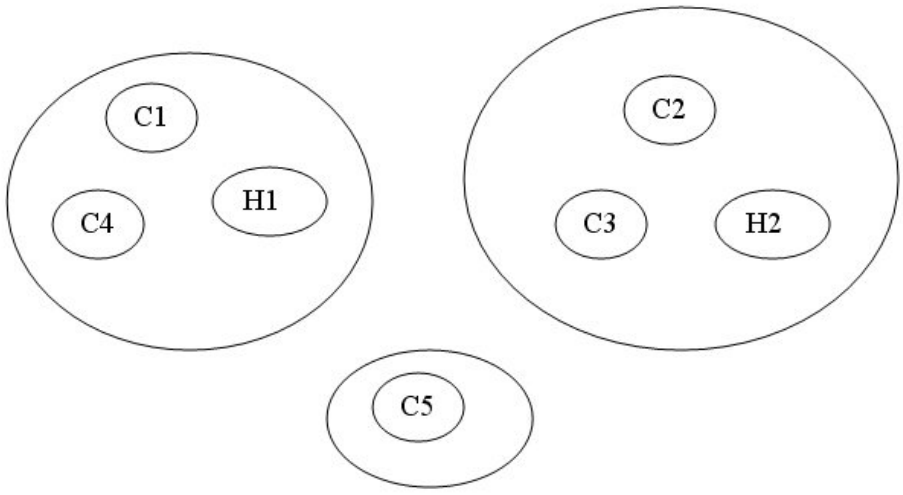


FIGURE 6.1: The searching steps of our coalition structure generation in the operation phase of the scenario.

- After round 2, (searching SL(7,4)) the result coalition structure is $\{(C1, C2, C3, H1), (C4, H2), (C5)\}$. The group payoff is 87.00.[1]

Suppose we stop the algorithm after round 2 (with $n = 7$, our algorithm has only 2 rounds), we then have the following coalition structure:



[1]We also try searching through all the coalition structures and get the optimal one as follows: $\{(C1, C4, H1), (C2, C3, H2), (C5)\}$ with payoff 89.00.

6.3 The Maintenance Phase

As stated in chapter 1, once the VO is in operation, there are two main situations that may arise in an uncertain environment:

- Some members fail or withdraw from the VO: in such cases, the VO will have to find the substitutes to add into the VO.
- The situation changes, but the members remain unchanged. This means that the work distribution between VO members needs to be reorganised.

Each of these will now be dealt with in turn.

6.3.1 Adding New Members into The VO

In this case, some members fail or withdraw from the VO which means that the VO will have to find the substitutes to add into the VO. To illustrate this point, suppose member PA1 withdraws from the VO. Now the VO will use a reverse auction mechanism (similar to that used in the creation phase) to find substitutions for this member. Specifically, the VO follows the 3 steps similar to those in section 6.1.

Step 1

The VO contacts potential VO partners and sends them a call for bids. The detail of the call for bids (the required services and the associated quantities) is as follows (this is what PA1 provided in the VO):

Air travel:	for 400 persons
Accommodation:	200 rooms

Formally, the call for bids can be written as (q_1, q_2) with:

$$\begin{cases} q_1 = 400 \\ q_2 = 200 \end{cases}$$

Step 2

The potential VO partners (PA7, PA8, PA9, PA10) sends the bids back to the VO.

Agent	Air tickets	Accommodation
PA7	300	150
PA8	200	250
PA9		300
PA10	250	

Using our formal notations, this means:

$u_1^1 = 300, u_1^2 = 150$

$u_2^1 = 200, u_2^2 = 250$

$u_3^1 = 0, u_3^2 = 300$

$u_4^1 = 250, u_4^2 = 0$

Suppose the details of the bids are as follows:

PA7 can provide air travel for 300 persons and accommodation for 150 persons with the following price function:

- Air travel: from 1 — 20 persons: price is 151/person; from 21 — 300 persons: price is 148.5/person.

  Formally:

  $$\begin{cases} P_1^1(r) &= 151 * r, \forall 1 \leq r \leq 20 \\ P_1^1(r) &= 148.5 * r, \forall 21 \leq r \leq 300 \end{cases}$$

- Accommodation: from 1 — 25 persons: price is 26/person; from 26 — 150 persons: the price is 25.5/person.

  Formally:

  $$\begin{cases} P_1^2(r) &= 26 * r, \forall 1 \leq r \leq 25 \\ P_1^2(r) &= 25.5 * r, \forall 26 \leq r \leq 150 \end{cases}$$

- For a package which comprises of at least 21 air tickets and at least 26 rooms, the total price of the package is $0.99 * [P(airtickets) + P(accommodation)]$ otherwise the total price is $P(airtickets) + P(accommodation)$.

  Formally:

  $P_1(r_1, r_2) = \omega_1(t_1, t_2)(P_1^1(r_1) + P_1^2(r_2))$ with the correlation function ω_1 calculated as follows:

  $$\begin{cases} \omega_1(2,2) & = & 0.99 \\ \omega_1(t_1,t_2) & = & 1, \text{ if } t_1 \neq 2 \text{ or } t_2 \neq 2 \end{cases}$$

PA8 can provide air travel for 200 persons and accommodation for 250 persons with the following price function:

- Air travel: from 1 — 10 persons: price is 152/person; from 11 — 200 persons: price is 148/person.

  Formally:

  $$\begin{cases} P_2^1(r) & = & 152 * r, \forall 1 \leq r \leq 10 \\ P_2^1(r) & = & 148 * r, \forall 11 \leq r \leq 200 \end{cases}$$

- Accommodation: from 1 — 20 persons: price is 25.4/person; from 21 — 250 persons: the price is 25/person.

  Formally:

  $$\begin{cases} P_2^2(r) & = & 25.4 * r, \forall 1 \leq r \leq 20 \\ P_2^2(r) & = & 25 * r, \forall 21 \leq r \leq 250 \end{cases}$$

- For a package which comprises of at least 11 air tickets and at least 21 rooms, the total price of the package is $0.98 * [P(airtickets) + P(accommodation)]$ otherwise the total price is $P(airtickets) + P(accommodation)$.

Formally:

$P_2(r_1, r_2) = \omega_2(t_1, t_2)(P_2^1(r_1) + P_2^2(r_2))$ with the correlation function ω_2 calculated as follows:

$$\begin{cases} \omega_2(2,2) & = & 0.98 \\ \omega_2(t_1,t_2) & = & 1, \text{ if } t_1 \neq 2 \text{ or } t_2 \neq 2 \end{cases}$$

PA9 can provide accommodation for 300 persons: from 1 — 20 persons: the price is 26.5/person; from 21 — 300 persons: the price is 25.5/person.

Formally:

$$\begin{cases} P_3^2(r) & = & 26.5 * r, \forall 1 \leq r \leq 20 \\ P_3^2(r) & = & 25.5 * r, \forall 21 \leq r \leq 300 \end{cases}$$

PA10 can provide air travel for 250 persons: from 1 — 20 persons: the price is 151/person; from 21 — 300 persons: the price is 147/person.

Formally:

$$\begin{cases} P_4^1(r) & = & 151 * r, \forall 1 \leq r \leq 20 \\ P_4^1(r) & = & 147 * r, \forall 21 \leq r \leq 300 \end{cases}$$

Using our clearing algorithms, we get the following results:

- Using the polynomial algorithm:
  - At round 1: Consider all the biggest packages offered by the bidders (after truncating them according to the auctioneer's demand) and their average unit price:
    - Bidder PA7: the biggest package offered is (300, 150) (300 air tickets, 150 rooms) with average unit price 107.5
    - Bidder PA8: (200, 200) with average unit price 86.5
    - Bidder PA9: (0, 200) with average unit price 25.5
    - Bidder PA10: (200, 0) with average unit price 147

    Thus, PA9 is selected to provide (0, 200).
  - At round 2: The new auctioneer's demand is (400, 0). Again, consider all the biggest packages offered by the bidders (after truncating them according to the auctioneer's demand) and their average unit price:

    - Bidder PA7: the biggest package offered is (300, 0) with average unit price 148.5
    - Bidder PA8: (200, 0) with average unit price 148
    - Bidder PA10: (200, 0) with average unit price 147

  Thus, PA10 is selected to provide (200, 0).

- At round 3: The new auctioneer's demand is (200, 0). Again, consider all the biggest packages offered by the bidders (after truncating them according to the auctioneer's demand) and their average unit price:
    - Bidder PA7: the biggest package offered is (200, 0) with average unit price 148.5
    - Bidder PA8: (200, 0) with average unit price 148

  Thus, PA8 is selected to provide (200, 0).

To this end, we have the following winning agents:

PA8 (200 air tickets)

PA9 (200 rooms)

PA10 (200 air tickets)

for a total bid price of 64100.

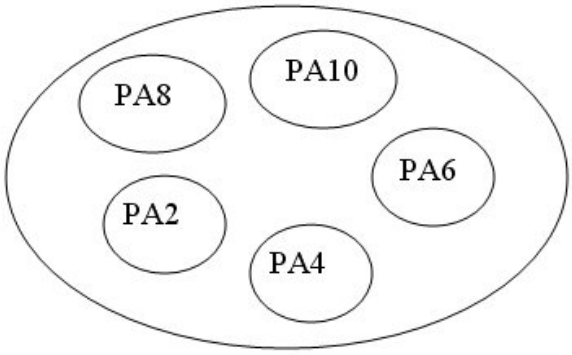
- Using the optimal algorithm: by searching through the dominant set, the winning agents are:

  PA8 (200 air tickets and 200 rooms)

  PA10 (200 air tickets)

  for a total bid price of 64000.

In this case, we will use the optimal result and so the members of the new VO winning agents are: PA2, PA4, PA6, PA8 and PA10.



6.3.2 Re-organising The Work

In this case the situation changes in some way (e.g. the production cost of a product/service increases in some regions) but the members remain unchanged. This means that the work distribution between VO members may need to be reorganised in order to maximise the profit of the VO as a whole. To illustrate this, consider the case in which there is a major work road in the city in progress. This forces the tour cars to change their route, and so the values of all the coalitions change. In response to this situation, the VO re-calculates the coalition values and re-runs our coalition structure generation to re-organise the work.

Suppose the new values for the coalitions are (the values those have been changed are highlighted in **bold**):

V(C1,H1) = 20	**V(C2,H1) = 22**
V(C3,H1) = 33	V(C4,H1) = 21
V(C5,H1) = 23	V(C1,H2) = 22
V(C2,H2) = 31	**V(C3,H2) = 43**
V(C4,H2) = 35	V(C5,H2) = 20
V(C1,H1,H2) = 30	V(C2,H1,H2) = 31
V(C3,H1,H2) = 28	V(C4,H1,H2) = 29

V(C5,H1,H2) = 33	V(C1,C2,H1) = 31
V(C1,C3,H1) = 32	**V(C1,C4,H1) = 34**
V(C2,C3,H1) = 33	**V(C2,C4,H1) = 44**
V(C3,C4,H1) = 35	V(C1,C5,H1) = 32
V(C2,C5,H1) = 41	**V(C3,C5,H1) = 27**
V(C4,C5,H1) = 31	V(C1,C2,H2) = 32
V(C1,C3,H2) = 43	V(C1,C4,H2) = 34
V(C2,C3,H2) = 45	**V(C2,C4,H2) = 43**
V(C3,C4,H2) = 34	V(C1,C5,H2) = 33
V(C2,C5,H2) = 31	V(C3,C5,H2) = 42
V(C4,C5,H2) = 34	V(C1,C2,H1,H2) = 37
V(C1,C3,H1,H2) = 38	V(C1,C4,H1,H2) = 37
V(C2,C3,H1,H2) = 38	V(C2,C4,H1,H2) = 38
V(C3,C4,H1,H2) = 37	V(C1,C5,H1,H2) = 36
V(C2,C5,H1,H2) = 37	**V(C3,C5,H1,H2) = 42**
V(C4,C5,H1,H2) = 38	**V(C1,C2,C3,H1) = 34**
V(C1,C2,C4,H1) = 31	V(C1,C3,C4,H1) = 30
V(C2,C3,C4,H1) = 31	V(C5,C1,C2,H1) = 30
V(C5,C1,C3,H1) = 31	V(C5,C1,C4,H1) = 29
V(C5,C2,C3,H1) = 32	V(C5,C2,C4,H1) = 28
V(C5,C3,C4,H1) = 31	V(C1,C2,C3,H2) = 31
V(C1,C2,C4,H2) = 47	V(C1,C3,C4,H2) = 30
V(C2,C3,C4,H2) = 31	V(C5,C1,C2,H2) = 31
V(C5,C1,C3,H2) = 32	V(C5,C1,C4,H2) = 29
V(C5,C2,C3,H2) = 30	V(C5,C2,C4,H2) = 29

V(C5,C3,C4,H2) = 31

V(C1,C2,C4,H1,H2) = 30

V(C2,C3,C4,H1,H2) = 34

V(C5,C1,C3,H1,H2) = 31

V(C5,C2,C3,H1,H2) = 32

V(C5,C3,C4,H1,H2) = 30

V(C1,C2,C3,C5,H1) = 31

V(C1,C2,C4,C5,H1) = 32

V(C1,C2,C3,C4,H2) = 38

V(C1,C3,C4,C5,H2) = 41

V(C2,C3,C4,C5,H2) = 42

V(C1,C2,C3,C5,H1,H2) = 44

V(C1,C2,C4,C5,H1,H2) = 44

V(C1,C2,C3,C4,C5,H1) = 30

V(C1,C2,C3,C4,C5,H1,H2) = 48

V(C1,C2,C3,H1,H2) = 34

V(C1,C3,C4,H1,H2) = 33

V(C5,C1,C2,H1,H2) = 31

V(C5,C1,C4,H1,H2) = 34

V(C5,C2,C4,H1,H2) = 28

V(C1,C2,C3,C4,H1) = 32

V(C1,C3,C4,C5,H1) = 30

V(C2,C3,C4,C5,H1) = 31

V(C1,C2,C3,C5,H2) = 42

V(C1,C2,C4,C5,H2) = 40

V(C1,C2,C3,C4,H1,H2) = 44

V(C1,C3,C4,C5,H1,H2) = 43

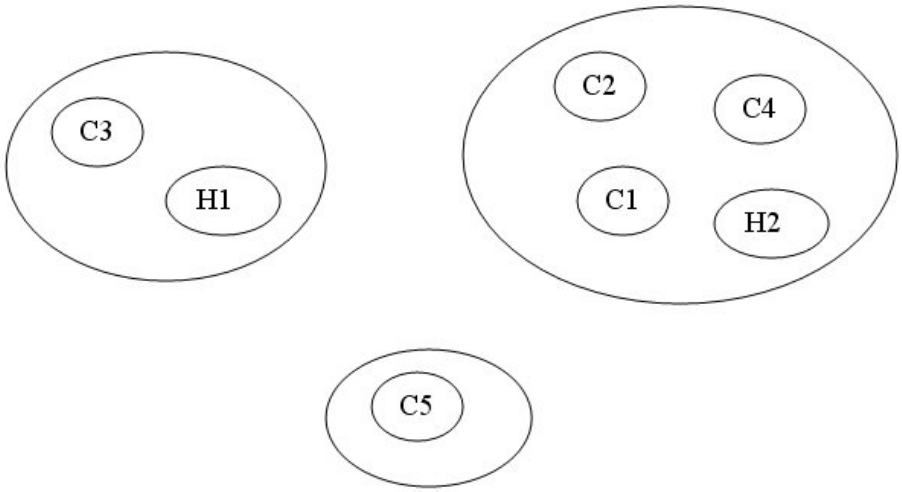
V(C2,C3,C4,C5,H1,H2) = 45

V(C1,C2,C3,C4,C5,H2) = 32

Here is the result:

- After round 1 (searching layer 1, 2 and 7): the result coalition structure is: $\{(C2, C4, H1), (C1, C3, C5, H2)\}$. The group payoff is: 76.00.
- After round 2 (searching SL(7,4)): the result coalition structure is $\{(C3, H1), (C1, C2, C4, H2), (C5)\}$. The group payoff is 80.00.

We then have the following coalition structure:


C3
H1
C2
C4
C1
H2
C5

Chapter 7

Conclusions

This thesis developed efficient mechanisms to automate the creation, operation and maintenance phases of the VO lifecycle. Specifically, novel algorithms were developed for the problems of:

- partner selection
- task distribution

In the case of partner selection, the thesis used an auction mechanism, specifically, combinatorial auctions, to select which agents should be part of the VO. To do this, however, we needed to develop new clearing algorithms for the types of combinatorial auctions that are most suitable to the VO environment. In particular, these algorithms had to deal with multi-unit combinatorial auctions with bids in form of demand/supply functions. Specifically, two sets of algorithms were developed: polynomial clearing algorithms and optimal clearing algorithms. The former operate in polynomial time and can establish a worst-case bound from the optimal, but are not completely efficient (because the solutions that they generate may not be optimal). In contrast, the later are guaranteed to find the optimal solutions, but are not polynomial.

In more detail, chapter 3 developed, for the first time, polynomial algorithms for clearing multi-unit combinatorial auctions with demand/supply functions. While previous

work has focused on single-item auctions with demand/supply curves or combinatorial auctions with atomic propositions, we generalised the problem to multi-unit single-item and multi-unit combinatorial auctions with discount and free disposal demand/supply functions. For this very general case, we showed that our algorithms are of polynomial complexity and can generate solutions that are within a bound of the optimal. Our empirical evaluation showed that our algorithms can produce solutions that are within a bound of the optimal that is much lower than the theoretically proved bound. Specifically, all of the bounds are within the range $[1, 1.025]$, while the theoretical bound is n in the multi-unit single-item case, and is $2nK^{m-1}$ in the multi-unit combinatorial case (n is the number of bidders, m is the number of items, and K is a constant). This generalisation and the tractability of these algorithms enable the efficient application of combinatorial auctions in VO partner selection, and more generally, they are important steps toward realising the full application potential of combinatorial auctions since it enables us to deal with a maximally flexible and efficient scheme in a computationally tractable manner.

On the other hand, chapter 4 presents, for the first time, optimal clearing algorithms for multi-unit single-item and multi-unit combinatorial auctions where bids are expressed through supply/demand functions. Specifically, we consider two classes of supply/demand functions where the demand/supply curves for each individual commodity are piece-wise linear (an important and often considered case) and where the demand/supply curves are monotonic one-unit-difference (a less common case). Again, the expressibility of our auction setting and the optimality of the solutions generated by our algorithms are an important step towards the use of combinatorial auctions in VO partner selection, as well as in the general area of e-commerce applications.

In the case of task distribution, the thesis used coalition formation mechanisms to partition the set of VO members into several subsets working on various activities in order to maximise the payoff of the whole VO. Again, as one of the main reasons that hinders the wide spread use of coalition formation is the computational complexity of the coalition structure generation, we developed an algorithm to overcome this shortcoming. Specifically, chapter 5 developed an anytime algorithm for coalition structure generation

that can produce solutions within a finite bound from the optimal. As it is an anytime algorithm, it can be interrupted at any time and the bound keeps improving with an increase in execution time. We then benchmarked our algorithm against [Sandholm *et al.*, 1999] which is the only other known algorithm for this task that can also establish a worst-case bound from the optimal. This comparison showed our algorithm to be significantly faster; for example, being up to 10^{379} times faster for systems containing 1000 agents for small bounds. With this significant jump forward, we believe coalition formation algorithms can now be used in the VO operation phase to partition the VO's members into several subsets working on different activities in order to maximise the VO's payoff.

Building on this work, there are still a number of areas for further work in terms of both auction clearing and coalition formation.

In the area of auction clearing, further research is needed to try to lower the bound from the optimal of the solutions of the polynomial algorithms and lower the computational complexity of the optimal algorithm. In the former case, one way to achieve this is to narrow down the classes of bidding functions. That is, find special classes in which our algorithms perform better than in the general settings investigated in this thesis. Another direction could be to modify the way our algorithms select winning bids. In the later case, it may be possible to apply heuristic techniques such as Branch-and-Bound. Although when we apply these techniques, the algorithm may not be guaranteed to find the optimal solution, the improved complexity may make it more broadly applicable in real-life scenarios.

Another important area of work is to try to obtain better benchmark platforms for evaluating clearing algorithms. Specifically, although the problem generator used in this thesis is claimed to resemble real-life problems, it is still artificial. Ideally, we would like to benchmark the algorithms on real-life data in order to give demonstrating more realistic results.

In the area of coalition formation, further research is needed to lower the complexity of the coalition structure generation algorithms still further. It would also be desirable to try to determine some lower bound and/or some upper bound on the computational

complexity of coalition structure generation algorithms in order for them to establish a specific bound from the optimal. The other direction is to develop heuristic algorithms that perform well in common cases and are applicable to large multi-agent systems (that contains hundreds of agents). Another important area is to see whether we can modify our algorithms so that they will be more suitable to dynamic environments in which coalition values may change and agents can enter/leave the environment. To adapt to this environment, we may have to frequently re-calculate the coalitional values and make the coalition formation algorithm fully distributed and more robust.

More generally speaking, further work is needed in a number of areas before the full vision of agile VOs created on demand to fulfill a particular niche is truly met. Such work includes the ability to determine, on the fly, how a particular service can be composed out of the available constituent sub-services, how the need or opportunity for a new VO can be automatically determined, and how a VO can determine when it should be disbanded. Nevertheless, our work on efficient mechanisms for partner selection and task distribution provide an important piece of this complex landscape.

Bibliography

[Beer *et al.*, 1990] M. Beer, R.A. Eisenstat, and B. Spector. Why change programs don't produce change. *Harvard Business Review*, 68(6):158–166, 1990.

[Camarinha-Matos and Afsarmanesh, 1999] Luis M. Camarinha-Matos and Hamideh Afsarmanesh. Virtual enterprises: Life cycle supporting tools and technologies. In *Proceedings of The First IFIP/PRODNET Working Conference on infrastructures for industrial virtual enterprises*, 1999.

[Contractor and Lorange, 1988] F. Contractor and P. Lorange. *Why Should Firms Co-operate?* Lexington Books, 1988.

[Dang and Jennings, 2002] V. D. Dang and N. R. Jennings. Polynomial algorithms for clearing multi-unit single item and multi-unit combinatorial reverse auctions. In *Proceedings of The Fifteenth European Conference on Artificial Intelligence*, pages 23–27, 2002.

[Dang and Jennings, 2003] V. D. Dang and N. R. Jennings. Optimal clearing algorithms for multi-unit single item and multi-unit combinatorial auctions with demand/supply function bidding. In *Proceedings of the Fifth International Conference on Electronic Commerce*, pages 25–30, 2003.

[Dang and Jennings, 2004a] V. D. Dang and N. R. Jennings. Generating coalition structures with finite bound from the optimal guarantees. In *Proceedings of the Third International Conference on Autonomous Agents and Multi-Agent Systems*, pages 564–571, 2004.

[Dang and Jennings, 2004b] V. D. Dang and N. R. Jennings. Polynomial algorithms for clearing multi-unit single-item and multi-unit combinatorial auctions. *Artificial Intelligence Journal*, 2004. Under review.

[Davenport and Kalagnanam, 2001] A. Davenport and J. Kalagnanam. Price negotiations for procurement of direct inputs. IBM Research Report RC 22078, 2001.

[Daviddrajuh and Deng, 2000] R. Daviddrajuh and Z. Deng. Identifying potential suppliers for formation of virtual manufacturing systems. 2000.

[Eso *et al.*, 2001] M. Eso, S. Ghosh, J. Kalagnanam, and L. Ladanyi. Bid evaluation in procurement auctions with piece-wise linear supply curves. IBM Research Report RC 22219, 2001.

[Fischer *et al.*, 1996] Klaus Fischer, Jorg P. Muller, Ingo Heimig, and August-Wilhelm Scheer. Intelligent agents in virtual enterprises. In *Proceedings of the First International Conference and Exhibition on the Practical Application of Intelligent Agents and Multi-Agent Technology*, pages 205–223, 1996.

[Gonen and Lehmann, 2000] Rica Gonen and Daniel Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *Proceedings of The Second ACM Conference on Electronic Commerce*, pages 13–20, 2000.

[Hardwick and Bolton, 1997] Martin Hardwick and Richard Bolton. The industrial virtual enterprise. *Communications of the ACM*, 40:59 – 60, 1997.

[Jennings and Wooldridge, 1995] Nicholas R. Jennings and M. Wooldridge. Applying agent technology. *Applied Artificial Intelligence: An International Journal*, 9:351–361, 1995.

[Ketchpel, 1994] Steven P. Ketchpel. Forming coalitions in the face of uncertain rewards. In *Proceedings of the Twelth National Conference on Artificial Intelligence*, pages 414–419, 1994.

[Klusch and Gerber, 2002] M. Klusch and A. Gerber. Dynamic coalition formation among rational agents. *IEEE Intelligent Systems*, 17(3):42–47, 2002.

[Leyton-Brown *et al.*, 2000] Kevin Leyton-Brown, Yoav Shoham, and Moshe Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *Proceedings of The Seventeenth National Conference on Artificial Intelligence (AAAI)*, pages 56–61, 2000.

[Martello and Toth, 1990] Silvano Martello and Paolo Toth. *Knapsack Problems: Algorithms and Computer Implementations (Wiley-Interscience Series in Discrete Mathematics & Optimization)*. John Wiley and Sons Ltd, 1990.

[McAfee and McMillan, 1987] McAfee and McMillan. Auctions and bidding. *Journal of Economic Literature*, 25:699–738, 1987.

[Neubert *et al.*, 2001] Ralf Neubert, Oliver Langer, Otmar Gorlitz, and Wolfgang Benn. Virtual enterprises - challenges from a database perspective. In *Proceedings of the Workshop on Information Technology for Virtual Enterprises*, pages 98 – 106, 2001.

[NIIIP, 1998] NIIIP. National industrial information infrastructure protocols (NIIIP) reference architecture, 1998.

[Nisan, 2000] Noam Nisan. Bidding and allocation in combinatorial auctions. In *Proceedings of The Second ACM Conference on Electronic Commerce*, pages 1–12, 2000.

[Norman *et al.*, 2003] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, A. Gray, and N. Fiddian. Conoise: Agent-based formation of virtual organisations. In *Proceedings of the Twenty-Third SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence*, pages 353–366, 2003.

[Norman *et al.*, 2004] T. J. Norman, A. Preece, S. Chalmers, N. R. Jennings, M. Luck, V. D. Dang, T. D. Nguyen, V. Deora, J. Shao, A. Gray, and N. Fiddian. Agent-based formation of virtual organisations. *International Journal of Knowledge Based Systems*, 17(2-4):103–111, 2004.

[Porter, 1980] M. Porter. *Competitive strategy*. New York: Free Press, 1980.

[Rabelo *et al.*, 1998] Ricardo J. Rabelo, Luis M. Camarinha-Matos, and Hamideh Afsarmanesh. Multiagent perspectives to agile scheduling. In *Proceedings of IEEE/IFIP International Conference on Balanced Automation Systems*, 1998.

[Rapoport and Kahan, 1984] Amnon Rapoport and James P. Kahan. *Theories of Coalition Formation*. Lawrence Erlbaum Associates, 1984.

[Rocha and Oliveira, 1999] Ana Paula Rocha and Eugenio Oliveira. An electronic market architecture for the formation of virtual enterprises. In *Proceedings of the IFIP/PRODNET Conference on Infrastructures for Industrial Virtual Enterprises*, 1999.

[Roman, 1984] Steven Roman. *The Umbral Calculus*. Academic Press, 1984.

[Rothkopf *et al.*, 1998] Michael H. Rothkopf, Aleksandar Pekec, and Ronald M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44:1131–1147, 1998.

[Sandholm and Suri, 2001] Tuomas Sandholm and Subhash Suri. Market clearability. In *Proceedings of The Seventeenth International Joint Conference on Artificial Intelligence*, pages 1145–1151, 2001.

[Sandholm *et al.*, 1999] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohm. Coalition structure generation with worst case guarantees. *Artificial Intelligence*, 111(1-2):209–238, 1999.

[Sandholm *et al.*, 2002] Tuomas Sandholm, Subhash Suri, Andrew Gilpin, and David Levine. Winner determination in combinatorial auction generalizations. In *Proc. 1st Intl. Joint Conf. on Autonomous Agents and Multiagent Systems*, pages 69–76, 2002.

[Sandholm, 1999] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of The Sixteenth International Joint Conference on Artificial Intelligence*, pages 542–547, 1999.

[Shehory and Kraus, 1995] Onn Shehory and Sarit Kraus. Coalition formation among autonomous agents: Strategies and complexity. *Lecture Notes in Artificial Intelligence*, (957):57–72, 1995.

[Shehory and Kraus, 1996] Onn Shehory and Sarit Kraus. A kernel-oriented model for coalition-formation in general environments: Implementation and results. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, pages 134–140, 1996.

[Shehory and Kraus, 1998] Onn Shehory and Sarit Kraus. Methods for task allocation via agent coalition formation. *Artificial Intelligence*, 101(1-2):165–200, 1998.

[Sieber and Griese, 1998] Pascal Sieber and Joachim Griese. Organizational virtualness. 1998.

[Stephens, 1999] Jeff Stephens. Virtual enterprises using groupware tools and distributed architectures public final report. Technical report, 1999.

[Tennenholtz, 2000] Moshe Tennenholtz. Some tractable combinatorial auctions. In *Proceedings of The Seventeenth National Conference on Artificial Intelligence (AAAI)*, pages 98–103, 2000.

[Wurman, 2001] Peter R. Wurman. Dynamic pricing in the virtual marketplace. *IEEE Internet Computing*, 5(2):36–42, March/April 2001.

[Zlotkin and Rosenschein, 1994] G. Zlotkin and J.S. Rosenschein. Coalition, cryptography and stability: Mechanisms for coalition formation in task oriented domains. In *Proceedings of the Twelth National Conference on Artificial Intelligence*, pages 432–437, 1994.