Computation Market Design and Experiments

by

Chunming Chen

A thesis submitted to the Faculty of Graduate Studies

in partial satisfaction of the

requirements for the degree of

Master of Science

Department of Computer Science

Faculty of Graduate Studies

University of Manitoba

0-612-76911-9

THE UNIVERSITY OF MANITOBA

FACULTY OF GRADUATE STUDIES

*****

COPYRIGHT PERMISSION

COMPUTATION MARKET DESIGN AND EXPERIMENTS

BY

Chunming Chen

A Thesis/Practicum submitted to the Faculty of Graduate Studies of The University of Manitoba in partial fulfillment of the requirement of the degree

of

MASTER OF SCIENCE

Chunming Chen © 2002

Abstract

Computation Market (CM) is an auctioning based resource trading/acquiring system that can be deployed in wide-area computing systems such as Grid systems. This study mainly focuses on the auction market design and experiments. The auction market design entails designing an economic model, i.e., protocols, rules, policies, and strategies, by using game theory. The CM adopts a multi-unit computational auction (MUCA) which allows the users to bid on multiple items at one time and then allocate in an "all or none" manner. The MUCA offers the CM the capability to allow the users to bid flexibly, efficiently, and easily. However, it also brings a problem of computational complexity–determining the winners for a MUCA problem is NP-hard. In this thesis, we propose a set of heuristics in conjunction with defining rules and policies to solve the MUCA problem. Simulations are performed and results are analyzed to evaluate the heuristics.

Acknowledgements

I would like to thank my supervisors, Dr. M. Maheswaran and Dr. M. Toulouse, for your valuable guidances and advices throughout this thesis work.

I would like to express my gratitude to Dr. J. Rueda, for your support and understanding. Without your support, this thesis could take longer to complete.

I want to thank my friends, Vasee, Farag, Indratmo, Tep, XinLiang, and Lei, for your enlightening discussions and great help.

I wish to express my hearty thanks to my husband, Yao Yao, for your love and support.

I also would like to thank my parents and other family memebers who support and encourage me all through.

TRLabs not only supported my research, but also offerred me great opportunities to share ideas with, and to learn from, others.

Thanks also to all members of the Department of Computer Science. You have been a great support and help.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The rapid advancements in computer and networking technologies, coupled with the emergence of new classes of applications with significantly increased resource requirements, tend to rapidly depreciate the value of state-of-art computer and networking equipment. Therefore, it has become increasingly important to maximize the utilization of the resources so that users can recoup their investment. However, maximizing the utilization of resources is a challenging task. The challenge arises from two sides: resources and applications. Firstly, resources are distributed in nature and may be restricted by varying administrative policies, hence with different access privileges. Resources also differ in their computational capabilities. On the other hand, applications too have varying computational, data, and functional requirements. Thus, applications have varying affinities towards different resources. Due to the above reasons, some resources might get overused while others remain idle.

This work presents computation market (CM), an online auctioning system that is designed to help maximize the usage of the already existing computing and networking resources. The CM allows users to "sell" their surplus resources for certain periods of time, and to "buy" various computing and networking resources for temporary use in wide-area networks. More specifically, a user can "sell" his idle resource times, or "buy" resources for use on-demand, simply by using the services provided by the

CM's online auction system. Here, the terms– "buy" and "sell", are adopted from the microeconomic to describe a model that is similar to that of human economy. It does not imply whether real money will be used for buying and selling. Like other projects using computational economic mechanism, we assume that virtual currency will be used to coordinate the buy and the sell activity in the CM.

The CM is a virtual market for wide-area networks. It solves the resource management problem by combining the Internet technology with an economy-based model (e.g., auction mechanism). The CM approach is a result of the following observations:

1. The auction mechanism, originated from game theory and existed as a form of gambling, has proven to be an effective, efficient, flexible and exciting solution for resource allocations. The power of the auction mechanism in solving the resource allocation problem can be demonstrated both from the human and computational societies.

   The human society is essentially a decentralized, dynamic, rapidly changing and geographically distributed system. " A fundamental question in any society is how does one allocate and produce goods and resources? " [21]. Although solving this fundamental problem is very complex due to the size of the human society, the market mechanisms (such as auctions) still exhibit robust solutions. As a result, auctions have been widely used to sell art, collectibles, and a variety of used goods from ancient ages. Auctions are playing an even more important role in modern economy. Auctions offer proven remarkable solutions in: trading stock securities and financial instruments, contracting construction projects, allocating the Personal Communication Services (PCS) spectrum as held by the Federal Communications Commission(FCC)[18], to just list a few.

   Like the human society, the computing systems are also dynamic, rapidly changing, and distributed in nature. The great similarities between the two systems have led many researchers to apply the market mechanisms as a solution to the resource management problem in wide-area computing systems. This category of research, known as computational economy approach, are noted for the capability of achieving flexible, efficient and robust solutions for the resource management problem in wide-area computing systems [2, 7, 24, 16, 4, 5, 11, 13]. The computational economy approach introduces an auction mechanism to coordinate the supply and the consume of computing resources. The computational economy approach are usually formed by resource suppliers and consumers. The supplier agent refers to machines that provide resources such as CPU, memory, disk capacity and I/O bandwidth, while the consumer agent refers to tasks that have some money used to pay for their executions [7]. The suppliers and consumers only interact with those physically close ones, and react based on each individual's own interests without considering the social outcomes. But using a flexible pricing and competitive mechanism, the computational economy approaches appear to fit well with the distributed feature of the wide-area network systems and yield a desirable global resource allocation.

2. The universal connectivity made available by the Internet provides an ideal setting for the deployment of a system such as the CM. Through the Internet, the CM are available to the user community anywhere, anytime. Therefore, the Internet auctions form a robust e-commerce model in today's new economy. This has been demonstrated by the sharp contrast between the success of the Internet auction companies such as eBay, Amazon, Yahoo, and the failure of the thousands of dot.coms. In fact, auctions offer an extremely fun and exciting way of exchanging commodities and play an important part in today's economy. The Internet makes auction experiences more convenient and enjoyable, hence, attracting more and more users.

Despite the popularity of Internet auctions, one would find that none of the auctions offer the similar services as proposed by the CM. This is due to the fact that Internet auctions are still in their early stages. Currently, the commodities traded in these Internet auctions are still narrow in a sense that the majority of the traded commodities are limited to the daily used goods. To take full advantage of the Internet auction mechanisms, the types of commodities traded need to be extended. For example, supporting the exchange of commodities such as time-constraint media spaces, computing and networking resources, etc. The design and implementation of auctions to support these new varieties of commodities is very complex and involves multi-dimensional technologies i.e., a combination of technologies including auction market design, distributed computing, security, and Quality of Service, etc. This will be the focus of the next generation auction design and research.

Although the CM is similar to a wide-area resource management system [Wei98], there are significant differences. A resource management system is concerned with resource allocation and scheduling that involves deciding where a task should be scheduled for execution and when the execution should be started. The resource management system assumes that the virtual "machine" has certain resources at any given time. Typically, the resource attributes of the virtual machine are known before the resource management decisions are made. On the other hand, the CM deals more with offering resource allocation services. The virtual machines are formed from the pool of resources that are made available by the user community. Driven by the needs of the users, multiple virtual machines could be synthesized. Further, unlike the traditional resource management systems, resources in CM are associated with market values that vary based on the demand and supply. Specifically, a resource may be valued more if it is made available when the demand is greater than the supply and may be valued less otherwise.

Once a virtual machine is created by acquiring the resources through the CM, a traditional resource management system may be instantiated on the virtual machine to manage it. The model provided by the CM supports on-demand acquisition of resources for temporary use. Such a facility is crucial for the deployment of next generation Internet services [25].

As discussed earlier, the design of CM demands solutions involving technologies from multi-domains. This work, however, is mainly concerned with the CM auction market design and experiments. The CM design includes the design of auction market and the winner determination algorithm. The auction market design considers the design and the use of incentive mechanisms based on game theory. The auction design theory suggests that the design of incentive mechanisms should be directed by the usage of the auction, and could vary fundamentally, depending on the goods traded on the auction, i.e., the fundamental difference of the incentive mechanisms for oil leases and for ship construction [21]. The literature also indicated that the understanding of the impacts of such incentive mechanisms are far more complicated than they frequently appeared to be. This suggests that a large amount of experimental study and thorough analysis are critical for determining the appropriate rules and strategies for an auction market. To evaluate the design practically and experimentally, two major measures are typically used: efficiency and revenue [21, 18]. The higher efficiency and revenue, the better. Since one of the purposes of the CM is to maximize the usage of computing and networking resources, the utilization of resources should also be considered as an important measure. The experiments conducted by this work, therefore, intensively adopt the above three parameters to assess the CM design.

In an attempt to achieve a greater degree of efficiency and flexibility, the CM adopts a multi-unit combinatorial auction (MUCA) mechanism. The MUCA allows the users to bid on combinations of resources, which reflects how the computing and networking resources are most likely being used. However, solving the MUCA problem (i.e., determining the winners) is NP-hard [22, 9, 10, 26]. This work tackles the problem using heuristics. Simulations are conducted and results are analyzed to evaluate the heuristics.

Chapter 2 introduces the related work in the literature. Chapter 3 presents the CM's architecture as well as the auction market design, i.e., the rules, policies, and strategies. Chapter 4 formally defines the MUCA in the CM, thoroughly discusses the heuristics designed, and analytically compare the experimental results obtained. Chapter 5 describes two different approaches to achieve the resource allocation for the MUCA problem and presents the simulation results with emphasis on the performance of utilization under various configurations. It also illustrates the bid generation methodologies for the simulation performed. Chapter 6 gives a conclusion and the future work.

Chapter 2

Related Work

The economic models have been successfully used in resource management of human societies. To testify the power and flexibility of economic models for solving resource allocation problems in a wide-area network, a variety of projects have been designed and implemented, a number of experiments are conducted, and the results of the experiments are analyzed. This chapter reviews auction theory, projects using auctioning approach and researches in the area of Internet auction and combinatorial auction design.

2.1 Auctions

Auctions are very widely studied mechanism in the economic and game theory area. The study of auctions can be dated back to the 1960's. As auctions offer an efficient and distributed mechanism for solving the task and resource allocation problem in multi-agent systems, different types of auction protocols and various agents' strategies in auctions have been researched in auction theory [San96]. This section briefly discusses different types of auctions documented in auction theory. An auction could be ascending or descending, open or sealed, depending on the rules that bidders should follow when bidding and the knowledge of other's bidding prices. Table 2.1

Table 2.1: Auction Types and Attributes

auction types	ascending	descending	open-outcry	sealed-bid
English auction	√		√	
Dutch		√		√
Vickery auction				√
First-price sealed auction				√

shows some of the well-known auction types, including: English (first-price open out-cry) auction, first-price sealed-bid auction, Dutch (descending) auction, and Vickery (second-price sealed-bid) auction. Different auction types are designed with distinct features to serve different purposes, and therefore, are commonly used to sell different types of goods. For example, the English auction is mostly used in selling art and other collectibles.

Conceptually, the English (first-price open out-cry) auction is an ascending auction. The seller submits its item for bid with a minimum required price called reserve price. The first bid's offer price should be higher than the reserve price. Each bidder bids iteratively for an item, it is required that each subsequent bid should be higher than the current highest price. The auction ends (called clears) with the highest bidder winning the auctioned item at the price it bids.

In the first-price sealed-bid auction, each bidder places its bid based on its own bidding strategy without knowing what others bid. Unlike the English auction, the first-price sealed-bid auction does not have iterative bidding process, each bidder only needs to bid once. The one that bids with the highest price wins at the price it offers.

In Dutch (descending) auction, the seller keeps on descending its sell price until one bidder gives the sell price. Similar to the first-price sealed-bid auction, each bidder only needs to bid once, and they don't know the others' bids. The highest bid wins at the price it submits. The Dutch auction differentiates from the English auction in that the Dutch auction is descending.

In the Vickery (second-price sealed-bid) auction, bidders bid for an item following exactly the same rules as in the first-price sealed-bid auction. The only difference between them is the winning price. The highest bidder wins at the price of the second highest bid in the Vickery auction. The strategy in Vickery auction is to bid one's true valuation [San96]. Vickery auctions are not widely adopted by auctions among humans due to the fact that the Vickrey auction can't efficiently avoid the cheating activity of the auctioneer. For example, an auctioneer can overstate the second highest bid price without being known, which causes a winner to pay more than he should. However, Vickrey auctions are commonly used in computational multi-agent systems. For example, the first computational multi-agent system, Spawn, used the Vickery auction.

Auctions can also be classified by how bidders figure out the value of the auctioned items in an auction. Based on this criteria, there exists three types of auctions: private value auctions, common value auctions and correlated value auctions. Private value auctions are auctions where the value of a good is entirely depended on bidder's individual preferences, which happens in a situation where the winner won't resell the good. On the contrary, in common value auctions, a bidder's valuation of a good totally relies on other bidders' values of it. The private value auctions and common value auctions are two extremely different cases. Unlike the above two, the correlated value auctions are medium ones where a bidder's value of a good depends on both its own preferences and other's values, which is the case for most auctions.

It is widely observed and accepted that English auctions can lead to the greatest revenue with optimal resource allocation. The open nature of the English auction provides three advantages:

- Avoid the counter-speculating activities among the bidders.
- Guarantee no lying auctioneer.
- Bidders gain excellent information about the value of target, which encourages them to bid aggressively. Therefore, leading to an efficient auction.

English auctions are especially suitable for Internet auctions and are adopted by most of the online sites. CM also intends to use English auction, but designing with some special bidding rules.

2.2 Internet Auctions

Internet and the emergence of electronic commerce infuse new life into auctions, leading to the popularity of Internet auctions. Auctions run by eBay, Amazon, and Yahoo are some of the most famous Internet auctions. Making use of agents, Internet auctions typically run for several days, lasting much longer than other forms of auctions. Internet auctions typically attract both a large number of bidders and sellers because of the easy access feature of the Internet. Most of the Internet auction sites auction a large number of categories of goods, from antiques and arts, books, dolls and bears, to real estates, to just list a few. But some Internet auctions are specialized in one particular category, for example cars, or computers. Given the large number of participants in the Internet auctions, one can imagine that the number of auctioned items could be enormorous, too. Thus, determining the resource allocation for Internet auctions typically involves more efforts on complex game-theoretic strategy and computation [22]. Although Internet auctions can be designed using any type of auction protocols, most Internet auctions are based on English auction protocols, but each with some different rules.

Different auctions are designed to serve different applications. It is absolutely true that an auctioneer wants to sell his item (items) as high as possible while each bidder wants to win at the lowest possible price(s). It seems that a good auction is the one with results satisfying both the auctioneer and the bidders.

Therefore, which type of auction to choose depends on the circumstances that will be used. In general, an auction's suitability of serving its purpose can be decided by the following measures: How efficient is the auction? Do the bidders waste effort on counter-speculating what others bid? Does the auction mechanism efficiently prevent lying activities of the auctioneers? Is the auction flexible and robust? In [26], the reasons for these measures and the problems occurred in each type of auction are very well discussed. With the emergence of Internet auctions, I would like to add two more measures: how easy is the auction for users to use? Are users well-informed and receiving instant feedback? These are the key issues that must be taken into consideration in designing an auction, which will be discussed and demonstrated later in the auction protocol design of CM.

Currently, all Internet auctions apply single-item auction designs where bidders can only bid on one type of goods. Because most bidders in the CM are supposed to be interested in acquiring more than one item of computing resources, the single-item auction is not appropriate for the CM. We are thus seeking the more flexible and efficient auction type, the combinatorial auction, where bidding for a bundle of resources is possible. The next section discusses the related work in combinatorial auctions.

2.3 Computation Economy Projects

Spawn [4] is an implementation of a distributed computational economy. Similar to the CM approach, the Spawn economy uses currency, funds (money), price and auction as the mechanisms of exchange of resource. It is organized as a market economy formed by interacting buyers and sellers. However, the CM approach is distinct from Spawn in four primary aspects. First, Spawn is not an Internet-based system. Secondly, the two systems apply different types of auctions. The Spawn uses Vickery (second-price sealed-price) auction, while computation market uses English (first-price open-cry) auction. Thirdly, the deployment of the auction mechanism is different. In Spawn, each machine maintains an auction mechanism, while the CM uses a domain-based auction mechanism with each local market having its own auction server. Here, I argue that the auction mechanism of Spawn would unavoidably decrease the system's overall scalability. As the system increases in size, the bidding overhead between the tasks and the machines would greatly degrade the overall performance of the system. Also, to control the communication overhead, Spawn constrains each task such that it can only communicate with the "nearest-neighbor", this prevents a task from finding and exploiting the computational resources located in distant regions. Nevertheless, as an early application of a microeconomic approach to resource management [4], Spawn highlights some of the economic concepts and their applicability in dealing with the complexity of resource management in large network computing systems.

Mariposa [15, 16], developed at the University of California at Berkeley, implemented an economic paradigm to demonstrate the power of economic models for managing query execution and storage management in a wide-area distributed database system. In Mariposa, each query is submitted with a budget that is used to pay for solving the query, and each processing site attempts to maximize its revenue by buying and selling fragments and by processing queries. To decide where to run a query, a distributed advertising service is designed to provide information for finding sites that might want to bid on a query or portions of a query. During the bidding process, two protocols could be involved, one is a two-phase based expensive bid protocol, and the other one is a purchase order protocol. In the first phase of expensive bid protocol, requests for bids are sent out to bidding sites. In the second phase, both the winners and losers of the bidding are notified. The purchase order protocol is much faster than the expensive bid protocol, as fewer messages are required. It directly sends the query (or subquery) to the processing site that would be most likely to win the bidding process. The site receives the query and processes it and then returns the answer with a service charge. Or if the site refuses the subquery, it can either return it back or pass it on to a third processing site.

My approach borrows some ideas from Mariposa. I also apply different protocols in bid processing. Either two-phase iterative or single-phase non-iterative protocol can be used in an effort to decrease the communication overhead among the domains.

Mark [12, 13, 14], an ongoing project at University of Michigan, uses market-based adaptive architectures for information survivability. Mark shares many important features with the CM. Mark also implements an on-line market architecture, which is built on the top of a general purpose Internet auction server: AuctionBot. Similar to the CM, agents are also served as the fundamental entities to trade tasks and resources at prices determined by an auction protocol. The interactions of the consumer and producer agents build supply chains. Each agent makes decisions based on its personal knowledge base. To protect the system from attack, a two-level market model is constructed. Mark also introduces negotiation protocols to achieve Internet resource reservation. However, the CM differentiates the Mark in the purpose of the project: the CM target towards providing a service framework for government, business, organizations and individuals; the Mark focus on information survivability.

The Java Market project [30], developed at the Johns Hopkins University, is a working system that aims to transform the Internet into a meta-computing system. It combines the recently developed web technology and the Java Applet architecture. Like our system, the Java Market allows the producers and consumers to access the Java Market web pages anywhere on the Internet by simply running secure Java Applets on the web browser. The major difference between their model and ours is that their model lacks a hierarchical structure. This significantly limits the scalability of their system. Also, their model is not developed to provide an efficient solution for aggregating resources that are located in the same network neighborhood for parallel computation.

2.4 Combinatorial Auctions Design and Algorithms

Combinatorial auctions allow bidding on bundles of goods and allocate in an "all if any" manner. Combinatorial auctions can be classified into two categories in general. Combinatorial auctions typically refer to those in which all auctioned items are different. Multi-unit combinatorial auctions are those with identical items for some of the goods. Determining an optimal set of bids is NP-hard in a combinatorial auction, and it is even harder for a multi-unit combinatorial auction. Due to the computational complexity of combinatorial auctions, few auctions actually applied combinatorial auction theoretically or practically, and none apply the multi-item combinatorial auction design [27, 28, 20]. Researchers working with combinatorial auctions typically attempt to solve this problem by three approaches: by heuristics [10, 22, 27], by defining special bidding procedures of market design [28], or by enforcing some special restrictions on the bidding mechanism to make the NP-complete problem tractable. This section discusses related work in using the above approaches to combinatorial auction problems.

The Federal Communications Commission (FCC) spectrum auctions and Adaptive User Selection Mechanism (AUSM)) are examples in which special bidding restrictions are enforced to make the combinatorial auction problem tractable. The FCC spectrum auctions are used to assign the electromagnetic spectrum for Personal Communication Services (family of mobile communications services that allow people access the Public Switched Telephone Network). Although the FCC spectrum auction and the AUSM auction share some features, their designs actually differ dramatically.

The FCC spectrum auctions are essentially simultaneous multiple round auctions with eligibility requirements, namely "a use-it-or-lose it" rule. The bidders can't place combinatorial bids. In fact, only single-item bids that improve the leading bids by at least minimum increment requirement are accepted. Bid withdrawals are allowed with penalty. The FCC spectrum auctions consist of three stages. The stage proceeds to a higher level when some particular situation occurs. For example, the transition from stage 1 to stage 2 happens when there are no bids on more than T1 percent of the population base for three consecutive rounds; from stage 2 to stage 3 occurs when there are bids on no more than T2 percent of the population base. Using the eligibility based stopping rules, the FCC spectrum auctions close with the winners paying what they bid. Since the bidders can't place combinatorial bids, the FCC spectrum auctions are faced with exposure problems. The disadvantages are: the auction may potentially take a very long time to run, thus not very efficient. Because the auction forbids placing combinatorial bids, bidders suffer from less profit. [6]

The AUSM is the only auction that permits all possible combinatorial bids. The bidders are free to submit or withdraw bids at any time before auction closed. The auction closes according to some pre-defined rules. One distinct feature of the AUSM is, the bidders can use a standby queue to announce their willingness to offer a particular price for a particular combination. The standby queue is used as a mechanism to avoid the "threshold problem". However, some researchers consider the standby queue a too complex mechanism for bidders, despite its capability of combating the threshold problem.

The auction stopping rule design is worth some researching effort. Traditionally, some auctions use hard closing time, some auctions use conditional stop rule, and some auctions even use random stop rule. It is observed that different stop rule designs can lead to totally different behavior of the bidders, which impacts on the benefit of both the auctioneer and bidders to some extent. A good demonstration of this is the totally different behavior from bidders of Internet auction sites: eBay and Amazon. eBay closes its auctions at a pre-defined fixed time, while Amazon continues its auction until no new bids are submitted within ten minutes. The different stopping rule leads to significantly more late bidding (the so called "last minute bid") on eBay than on Amazon [3]. The experiment from [3] also shows that the experienced bidders on eBay contribute more late bids than do less experienced bidders, while the case on Amazon goes to the opposite.

Tuomas Sandholm presents a search algorithm for optimal winner determination [27]. The search algorithm consists of four preprocessing steps and the main search. A special bid tree is used in the main search. The bid tree is a binary tree with the bids inserted up front as the leaves. The algorithm also uses an iterative deepening A* search strategy to fast the main search. Similar to their algorithm, CM also uses heuristics to tackle the winner determination problem in combinatorial auctions. What makes our bid tree different from that of [27] is, the leaves in our bid tree are linked list of bids that consist of the same combination of items. The linked-list is organized based on the rate of each bid with the higher rate bid comes before the lower rate bid.

Recently, Branch-and-bound search technique has attracted a lot of research efforts to solve Combinatorial Auction problem [10, 22, 27]. The branch-and-bound search consists of two steps: a depth-first search, and a backtrack search to cover the whole search space. Each time we add a bid that has no conflict with the current partial solution until a full allocation is constructed, then we backtrack. The backtracking depends on a good upper bound for the value of the optimal solution corresponding to the still unallocated items. If the sum of the upper bound and the values of the bids of our partial solution is not larger than the value of the best solution found so far, we may backtrack immediately.

An algorithm called CAMUS (Combinatorial Auction Multi-Unit Search) is presented in [10]. CAMUS introduces a branch-and-bound technique and an estimate function o() that will always provide an upper bound on the actual revenue. They also show that the CAMUS is guaranteed to find optimal allocations. Similar to our algorithm, their backtrack also removes the most recently added bid from the partial allocation. But our algorithm lacks the estimate function. Another interesting similarity between their algorithm with ours is, we both consider the order of the goods (that is, which good corresponds to the first bin, which corresponds to the second, etc) and the order of the bids within bins impact the result of the algorithm significantly.

Another research [22] also uses branch-and-bound technique to solve the Multi-Unit Combinatorial Auction problem. Their experiment results shows that the branch-and-bound techniques require both a way to bound from above the value of the best allocation and a good criterion to decide which bids are to be tried first. They suggest making use of average price per unit or average price per unit related criteria in a branch-and-bound algorithm, which is quite similar to the use of rate to rank the bids in our bidding selection algorithm.

A significant difference between our algorithm and algorithm discussed in [10, 22] is, we require bidders to indicate the unit prices for each kind of good demanded instead of a total price for the whole bid, which are the case in [10] and [22]. We consider this rule is more suitable for an auction for computing and networking resources. This rule ensures that all the winners pay the same price for the same kind of resources in the same auction no matter how much their offers are. This rule is fair for all the bidders and encourages bidders to offer aggressive prices without regretting.

Another initiative is the study of combinatorial auctions for supply chain formation

[29]. A supply chain refers to complex business negotiations involving interrelated exchange relationships among multiple levels of production. A particular combinatorial protocol consists of a one-shot auction and a strategic bidding policy. They perform experiments to analyze the importance of strategic bidding regarding the efficiency and producer surplus. The results show that producers can sometimes gain significantly by bidding strategically. The robustness of the combinatorial protocol is also studied.

PAUSE (Progressive Adaptive User Selection Environment), presented in [28], designs a two-stage procedure to solve the general combinatorial auction problem. The first step is a simultaneous, multiple-round auction. In stage 1, bidders are required to bid on individual goods with progressive eligibility requirements and an improvement margin requirement. Stage 2 is also a simultaneous, multiple-round auction. The bidders can bid on composite goods with progressive eligibility requirements and an improvement margin requirement. By designing the market with two progressive adaptive stages, PAUSE presents another way to control the complexity of combinatorial auction.

Chapter 3

Computation Market Design

The design of the computation market involves two different domains: the system infrastructure design and the auction market design. The system infrastructure design provides a platform and a mechanism on which the auction activities can be conducted, while the auction market design specifies the set of rules followed by the auctioneers and the bidders. The system infrastructure design involves technologies in the networking and e-commerce areas, requiring an architecture that scales well in a wide-area networking environment. The auction market design is basically an economic topic, entailing designing an economic model (e.g., online auctioning mechanism), a set of protocols, rules, policies, and strategies, by using game theory. Although from distinctly different areas, the system infrastructure design and auction market are actually related and tightly coupled. We design the market by taking the current network and technology capabilities into consideration, and then design the system architecture reflecting the bidding activities that are decided by the market design.

3.1 Assumptions and Objectives

Due to the lack of an existing model for CM, several assumptions are made to guide the CM design.

- As an Internet auction system, we would expect both a large number of sellers and buyers.
- Examining how the resources are used and why the users desire the services of CM, the second assumption is that most of the users of the CM prefer bundles of resources rather than single units.
- In addition, there will be a large amount of distributed resources available simultaneously. The availability of the resources is typically constrained within certain period of time (i.e., their idle times).

The above assumptions lead to the establishment of a set of design objectives for the CM:

- Easy to use. This is a prerequisite for a success Internet auction. The users should be able to find their bidding targets (i.e., the resources to be bid on) and to indicate their bidding prices easily.
- Low communication overhead are involved. The large amount of traffic generated by the bidding activities coupled with transferring tasks to be executed remotely could decrease the performance dramatically. This issue should be addressed both at the design of system architecture level and at the design of auction market level.
- Be able to aggregate resources (especially those geographically close resources) to form clusters, and allow for bundle bidding (i.e., all or none allocation). Applications such as parallel computing and multimedia demands a huge amount of resources, which is exactly the value of the services of the CM are highlighted. Therefore, the resource aggregation and bundle bidding feature is crucial.

- Be efficient to determine winners. The large number and the time-constrained feature of resources, combining with the large number of bids, make the winner determination extremely challenging. A long winner determination process could affect the feasibility of the solution.

- Provide a good balance among the efficiency, revenue maximization, and resource utilization. The CM is designed for maximizing the utilization of resources. Therefore, this adds extra requirement for auction market design and winner determination, although most auction designs focus mainly on efficiency and revenue.

3.2 System Architecture

Computation market divides the wide-area network into regions called local markets. Hence, the overall architecture of computation market (as shown in Figure 3.1) is composed of two levels, which are presented as an interconnection of local markets and a global market that is built on top of all the local markets. The first level of computation market focuses on intra-market flows and the second level handles the inter-market resource flows. In the computation market, a local market consists of an auction server (AS), local brokers (LBs), supplier agents (SAs), and consumer agents (CAs). The supplier agents represent machines that provide resources such as CPU, memory, disk capacity and I/O bandwidth, while the consumer agents represent clients that are willing to pay in currency for those resources. An auction server provides a virtual market where the local supplier agents and consumer agents can trade the computation resources. The currency is used to encapsulate the resource rights and price to reflect the demand and supply. The interaction of the supplier agents and consumer agents causes the fluctuation of the prices, which in turn results in a balance of the supply and demand of the computation resources.

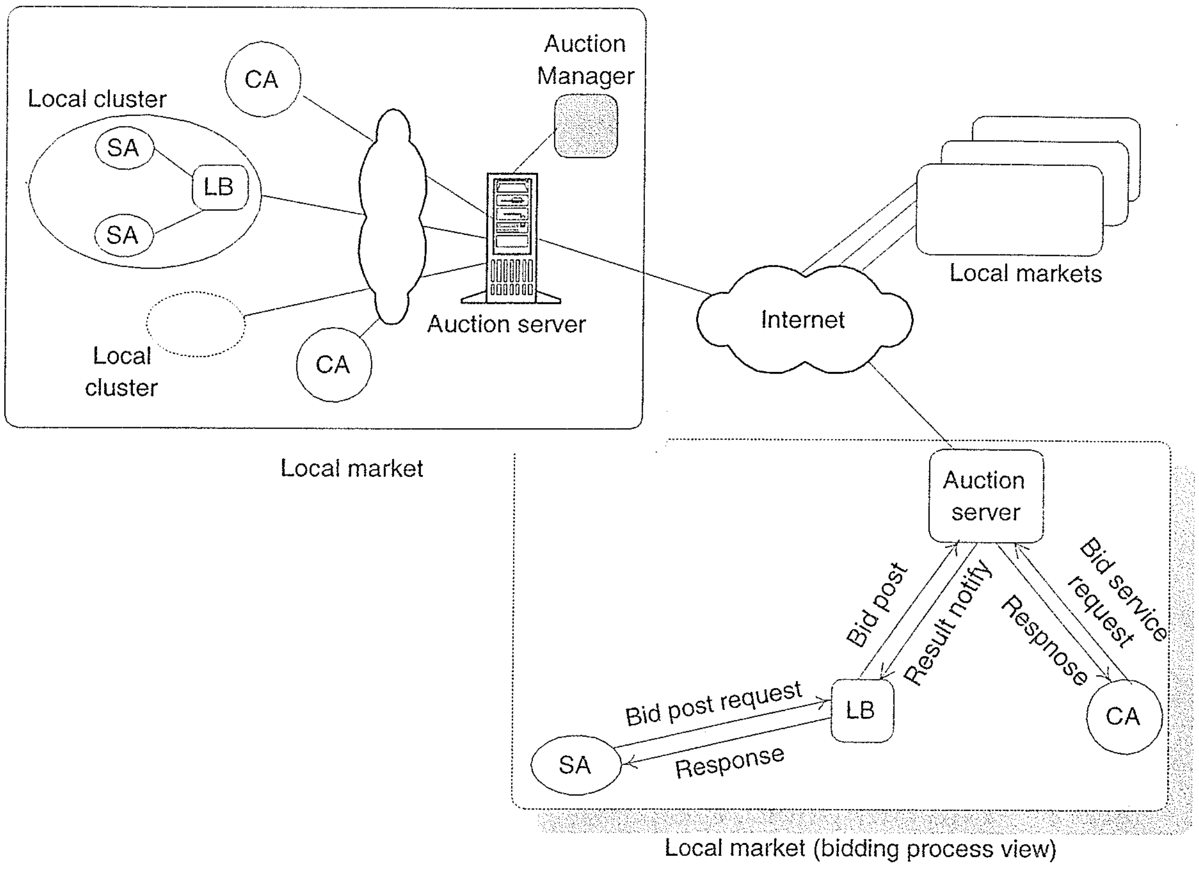


Figure 3.1: Two-level Architecture of Computation Market

3.2.1 The Auction Server(AS)

The auction server is a key component in computation market. An auction server services its local clients by offering three major functions: (a) providing a virtual market; (b) importing resources from, or exporting resources to, the global market; and (c) selecting of the winning bids. An auction server does not provide its services for free. In fact, it makes money, which comes from three sources: (a) transaction fees charged on its local SAs and CAs; (b) profit from selling cheap remote resources in its local market; and (c) profit from selling surplus local resources in the global market. Money oriented, an auction server always tries to maximize the profits from the services it offers.

The auction server relies on its three managers to fulfill its functionalities:

- The **auction manager** determines the winners of the auction according to the bidding selection heuristics, which will be discussed in the next chapter.
- The **market manager** acts as both a resource supplier and a resource consumer in the global market. If the demand is greater than the supply in the local market, the market manager tries to export resource surplus by posting bid(s) to sell local resources in the global market; otherwise, it tries to import resources by bidding on remote resources in the global market.
- The **schedule manager** ensures the auction opens and clears happening at the appropriate time sequence. The schedule manager also triggers sending emails to provide feedback to the sellers and the buyers.

3.2.2 The Supplier Agent(SA)

The supplier agent is a seller representing a machine with idle resources. The supplier agent is responsible for handling the submission of resources. It sends bid post request for the resource it attempts to sell, providing the following attributes: resource attributes, available start time, end time, and a reserved price (the lowest acceptable price). Although the auction server is the real market that holds the auction, the bid post request is actually sent to a special local broker rather than to the auction server.

3.2.3 The Consumer Agent (CA)

The consumer agent is responsible for handling consumption of resources. A consumer agent is a buyer bidding iteratively by taking the performance and the desirability of the resources into consideration. The consumer agent sends its bid service request to the auction server and receives bid services, such as viewing the current bids information, placing, modifying or withdrawing a bid.

3.2.4 The Local Broker (LB)

The local broker represents a set of geographically close supplier agents. The local broker has two responsibilities. One is to post bids on behalf of the supplier agents it represents. The other one is aggregation, that is, it combines the collection of bids with the common available time by examining all the post bid requests, and then post bids to sell a cluster (or several clusters). The local broker is used here as a mechanism to perform the aggregation function in such a way so that the opportunity of selling resources from the supplier agents can be optimized.

The local broker allows independently managed but co-located resources to be grouped to form clusters. The aggregation of resources in this manner provides an efficient solution to the co-allocation problem [8], and could be a cost-effective approach to parallel computing. The aggregation performed by the local brokers does not enforce the resources to be allocated as a block, i.e., the AS can "unbundle" the resources when it sees fit, thus offering more flexibility.

3.2.5 The Bidding Protocols in a Local Market

To enhance the scalability of the system, the computational market adopts a two-level architecture. The first level of the architecture handles the resource management within a domain. This level uses a two-phase iterative bidding protocol, which consists of 3 steps:

1. Initially, a supplier agent sends a bid post request to its local broker, which in turn aggregates and posts the bid to the auction server.
2. The consumer agents can bid on either an item or combination of items that

are posted at the auction server with a price (or prices) higher than the current price(s).

3. The highest bidding price(s) is reflected at the auction server. The consumer agents can query the current highest price(s) on the auction server.

The step 2, 3 may be iterated if the consumer agents are willing to offer a higher price. The highest bidder(s) wins at the close of the bidding. The auction server notifies the participating agents of the result at the end of the auction.

The two-phase iterative bidding protocol can be communication intensive depending on the number of iteration performed until closing the bids. Thus, the CM only allows local consumer agents to bid on resources at the local auction server. The inter-market resource flows are managed by another auction called the global auction.

3.3 Market Design

The auction market design involves designing an economic model (e.g., online auctioning mechanism) and a set of protocols, e.g., the bidding rules and policies that are followed by the auction users. As an Internet auction system with a large number of users, one of the most important features is to provide efficient, flexible, and easy to use means for users to post and place bids. Therefore, one of the design goals of computation market is to allow users to acquire the computing and networking resources efficiently and flexibly. The computation market achieves this design goal through several approaches: creating a special bid type, enforcing resource classification, applying two-level bidding protocol, and using combinatorial auction mechanism. The computation market design also reflects the users needs and fits the attributes of the resources. This section discusses how the computation market is designed to meet users' need.

3.3.1 Any-bid

Any bid is a special bid type that is designed to offer a flexible and convenient way for placing a bid. The current existing auction systems only allow a bidder to specify his bidding targets (i.e, the resources that a bidder is interested in) explicitly. Given the large number of buyers and sellers in an Internet auction system, locating the suitable bidding targets can be a very complicated task. To make placing bids easier, computation market designs a special type of bid called "any-bid", which basically does not constrain a buyer to bid on any particular auctioning object. Instead, object(s) requested by a any-bid are abstract in a sense that they could be any objects given that the objects satisfying the computing and budget constraints requested by a bidder. A user places a bid by simply indicating the computing and budget requirements, and the system will be responsible for allocating the resources that match the user's requirements, if there are any. Using the any-bid, a user can place a bid easily and flexibly without suffering from the tedious bidding process. In addition, using the any-bid provides a bidder with more chances to obtain his bidding targets and a seller with more chances to sell.

3.3.2 Resource Classification

Resources, including computing and networking resources, are actual commodities in the computation market. Generally, in a market, the value of a commodity is decided by its usage and social desirability. In particular, the usages of computing and networking resources are decided by their computing capacities and transmitting capacities respectively. In CM, the value of a resource, therefore, is decided by its social desirability and its capacity. Attributes, such as CPU speed, memory size, disk capacity, and IO bandwidth, contribute most to the computing capacity. Another key attribute that affects the usage of a computing resource is the platform, such as operating system and compiler support. For example, a user wants to buy resources to run applications under Windows NT platform, obviously, it is no use if the resources are from any other platforms. For networking resources, bandwidth

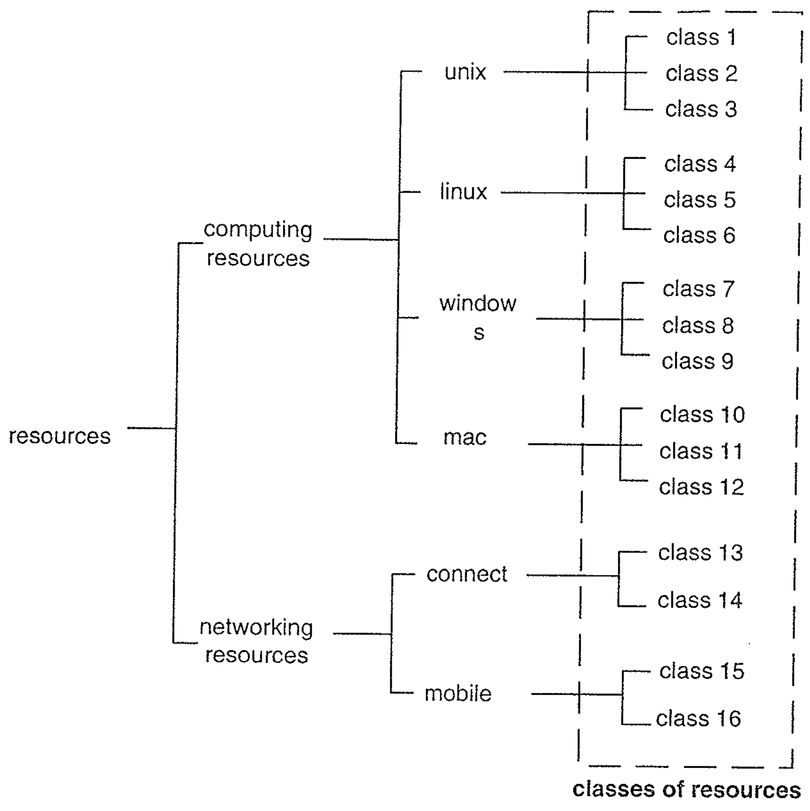


Figure 3.2: Resource Classification Based on Attributes

and speed are major attributes that affect the transmission capacity. Consider, if the users are allowed to bid randomly for any combinations of attributes, the number of possible combinations provided by the users will be huge. Allocating the resources that matches the specified attributes will be difficult and inefficient practically. Also this would make the determination of the winners in an auction too complex to handle. Thus, to simplify the winner determination problem and promote the efficiency of CM, it's desirable to have some methods to control how a bidder indicates his resource requirements. The solution here is to define resources with roughly equal capacity (can be either computing capacity or networking capacity) into one class, so the set of resources consist of several classes of resources. In this way, users may specify the attributes of the desired resource using attribute-value pairs [23].

Figure 3.2 shows the hierarchies of resources, classified by resource attributes. According to their different usages, the set of resources to be auctioned consist of two categories: computing resources and networking resources. Computing resources are then divided into four groups, depending on resource platforms. Under each platform, the resources can further be classified into 3 different classes based on their computing capacities. The computing capacity decreases from class 1 to class 3. Based on the connection methods, networking resources are divided into connect and mobile resources, which can further be classified into 2 classes due to their transmission capacities. The transmission capacity of class 1 resources is less than that of class 2 resources.

3.3.3 Auctions

Two different auction types are used in the CM: an open-outcry/ascending auction (a variety of the English auction) for its local markets, and a sealed/second-price auction (Vickery auction) for the global market. As described earlier, the bidding protocol in an open-outcry/ascending auction is iterative in nature, hence can be communication intensive. To achieve a better scalability, the global market avoids the iterative bidding protocol by using a sealed auction, where each bidder only submits his bid once.

Although conceptionally an English auction, the local auction is actually designed with some customized auction rules and policies. The customized rules and policies are designed mainly for three purposes:

1. Fit better with the properties of goods auctioned in the CM;

2. Provide convenient methods for a user to indicate his bidding requirements;

3. Reduce the computation complexity for the winner determination problem.

3.3.4 Auction Rules for a Local Market

[20] suggests that the auction rules can be described as three axes: rules for receiving bids, rules for auction clear, and rules for information revelation. Rules for receiving bids decides constraints that a bid should satisfy in order to be accepted as valid. *Auction Clear* is a terminology referring to the close of an auction. Rules for auction clear, also known as stopping rule, therefore, determines under which conditions shall an auction complete. Such conditions for auction clear include two types in general: a fixed end time and the bidding behavior of the bidders. *Quote*, a concept related to the information revelation, is used to describe the summary information announced by the auction. Quote usually provides insightful information regarding the potential outcomes of the auction. The information revelation strategy differs among auctions of different types, and even among auctions of the same type. Normally, English auctions reveal the quote information in a timely manner, and most sealed-bid auctions disclose the quote information after the auction clears. This section presents the auction rules for the CM from the above perspectives.

A local market in the CM adopts an open-outcry/ascending multi-unit combinatorial auction mechanism. This demands terminologies that a seller or a buyer used are capable of expressing multi-unit requirements.

Let n be the number of resource classes in the CM. Rules for sellers and buyers are defined as follows:

Rules for sellers

A seller i shall use the following terms to indicate his willingness of auctioning off a bundle of resources:

$$\textit{selling request } (i) = (C_i, P_i), \tag{3.1}$$

Where $C_i = (c_{1i}, c_{2i}, ..., c_{ni})$, and $P_i = (p_{1i}, p_{2j}, ..., p_{ni})$ are auction items, and reserve prices for resource class [1,2,.., n] respectively.

Rules for buyers

To use an English auction in a multi-unit combinatorial auction (MUCA), several issues have to be addressed. As stated previously, an English auction requires a new bid to be superior to the previous bid by at least a minimum increment ϵ. The reason for the minimum increment rule is based on the fact that it ensures certain speed for the auction [28], and thus enhances efficiency of the auction. As English auction is originally designed for single unit auctions, enforcing increment rule in a MUCA could be confusing, although it is trivial for a single unit auction. To introduce the price increment rule into a MUCA, several issues have to be addressed first:

1. Shall a bidder offer a combined price, or state unit prices for each single type of resource demanded?
   Our decision is to require unit prices for each type of resource instead of a total price for all resources. The decison is based on two main reasons: Given the large number of users for the Internet auction, determing an appropriate price offer is a complex task for bidders. Using unit price makes bidding easy and straight-forward: a bidder simply increases the current highest unit prices without the need of complicated mathematics computations. Secondly, a unit price-based rule facilitates simplifying the minimum price increment rule, a requirement for an English auction. In the end, this also makes the winner determination problem more manageable.

2. Given the unit price based rule, there still exists a few options for enforcing the minimum price rule. To illustrate the scenario, assume $P_h = (p_{h1}, p_{h2}, ..., p_{hn})$ is the current highest bidding price vector, and $P_{new} = (p_{new_1}, p_{new_2}, ..., p_{new_n})$ is a new bid's bidding price vector, for resources [1, 2, ..., n], respectively. The options existed for enforcing the minimum bidding price can be described as follows:

   - **Beat all** – Beat the current highest for all the asking resource types. That is, the bidding price $P_{new_j} = \{p_{new_j} | p_{new_j} > p_{h_j}, \forall p_{new_j} > 0\}$, where

$j \in [1..n]$.

- **Beat at least one** – Beat at least one of the current highest and no less than the other current highest. Formally, $P_{new_j} = \{p_{new_j} | \exists p_{new_j} > p_{h_j} \wedge \forall p_{new_j} \geq p_{h_j}, \forall p_{new_j} > 0\}$, where $j \in [1..n]$.
- **Beat overall** requires that $\sum p_{new_j} > \sum p_{h_j}$, where $j \in [1..n]$.

Obviously, the *beat all, beat at least one,* and *beat overall* interprets the minimum price requirement rule in three levels, with the *beat all* presenting the strictest requirement, and the *beat overall* the least strict requirement. The *beat all* is not suitable for the CM as the large number of bidders, combined with a very strict rule, could cause the price to increase too fast and therefore prevent the bidders continuing to offer higher prices potentially. On the other hand, the *beat overall* is not straight-forward and is difficult to use. In addition, the *beat overall* rule may lead to a much more complex winner determination problem.

The CM considers the *beat at least one* as the most appropriate and easy to use rule. The CM enforces the *beat at least one* rule among bids that ask for the same combination of resources. Here, "bidding for the same combination of resources" refers to bids that ask the identical set of resource types, while the number of bid items for each resource type asked by these bids could differ. Let G be such bids, it follows that:

$$G = \{C \subseteq B | \forall i \in C \wedge \forall j \in C, w_{ki} > 0 \wedge w_{kj} > 0\} \tag{3.2}$$

where B is the set of bids in the auction, k$\in [1..n]$.

In real world auctions, it's frequently observed that certain combinations occur more often than others [9]. Therefore, it's reasonable and more meaningful to enforce the *beat at least one* rule among bids with the same combination of resources, as they can conceptually be considered as bidding for the same resource as in a single unit auction.

Therefore, a bidder j submits a bid offer using the following form:

$$buy\ offer\ (j) = (i, W_j, P_j), \tag{3.3}$$

Where j is the seller ID, $W_j = (w_{1j}, w_{2j}, ..., w_{nj})$, and $P_j = (p_{1j}, p_{2j}, ..., p_{nj})$ are bid items (also known as bidding sizes), and offer prices for resources of class 1, 2, ..., n respectively. Notice that the bidding prices satisfy the *beat at least one* rule among bids with the same combination of resources.

Winning prices rule

All the winners pay the same price for each resource type. The price is the minimum unit price of the type of resource among all auction winners. This rule is different from most other multi-unit combinatorial auctions, where each bidder pays at his own offer prices. We consider "all buyers pay the same price for the same resource" as a more appropriate rule for the CM. This rule encourages bidders to offer higher prices and is fair for all the winners. The buyers do not need to regret having bought something too expensive, hence will be willing to offer better prices with no hesitation in the future. Let B be all bids in the auction, Let Q be the set of winning bids, $P_{final} = (p_{1f}, p_{2f}, ..., p_{nf})$ be the final selling prices for resources of class 1, 2, ..., n, correspondingly. The winning prices rule follows:

$$Q = \{S \subseteq B | \forall j \in S, p_{kj} \geq p_{kf}, k \in [1..n]\} \tag{3.4}$$

Stopping rule

The CM adopts a fixed stop rule: the auction clears at a pre-defined time that is indicated by the seller.

Feedback rating policy

The bidding rules in conjunction with the credit policies provides useful feedback to the bidders. This also encourages healthier bidding behaviors and market.

The Internet auction provides a virtual market that allows auctions available to the user community anywhere, anytime. The trades are conducted both domestically and internationally. This high availability feature of the Internet auction also brings a potential problem – the dis-honest behavior of the sellers. A bad seller can untruthfully claim the quality of their resources and hurt the interests of the consumers. To overcome this problem, almost all Internet autions use a feedback rating system. For instance, both amazon.com and eBay use "feedback rating" policy, which includes a tally of the comments other users have made about the seller, and the detail comments.

The CM also applies the similar feedback rating policy. For each transaction, a seller's feedback rating will:

- increase 1 point (+1) for each positive comment received;
- not change (0) for each neutral comment received;
- decrease 1 point (-1) for each negative comment received;

A seller's rating score is defined as:

$$rating\ score = \frac{total\ rating\ points\ recieved}{the\ total\ number\ of\ comments\ received} \tag{3.5}$$

Bid withdrawal rule

Following most of other auction designs, the CM allows bid withdrawal, but penalties those withdrawal bidders. The bidder should pay the difference between his bidding price and the final selling price. Let a bidder's buy offer as follows: $W_j = (w_{1j}, w_{2j}, ..., w_{nj})$, and $P_j = (p_{1j}, p_{2j}, ..., p_{nj})$ are bid items, and prices for resources of class (1, 2, ..., n), respectively. Let $P_{final} = (p_{1f}, p_{2f}, ..., p_{nf})$ be the final selling prices. The penalty paid by the bidder equals to:

$$penalty\ (bidder\ j) = \sum_{i \in [1..n]} (p_{ij} - p_{if}) w_{ij} \tag{3.6}$$

3.3.5 Global Bidding Strategies

To efficiently direct inter-market resource flows, market managers should respond strategically under different market situations. A market manager needs to be designed to know: How to identify the needs for importing (buying) or exporting (selling) resources? what kind of resources? How much to buy or sell? The basic functions of a market manager can be categorized into two parts: calculation and decision making.

The auction mechanism used by a global market is different from the local markets. A global market uses a Vickery auction, also known as second-price sealed-bid auction. The Vickery auction is essentially a single-phase non-iterative protocol. Without knowing how much others' bids are, a bidder's offer price is based on his private value and prior beliefs of others' valuation. Because all bids are sealed, which means each bidder has no idea how much the others bid in the current session, the dominant strategy in a Vickery auction is to bid one's true value. So, bidding in a Vickery auction is much simpler and more straightforward than in other auctions, there's no need for bidders to counter-speculate what others' bids, therefore avoiding the long tedious bidding process.

The auction mechanism for the global market is essentially a sealed, single-phase non-iterative. Different types of resources can not be sold as a bundle. A market manager can only auction off multiple resources of the same type. The following is the bidding process in a global market:

1. A market manager submits a bid post request to the global market, providing a triple: $(p^k_{reserved}, k, m^k)$, which basically indicates its willingness to sell m^k items of class k resource with price higher than its sealed reserved price $p^k_{reserved}$.
2. The bid post request is accepted and the resources are reflected in the global auction. However, the reserved price $p^k_{reserved}$ is hidden from the bidders since the auction is sealed. The first bidder j can bid on auctioned resources using a triple: (p^k_j, k, m^k_j), where p^k_j is the bidding price, m^k_j is the bidding item and k

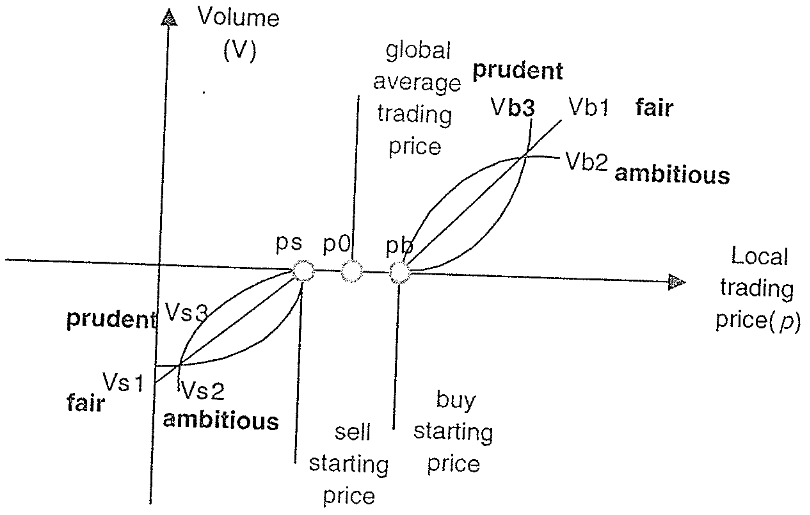


Figure 3.3: A Market Manager's Bidding Strategies

is the resource class, and $m_j^k <= m^k$.

The global market also maintains a history of the trading prices for all the local markets. The trading price history provides intuitive information on the possible outcomes of the trading prices trend. It helps the market managers decide where to buy the resources he needs. Determining from where to bid is only part of the strategy. Another aspect of the bidding strategy is to decide how much to bid?

Figure 3.3 illustrates the bidding strategies that a market manager could potentially have. A market manager adjusts his bidding strategy according to three parameters from previous sessions: a global average trading price p_0, a buy starting point p_b, and a sell starting point p_s. Note that p_0 is information published by the global market, accessible to all local market managers, p_b and p_s are private values used as measures by each market manager to quantitatively determine whether buying resources from or selling resources to the global market is necessary. For a class of resource i, a market manager compares the local market price p^i with the global average trading price p_0^i, and determines his buy starting point p_b^i and sell starting point p_s^i, and acts accordingly.

A market manager classifies its local market's desirability of a resource i into three

Table 3.1: A market manager's strategy and potential impacts

state	*strategy*	*potential impact*
equilibrium state	do nothing	no impact
buying state	bid on resources from the global market	local price decreases
selling state	posting a bid to sell resource	local price increases

states:

- Equilibrium state, if $p_s^i < p^i < p_b^i$;
- Buying state, if $p^i \geq p_b^i$;
- Selling state, if $p^i \leq p_s^i$.

Intuitively, an equilibrium state reveals that the local trading price is relatively close to the global average trading price, indicating that neither buying nor selling activities are able to bring any profit. Buying state represents a situation where a market manager could probably benefit from buying resource i from the global auction market, because the price difference makes buy cheaply and sell high possible. Similarly, selling state indicates a possibility of making profits by selling local resources to the global market due to a relatively lower local price.

As shown in Table 3.1 , the strategy of a market manager can potentially result in the increase or decrease of a resource's local trading price. That is, for a resource i, if the local trading price p^i is much higher than the global average trading price p_0^i, we denote, resource i is under buying state, meaning a market manager's best strategy is to bid on resource i from the global market, and if a market manager wins the bid(s), this would cause the local trading price for resource i decreases. Otherwise, if the local trading price p^i is much lower than p_0^i, resource i is under selling state, meaning a market manager's best strategy is to sell surplus resource i in the global market, and if the sell offer is successful, the local trading price for resource i would probably increase in the next session. And if the local trading price p^i is close to the global average trading price p_0^i, resource i is under equilibrium state, meaning a

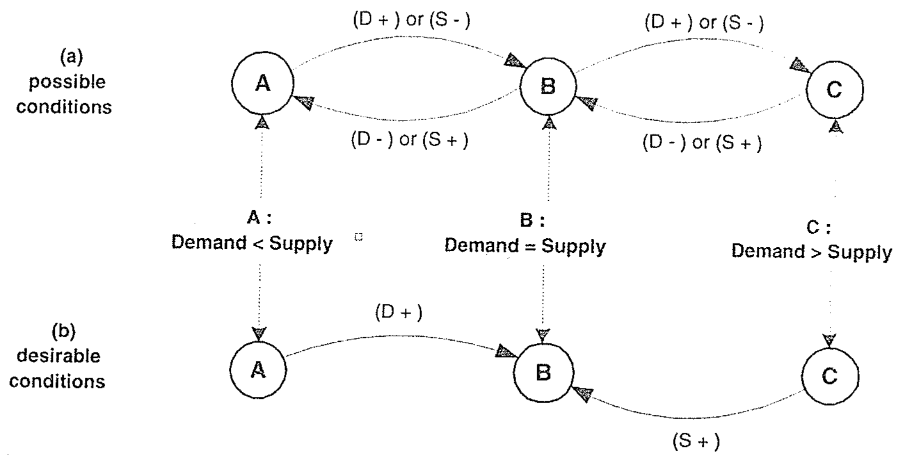


Figure 3.4: Demand and Supply State Transition Diagram

market manager's best strategy is to do nothing, hence no impact on the local trading price.

A market manager determines whether to buy or sell based on the price comparison between the local market and the global market, focusing on a price balance between them. The motivation of this price-centric design, though, is to maintain equilibrium of demand and supply of a resource. Figure 3.4 examines how the CM market is designed to generate the desirable conditions to maintain the equilibrium.

Figure 3.4 interprets the relationship between the demand and supply of a resource into three situations: State A, B, and C, which represent states where demand is less than, equal to, and greater than, supply respectively. The transition of state A to B, or from B to C, could be a result of: (1) an increase in demand (or, equivalently, the increase in demand is greater than the increase in supply) (2) a decrease in supply (or, equivalently, the decrease in demand is less than the decrease in supply). On the contrary, the transitions of state C to state B, or from state B to A are caused by: (1) a decrease in demand, (or, equivalently, the decrease in demand is greater than the decrease in supply) (2) an increase in supply (or, equivalently, the increase in demand is less than the increase in supply). Note that an increase in demand is essentially equivalent to a decrease in supply, and visa versa.

The (a) in Figure 3.4 lists all the conditions that could cause the transitions among the states A, B, and C. However, only the conditions in (b) are the desirable conditions that we want the market to present. As illustrated earlier, the market design desires equilibrium of demand and supply and abandons otherwise. Therefore, if a local market is in state B, a market manager requires no action; if a local market is in state A, a market manager needs to increase the demand, i.e., sell resources to the global market; If a local market is in state C, a market manager needs to increase the supply, i.e., buy resources from the global market. Eventually, the change in demand and supply will trigger the change of the trading price in a local market. Hence, a price-based decision is completely consistent with and reflects the relationship between demand and supply.

3.3.6 Bidding Price and Volume Strategy

An important aspect of the strategy is how a market manager determines the appropriate price and quality offered when buying or selling. As depicted in Figure 3.3, the CM market currently provides three distinct strategies, namely: prudent, fair, and ambitious - corresponding to a logarithmic, linear, and exponential curse respectively [1], which defines the quantity of buying or selling volumes V in terms of a function of local trading price p. The functions can be stated as:
If a market is under equilibrium, $p_s^i < p^i < p_b^i$, then

$$V = 0$$

If a market is under buying state, $p^i \geq p_b^i$, then

$$V = \begin{cases} \gamma_1^i \log(p^i - p_b^i + 1) & \text{if prudent strategy is used} \\ \gamma_2^i (p^i - p_b^i) & \text{if fair strategy is used} \\ \gamma_3^i \exp(p^i - p_b^i) & \text{if ambitous strategy is used} \end{cases}$$

where γ_1^i, γ_2^i,and γ_3^i are weights used to control the buying volume of resource i.

If a market is under selling state, $p^i \leq p^i_s$, then

$$V = \begin{cases} -\xi^i_1 \log(-(p^i - p^i_s + 1) & \text{if prudent strategy is used} \\ -\xi^i_2(-(p^i - p^i_s)) & \text{if fair strategy is used} \\ -\xi^i_3 \exp(-(p^i - p^i_s)) & \text{if ambitous strategy is used} \end{cases}$$

where ξ^i_1, ξ^i_2, and ξ^i_3 are weights used to control the selling volume of resource i.

Chapter 4

Bid Selection Heuristics

A sequential auction is an auction where items are auctioned one at a time. The sequential auction has been used extensively and is simple to implement. However, the sequential auction is not suitable for the CM for two reasons. First, most users in CM are interested in buying multiple unit of resources (i.e., bundles of resources that are allocated in an "all or none" manner), the sequential auction is not efficient enough to handle this. Second, the user requests can specify different combinations of the various resources at different quantities. Therefore, the CM adopts a multi-unit combinatorial auction (MUCA) in the local markets, in which users can bid for multiple items at one time and then allocate in an "all or none" manner. The MUCA auction offers the bidders the capability to bid flexibly, efficiently, and easily. However, it also brings a problem of computational complexity – determining the winners for a MUCA is NP-hard. Consequently, this problem can not be solved exactly, rather approximation algorithms or heuristics come handy here. Furthermore, the time available for selecting the winners is constrained by the start time of the next auction session and the "available" times of the resources. When the auctions are proceeding continuously with periodic closings, as it is here, the bid selection should be performed within a tight time interval. This makes solving the MUCA problem even more challenging.

When designing heuristics or approximation algorithms to handle NP-hard problems, we often find that maximizing the quality of the solution while minimizing the computational time are often two conflicting goals. We are facing this problem when designing heuristics which determines the winners of a MUCA process. We wish to maximize the revenues for the auctioning process, but this must be done inside the time window allowed by the closing periods. In the literature, revenue and efficiency are considered as the two most important measures for both auction market design and performance evaluation [19, 21]. Thus, most market designers typically chooses one, either revenue or efficiency, as the primary goal, depending on which measure is more important for that particular auction. In Federal Communication Commision (FCC), for instance, efficiency is considered as the primary goal because the FCC auction needs to assign the licences to those firms in a timely manner[19]. The reason is that if the auction takes very long to run, the licences can not be used by the firms, eventually leading to a poor revenue. Because the CM is designed to maximize the usage of computing and networking resource, the utilization, is identified as an important measure.

To solve the MUCA problem within the tight time schedule, heuristics are designed, seeking to achieve a good balance of efficiency, revenue, and utilization. Simulations are performed to assess the solutions. This section gives a formal model of the MUCA problem, and presents the bid-selection heuristics and the experimental results.

4.1 Multi-Unit Combinatorial Auction

The multi-unit combinatorial auction is essentially a multiple knapsack problem. We define the problem, formally, as follows:

Let $R = \{1, ..., N\}$ be a set of N classes of resources (knapsacks) to be auctioned, with capacities $(c_1, c_2, ..., c_N)$. Let $B = \{1, ..., K\}$ be a set of K bids. A bid $j \in [1..K]$ is denoted by $b(j) = (W_j, P_j)$, where $W_j = (w_{1j}, w_{2j}, ..., w_{Nj})$ represents the bidding quantities, and $P_j = (p_{1j}, p_{2j}, ..., p_{Nj})$ represents the bidding prices, for resources of class $1, 2, .., N$, respectively.

A feasible solution determines a subset $U \subseteq B$ as the winning bids. The total bid sizes $(w_1, w_2, ..., w_N)$ of the winning bids satisfy the constraint:

$$(w_1, w_2, ..., w_N) = \sum_{i \in U} (w_{1i}, w_{2i}, ..., w_{Ni}) \leq (c_1, c_2, ..., c_N) \tag{4.1}$$

The trading price $P_{\text{feasible}} = (p_1, p_2, ..., p_N)$ are the minimum unit price among the winning bids:

$$P_{\text{feasible}} = \{p_i \mid p_i \leq p_{ij}, \forall j \in U \text{ , and } i \in [1..N]\} \tag{4.2}$$

Consequently, the feasible solution U has a revenue of f equals to,

$$f(\text{ U}) = (w_1, w_2, ..., w_N) \times (p_1, p_2, ..., p_N) \tag{4.3}$$

An optimal solution U^* is a feasible solution that maximizes the revenue:

$$f(U^*) = \{f(U^*) | f(U^*) \geq f(U), U \subseteq B \text{ , and } \sum_{i \in U} w_{ji} \leq c_j, j \in [1..N]\} \tag{4.4}$$

4.2 Bid Selection Heuristics

As discussed earlier, the goal of the bid selection heuristic is to find a feasible solution that balances the revenue, efficiency, and utilization. That is, the heuristic should be able to find a feasible solution that gives as much revenue as possible within the tight time schedule.

According to equation 4.3, a greater revenue f can be achieved in two ways:

- by maximizing the total number of items to be placed in the knapsacks,

- by improving the trading prices, given that the number of items placed in the knapsacks is the same.

Based on this, the algorithm in selecting the winning bids is designed with two search stages:

1. The primary phase(PP) : aims to place as many "high fitness" bids as possible in the knapsacks;
2. The refinement phase(RP): aims to improve the trading price by removing the lower price bids out of the solutions and replacing with higher price bids.

Here, how the "fitness" of a bid is measured will be discussed in the next section.

4.2.1 Data Structure Used in Bid Selection Heuristics

The heuristics uses *rate* and *bid tree* to help choose the winning bids wisely and efficiently. This section discusses the concepts of *rate* and *bid tree* and the motivation of the design.

Rate

In a combinatorial auction, the resources asked and the prices offered differ among the bids. Determining which bid offers a better price is not trivial in MUCAs. But knowing how promising a bid is in maximizing the revenue is critical for an efficient bid selection. To illustrate the scenario:

Case 1: *It is very hard to tell which bid is higher if different combinations of resources are asked.*
As an example, consider two bids: *Bid A* offers price vector $(3.5, 2.6, 0)$, *Bid B* offers price vector $(3.4, 0, 1.0)$. Obviously, deciding which one, Bid A or Bid B, is more

promising in maximizing the revenue is not trivial. As the number of resource types increases, judging which bid has a better price becomes even more difficult.

Case 2: *It becomes even more tricky to compare how promising a bid's prices among bids asking for the same combination of resource types.*
Consider, for instance, a scenario as follows: a total of $(10, 20, 15)$ of resources are auctioned. Assume there already existed 5 bids in the auction, with a total number of bid items: $(t_1, t_2, t_3) = (20, 30, 15)$, and average prices $(p_1, p_2, p_3) = (2.5, 1.5, 1.0)$. Let's consider two new bids: Bid A and Bid B (as shown in Table 4.1) to determine which bid is more promising in improving revenue. To compare, it's intuitive to consider Bid A and Bid B's contribution to the average prices separately. That is, we consider the final average prices in cases when only bid A or bid B is added to the auction. Notice that the average prices are not actually used in CM as the winning price rule. Instead, the minimum prices among the winning bidders are used to determine the final selling prices. However, it is generally observed and understood that competitive bidding behavior among the bidders will usually lead to higher average prices, therefore better revenue for an auction. Therefore, we consider the average prices do provide intuitive information regarding improving the revenue of an auction, although we can't prove it mathematically.

Table 4.1: An example of a bid's contribution to the total average prices

Bids	**Bid item vector**	**Bid price vector**	**Avg. prices after Bid A or B is added**
Bid A	$(w_{1A}, w_{2A}, w_{3A}) = (10, 2, 0)$	$(p_{1A}, p_{2A}, p_{3A}) = (2.3, 2.0, 0)$	$(2.43, 1.53, 1.0)$
Bid B	$(w_{1B}, w_{2B}, w_{3B}) = (2, 10, 0)$	$(p_{1A}, p_{2A}, p_{3A}) = (2.2, 2.0, 0)$	$(2.47, 1.63, 1.0)$

Only add Bid A to the auction will change the average prices (p_1^0, p_2^0, p_3^0), to:

$$
\begin{aligned}
(p_1^{0}, p_2^{0}, p_3^{0}) &= \left(\frac{t_1 \times p_1 + w_{1A} \times p_{1A}}{t_1 + w_{1A}}, \frac{t_2 \times p_2 + w_{2A} \times p_{2A}}{t_2 + w_{2A}}, \frac{t_3 \times p_3 + w_{3A} \times p_{3A}}{t_3 + w_{3A}}\right) \\
&= \left(\frac{20 \times 2.5 + 10 \times 2.3}{20 + 10}, \frac{30 \times 1.5 + 2 \times 2.0}{30 + 2}, 1.0\right) \\
&= (2.43, 1.53, 1.0)
\end{aligned}
$$

Similarly, if only Bid B is added to the auction, the average prices becomes:

$$
\begin{aligned}
(p_1^{(0)}, p_2^{(0)}, p_3^{(0)}) &= \left(\frac{t_1 \times p_1 + w_{1B} \times p_{1B}}{t_1 + w_{1B}}, \frac{t_2 \times p_2 + w_{2B} \times p_{2B}}{t_2 + w_{2B}}, \frac{t_3 \times p_3 + w_{3B} \times p_{3B}}{t_3 + w_{3B}}\right) \\
&= \left(\frac{20 \times 2.5 + 2 \times 2.2}{20 + 2}, \frac{30 \times 1.5 + 10 \times 2.0}{30 + 10}, 1.0\right) \\
&= (2.47, 1.63, 1.0)
\end{aligned}
$$

Surprisingly, the contribution of Bid A and Bid B to the average prices appears to be not consistent with their price offers. For class 1 resource, Bid A outbids Bid B by 0.1\$; They offer the same price for class 2 resource, and both have no offer for class 3 resource. It seems that Bid A should lead to higher average prices than Bid B, or at least for class 1 resources. But the calculation results showed that Bid B actually contributes more on average price for both class 1 and class 2 resources. The reason behind the scene is that how promising a bid is does not only depends on the bidding prices. In fact, it's a combination of the following three factors:

1. A bid's offer prices.
2. The composition of the bid resource types and the number of items for each resource type.
3. The competition condition for each resource type in the auction. How competitive is the bid against others in overall?

Therefore, the CM needs a mechanism to quantitatively measure the above factors and to direct the bid selection process. **Rate** is a measure to help in determining how promising a bid is regarding revenue maximization, which is defined as the value based on a bid's own offered prices divided by the value of the bid based on the average bid prices in the auction, formally as follows:

$$Rate(i) = \frac{\sum_{k=1}^{k=N} p_{ik} \times w_{ik}}{\sum_{k=1}^{k=N} a_k \times w_{ik}} \tag{4.5}$$

where i represents a Bid's ID, N is the number of resource classes in this auction, k is the resource class, p_{ik} and w_{ik} are Bid i's bidding price, and items for class k resource, respectively; a_k is the overall average bidding price for resource k in the auction.

Intuitively, the rate combines the prices, items composition, and competition factors, in the auction, allowing to measure the "quality"(or "fitness") of a bid quantitatively. The bid selection heuristic then will choose those bids with higher rate first, aiming at finding a good solution quickly. To testify the efficiency of the rate, experiments are performed to compare it with different measures such as total bidding prices, or bidding items. The results will be discussed later in this chapter.

Bid Tree

The bid selection algorithm uses a bid tree[27] that is essentially a binary tree with its depth equal to the number of resource classes. Our bid tree (as shown in 4.1) is different from that of [27] because in the CM each resource class can have multiple items. Therefore, the leaves in our bid tree are a linked list of bids with the same combination of resource types. The linked-lists in the leaves are ordered based on the *rate* of bids starting from the highest rate bids. The path from the root to a leaf represents the classes of resources in a bid at the leaf. From any node, the left branch leads to a bid with the corresponding class of resource; the right branch leads to a bid without the corresponding class of resource. Figure 4.1 is an example of how bids (as shown in Table 4.2) are organized in a bid tree.

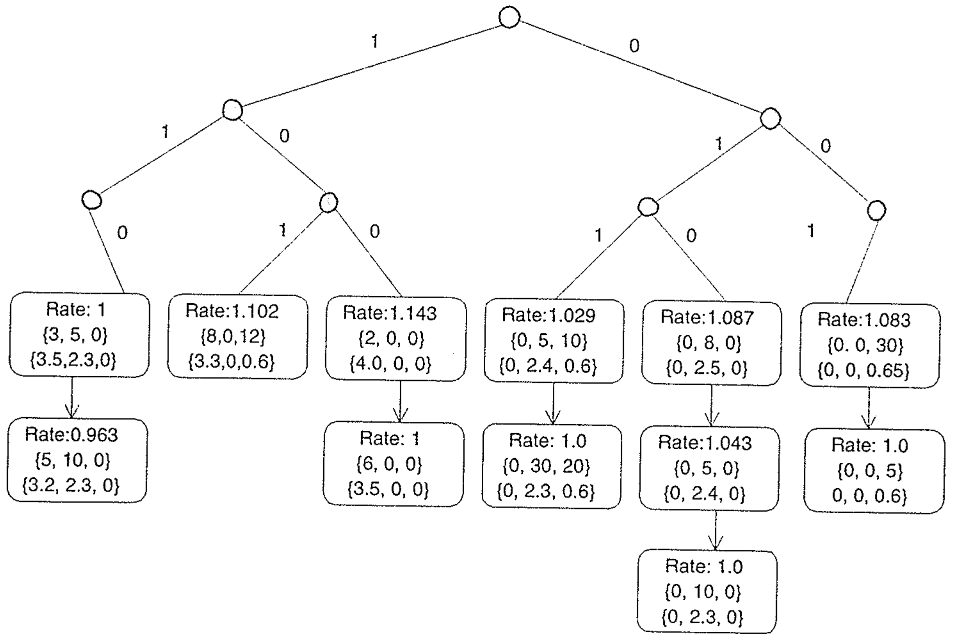


Figure 4.1: The bid tree data structure for the bids in Table 4.2

Table 4.2: An example of bids submitted to the Auction

Bid No.	Bidders	Bid item vector	Bid price vector	Bid rate
1	Bidder 3	$\{5, 10, 0\}$	$\{3.2, 2.3, 0\}$	0.963
2	Bidder 5	$\{3, 5, 0\}$	$\{3.5, 2.3, 0\}$	1.0
3	Bidder 1	$\{8, 0, 12\}$	$\{3.3, 0, 0.6\}$	1.012
4	Bidder 6	$\{2, 0, 0\}$	$\{4.0, 0, 0\}$	1.143
5	Bidder 12	$\{6, 0, 0\}$	$\{3.5, 0, 0\}$	1.0
6	Bidder 4	$\{0, 10, 0\}$	$\{0, 2.3, 0\}$	1.0
7	Bidder 8	$\{0, 5, 0\}$	$\{0, 2.4, 0\}$	1.043
8	Bidder 2	$\{0, 8, 0\}$	$\{0, 2.5, 0\}$	1.087
9	Bidder 10	$\{0, 30, 20\}$	$\{0, 2.3, 0.6\}$	1.0
10	Bidder 7	$\{0, 5, 10\}$	$\{0, 2.4, 0.6\}$	1.087
11	Bidder 9	$\{0, 0, 5\}$	$\{0, 0, 0.6\}$	1.0
12	Bidder 11	$\{0, 0, 30\}$	$\{0, 0, 0.65\}$	1.083

4.2.2 Bid Selection Heuristics

As mentioned earlier, the bid selection heuristics use two stages in determining the winning bids: the primary phase and the refinement phase. We assume that the resource classes are arranged in a descending order by their expected prices, i.e., resources of class 1 are expected to be sold at a higher price than resources of class 2. The primary phase starts with class 1 resources and adds bids until it can't find any such bids or no bids can be included due to resource exhaustion. Then it adds class 2 resources and so on.

A "selection mask" of size N is used to implement the primary phase, acting as a mechanism to indicate the search space as the algorithm progresses. The n-th location of the mask corresponds to class n resource, where $n \in [1..N]$. A location in the mask has three states: "*enabled*", "*disabled*", and "*any*". The values in the selection mask determine the search space of the corresponding resource class. If a class's corresponding mask value is:

- *enabled* : the search space includes those bids that consist of this resource class;
- *disabled*: the search space excludes those bids that consist of this resource class;
- *any* : this resource class can be in or out of the search space.

The primary phase algorithm is given below:

1. The search starts with the first value "enabled" in the selection mask and others with "any". In this case, the selection mask values { *"enabled", "any", "any"*}, indicating the search will consider those bids asking for resources of class 1,

with or without asking for resources of class 2 and class 3. Consequently, the search space includes the bids that can be reached by traversing the paths {*111,110,101,100*} in the bid tree. Therefore, the following bids are under the scope of the search space of this stage:

- Bid No. 1; $\{Rate = 1, (3, 5, 0), (3.5, 2.3, 0)\}$
- Bid No. 2: $\{Rate = 0.963, (5, 10, 0), (3.2, 2.3, 0)\}$
- Bid No. 3: $\{Rate = 1.102, (8, 0, 12), (3.3, 0, 0.6)\}$
- Bid No. 4: $\{Rate = 1.143, (2, 0, 0), (4.0, 0, 0)\}$
- Bid No. 5: $\{Rate = 1, (6, 0, 0), (3.5, 0, 0)\}$

2. The search determines the next bid to be added by considering the bid with the highest "fitness" in the search space. By choosing different parameters such as price, *value*, or *rate*, we can create a family of heuristics. If the available resources are enough to be allocated to the current highest rate bid, then move the bid to the *solution-list*.

3. Repeat step 2 until no bids can be added to the *solution-list*.

4. Change the current "*enabled*" selection mask variable to "*disabled*" and set the next selection mask location from "*any*" to "*enabled*". The search space changes due to the changes in the selection mask. Consequently, those bids that include the class with "*disabled*" in their corresponding selection masks will be excluded from the next stage of search because they've already been considered in the previous stage. Follows the bids listed in Figure 4.1, below is the set of bids that will be included in the search space when selection mask values equal to {*"disabled"*, *"enabled"*, *"any"*}:

- Bid No. 8: $\{Rate = 1.087, (0, 8, 0), (0, 0.25, 0)\}$
- Bid No. 7: $\{Rate = 1.043, (0, 5, 0), (0, 0.24, 0)\}$
- Bid No. 10: $\{Rate = 1.029, (0, 5, 10), (0, 2.4, 0.6)\}$
- Bid No. 9: $\{Rate = 1.0, (0, 30, 20), (0, 2.3, 0.6)\}$

5. Repeat step $2, 3$, and 4 until all the selection mask locations are "*disabled*". The primary phase terminates.

Theoretically, the bid tree data structure used in the primary phase can equivalently be replaced by using a linked-list. However, bid tree is used to organize the bids nicely based on the compositions and the rate of the bids. It makes evolution of search space from one stage to the other clear and straight-forward.

The primary phase achieves a feasible solution for the auction. The solution obtained by the primary phase is improved by a refinement phase, which focuses on trading price improvement. The algorithm records the minimum bid *rate(=b)* in the *solution-list* after the primary phase. During the refinement phase, we only consider those bids that remain in the bid tree with rate higher than b. Those bids are moved from the bidtree to a *back-list*. The *back-list* can be sorted by *rate*, *value*, or *price*, depending on which parameter is used in the refinement phase. Assume after the primary phase, the revenue is m. The following is a brief description of the refinement phase:

The refinement phase examines the bids in the *back-list* starting at the top of the list. For each bid, it determines whether sufficient resources are available to accommodate the bid. If resources are found, the bid is inserted into a new *solution-list*; Otherwise, bids are removed from the previous *solution-list* to make room for the new bid. Once enough space is found, the bid from the *back-list* is inserted into the *solution-list*.

There may exist some space for re-inserting some of the removed bids or some more bids from the *back-list* into the *solution-list* because of the removal of bids from the *solution-list*. A series of adds and removes may have been performed on the *solution-list* at this time. To determine the impact of these changes, the total revenue provided by the current version of the *solution-list* is evaluated. Let this revenue be p. We note down this revenue in a *history-list* and if $p > 0.8m$, the search continues examining the rest of the bids on the *back-list*.

The refinements performed to the *solution-list* are tagged by the last known revenue and recorded in a *history-list*. Once the refinement phase terminates due to the revenue falling below $0.8m$ or empty *back-list*, the *history-list* is used to roll-back to the best solution that was found during the refinement phase.

4.2.3 Simulation Results and Discussion

In this section, we present the results from the simulations that evaluate the performance of the bid selection heuristics under different scenarios. All heuristics are based on the primary and refinement phases explained in Section 4.2.2. The performance tests are based on three parameters: **bid rate (R)**, **bid value (V)**, and **bid price (P)**. Different combinations of thsee parameters (shown in 4.3) are used to study the impact of these parameters.

Table 4.3: Heuristics Using Different Combinations of Parameters

Heuristics name	**Primary phase parameter**	**Refinement phase parameter**
VR	value	price
RV	rate	value

Table 4.3: Heuristics Using Different Combinations of Parameters

RP	rate	price

For each parameter combination, three test cases (**A**,**B**, **C**) are designed and the configurations of each test case are listed in Table 4.4 and 4.5.

Table 4.4: Common configurations used in all test cases

Configuration name	**Value**
Number of resource classes:	15
Number of simulation runs:	100
Number of resources per class:	uniformly from the range [2000-2250].
Reserve prices for resources:	uniformly from the range [0.5-4.5].

Table 4.5: Distinguish configurations of the test cases

Test case	**Number of bids**	**Mean items/bid**	**Mean classes/bid**
A	1000	500	4
B	1000	100	4
C	1000	100	2

Table 4.6, 4.7, 4.8 shows the relative performance of the heuristics for test case A, B, and C respectively.

Table 4.6: Performance of the bid selection heuristics for test case A

Heuristics	Upper bound	Primary phase	PP as percent of upper bound	Refinement phase	Improvement through refinement	Final result as percent of upper bound
VR	99226	79557	80.99%	84411	6.10%	85.94%
RV	98226	84886	86.42%	85388	0.60%	86.93%
RP	98226	84886	86.42%	87265	2.80%	88.84%

Table 4.6: Performance of the bid selection heuristics for test case A

Table 4.7: Performance of the bid selection heuristics for test case B

Heuristics	Upper bound	Primary phase	PP as percent of upper bound	Refinement phase	Improvement through refinement	Final result as percent of upper bound
VR	92001	79001	85.87%	81847	3.60%	88.96%
RV	92001	83795	91.08%	84005	0.25%	91.31%
RP	92001	93795	91.08%	85231	1.71%	92.64%

Table 4.8: Performance of the bid selection heuristics for test case C

Heuristics	Upper bound	Primary phase	PP as percent of upper bound	Refinement phase	Improvement through refinement	Final result as percent of upper bound
VR	102516	80870	78.88%	87924	8.72%	85.76%
RV	102516	88851	86.67%	89532	0.77%	87.33%
RP	102516	88851	86.67%	92428	4.02%	90.16%

The results show that the bid-value-based primary phase has the worst results. For test cases A, B, and C, bid-value-based primary phase results are (80.99%, 85.87%, 78.88%) as compared to (86.42%, 91.08%, 86.67%) for rate-based primary phase. Particularly interesting is the observation that when average bid size is small(test case B), the primary phase yields better results than with bigger average bid sizes (test cases A and C). Further, the rate-based primary phase combined with price-based refinement performs the best in all three test cases. This suggests that price (or price related factor such as rate) has the most impact(dominates) on the revenue.

The VR approach has the biggest refinement (6.10%, 3.60%, 8.72%) in any of the test cases. It can be observed that the bigger the bid size is, the more benefits achieved by the refinement phase. However, irrespective of the refinement improvement, the final results of VR are always worse than RV and RP. Intuitively, this suggests that considering the most promising bids in terms of producing the revenue first has a significant impact on the final result.

4.3 Generating Bids

Using simulation methods to model real-world problem demands generating data that reflects real-world problems. Particularly, generating bids that can match the realistic bidding behavior would help us better understand and evaluate our algorithms. However, as there are no auctions for exchanging computing and networking resources in the real world, we thus lack such realistic data. What we can do is to generate bids that we believe would most likely appear in a realistic situation, based on our knowledge and understanding of the problem.

4.3.1 Required Parameters

The parameters needed for modeling the auction can be divided into two groups: problem modeling parameters, and bid parameters. The problem modeling parameters relate to MUCA auction modeling. The bid parameters relate to bid generation in a MUCA.

Modeling the MUCA problem in the CM needs the following parameters:

- numLBs: the number of local brokers;
- numClasses: the number of resource classes;
- numbids[numLBs]: the number of bids generated for each local broker;
- numItems[numLBs][numClasses]: the number of auctioned items of each class for each local broker;
- numCoBids: the number of CoBids;

Generating a bid for a MUCA involves the following parameters:

- lbID: local broker ID indicates which local broker this bid belongs to.
- bidItems[numClasses]: the array indicates the number of item for each resource class.
- numPrices[numClasses]: the array indicates the bid price for each resource class.

4.3.2 Generating the parameters

For the required parameters listed in 4.3.1, some parameters (such as numLBs, numClasses) are directly provided, while others are programmably generated by simulation. This section focuses on how to programmably generate the parameters and which distribution function to be used.

The following shows how the modeling parameters are generated:

- numBids[numLBs] Input parameters: **startNumBids, endNumBids**

  numBids[i] is randomly distributed from [startNumBids, endNumBids], where $i \in [1..numLBs]$.

- numItems[numLBs][numClasses]

  Input parameters: **startNumItems, endNumItems**

  numItems[i][j] is randomly distributed from [startNumItems, endNumItems], where $i \in [1..numLBs]$, and $j \in [1..numClasses]$.

Generating a bid is equal to answering the following questions: (1) How many items does a bid have? (2) how many classes of resources in this bid? (3) how many items in each class and which class? (4) what are the prices for these items? Here's the answers to the above questions:

1. How many total bid items (ttlBidItems) in a bid?

   Input parameters: **meanBidItems** indicates the average bid items for all bids generated.

   Output parameters: **total bid items** in a bid is generated as exponential distribution with an average of meanBidItems:

   expBidItems = new Exponential (meanBidItems, bidStream);

   ttlBidItems = expBidItems.sample();

2. How many classes (classNum) in a bid?

   Input parameters: **meanClassNum** indicates the average classes for all bids generated;

Output parameters: **number of classes** in a bid is generated as exponential distribution with an average of meanClassNum.

3. How many items for each class and which class?

```
/*
  generateBidItems Function that generate the bid items for a bid.
  variables:
  classNum      : number of resource classes this bid has;
  ttlBidItems : total number of items (for all resource classes)
                     asked by this bid;
  allocItems   : number of items already allocated by this bid;
  classPos      : the resource class number;
*/
Function generateBidItems(classNum, ttlBidItems)
   allocItems = 0;
   n = 1;
   While (n ¡= classNum)
      //classPos indicates this class has bid items
      Randomly choose a classPos from [0, classNum-1];
      Randomly generate a number p from [0, 1];

      // If there's no item for this classPos resource
      If (bidItems[classPos] == 0)
         If (n ¡ classNum)
            //if this is not the last class that
            bidItems[classPos] = (ttlBidItems - allocItems) * p;
         Else
            bidItems[classPos] = ttlBidItems - allocItems;
         End If
         allocItems += bidItems[classPos];
         n = n + 1;
      End If

   End While
End Function
```

4. What price for each class?

Input parameters:

resPrices[numClasses] indicates the reserved price for each class.
incUnits[numClasses] indicates the minimum increase unit for each class.
prBidInc[5] = [0, 0.2, 0.8, 0.9, 1.0] indicates the probability of price increasement ε, the price increase by $(1*\varepsilon, 2*\varepsilon, 3*\varepsilon, 4*\varepsilon)$ with probability of (0.2, 0.6, 0.1, 0.1) respectively. This is based on the observation and understanding that most of the bidders will bid rationally – 80% of bidders only increase price by 1 or 2 times of minimum price improvement requirement.

```
/**
 * getIncRate:              function that generates the price increase
 *                            rate for bid price.
 */
Function getIncRate()
    rate = 1;
    rand = Generate a random number from [0, 1];
    For i = 0 to prBidInc.length
        If (prBidInc[i] ¡ rand ¡= prBidInc[i+1] )
            rate = rate + i;
        End If
        Return rate;
    End For
End Function

/**
*   generateBidPrices       function that generates the bid prices
*                             vector based on the resPrices and
*                             incUnit.
* resPrices[numClasses]: the reserved price vector
* incUnits[numClasses]:  the minimum price increasement requirement
*                              for each resource class
*/
Function generateBidPrices(resPrices[numClasses], incUnits[numClasses])
    curHighestPrices = get current highest bid from bids with
                          the same combination of classes;
    If (classNum == 1)
        rate = getIncRate(incUnit);
        For n = 0 to numClasses-1
            If (bidItems[n] ¿ 0)
                BidPrices[n] = curHighestPrices[n] + incUnits[n] * rate;
            End If
        End For
    Else
        All classes has the same probability of increase bid price
    End If
End Function
```

Chapter 5

Heuristics for Resource Co-allocation

One of our design goals is to allow bidders to indicate their bidding requirements flexibly, efficiently, and easily. Different bid types are designed to achieve this goal. To support high resource-consuming tasks such as: bio-computing, multimedia applications, and parallel computing, resource co-allocation is highly desirable. As described in market design, CoBids makes bid concurrently from several auctions and then allocates in an "all or none" manner possible.

A bidder specifies his requirement on the locations of the resources by placing three kinds of bids:

LB specific bid: resources should be from a specific local broker.

CoBids: resources are from several local brokers instead of one local broker.

Any-bid: no constraints on resource location.

Because *any-bid* have the loosest constraints on resource locations, normally, they have more chances to win.

This chapter discusses the extension of the bid-selection heuristics in chapter 4 to two new heuristics to address the resource co-allocation issue when different bid types are jointly used. The two new heuristics, namely, *CoBids first approach*, and *no preference approach*, are designed mainly to: (1) experiment how to allocate the CoBids; (2) research to help in understanding how the CoBids impacts the utilization and the revenue.

5.1 CoBids First Approach

The basic idea of the CoBids first approach is to allocate CoBids first because CoBids have more constraints than other bids.

Figure 5.1 shows the sequence of the CoBids first approach. To better illustrate the approach, we make the following assumptions: assume LBs (A, B, C, D, E, F) are posting sell bid offers, and a set of bidders $N = \{1, 2, ..., n\}$ place various bids, including CoBids, LB specific bids, and any-bid, bidding on resources from LBs. As indicated in figure 5.1, Bidders [1..5] place CoBids, while other bidders place LB specific bids and any-bids. Let $\{a_k\}$ denotes bidder k place a bid on resources from LB A.

The CoBids first approach involves six steps:

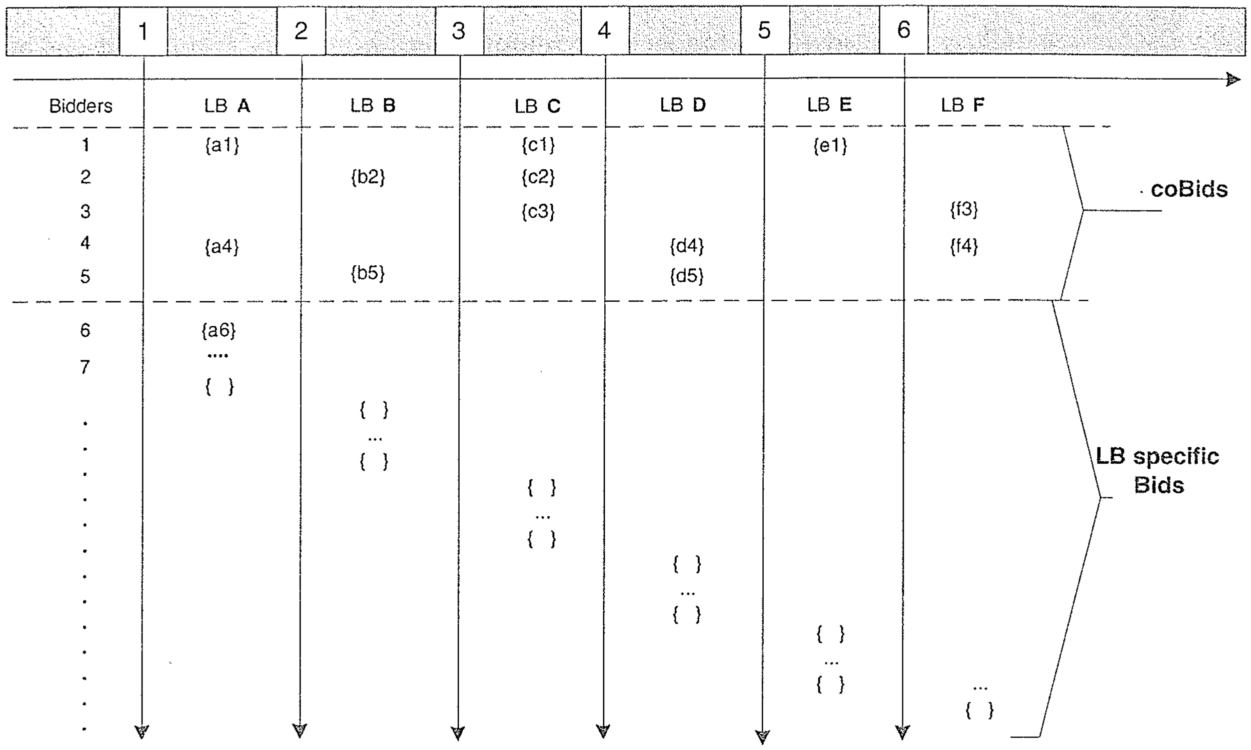


Figure 5.1: Figure CoBids First Approach Algorithm

Step 1: Allocate CoBids first using the primary phase (PP) algorithm.

Step 2: Allocate the LB specific-bids using the PP algorithm.

Step 3: Backtrack for all the CoBids related bids as follows:

Assume, after *Step 1* and *Step 2*, the revenues for LB (A, B, C, D, E, F) are (*revenue(A), revenue(B), revenue(C), revenue(D), revenue(E)*) correspondingly.

Backtrack uses an arbitrary order for the LBs. Let's assume the sequence of backtrack is from LB A to LB F. Within each LB, consider bids with rates from the lowest to the highest. Given, for instance, $rate(a_1) < rate(a_4)$ and $rate(b_2) < rate(b_5)$, the order of backtracking is: $bidder1 \rightarrow bidder4 \rightarrow bidder2 \rightarrow bidder5 \rightarrow bidder3$. For each CoBids, the backtrack is performed as follows: (as an example, let's consider bidder 1)

I. Check the bids from bidder 1 by removing a_1, c_1, and e_1 from solutions of LB A, C, and E correspondingly;

II. Reschedule and get new revenues for LB (A, C, E), which are (*new_revenue(A)*, *new_revenue(C)*, *new_revenue(E)*), correspondingly.

III. If $\sum_{lb \in \{A,B,C\}} new\_revenue(lb) > \sum_{lb \in \{A,B,C\}} revenue(lb)$, then remove bids a_1, c_1, e_1 from their corresponding solution sets; Otherwise, fix all bids offered by bidder 1.

Step 4: Allocate the any-bids using the PP algorithm. For each LB with available capacities, check if there's any unallocated any-bids can be allocated to this LB, and accept those any-bids that improve the total revenue.

Step 5: Keep on doing 4 until no capacity is left for all LB, or no un-allocated any-bid can lead to an increase of the total revenue.

5.2 No Preference Approach

Different from CoBids first approach, *no preference approach*, as depicted by its name, is to treat all the bids with the same priority no matter what type of bid it is. The no preference approach involves four steps:

Step 1: Allocate all the bids equally with no preference for CoBids using the primary phase algorithm. Let $S_A, S_B, S_C, S_D, S_E, S_F$ be the set of bids that are included in the solutions for LB A, B, C, D, E, and F correspondingly. Let $R_A, R_B, R_C, R_D, R_E, R_F$ be the revenues from their corresponding solutions $S_A, S_B, S_C, S_D, S_E, S_F$ respectively.

Step 2: Check if each coBid's related bids are in their corresponding solution_sets. The order of the checking process starts from the highest CoBids to the lowest CoBids (Consider CoBids with bid for LB A first, then B, then C, ..., to F)

For LB A, B, C, D, E, F, we assume:

- Current solution sets: S_A, S_B, S_C, S_D, S_E, and S_F.
- Revenues: R_A, R_B, R_C, R_D, R_E, and R_F.

The following is the pseudocode for the no preference approach:

```
Let $G$ = the set of CoBids;
While ($G \neq \{\}$)
    Let $aCoBids$ = remove a CoBids from $G$;
    Let $B$ = all bids in aCoBids;

    //if all bids in $B$ are in the current solution sets
    If ($\forall b \in B$ are in the solution sets $[S_A..S_F]$) Then
        Mark all bids in $B$ with fix tags;
    Else
        // all bids in B are seperated into two sets: Y and N
        // Y: those bids that are included in the current solution sets
        Let $Y = \{Y \subseteq B | \forall y \in Y \wedge y \in \bigcup_{i \in [A..F]} S_i\}$
        // N: those bids that are not included in the current solution sets
        Let $N = \{N \subseteq B | \forall n \in N \wedge n \notin \bigcup_{i \in [A..F]} S_i\}$;

        Try 1:
            Add all $n \in N$ to the solutions by removing bids without fix
tags
            as needed, Evaluate solutions after the replacement,
            Calculate revenues from the new solution sets as:
            $(R_A^N, R_B^N, R_C^N, R_D^N, R_E^N, R_F^N)$.

        Try 2:
                Remove all $y \in Y$ from the solutions and add some bids as needed,
                Evaluate the solutions after the replacement,
                Calculate revenues from the new solution sets as:
                $(R_A^Y, R_B^Y, R_C^Y, R_D^Y, R_E^Y, R_F^Y)$.

            *Evaluate Try 1 and Try 2:*
                If $(\sum_{i \in [A..F]} R_i^N)$ ¿ $(\sum_{i \in [A..F]} R_i^Y)$ Then
                    Accept solutions from *Try 1*:
                    Update solution sets $S_A, S_B, S_C, S_D, S_E, S_F$ accordingly;
                    Update revenues: $R_i = R_i^N$, for all $i \in [A..F]$;
                    Label all $b \in B$ with fix tags.
                Else
                    Accept solutions from *Try 2*:
                    Update solution sets $S_A, S_B, S_C, S_D, S_E, S_F$ accordingly;
                    Update revenues: $R_i = R_i^Y$, for $i \in [A..F]$;
                    Discard all $b \in B$.
                End If
            End If
        End While
```

Step 3: Allocate the any-bids using the PP algorithm. For each LB with available capacities, check if there's any unallocated any-bids can be allocated to this LB, and accept those any-bids that improve the total revenue.

Step 4: Keep on doing 3 until no capacity is left for all LB, or no un-allocated any-bid can lead to an increase of the total revenue.

5.3 Simulation Results

The intent of the simulation is to achieve a better understanding of the impact of the *CoBids first approach* and the *no preference approach* , and to evaluate the design of the algorithms. A group of simulation test cases are designed to compare the performance measures between the two approaches. To assess the auction market design practically and experimentally, efficiency and revenue maximization are extensively used in literature [21, 18]. As the CM is designed to maximize the utilization of the computing and networking resources, we also consider the utilization as an important performance measure for resource allocation. Therefore, unlike most other auction market designs, this group of simulations use *utilization* and *revenue* to evaluate the algorithms. The decision of not measuring the efficiency of the algorithms is based on the observation that the heuristics are always capable of solving the MUCA problem quickly during the simulations.

The simulations are based on the following parameters:

- Percentage of CoBids;
- Number of bids;
- Number of LBs;
- Average of LBs per CoBids.

Using the above parameters, the performed simulation, therefore, enables us to answer the following questions:

(1). Which approach is capable of achieving a better utilization?

(2). How is the utilization affected by the above parameters?

(3). How is the revenue affected by the above parameters?

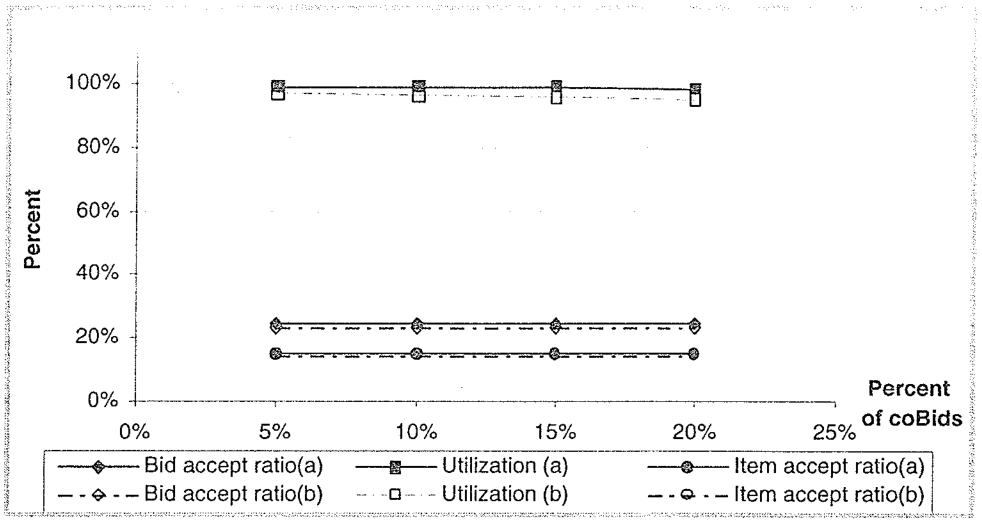


Figure 5.2: Bid accept ratio, utilization, and item accept ratio versus percentage of CoBids. Simulation configurations: the number of LBs = 20, the number of bids for each LB = [200-300]

5.3.1 Simulation results: utilizations

The utilization related performance measures used in the simulations include: bid accept ratio, utilization, and item accept ratio, which are defined as follows:

(5.1)

(5.2)

(5.3)

The simulation results are as follows:

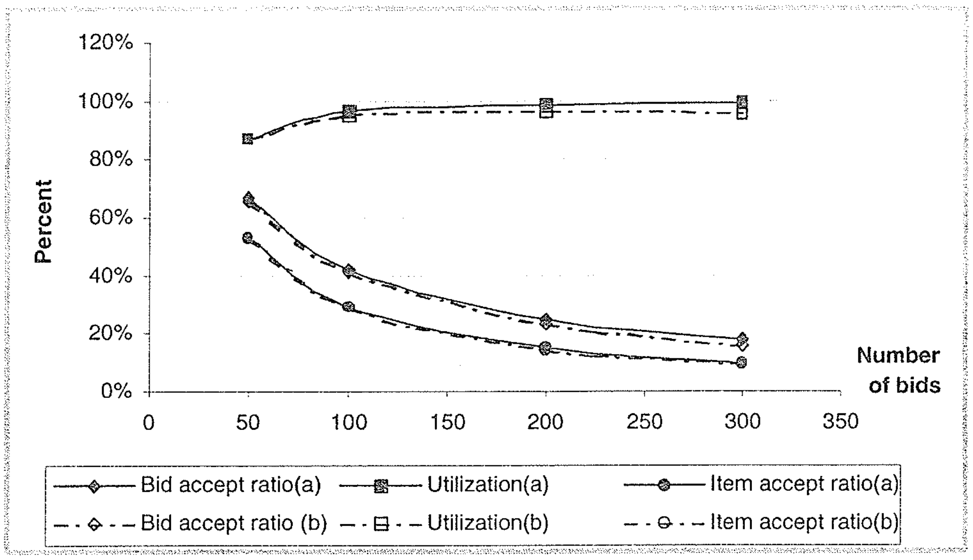


Figure 5.3: Figure Bid accept ratio, utilization, and item accept ratio versus the number of bids. Simulation configurations: the number of LBs = 20, the percentage of CoBids = 10%.

Figure 5.2 shows the performance measures of the two approaches under different percentage of CoBids. The simulation uses 20 local brokers, 200-300 bids are randomly generated for each local broker. The tests are performed based on a set of bids in which 5%, 10%, 15%, and 20% of the bids are CoBids. For both approaches, the bid accept ratio and item accept ratio are stable when the percentage of CoBids varies from 5% to 20%, with the CFA (a) performs slightly better than the NPA (b). The utilization decreases with an increase in the percentage of CoBids for both approaches. The decrease of the utilization is caused by more constraints on allocating the resources, hence causing some resources not able to be allocated. The CFA exhibits a much better utilization comparing with that of the NPA when the percentage of CoBids increases, indicating that considering the bids with more constraints help allocate more resources.

Figure 5.3 shows the performance measures when the number of bids generated varies. The bid accept ratio and the item accept ratio decrease with the increase of the number of bids for both approaches. Notice that the two approaches have almost identical curves for the item accept ratio, while the CFA yields a slightly better bid accept ratio when the number of bids increases. The better bid accept ratio suggests that evaluating the CoBids first could potentially include more bids in the solution. And because the item accept ratios are the same, therefore, suggesting that small size bids (bids asking for a smaller number of items) have more chance to win in the CFA. The utilization increases with an increase in the number of bids when the number of bids are within [50-100] range for both approaches. When the number of bids are between [100-300], the utilization of the CFA is stable and close to 100% , while the utilization of the NPA decreases slightly with the increase of the number of bids. The reason is that the CFA tends to include more small-size bids in the solution, therefore, leading to a better utilization.

Figure 5.4 shows how the number of LBs affects the performance measures. The simulation randomly generates [200-300] bids with 20% of which are CoBids. The tests are performed when the number of LBs are 5, 10, 25, and 20. For both approaches, the item accept ratio is almost the same and stable; the bid accept ratio is stable too, but the bid accept ratio for the CFA is slightly better than that of the NPA. The results are consistent with those shown in Figure 5.3, the small-size bids have more chances to win the auction no matter what the size of the number of LBs is. For both approaches, the utilization increases with the increase in the number of LBs when the number of LBs is within [5-10]. When the number of LBs is between 10-20, the utilization of CFA is stable and close to 100%; while the utilization of the NPA is also stable, but slightly less than that of the CFA, because more constraints exist

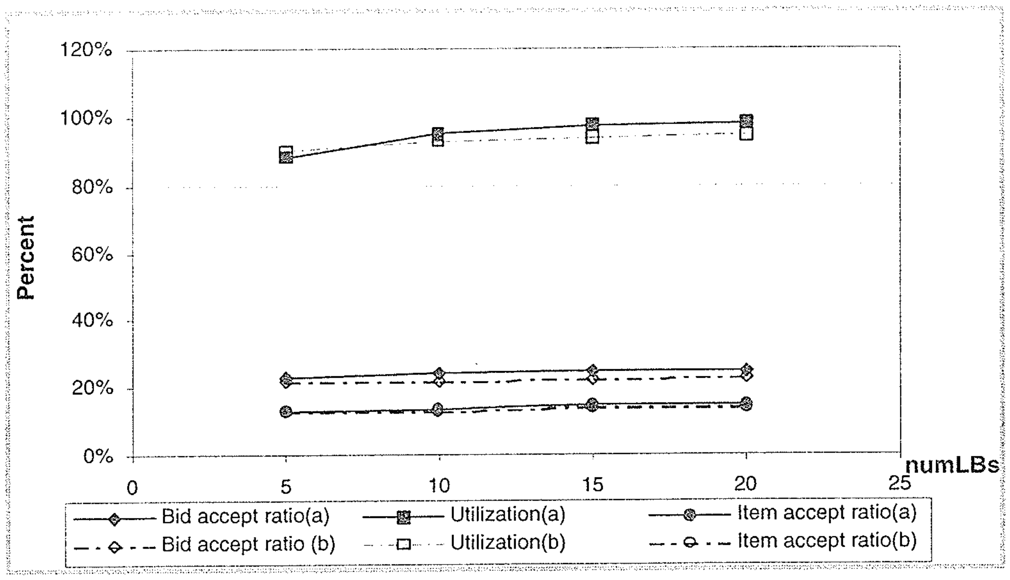


Figure 5.4: Figure Bid accept ratio, utilization, and item accept ratio versus the number of LBs. Simulation configurations: the number of bids for each LB: [200-300], the percentage of CoBids = 20%.

among the LBs caused by the CoBids and the bigger size bids in the NPA.

AvgLBs/CoBids is a measure used to indicate the average number of LBs in which a CoBids spans. The larger the AvgLBs/CoBids, the more constraints for resource allocation, hence, the worse the utilization. This is demonstrated in Figure 5.5. The simulation configurations are: the number of LBs=20, 200-300 bids are randomly generated by each local broker, 10% of which are CoBids. It is shown that the utilization of CFA decreases only slightly with the increase of AvgLBs/CoBids; while the utilization of NPA decreases dramatically with the increase of AvgLBs/CoBids, i.e., from 98% to 82% corresponding to the AvgLBs/CoBids is 2.3 to 8.5, respectively. The increase of AvgLBs/CoBids has almost no impact on both the bid accept ratio and bid accept ratio. In fact, the bid accept ratio and item accept ratio, are pretty close for the two approaches.

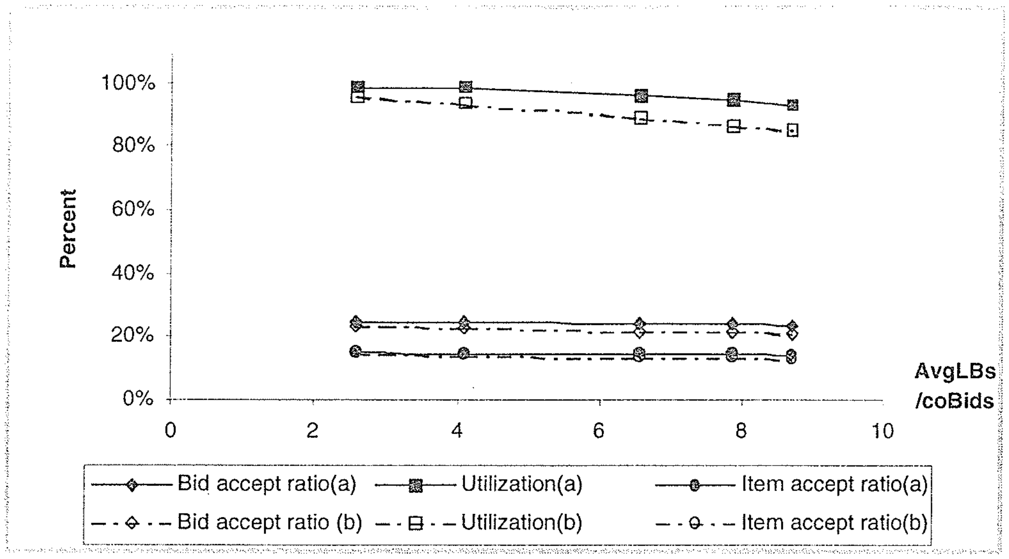


Figure 5.5: Figure Bid accept ratio, utilization, and item accept ratio versus AvgLBs/CoBids, Simulation configurations: the number of LBs = 20, the number of bids generated is uniformly distributed within [200-300], the percentage of CoBids =10%.

The simulation in Figure 5.6 uses different percentage of CoBids (=20%) than that in Figure 5.5. The purpose is to compare the performance measures when the percentage of CoBids doubles. The two approaches still exhibit very stable and close curves for the item accept ratio. The bid accept ratio, declines for both approaches, but is more stable in the CFA. The result of the utilization is very interesting. In Figure 5.5, the utilization for the CFA is observed to be slightly decreasing with the increase in the AvgLBs/CoBids, which is not the case when the percentage of CoBids increases to 20%. For the CFA, the utilization goes from 98% down to 80% when the AvgLBs/CoBids rises from 2.3 up to 8.4, correspondingly. No exception, the utilization for the NPA, too, declines dramatically with the increase of the AvgLBs/CoBids, and is 2% worse in average than that of the CFA.

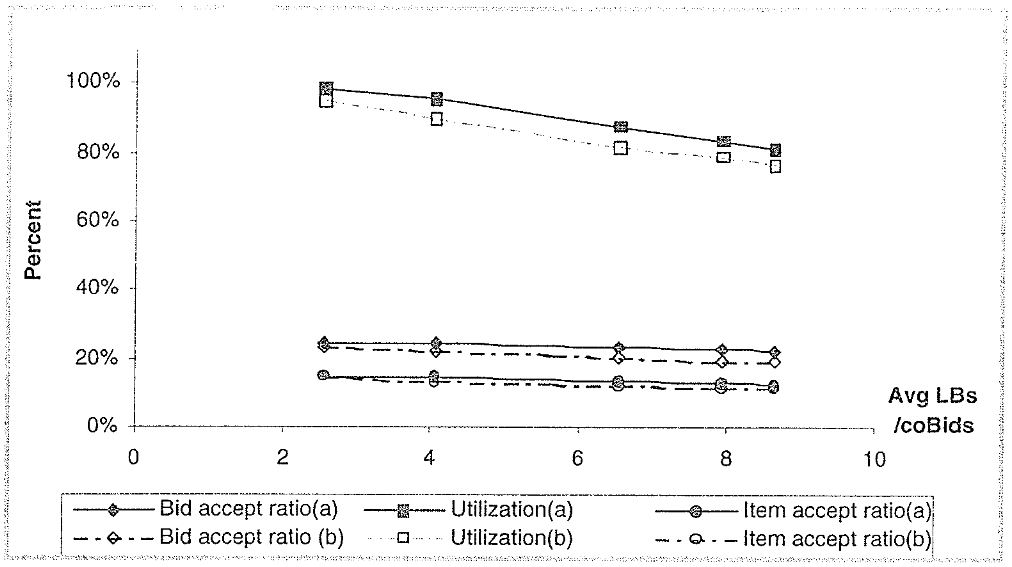


Figure 5.6: Figure Bid accept ratio, utilization, and item accept ratio versus AvgLBs/CoBids, Simulation configurations: the number of LBs = 20, the number of bids generated is uniformly distributed within [200-300], the percentage of CoBids =20%

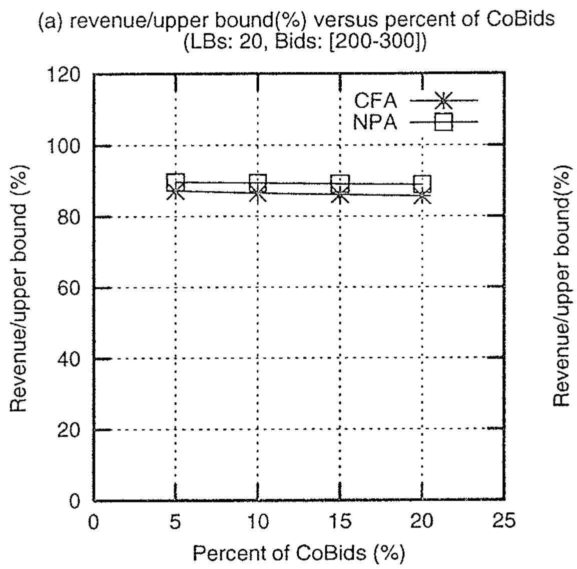
In summary, the CFA achieves better utilizations than the NPA for all simulation configurations. Intuitively, this means that considering the bids with more constraints first is a useful strategy, and could potentially lead to a better utilization. However, in cases where the constraints for resource allocation (i.e., the percentage of CoBids and the AvgLBs/CoBids) exceeds some threshold, the utilization will suffer for both approaches.

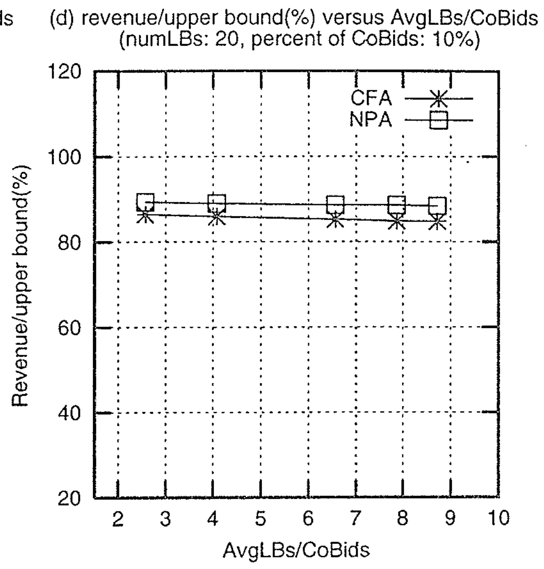
5.3.2 Simulation results: revenues

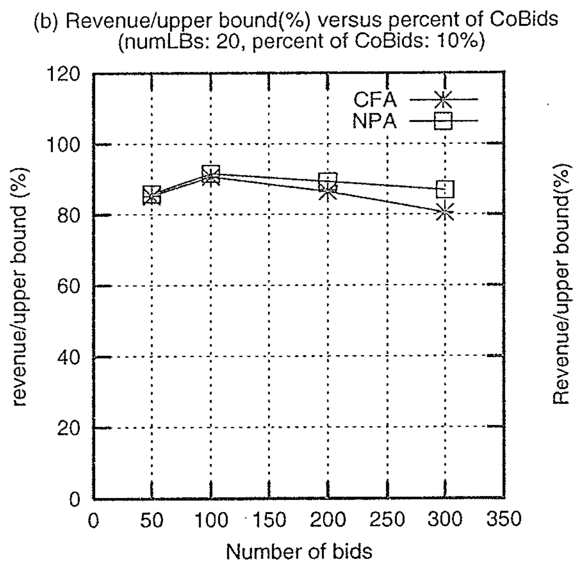
As mentioned earlier, revenue is an important measure in evaluating the market design. To compare and evaluate the revenues generated from different simulation configurations using different algorithms (CFA and NPA), we use percent of revenues/upper bound as a measure instead of using revenues directly. As different simulations differ in their maximum revenues, the percent of revenues/upper bound is more meaningful and intuitive to compare the capability of different algorithms in maximizing the revenue.

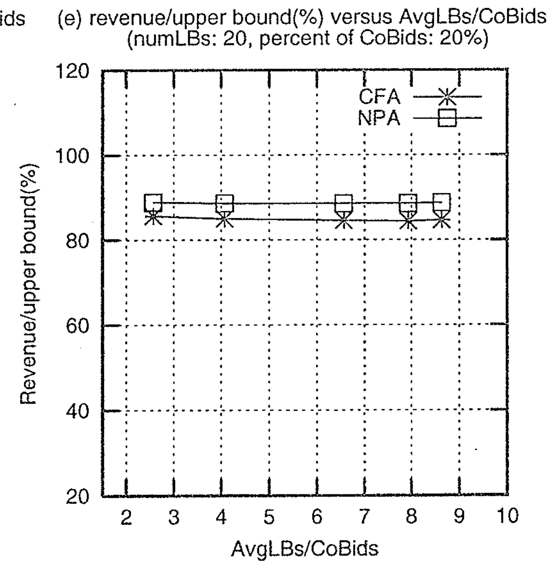
Figure 5.7 shows the performance of revenues under different simulation configurations. The simulation results of utilization measures showed in section 5.3.1 suggest that the CFA performs better than the NPA. However, the performance of revenues appears to be reversed. The NPA achieves very stable revenues ([86% – 90%]) under all simulation configurations, and outperforms the CFA by 3% in average. The revenues from the CFA, although worse than that of the NAP, are stable for most of the test cases, except when the number of bids increases dramatically, (as shown in figure 5.7 (b)). This indicates that when the competition becomes intensive, the CFA tends to make a decision of fixing some of the CoBids too early, which in turn affects the final selling price. This is reflected in a significant decline in revenue (from 90.65% decreases to 80.54% corresponding to the number of bids of 100 to 300). Due to the same reason, the revenue of the CFA also tends to be affected more by the AvgLBs/CoBids (test case *(d)* and *(e)*) than the NPA does.

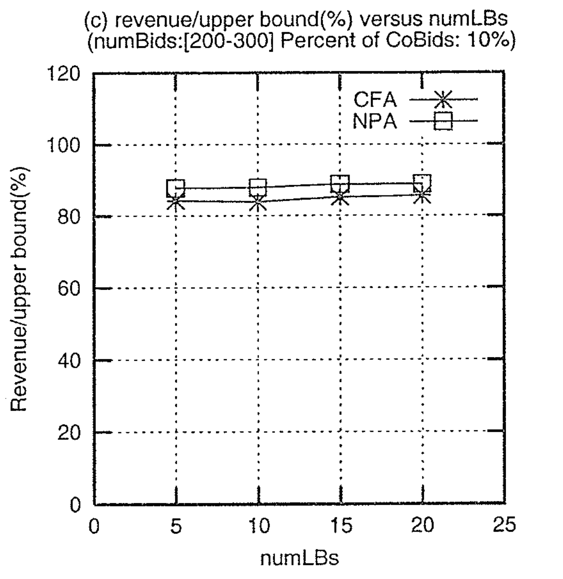
The simulation results of utilizations and revenues pose two sides of the CFA and the NPA. The CFA achieves better utilization while the NPA tends to give better revenue. The simulation results allow us to have a better understanding of the behaviors of the two approaches. Hence, depending on the degree of the importance of the utilization and the revenue, we can make decisions on which approach to use later on.


(a) revenue/upper bound(%) versus percent of CoBids
(LBs: 20, Bids: [200-300])
CFA
NPA
Revenue/upper bound (%)
Percent of CoBids (%)


(d) revenue/upper bound(%) versus AvgLBs/CoBids
(numLBs: 20, percent of CoBids: 10%)
CFA
NPA
Revenue/upper bound(%)
AvgLBs/CoBids


(b) Revenue/upper bound(%) versus percent of CoBids
(numLBs: 20, percent of CoBids: 10%)
CFA
NPA
revenue/upper bound (%)
Number of bids


(e) revenue/upper bound(%) versus AvgLBs/CoBids
(numLBs: 20, percent of CoBids: 20%)
CFA
NPA
Revenue/upper bound(%)
AvgLBs/CoBids


(c) revenue/upper bound(%) versus numLBs
(numBids:[200-300] Percent of CoBids: 10%)
CFA
NPA
Revenue/upper bound(%)
numLBs

Chapter 6

Conclusions and Future Work

This work introduces an online auction market framework and design that is targeted towards helping business, organization, government, and individuals to maximize the utilization of the already exisited computing and networking resources. The study focuses on several aspects: (1) auction market design (i.e., the design of rules, policies, and strategies for auction market); (2) solve the winner determination problem in a multi-unit combinatorial auction problem using heuristics; (3) design and perform computational experiments,. to assist in the algorithm design, and to help understand and study the behavior of the auction design; (4) evaluate the design systematically and analytically using performance measures such as revenue, efficiency, and utilization.

We consider the design and experiment of the CM contributes in three areas:

1. The Internet auction market design: this research extends the application of the online auction market to computing and networking resources, a new kind

of commodity exchanged on the Internet auctions. The CM is designed as a service provider framework for exchanging computing and networking resources. This differentiates itself from the normal computational economy researches, targeted at designing approaches for resource management. To fit better with the features and usage of the new commodity, this work initiates rules, methods, and concepts to leverage the usability and user experiences of the CM. It is worth mentioning that we demonstrated the methods and concepts (such as beat at least one rule, resource classification, any-bid, and CoBids, etc) to the public during TRLabs' TechForum held in October 2001, the feedback from the people we talked to showed that they liked the design and thought that the concepts and methods are very reasonable and really make bidding on a complex Internet auction easier and more convenient.

2. The multi-unit combinatorial auction study and experiment. This work designs "rate" to quantitatively measure the quality of the bid in terms of producing better revenues, and the experimental results have shown it to be effective in helping the heuristics achieve a good solution efficiently.

3. The resource management in wide-area networking systems. As stated in the introduction, although the focus of the CM differs significantly from a traditional resource management system, we consider the CM as a new application that can extend the usage of the traditional resource management. In fact, the combination of the two technologies exhibit a very promising and valuable opportunity in the future.

Although systematical experiments are conducted to help us accomplish a prototype for the CM, we realize that the study of the behavior of the auction market design is still very limited. As in many other studies on the MUCA problem, the experiments of the CM also suffer from the lack of realistic data[9, 17]. As indicated by the auction market design theory, empirical data offers a valuable complement to the study of the auction market design[3]. Therefore, implementing the prototype and putting it into practical use could allow us to obtain valuable empirical data for the problem. Based on the empirical data, we can further test and refine the auction market design. We identify this as a future work.

Bibliography

[1] A. Chavez and P. Maes. Kasbah: An agent marketplace for buying and selling goods. In *In Proceedings of the First International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology*, London, UK, April 1996.

[2] A.Chavez, A.Moukas, P. Maes. Challenger: A multi-agent system for distributed resource allocation. In *Agents '97 Conference Proceedings.* ACM, 1996.

[3] Alvin E. Roth, Axel Ockenfels. Last minute bidding and the rules for ending second-price auctions: Theory and evidence from a natural experiment on the internet. Technical report, Harvard University University of Magdeburg, 2000.

[4] C. A. Waldspueger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta. Spawn: A distributed computational economy. *IEEE Transactions On Software Engineering*, Vol. 18, No.2, February 1992.

[5] C. A. Waldspurger, W. E. Weihl. Lottery scheduling: Flexible proportional-share resource management. In *First published in Proceeding of the First Symposium on Operating Systems Design and Implementation.* Usenix Association, November 1994.

[6] C. DeMartini, A. M. Kawsnica, J. O. Ledyard, D. Porter. A new and improved design for multi-object iterative auctions. *Social Science Working Paper 1054*, 1999.

[7] D. F. Ferguson, C. Nikolaou, J. Sairamesh, Y. Yemini. Economic models for allocating resources in computer systems. In *Market based Control of Distributed Systems.* Ed. Scott Clearwater, World Scientific Press, 1996.

[8] I. Foskev and C. Kesselman. *The Grid: Blueprint for a New Computing Infrastructure.* Morgan Kaufmann, San Francisco, CA, 1999.

[9] K. Leyton-Brown, M. Pearson, and Y. Shoham. Towards a universal test suite for combinatorial auctions. In *2000 ACM Conference on Electronic Commerce (EC'00)*, 2000.

[10] K. Leyton-Brown, Y. Shoham, and M. Tennenholtz. An algorithm for multi-unit combinatorial auctions. In *17th National Conference on Artificial Intelligence*, 2000.

[11] M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research*, pages 1–23, 1 (1993).

[12] M. P. Wellman, J.K. Mackie-Mason, Sugih Jamin. Market-based adaptive architectures for information survivability. Technical report, http://www.darpa.mil/ito/psum1998, 1998.

[13] M. P. Wellman, P.R. Wurman. Real time issues for internet auctions. In *First IEEE Workshop on Dependable and Real-Time E-Commerce Systems (DARE-98)*, Denver, CO, USA, June 1998.

[14] M. P. Wellman, W. E. Walsh, P. R. Warman, J. K. MacKie-Mason. Auction protocols for decentralized scheduling. In *Eighteenth International Conference on Distributed Computing Systems*, Amsterdam, May 1998. Revised and extended version of "Some economics of market-based distributed scheduling".

[15] M. Stonebraker, P. M. Aoki, A.Pfeffer, A. Sah, J. Sidell, C. Staelin and A. Yu. Mariposa: A wide-area distributed database system. *VLDB Journal 5*, pages 48–63, 1 (Jan. 1996).

[16] M. Stonebraker, R. Devine, M. Kornacker, W. Litwin, A. Pfeffer, A. Sah, and C. Staelin. An economic paradigm for query processing and data migration in mariposa. *IEEE Computation Society Press*, pages 58–67, 1994.

[17] Matthew B. Wall. *A Generic Algorithm for Resource-Constrained Scheduling.* PhD thesis, Massachusetts Institute of Technology, 1996.

[18] C. Peter. The fcc spectrum auctions: An early assessment, 1997.

[19] Peter C. Cramton. The fcc spectrum auctions: An early assessment. *Journal of Economics and Management Strategy*, pages 6:3, 431–495, 1997.

[20] Peter R. Warman, Michael P. Wellman, William E. Walsh. A parametrization of the auction design space. *Journal of Economic Literature Classification Numbers: C70, D44*, 1999.

[21] R. Engelbrecht-Wiggans, M. Shubik, R. M. Stark. *Auctions, Bidding, and Contracting: Uses and Theory*, chapter 2, page 33. New York University Press, 1983.

[22] R. Gonen and D.J. Lehmann. Optimal solutions for multi-unit combinatorial auctions: Branch and bound heuristics. In *ACM Conference on Electronic Commerce*, pages 13–20, 2000.

[23] R. Raman and M.Living. Matchmaking: Distributed resource management for high throughput computing. In *7th IEEE Int'l Symposium on High Performance Distributed Computing*, July, 1998.

[24] H. G. Rotithor. Taxonomy of dynamic task scheduling schemes in distributed computing systems. *IEE Proceedings on Computer and Digital Techniques*, 141(1):1–10, Jan. 1994.

[25] S. D. Gribble, M. Welsh, R. von Behren, E.A. Brewer, D. Culler, N. Borisov, S. Czerwinske, R. Gummadi, J. Hill, A. Joseph, R.H. Katz, Z.M. Mao, S. Ross, and B. Zhao. The ninja architecture for robust internet-scale systems and services. In *Computer Networks (Special Issue on Pervasive Computing)*, 2000.

[26] T. Sandholm. Limitations of the vickrey auction in computational multiagent systems. In *Second International Conference on Multiagent Systems (ICMAS-96)*, pages 299–306, Keihanna Plaza, Kyoto, Japan, December 1996.

[27] T. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *International Joint Conference on Artificial Intelligence (IJCAI)*, pages 542–547, 1999.

[28] Richard Steinberg. A general combinatorial auction procedure. Technical report, University of Cambridge, The Judge Institute, Cambridge, England CB2 1AG, 2000.

[29] W. E. Walsh, M. P. Wellman, F.Ygge. Combinatorial auctions for supply chain formation. In *EC'00*, pages 17–20, Minneapolis, Minnesota, October 2000. Copyright 2000 ACM 1-58113-272-7/00/0010.

[30] Y. Amir, B. Awerbuch, R. S. Borgstrom. The java market: Transforming the internet into a metacomputer. Technical report CNDS-98-1, Department of Computer Science, the Johns Hopkins University, 1998.