IT Licentiate theses
2003-007

Market Based Programming and Resource Allocation

MARIA KARLSSON

UPPSALA UNIVERSITY
Department of Information Technology

UPPSALA
UNIVERSITET

Market Based Programming and Resource Allocation

BY
MARIA KARLSSON

June 2003

COMPUTING SCIENCE DIVISION
DEPARTMENT OF INFORMATION TECHNOLOGY
UPPSALA UNIVERSITY
UPPSALA
SWEDEN

Dissertation for the degree of Licentiate of Philosophy in Computer Science
at Uppsala University 2003

Market Based Programming and Resource Allocation

Maria Karlsson
Maria.Karlsson@msi.vxu.se

Computing Science Division
Department of Information Technology
Uppsala University
Box 337
SE-751 05 Uppsala
Sweden

http://www.it.uu.se/

ISSN 1404-5117
Printed by the Department of Information Technology, Uppsala University, Sweden

Abstract

The subject of this thesis is the concept of market-oriented programming and market protocols. We want to solve an allocation problem where some resources are to be divided among a number of agents. Each agent has a utility function telling how much the current allocation is worth for it. The goal is to allocate the resources among the agents in a way that maximizes the sum of the utilities of all agents. To solve this problem we use the concept of markets to create mechanisms for computational implementation.

To achieve the advantages of market oriented programming, we have to consider the conceptual view of the problem a main design issue. We want to investigate the possibilities to build computationally effective mechanisms which maintain the intuitive, easy-to-understand structure of market-based approaches. In the first paper we look at two examples from the literature and show that conceptual improvements of the approaches will make agent behavior more realistic. This will also make the examples fit into a more general theory. In the second paper we create a market mechanism for handling combinatorial markets. The mechanism includes an auction, where each iteration runs in polynomial time. The mechanism shows good performance when the number of resources is relatively small compared to the number of agents.

Contents

1

Summary

The subject of this thesis is the concept of market-oriented programming and market protocols. We want to solve an allocation problem where some resources are to be divided among a number of agents. Each agent has a utility function telling how much the current allocation is worth for it. The goal is to allocate the resources among the agents in a way that maximizes the sum of the utilities of all agents. To solve this problem we use the concept of markets to create mechanisms for computational implementation. There are a number of advantages of this approach:

- Previously developed theories from mathematical economics can be used as guidelines for modelling computational solutions to resource allocation problems.
- Markets and auction mechanisms are easy to understand and based on everyday experience. In this way, the mechanisms of the algorithm should be easily explainable to anybody.
- Market approaches enable a natural decomposition of the problem, which is well suited for distributed environments.

To achieve the advantages above, we have to consider the conceptual view of the problem a main design issue. We want to investigate the possibilities to build computationally effective mechanisms which maintain the intuitive, easy-to-understand structure of market-based approaches. Even though in all the examples of this thesis the system has full control over agent behavior, it is therefore important that they act in a realistic way. Furthermore, this is a necessary condition if we want the methods to be useful in the case where computational agents can be replaced by humans.

Paper I

In this article two examples of market-based approaches from the literature are investigated. We argue that even though the authors claim that their ap-

proaches are market-based, these two examples lack important concepts needed to achieve all the advantages of market-based programming. The important issue is that the methods need to have *realistic agent behavior* in order to be based on everyday experience. We propose conceptual improvements of the models and show that the improved models fit into a more general theory. The theory is defined and it is shown that the specific theorems of the examples now can be stated as more general theorems.

Main points and contributions:

- Since one of the main advantages of market-oriented programming is a pedagogical one, it is very important that the conceptual considerations are emphasized.
- In the first example, by Kurose & Simha, we show that introducing the concept of money makes the agent behavior more realistic. We also point out that changing from a *resource-oriented* to a *price-oriented* mechanism will have computational advantages.
- For the example of Bertsekas we create variants of the original model that better correspond to the proposed agent behavior from a conceptual point of view. Theoretical discussions and a practical implementation suggest that there could be a trade-off between economic realism and computational efficiency.
- We present general theory from the literature and enhance the theory to the case of *non-separable* utility functions. We then show how the examples fit into this general theory.

Paper II

This article focuses on combinatorial auctions. In a combinatorial auction the utility of an agent is dependent on its allocation of *all* the commodities. These kinds of dependencies between resources make the problem hard to solve.

We build a market mechanism to handle the case of an iterative, double-sided combinatorial auction for divisible commodities. Since the problem of deciding which bids to implement in general is $\mathcal{NP}$-hard, we create a mechanism which finds an approximate solution in polynomial time. On a very high level the mechanism could be stated as follows:

1: Given an initial allocation and an initial set of prices.
2: **repeat**
3: All agents are allowed to put in combinatorial bids.
4: The mechanism selects bids to be implemented.
5: The selected bids give rise to a new allocation and a new set of prices.
6: **until** no more surplus increment can be found.

To be able to perform step 4 in polynomial time, we allow the bids to be implemented in part. In this way we can formulate the selection problem as a linear programming problem to be handled by a commercial lp-solver.

Main contributions:

- The main principles of the created market protocol are simple and easy to understand, based on everyday experience.
- Each iteration of the mechanism takes polynomial time.
- The implementation of the mechanism shows good behavior for the tested instances.

2

Acknowledgements

The road to this thesis has been long and winding so there are many people I would like to thank.

- My supervisors Arne Andersson and Fredrik Ygge. You got me started, you believed in me. You have shared your knowledge with me, you gave me advice and your time. Thank you!
- My local supervisor Joakim Nivre. You have been my "academic advisor", you read my papers and gave fruitful comments. You also had the great ability to make me believe in myself again when my self-confidence was gone.
- To former and present members of the School of Mathematics and Systems Engineering at Växjö University. Some of you have contributed as my colleagues by discussing my research, arranging financial funding or helping me with LaTeX. Some of you are my friends and you have made my everyday life enjoyable during lunches and coffee breaks, but also on occasions out of work like pub evenings, dinners, "kräftskivor", barbecues and other parties. We have also been watching a lot of interesting movies together. I will not mention any names; I hope each and every one of you is aware of your contributions. Thank you!
- No rule without an exception; I like to send a special thanks to Ulrika Thörnqvist. I would not have been here if it were not for you. You opened the doors.
- To my friends outside the university, especially Matilda, Gunilla, Kalle and Christina. To my family and relatives. You have always been supportive and you believed in me. Thank you!

3

Papers

Market-based Approaches to Optimization

Maria Karlsson[†], Fredrik Ygge[‡], and Arne Andersson[‡]

[†]Uppsala University and Växjö University
School of Mathematics and Systems Engineering
Växjö University
SE - 351 95 Växjö, Sweden
www.msi.vxu.se/~mka
Maria.Karlsson@msi.vxu.se

[‡]Trade Extensions and Uppsala University
Computing Science Department
Information Technology
SE 751 05 Uppsala, Sweden
www.csd.uu.se/~{ygge,arnea}
{ygge,arnea}@csd.uu.se

Abstract. The use of markets has been proposed to a number of resource allocation/optimization problems, as such approaches often have a number of conceptual advantages. However, most examples found in the literature are rather ad hoc.
In this article we present a general definition of what constitutes a market-oriented approach to optimization. We demonstrate how this general framework can be used to conceptually improve two well-known approaches from the literature, and discuss computational properties of the different approaches. We also present some general theory and show that the theory of the two approaches under investigation are special cases of this theory.

1 Introduction

The use of markets has been proposed for different resource allocation/optimization problems. The main arguments found in the literature for applying market to these types of problems are:

- The numerous similarities between economic systems and distributed computer systems suggest that models and methods previously developed within the field of mathematical economics can serve as *blueprints* for engineering similar mechanisms in distributed computer systems [1].
- Auction algorithms are highly intuitive and easy to understand; they can be explained in terms of economic competition concepts, which are couched on everyday experience [2].
- Market approaches enable a natural decomposition, both from a software engineering perspective as well as from a computational perspective [3, Chapter 15], [4].

- Market approaches are very flexible in that they allow for ongoing addition and deletion of agents. No global changes are required—merely the demand/supply relation is altered [3, Chapter 15].
- Markets are informationally efficient in terms of information dimensionality [5], and the abstractions used are the most natural ones for the user [3, Chapter 15].
- The introduction of trading resources for some sort of money enables evaluation of local performance and valuation of resources, so that it becomes apparent which resources are the most valuable and which agents are using the most of these [3, Chapter 15].

This article focuses on the presentation of two approaches to optimization of the literature, based on some market abstractions. Kurose & Simha [1] investigate a file allocation problem. In this work the agents report their marginal utility (for having a certain amount of storage) and its derivative to an auctioneer, which reallocates the resource using a resource-oriented Newton-Raphson algorithm until an equilibrium has been reached in which the marginal utilities of all agents are the same. This procedure leads to an optimal allocation and is reported to be computationally efficient.

The assigning problem of allocating n objects to n users has been investigated by Bertsekas [2]. Each user has a valuation of each object, and cannot be assigned more than one object. It is reported that each user can be seen as an economic agent, and it is shown how an auction (which essentially is an English auction) results in an equilibrium. The assumed agent behavior is that an agent bids for the object that is most profitable given the agent's valuation of the objects and the current prices. The bidding increment depends on the valuation and the price of the most preferred object in relation to the valuation and the price of the second most preferred object. When the equilibrium is reached, the allocation is no more than a limited term away from being optimal.

Even though the two examples use ideas that are inspired by microeconomics, they rely on rather unrealistic agent strategies. Therefore we argue that neither of these approaches can be seen as fully market-oriented. For developing this argument, we give a general definition of what a market-oriented approach to optimization is, and show how the approaches under investigation can be conceptually improved to fit the more general framework. We also present some general theory (which interestingly also applies to non-separable resource allocation problems) and show how the rather ad-hoc theory of the examples constitute special cases of the general theory. Thus, the aim of the paper is to point out the pedagicical advantages of markets as important. We want to emphasize the concepts that support this view even though we show that this migth result in a loss of computational efficiency.

The article is organized as follows. In Section 2 we shortly describe the basic properties of market-oriented programming. In Section 3 the two examples of market-based applications are described in more detail. In Section 4 we generalize the theory, while Section 5 includes discussion and conclusions.

2 Market-oriented programming

Wellman [6] has introduced market-oriented programming as a general approach to deriving solutions to distributed resource allocation problems by a computational economy. Based on the models introduced by Wellman, we require that, for an optimization approach to be considered market-oriented, it must at least fulfill the following basic requirements:

- There must be a well defined *market mechanism* which includes some notion of prices (which often are expressed in terms of some monetary unit). The market mechanism is the rules for how the negotiations and trade is performed among the participating agents and hence tells how certain commodities can be traded for certain other commodities.
- There must be some arguments for why the *agent strategies are reasonably realistic*, given the market model. That is, from some model of information available for the different agents and a well-defined model for how they interact (the market mechanism), the strategies must be consistent with the agents' attempt to maximize utility, given bounded rationality.

We are used to different market mechanisms from our daily experience. At least in the industrial countries, *fixed price* markets are the most frequent. In a fixed price market, the customer faces preset prices for different commodities and has only the choice whether to buy or not. This type of markets is however of limited interest for application to optimization problems. Efficient market-oriented optimization must generally be based on a *dynamic price* market mechanism. In a dynamic price market the agents do not only chose which commodities to buy/bid for, but also at what prices.

Another important distinction is whether the commodities can be traded for in *discrete* or *continuous* amounts. An example where only discrete amounts make sense is an auction for flight tickets. Many other markets, however, allow for continuous amounts to be traded. Examples of such markets are markets for electricity, bandwidth and pollution rights.

An example of a dynamic price mechanism for discrete amounts is the English auction. In its most simple form, the auction consists of an object to be sold, an auctioneer (which keep track of the currently highest/winning bid), and a number of bidders. The auction may start at some minimum reserve price. A bidder can place a new bid by bidding at least ϵ above the currently winning bid. When no agent wants to make any further bid the auction stops. The objects are allocated to the agents with the highest bid on that object at the price reported by the winning bid. This type of market mechanism is very common and can, not least, be seen at a number of Internet sites.

An example of a dynamic price mechanism for continuous amounts is equilibrium (or clearing) markets [6]. An equilibrium market establishes a price at which supply meets demand. Each agent then typically declares a *demand/supply function*, telling how much it wants to buy/sell at different prices. The role of the auctioneer is then to search for the price such that all demand matches all supply. When the price is found the search terminates and the resources are

reallocated as described by the bids and the clearing price. This type of markets is less common, but one example of a market based on these principles is the main power market in the Nordic countries, NordPool [7].

Below we will illustrate both the use of discrete and continuous market mechanisms in the context of optimization problems.

Since we in this paper deal with the modelling of optimization problems as markets, we assume the designer to have full control of agent strategies. This means that even unrealistic agent behavior (like in the algorithm by Bertsekas [2]) can be considered for computional reasons. However, unrealistic agent behavior is a main conceptual flaw. For the concept of market-oriented programming to achieve the advantage of being easy to understand, the agents' behavior should be based on everyday experience. This paper will show that the conceptual gains of making the behavior more realistic can result in a loss in computional efficiency. We consider the trade-off between these two concepts an interesting question.

If the model is to be used in systems where the designer does not have control over agent strategies, then computional properties etc. must be demonstrated using realistic agent strategies.

However, we make the simplifying assumption that agents do not speculate about how the behavior of themselves or of other agents will affect the market. This is resonable if the market is of sufficient size and/or uncertainty is sufficienlty large [10, 11].

3 Examples of market-based optimization

3.1 The file allocation problem

Kurose & Simha [1] consider the file allocation problem in a distributed computer system. One file is fragmented into records so that it can be distributed to the different computers in a network. The fragmentation is done in a way that minimizes the communication costs and the average processing delay associated with a file access.

The system consists of n nodes interconnected through a communication network. The network is assumed to be fully connected. The processes running at each of the nodes generate accesses (queries and updates) to the file resource. If a process generates an access request which can not be satisfied locally, the access is transmitted to the node in the network that holds the requested information.

Somewhat more formally, the problem can be described as follows. Let x_i be the fraction of the file resource stored at node i. The goal is then to find the optimal $(x_1, x_2, \ldots, x_n)$. The overall expected cost of access to the file system is

given by (with variable definitions as described in Appendix A)

$$\begin{aligned} C &= \sum_{i \in N} (\text{cost of access to } x_i)\text{prob}(\text{accessing } x_i) \\ &= \sum_{i \in N} (C_i + KT_i)x_i \\ &= \sum_{i \in N} \Big(C_i + \frac{K}{\mu - \lambda x_i} \Big) x_i. \end{aligned}$$

In order to adapt to microeconomic theory, Kurose & Simha choose the equivalent problem formulation of maximizing the negative of the cost function. Thus, the utility function $U = -C$ is to be maximized.

Kurose & Simha present two approaches to the problem. The first approach starts with an arbitrary initial feasible allocation. Each node i computes its marginal utility, evaluated at the current allocation. Then each node transmits this value to a central node, which computes the average value. The result is broadcast back to the individual nodes and the file resources are reallocated. Nodes with marginal utility below average utility gets a proportional decrease in file resource, while nodes with marginal utility above average gets an proportional increase. These steps are iterated until the termination condition is met, i.e. the algorithm terminates then the differences between the marginal utilities of the nodes is below some ϵ for all pair of nodes. At each reallocation the algorithm must make sure that no node gets a negative allocation.

Let U_i' be $\partial U(x_1, \ldots, x_n)/\partial x_i$. Furthermore, let A be the set of nodes that could participate in a reallocation without getting a negative allocation. (For a description on how the set A is constructed, see Appendix B.) Now, we can state this algorithm more formally:

1. Initialization. An arbitrary feasible allocation $x_i, i \in N$ is made.
2. Iteration.

DO - Each node $i \in N$ calculates U_i' evaluated at the current allocation and sends U_i' and x_i to the designated central agent.

- The central agent computes the change in allocation for each node by:

$$\Delta x_i = \alpha \Big(U_i' - \frac{1}{|A|} \sum_{j \in A} U_j' \Big), \qquad \forall i \in A \tag{1}$$

where α is the stepsize parameter. $\forall i \notin A \Rightarrow \Delta x_i = 0$.

- Each node $i \in A$, sets $x_i = x_i + \Delta x_i$.

UNTIL $|U_i' - U_j'| < \epsilon, \forall i, j \in A$.

An alternative approach is to let the reallocation depend on the second partial derivatives. Then equation (1) in the algorithm above is replaced by

$$\Delta x_i = \alpha k_i \Big(U_i' - \frac{\sum_{i \in A} k_i U_i'}{\sum_{i \in A} k_i} \Big), \qquad \forall i \in A \tag{2}$$

where $k_i = 1/|\partial^2 U/\partial x_i^2|$ and the set A is constructed in a way similar to the first algorithm, see Appendix B for details.

Theorem 1. *Both algorithms have the following properties:*

- *When the algorithm converges the allocation is optimal.*
- *For sufficiently small values of the stepsize parameter, α, the algorithm converges to the optimal file allocation.*
- *Each iteration of the algorithm results in a strict increase of the systemwide utility.*
- *The algorithm maintains a feasible file resource allocation at each iteration.*

The two first properties were proved by Heal [8] while the two remaining ones were proved by Kurose & Simha [1].

Conceptual improvements Despite the title of their paper ("A Microeconomic Approach to Optimal Resource Allocation") Kurose & Simha's approach is—apart perhaps from the notions of utility and marginal utility—not very microeconomic. First, the algorithm of computing an allocation such that the marginal utility of each agent is the same for all agents can not really be called a market mechanism. To motivate a similar behavior the concept of money is needed. Secondly, there is no motivation for why an agent should truthfully report its marginal utility to the auctioneer.[1] As we will see below, there is a very natural way to formulate this approach as a proper market, giving a number of conceptual improvements.

Market mechanism We use a *clearing* or *equilibrium market* as market mechanism, c.f. [6, 3]. In an equilibrium market, a price is established such that supply meets demand. This computation is based on a demand function, $z(p)$, which represents an agent's requested *change* in allocation at different prices, further described below. (A supply is represented by a negative demand.) Given the demand function for each agent, $z_i(p)$, the clearing (or equilibrium) price is the solution to

$$\sum_{i=1}^{n} z_i(p) = 0, \tag{3}$$

where $z_i(p)$ is the demand function of agent i.

We now turn our attention to how $z_i(p)$ is determined.

Agent strategy Let $U_i(x_i)$ represent U's dependency on x_i, i.e. U as a function of x_i given some allocation to all nodes except i. Given $U_i(x_i)$, we introduce a utility function for each agent defined by

$$u_i(x_i, m_i) = U_i(x_i) + m_i, \tag{4}$$

[1] Admittedly, we also made mistakes similar to the ones made by Kurose & Simha in some of our earlier work, e.g. [9]

where m_i represents some "money". In microeconomics, x_i denotes the allocation after a trade. The initial allocation (called endowment) is denoted e, and the difference between the initial and the final allocation is denoted z, i.e. $z = x - e$. The local optimization problem of an agent is then to maximize $u_i(x_i, m_i)$ subject to that what is bought (z_i) must be paid for in m_i. Since the monetary term is linear we can hence write the optimization problem for an agent as

$$\max_{z_i} U_i(e_i + z_i) - p \cdot z_i.$$

As U_i is a concave function, the demand as a function of the price, $z_i(p)$, is obtained from[2]

$$\begin{array}{l} \frac{\partial U_i(e_i+z_i)-p\cdot z_i}{\partial z_i} = 0 \Leftrightarrow \\ U_i'(e_i + z_i) = p \Leftrightarrow \\ (U_i')^{-1}(p) = e_i + z_i \Leftrightarrow \\ z_i(p) = (U_i')^{-1}(p) - e_i. \end{array} \tag{5}$$

In the Kurose & Simha case we get[3]

$$z(p) = \frac{1}{\lambda}\left(\mu - \sqrt{\frac{\mu K}{-C_i - p}}\right) - e_i.$$

Discussion From Equation (3) and Equation (5) we see that the competitive equilibrium is equivalent to the outcome of the Kurose & Simha algorithm. Hence, by formulating the problem properly we have arrived at an approach (with equivalent allocations), which rightfully can be called market-oriented; it is based on a well-defined market mechanism and assumed realistic agent strategies given the mechanism. This gives some conceptual advantages. For example, it is fairly easy for most people to understand the principles: a number of agents make a trade-off between their usage of a file resource and money, and the resource is reallocated among the agents in such a way that supply meets demand.

Computational considerations The most straightforward way to implement the above mechanism is to let every agent submit a (sampled) demand function ($z(p)$ from Equation (5)), compute the clearing price (p from Equation (3)), and assign the resource in accordance with the bids. For example, if agent one submits $z(p) = 1/p - 4$ and agent two submits $z(p) = 1/p$, the clearing price is $\frac{1}{2}$ and 2 units is sold by agent one to agent two for 1 ($p \cdot z$) unit of money.

The search for the clearing price can be performed in many different ways of which the bisection method [12] (or binary search) is among the simplest, but often sufficiently fast in comparison to e.g. communication delays. More efficient algorithms include standard Newton-Raphson methods [12], e.g.

$$p^{it+1} = p^{it} - \alpha \frac{z(p^{it})}{z'(p^{it})}, \tag{6}$$

[2] This formulation apparently takes the price as given, even though an agent's choice actually has an impact on the prices.

[3] For the definitions of the different variables, see Appendix A.

where it is an iteration index and α is a step size. In heavily distributed environment it often also pays off to use some hierarchical algorithm [13]. One important design issue of all Newton-Raphson methods is the management of the step-size, here denoted α. Kurose & Simha derive a step-size for which the algorithm converges. This is however unnecessary since there are algorithms that automatically adjust the step-size through standard backtracking [12]. This does not only have the advantage of relieving the developer from determining the step-size at design time, but it also speeds up convergence significantly when the algorithm approaches the solution.

In the above, the price was used as the free search parameter. This is commonly referred to as a price-oriented method. However, as seen from Equation (1) and Equation (2) Kurose & Simha use a different method, commonly referred to as a *resource-oriented* method. In the Kurose & Simha algorithms the *allocations* (i.e. resources) are updated until all marginal utilities are equal. There is also a natural market interpretation of these algorithms: at the current allocation, ask the agents how much they are willing to pay for an additional infinitesimal amount of resource. Then the auctioneer tries different reallocations until the price that every agent is willing to pay for an infinitesimal amount of resource is the same. A standard resource-oriented Newton-Raphson approach [12] formulated in market terms looks like [14]

$$z_i^{it+1} = z_i^{it} - \alpha \frac{p(z_i^{it}) - \frac{\sum_{j=0}^{n} \frac{p(z_j^{it})}{p'(z_j^{it})}}{\sum_{j=0}^{n} \frac{1}{p'(z_j^{it})}}}{p'(z_i^{it})}, \tag{7}$$

where—as before—it is an iteration index and α is a step-size. As we see from Equation (5), $p(z_i) = U_i'(e_i + z_i)$, and hence Equation (7) is equivalent to Equation (2).

A detailed comparison between price-oriented and resource-oriented approaches can be found elsewhere [3], but some pros and cons of these two methods are outlined here:

- The price functions are closly related to the agents utility functions, and therefore the price function, $p(z)$, can be computed faster than the demand function, $z(p)$, in cases where the demand function can not be analytically derived.
- As resource-oriented algorithms merely reallocates the existing resource in each step, the reallocation of each iteration is feasible. Hence, gradually better allocations can be performed during the search for the optimal allocation. However, since for most networks the number of messages is more critical than the size of the messages, it is typically preferable to send the demand function as a sample vector. With such an approach, no (communication) iterations are required for computing the solution and the optimal solution will be very rapidly computed compared to the communication delays, and therefore there is no point in sending allocation updates to the agents during the search for the optimal solution.

- For resource-oriented algorithms he management of boundaries is a cumbersome issue. For each iteration, the auctioneer needs to make sure that no agents are assigned an allocation that is outside their boundaries. (This is often the result when the Newton-Raphson algorithms are used, particularly early in the search.) If the resource of one agent is outside its boundaries the excess (or shortage) must be reallocated among the other agents to obtain a feasible allocation.
- The resource-oriented algorithms are more complicated, cf. Equation (6) vs. Equation (7).
- Constant factors are higher for resource-oriented algorithms and therefore price-oriented approaches are significantly faster if the demand functions can be derived analytically.

The conclusion for the Kurose & Simha example is that—since the demand can be analytically derived—a price-oriented approach is highly preferred to a resource-oriented approach (as the one used by Kurose & Simha).

Discussion Above we have studied the file allocation approach by Kurose & Simha. It was described how this approach could be improved both conceptually and computationally.

The conversion from an approach that aimed at finding an allocation such that all partial derivatives are equal was replaced by a market mechanism that establishes a price such that supply meets demand. It was also described how the agents can be assigned proper utility functions, with which they made a natural trade-off between money and resource. Furthermore, given the market mechanism and the utility functions, an agent strategy leading to a globally optimal solution is realistic from an economics point of view (in bright contrast to the original approach, which gave no arguments for why the agents should report their true marginal utility). It is our belief that this type of improvement makes the system easier to understand and implement (as well as giving a number of other advantages as described in Section 1).

It was also shown that since the demand function can be derived analytically, a price-oriented approach can be used (instead of the resource-oriented approach proposed by Kurose & Simha). Since, price-oriented approaches are less complicated and more computationally efficient, they are typically highly preferable [3], and are recommended for the application at hand.

3.2 The assignment problem

Bertsekas [2] studies the *symmetric assignment problem*; matching n persons and n objects on a one-to-one basis, in such a way that the total benefit is maximized.

Original model The *market mechanism* is not explicitly given by Bertsekas [2]. However, there is one price, p_j per object, j, which is public to all bidders. Hence,

the mechanism is most reasonably interpreted as an open bid *English auction*, i.e. an auction in which the current winning bid for each object is public and the winner, at termination, pays p_j for object j.

The *agent strategy* is to bid the current running price of the most preferred object (given the market prices) plus the surplus (i.e. the agent's valuation of having the object minus the current running price) of the most preferred object minus the surplus of the second most preferred object. Somewhat more formally: Let $\mathcal{A}$ be the set of agent-object pairs (i, j) that can be matched. For each person i, the set of objects that can be matched with i is $A(i) = \{j|(i, j) \in \mathcal{A}\}$. Furthermore, let a_{ij} be the benefit of matching person i with object j. Then the surplus of matching object j to person i is $a_{ij} - p_j$, $(i, j) \in \mathcal{A}$. The surplus of the most preferred object, v_i is

$$v_i = \max_{j \in A(i)} \{a_{ij} - p_j\} \tag{8}$$

and the surplus of the second most preferred object, w_i, is[4]

$$w_i = \max_{k \in A(i), k \neq j} \{a_{ik} - p_k\}.$$

Hence, in each iteration an agent bids for object j (of Equation (8)) at price $p_j + v_i - w_i$.

The auction is run in iterations and, in each iteration, every object is assigned to the agent that bid the highest for that object. When every agent has been assigned an object the algorithm terminates. Unfortunately, this naive auction algorithm does not always work. If more than two objects offers maximum value for the bidder i the bidding increment is zero. Thus, several persons could compete for some objects without raising their prices and the algorithm would be trapped in an infinite loop.

The solution to this problem is to let every agent bid for its most preferred object, but at price $p_j + v_i - w_i + \epsilon$.

With this protocol, the following theorem holds.

Theorem 2. *A feasible assignment according to the auction algorithm together with some price vector is within $n\epsilon$ of being optimal [2].*

This means that when the algorithm stops the cost of the final assignment is within $n\epsilon$ of being optimal. An immediate conclusion is that if all benefits a_{ij} are integers and if $\epsilon < 1/n$ then the auction algorithm terminates in a finite number of iterations with an optimal assignment. This is true because if $n\epsilon < 1$ then any assignment within $n\epsilon$ of being optimal must be optimal. (This generalizes in a straightforward manner to any discretization, not only integers.)

With the mechanisms described above, if the problem is infeasible, the algorithm never terminates (since termination only occurs with a feasible assignment). When problems are not known to be feasible, detection of infeasibility is crucial. Bertsekas [2] shows several ways of handling this.

[4] If j is the only object in $A(i)$, then w_i is defined to be $-\infty$ (i.e. in practical computer implementations some large negative constant).

Even though the above gives a highly efficient and close to optimal algorithm, it has one main conceptual flaw: *the agent strategy is not realistic given the market mechanism*. Hence, it is not possible to view each person as an economic agent acting in its own best interest as suggested by Bertsekas [2]. The rational (dominant) strategy for a self-interested agent for the given market mechanism would be to bid p_j plus some minimal increment for the most preferred object (given the prices). For example, assume that we have two agents with valuations $a_{11} = 800$ and $a_{12} = 100$ for agent one and $a_{21} = 400$ and $a_{22} = 300$ for agent two. Then with the (sequential variant of) the above scheme agent 1 bids 700 for object one, and next agent two bids 600 for object two. However, with rational agent strategies for the given mechanism, the prices would have become 100 (optionally plus some minimal bid increment) for object one and zero for object two. (They would have only marginally overbid each other for object one until the price reached 100 and then agent two would bid for object two.) This algorithm would of course be very computionally inefficient. In the next sections we will discuss how a more realistic mechanism and strategies can be introduced at different computational expenses.

Mechanism variant 1 For every object, let there be two public prices, p_j^1 and p_j^2, which reflect the highest and the second highest bids for the object respectively. Further, require that a new bid for an object only is accepted if it is at least $p_j^1 + \epsilon$. Bids can not be withdrawn. When no more bids are received assign the objects to the highest bidder of the respective objects, but *charge* only p_j^2. (That is, use a second price auction [15].)

Now, with one exception that we return to below, the above agent strategy (with p_j^1 instead of p_j) is very realistic. An agent does not gain anything by bidding below $p_j^1 + v_i - w_i$, since it own bidding price does not affect what it will be charged. The only result of bidding too low is that there is a greater chance of losing the object.

The exception mentioned above is that an economic agent can be assumed to bid $p_j^1 + \max(v_i - w_i, \epsilon)$, rather than $p_j^1 + v_i - w_i + \epsilon$, i.e. it will not increase the bid more than what can be motivated by its preferences and the market mechanism—the latter saying that a bid must be an increase by at least ϵ. However, this typically has a relatively small effect on algorithm performance.

Hence, we have derived a new mechanism that better conforms to the market-oriented view, while making small sacrifices in terms of algorithm efficiency. This is in bright contrast to the original model, which has been referred to as an auction algorithm, motivated by the fact that it can be explained in competitive economics terms couched in our everyday experience, but where the agent strategies do not reflect our everyday experience of how people bid in standard auctions.

Though the new market mechanism indeed has made the assumed strategy much more realistic, it could still be in an agent's best interest to deviate from the above strategy. The problem is that if an agent with sufficiently high probability predicts that no other agent places a simultaneous bid, it can be beneficial to

bid *too high*. For example, assume that the sequential mechanism suggested by Bertsekas [2] is used. Then only one agent bids at the time, and it is a dominant strategy to bid ∞ for the most preferred object, as you then get to pay the current price for it and no other agent can outbid you. Of course, if agents can place simultaneous bids, there is an enormous risk in doing this—if another agent uses the same strategy it could be fatal. So if, for example, all agents update their bids synchronous in parallel, this need not be a problem.

In the next section we introduce an algorithm that eliminates this problem, but has lower computational efficiency.

Mechanism variant 2 For every object, let there be one public price, p_j^2, which reflects the second highest bid for the object. Further, require that a new bid for an object only is accepted if it is at least $p_j^2 + \epsilon$. When putting in a bid, the agent is told whether it is winning the object or not. When no more bids are received, assign the objects to the highest bidder of the respective objects, but charge p_j^2.

Then it is very realistic to assume that every agent bids $p_j^2 + v_i - w_i$ for its most preferred object j, since any price between p_j^2 and $p_j^2 + v_i - w_i$ will maximize agent i's surplus. There is no point for the agent to put in a higher bid than $p_j^2 + v_i - w_i$. If the unknown p_j^1 is higher than $p_j^2 + v_i - w_i$ then the agent will have to pay p_j^1 and thus it will be more profitable for him to bid for another object.

First some observations regarding the correctness of this mechanism and strategies: Since a new bid has to be at least $p_j^2 + \epsilon$, and p_j^2 of this scheme cannot be above p_j^1 of the above scheme, the algorithm will terminate. Finally, it is clear that the highest bid for each object has the same optimality features as p_j and p_j^1 above—if an agent wins an object it prefers it higher than all other objects given p_j^2, and then it also clearly prefers the object it is winning given the highest bid on every other object.

This approach has lower computational efficiency than the one above. This is not surprising: Generally we can not expect mechanisms that are based on the assumption that the agent only reveals the information that is most profitable for them to be as efficient as mechanisms when the agents' preferences are publicly available knowledge.

Discussion In this section we have discussed different aspects of Bertsekas' auction algorithm. We first discussed that no mechanism was originally given that motivated the proposed agent strategies. Then we introduced a new scheme (with slightly lower computational efficiency) that made the strategies more realistic and therefore made the entire approach more intuitive from an economics point of view. We showed that under certain circumstances we needed to modify the mechanism and agent strategies further to obtain an economically realistic model. The different mechanisms were implemented and tested for computational efficiency. The results, shown in Fig 1, indicates that methods that are

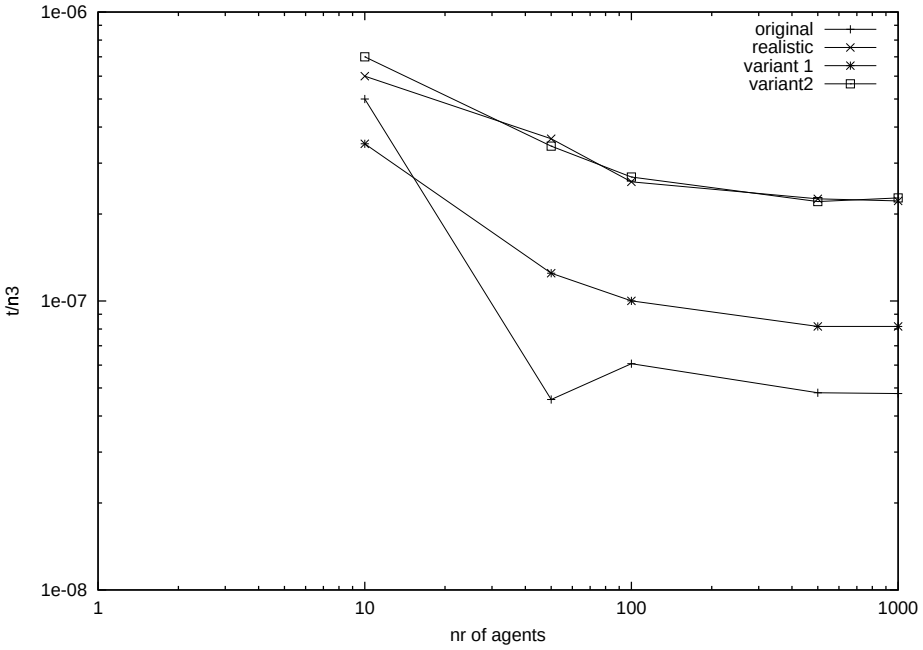

Fig. 1. Test of the computional efficiency of the different algorithms, comparing n, the number of agents, with the time of the computation, t, divided by n^3. *Original* is Bertsekas original model with bidding increment $v_i - w_i + \epsilon$, *realistic* has bidding increment ϵ, *variant 1* has bidding increment $\max(v_i - w_i, \epsilon)$ and finally in *variant 2* the new bid is $p_j^2 + \max(v_i - w_i, \epsilon)$. The agents valuations for the respective items are (independently) drawn from a rectangular distribution.

more realistic (like *realistic* and *variant 2*) are less effective than the less realistic methods (*original* and *variant 1*).

The main conclusion from this exercise is that there sometimes is a trade-off between computational efficiency and economic realism. In the case of a standard optimization problem with publicly available information, economic realism (with associated conceptual advantages) can of course be sacrificed for computational efficiency, by imposing more or less unrealistic strategies on the agents. However, if we assumed that we were to assign the objects to real self-interested persons, we must on the other hand assume that they act in their own best interest. Then we can not generally hope for obtaining efficient methods.

4 Generalizing the theory

The theory of the two examples [1, 2] examined in this article is very specific to the examples. In this section we show that properly formulating the optimization

problems as market problems not only has conceptual advantages; it also allows for the usage of some more general theory.

We first define a type of optimization problems to which the theory of this section can be applied.

Definition 1. *The maximization problem MP is given by*

$$\begin{array}{ll} \max_{\mathbf{x}} & \sum_{i=1}^{n} f_i(\mathbf{x}) \\ s.t. & \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X}, \end{array}$$

$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ik}]$, $x_{ij} \in S_{ij}$, $\mathbf{S}_i = [S_{i1}, S_{i2}, \ldots, S_{ik}]$, $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, $\mathbf{S} = [\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_n]$, $\mathbf{X} = [X_1, X_2, \ldots, X_k]$, $f_i : \mathbf{S} \mapsto \Re$, S_{ik} is any set of reals (e.g., all reals, all positive integers or $\{1.25, 2.77, 4\frac{1}{3}\}$) and X_i is the total available amount of commodity i. The problem is interpreted as a resource allocation problem with n agents and $f_i(\mathbf{x})$ as agent i's valuation of the allocation $\mathbf{x}$.

Next we give a proper market definition corresponding to the optimization problem of definition 1.

Definition 2. *With the notation from MP, the market M is a market of n agents and k commodities. All agents want to maximize their utility function defined by*

$$u_i(\mathbf{x}, m_i) = f_i(\mathbf{x}) + m_i, \tag{9}$$

$m_i \in \Re$, i.e. $u_i(\mathbf{x}, m_i)$ is quasi-linear with respect to m_i.

$\mathbf{X}$ is distributed as endowment among the agents. Let the endowment be $\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n]$, $\mathbf{e}_i = [e_{i1}, e_{i2}, \ldots, e_{ik}]$ and let the endowment of m be $\mathbf{m}^\circ = [m_1^\circ, m_2^\circ, \ldots, m_n^\circ]$. Let the market price of m be 1, i.e. let

$$m_i = m_i^\circ - \sum_{j=1}^{k} (x_{ij} - e_{ij}) p_j$$

where $p_j \in \Re$ denotes the respective market price for the k commodities.

Then we can state the following theorem.

Theorem 3. *Any Pareto-optimal allocation in M is a solution to MP.*

Proof. See Appendix C.

Hence, Theorem 3 implies that if we can devise a market mechanism that exploits all possible profitable reallocations, it will lead to a globally optimal solution. An important remark is of course that even though it sometimes is possible to find such a mechanism, the reallocation problem in general is $\mathcal{NP}$-hard [17], i.e. can be computationally intractable. Furthermore, the conditions that make problems hard to solve with markets, also tend to make them hard to solve by conventional means. As an example, concave objective functions (i.e. the functions $f_i(\mathbf{x})$ above) ensure existence of equilibrium and enable straight-forward market implementation. If the functions are non-concave however, the problem is less straight-forward to formulate in market terms and at the same time harder to solve conventionally [17].

Non-separable functions A utility function is said to be *separable* if the utility of one agent only depends on the same agent's allocation. The definitions and theorems above do also capture the case of non-separable functions. This deserves some further discussion.

Starting with Theorem 3, the notion of Pareto optimality is commonly discussed in the context of separable functions. However, the concept of Pareto optimality makes perfect sense also with non-separable functions. With separable functions Theorem 3 captures the fact that if there is another allocation with higher total utility, then there is always some agent (or set of agents) that is so much better off by changing its allocation, that it can compensate the agent (or set of agents) that gets worse off. With non-separable functions Theorem 3 instead tells that if there is another allocation with higher total utility, then there is always some agent (or set of agents) that is so much better off by changing its *or any other agent's* allocation, that it can compensate the agent (or set of agents) that gets worse off.

In a corresponding manner, also Theorem 4 below (and its corollaries) has a straight-forward interpretation in the non-separable case.

When generalizing the theory for the optimality of the assignment problem of Section 3.2, we have made the following interesting observation: When generalizing the problem to a high enough level, it becomes trivial. When having the general theorem, the theory of the investigated assignment problem (as well as some other theory of the literature) is obtained as trivial corollaries.

Theorem 4. *With basic definition from Definition 1, assume that we have a resource reallocation problem, where the current allocation* $\mathbf{e}$*, is to be reallocated to a new allocation* $\mathbf{x}$*. Furthermore, let there be a cost (that may be positive or negative) for each agent associated with moving from* $\mathbf{e}$ *to* $\mathbf{x}$*. Call this cost* $c_i(\mathbf{e}, \mathbf{x}_i)$*. Define the objective of the resource allocation as*

$$\begin{array}{ll} \max_{\mathbf{x}} & \sum_{i=1}^{n} f_i(\mathbf{x}) - c_i(\mathbf{e}, \mathbf{x}_i) \\ s.t. & \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X}. \end{array}$$

An upper bound on the objective is

$$\begin{array}{ll} \max_{\mathbf{x}} & \sum_{i=1}^{n} f_i(\mathbf{x}) - c_i(\mathbf{e}, \mathbf{x}_i) \\ s.t. & \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X} \end{array} \leq \begin{array}{l} \sum_{i=1}^{n} \max_{\mathbf{x}} f_i(\mathbf{x}) - c_i(\mathbf{e}, \mathbf{x}_i) \\ s.t. \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X}, \end{array}$$

Proof. The theorem is obvious; the maximized sum is clearly smaller or equal to the sum of the maximized terms.

Corollary 1. *If* $c_i(\mathbf{e}, \mathbf{x}_i) = \mathbf{p}(\mathbf{x}_i - \mathbf{e}_i)$*, then*

$$\begin{array}{ll} \max_{\mathbf{x}} & \sum_{i=1}^{n} f_i(\mathbf{x}) \\ s.t. & \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X} \end{array} \leq \begin{array}{l} \sum_{i=1}^{n} \max_{\mathbf{x}} f_i(\mathbf{x}) - \mathbf{p}(\mathbf{x}_i - \mathbf{e}_i) \\ s.t. \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X}, \end{array}$$

since $\sum_{i=1}^{n} \mathbf{p}(\mathbf{x}_i - \mathbf{e}) = 0$ *(i.e. all payments sum to zero, everything is transferred between the agents). In other words, if the only costs are the payments between the agents, the optimal solution is smaller than or equal to the sum of the utilities of the optimal choice for the respective agents, for any given price vector.*

Corollary 2. *By adding* $\sum_{i=1}^{n} f_i(\mathbf{e})$ *to the expression in Corollary 1, we get*

$$\begin{array}{ll} \max_{\mathbf{x}} \sum_{i=1}^{n} f_i(\mathbf{x}) - f_i(\mathbf{e}) \\ s.t. \quad \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X} \end{array} \leq \begin{array}{l} \sum_{i=1}^{n} \max_{\mathbf{x}} f_i(\mathbf{x}) - f_i(\mathbf{e}) - \mathbf{p}(\mathbf{x}_i - \mathbf{e}_i) \\ s.t. \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X}. \end{array}$$

That is, the total improvement can not be greater than the sum of the desired improvements of the respective agents at a given price vector.

From Corollary 2 the optimality of our version of the Kurose & Simha file allocation problem follows directly. We also see that the proof for upper bound on distance to global optimality for the assignment problem, Theorem 2, is a direct consequence of the corollary. So is the theorem on upper bound on distance to optimality by Walsh and Wellman [19].

There is another interesting comment to make about Corollary 2. If we generalize the notion of competitive equilibrium to properly incorporating non-separability, we will directly see that the competitive equilibrium is globally optimal. So, we generalize the definition of competitive equilibrium from "a price vector and allocation such that supply meets demand and at the given prices, no agent's utility can be increased with respect to those prices by changing its allocation (i.e. buy or sell the different commodities)" to "a price vector and allocation such that supply meets demand and at the given prices, no agent's utility can be increased with respect to those prices by changing its *or any other agent's* allocation". We observe that Corollary 2 captures the optimality of the generalized equilibrium.

Hence, we see that we have arrived at a very general theory, and that formulating the applications of Kurose & Simha and Bertsekas in proper market terms make them fit into a broader theoretical framework.

5 Discussion and Conclusions

In this article we have discussed market-oriented approaches to optimization. We have presented some general definitions of what a market-oriented approach to optimization is and argued that the basic requirements are that it should rely on some well-defined *market mechanism* and some *agent strategies* that are reasonably realistic given the market mechanism. Some rather general theory regarding the relations between standard optimization formulations and market formulations has also been presented.

To illustrate the general framework for market-oriented approaches to optimization, two examples from the literature—the microeconomic approach to the file allocation problem by Kurose & Simha [1] and the auction algorithm for the assignment problem by Bertsekas [2]—have been analyzed.

It was shown that a number of conceptual improvements could be made to the Kurose & Simha approach, making the model more "market-like" from the perspective of how people generally understand markets. Some computational discussions showed that the file allocation problem could be managed by a *price-oriented* approach rather than the *resource-oriented* approach presented

by Kurose & Simha. Generally, price-oriented approaches are simpler (easier to implement) and more computationally efficient when the demand functions can be derived analytically (as could be done in the analyzed case).

For the auction algorithm for the assignment problem, different mechanisms were discussed and reasonable strategies introduced. It was shown that the original approach does not reflect agent behavior in real auctions (as suggested by the motivation of the algorithm), but that there are more reasonable mechanisms and agent strategies with high computational efficiency. These mechanisms are hence also better suited for applications in real markets (i.e. when the different agents truly represent different interests).

In the case of the file allocation application, we showed that one could define market mechanisms and reasonable agent strategies that are highly efficient. However, there is a catch preventing straightforward application of these methods to real markets. The problem is the utility function. In Equation (4) and just above that equation, we simply assigned a utility function to each agent. However, in a market setting, this would not be the utility function of a self-interested agent (since the current utility function is based on the notion of global utility). The main problem is that the utilities are *non-separable*, i.e. dependent on the allocations of the other agents. In Section 4 we show that existing theory could be generalized to capture even this case.

One of the most interesting results from the comparison between different auction mechanisms for the assignment problem is that we found a trade-off between computational efficiency and economic realism. Whereas the economical efficiency aspects of strategic behavior are relatively well understood and described (see e.g. [20, 21] for good introductory overviews), the computational aspects of strategic behavior are much less so. The different variants above suggest that there may be an interesting connection that seems interesting to look into further.

References

1. J. F. Kurose and R. Simha. A microeconomic approach to optimal resource allocation in distributed computer systems. *IEEE Transactions on Computers*, 38(5):705–717, 1989.
2. D. Bertsekas. Auction algorithms for network flow problems: A tutorial introduction. *Computational Optimization and Applications*, (1):7–66, 1992.
3. F. Ygge. *Market-Oriented Programming and its Application to Power Load Management*. PhD thesis, Department of Computer Science, Lund University, 1998. ISBN 91-628-3055-4, CODEN LUNFD6/(NFCS-1012)/1-224/(1998) (Available from http://www.enersearch.se/ygge).
4. A. Th. Schreiber, J. M. Akkermans, and al. *Knowledge Engineering and Management*. The MIT Press, Cambridge, MA, 1999. In Press.
5. J. S. Jordan. The competitive allocation process is informationally efficient uniquely. *Journal of Economic Theory*, 28:1–18, 1982.
6. M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research (www.jair.org/)*, (1):1–23, 1993.

7. NordPool. The elspot market—The spot market, October 1998. (Available from www.nordpool.no).
8. G. Heal. Planning without prices. *Review of Economic Studies*, 36:346–362, 1969.
9. F. Ygge and J. M. Akkermans. Power load management as a computational market. In M. Tokoro, editor, *Proceedings of the Second International Conference on Multi-Agent Systems, ICMAS'96*, pages 393–400. AAAI Press, Menlo Park, CA, December 9–14 1996. (Available from www.enersearch.se/ygge).
10. D. J. Roberts and A. Postlewaite. The incentives for price-taking behavior in large exchange economies. *Econometrica*, 44(1):115–127, 1976.
11. T. W. Sandholm and F. Ygge. On the gains and losses of speculation in equilibrium markets. In *Proceeding of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI 97*, pages 632–638, August 23–29, 1997. (Available from http://www.enersearch.se/ygge).
12. W. Press, S. Teukolsky, W. Vetterling, and B. Flannery. *Numerical Recipies in C*. Cambridge University Press, 1994. Second Edition.
13. A. Andersson and F. Ygge. Managing large scale computational markets. In H. El-Rewini, editor, *Proceedings of the Software Technology track of the 31th Hawaiian International Conference on System Sciences (HICSS31)*, volume VII, pages 4–14. IEEE Computer Society, Los Alamos, January 1998. ISBN 0-8186-8251-5, ISSN 1060-3425, IEEE Catalog Number 98TB100216. (Available from http://www.enersearch.se/ygge).
14. F. Ygge and J. M. Akkermans. On resource-oriented multi-commodity market computations. *Autonomous Agents and Multi-Agent Systems*, 3(1), 2000. In press.
15. W. Vickrey. Counter speculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.
16. S. Bikhchandani and J. W. Mamer. Competitive equilibrium in an exchange economy with indivisibilities. *Journal of Economic Theory*, 74:385–413, 1997.
17. A. Andersson and F. Ygge. Efficient resource allocation with non-concave objective functions. Technical report, EnS 99:1, EnerSearch AB, 1999. Submitted for journal publication. (Available from http://www.enersearch.se).
18. A. Takayama. *Mathematical Economics*. Cambridge University Press, 1985.
19. W. Walsh and M. Wellman. Efficiency and equilibrium in task allocation economies with hierarchical dependencies. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI 99)*, pages 520 – 526. Morgan Kaufmann Publishers, Inc., July 1999.
20. R. B. Myerson. Bayesian equilibrium and incentive compatibility: An introduction. In L. Hurwicz, D. Schmeidler, and H. Sonnenschein, editors, *Social Goals and Social Organization*, chapter 8, pages 229–259. Cambridge University Press, 1985.
21. E. Muller and M. Satterthwaite. Strategy-proofness: The existence of dominant-strategy mechanisms. In L. Hurwicz, D. Schmeidler, and H. Sonnenschein, editors, *Social Goals and Social Organization*, chapter 8, pages 131–171. Cambridge University Press, 1985.

A The variables of the file allocation problem

- x_i is the fraction of the file resource at node i. Since there is only a single divisible resource, $\sum_{i=1}^{n} x_i = 1$. Assuming accesses are made on a uniform basis, x_i also represents the probability that a file access (from anywhere in the network) will be transmitted to node i for processing.
- λ_i is the average rate at which node i generates accesses to the file resource. The networkwide access generation rate is defined $\lambda = \sum_{i=1}^{n} \lambda_i$.
- c_{ij} is the communication cost of transmitting an access from node i to node j and transmitting the answer back to i from j. c_{ii} is taken to be zero.
- C_i is the average systemwide communication cost of making an access at node i. We take this as the weighted sum of the individual communication costs:
$$C_i = \sum_{j \in N} \frac{\lambda_j}{\lambda} c_{ji}.$$
- $1/\mu_i$ is the average service time for an access at node i. We will assume $\mu_i = \mu$ for all nodes i.
- T_i is the expected time delay associated with satisfying an access at node i. In terms of λ, x_i, and μ, we have:
$$T_i = \frac{1}{\mu - \lambda x_i}.$$
(μ is the average number of jobs processed at one node every time unit. λx_i is the average number of accesses to node i per time unit. Thus, $\mu - \lambda x_i$ is the decrease in the length of the queue of waiting jobs at node i per time unit and so $1/(\mu - \lambda x_i)$ is the expected time a job will have to spend in the queue at node i.)
- K is the relative importance of communication costs and access delays. K is here taken to be 1, meaning that communication costs and access delays are equally important.

B The set A of the file allocation problem

The construction of the set A is required to ensure that no node's allocation goes below zero in the reallocation process. The algorithm for computing A for the first derivatives algorithm, at each iteration:

1. For all i sort U_i'.
2. Set $A' = \{i | \text{node } i \text{ has largest } U_i'\}$.
3. do step 4. for each $j, j \notin A'$ in descending order of U_j'.
4. If j would receive a positive allocation x_j as a result of the reallocation defined by Equation (1) with $A = A' \cup \{j\}$, then set $A' = A' \cup \{j\}$.
5. Set $A = A'$.

If the second derivatives algorithm is used, Equation (1) is replaced by Equation (2).

C Proofs

Proof for Theorem 3 Let $\Delta f = \sum_{i=1}^{n} f_i(\mathbf{x}^b) - f_i(\mathbf{x}^a)$, $\sum_{i=1}^{n} \mathbf{x}_i^a = \mathbf{X}$, and $\sum_{i=1}^{n} \mathbf{x}_i^b = \mathbf{X}$. For any allocation $\langle \mathbf{x}^a, \mathbf{m}^a \rangle$ in M, $\langle \mathbf{x}^b, \mathbf{m}^b \rangle$ (where $m_i^b = m_i^a + u(\mathbf{x}^a, m_i^a) - u(\mathbf{x}^b, m_i^a) + \frac{\Delta f}{n}$) is a Pareto-improvement if $\Delta f > 0$. ($\sum_{i=1}^{n} m_i^b = \sum_{i=1}^{n} m_i^a$, because $\sum_{i=1}^{n} u_i(\mathbf{x}^a, m_i^a) - u_i(\mathbf{x}^b, m_i^a) = -\Delta f$.) In other words: if an allocation is not a solution to MP it is not Pareto-optimal in M. Thus, if an allocation is Pareto-optimal in M it is a solution to MP. □

A computationally efficient mechanism for complex market negotiations

Maria Karlsson[†] and Arne Andersson[‡]

[†]Uppsala University and Växjö University
School of Mathematics and Systems Engineering
Växjö University
SE - 351 95 Växjö, Sweden
www.msi.vxu.se/~mka
Maria.Karlsson@msi.vxu.se

[‡]Trade Extensions and Uppsala University
Computing Science Department
Information Technology
SE - 751 05 Uppsala, Sweden
www.csd.uu.se/~arnea
arnea@csd.uu.se

Abstract. The problem to be solved consists of a number of continuous dividable resources that are to be allocated to a number of agents. Each agent has a multi-dimensional utility function telling how much it is worth for it to have a certain (combined) amount of the resources. We want to use a market-based approach to allocate the resources among the agents in a way that maximizes the total utility. Since in our case, the utility of one resource is dependent on the agent's allocation of other resources this is not a trivial task.
We design an iterative, two-sided combinatorial auction to solve this problem. Although combinatorial auctions in general are $\mathcal{NP}$-hard, we create an iterative market mechanism that only requires polynomial computation complexity. Moreover, it is natural to use and our experimental evaluations indicate good behavior for instances of the problem where the number of agents is not too small relative to the number of resources.

1 Introduction

We are considering a market where k different resources are to be divided among n agents. Initially, all of the resources are divided among the agents as *endowment*. Each agent has a utility function telling how much it is worth for it to have different amounts of the resources. We want to use a market mechanism to allocate the resources among the agents in an optimal way, where the mechanism has the following properties:

1. The mechanism is natural and easy to use.
2. The involved computations are tractable.
3. The resulting allocation gives a high total utility.

If the utility of one resource for one agent is not dependent on the allocation of the other resources for that agent, the utility function of the agent consists of the sum of the utilities for each resource allocated to that agent. Hence, the utility functions are one-dimensional. If, in addition, the utility functions are concave the problem could be solved easily [1], [2]. If, on the other hand, the utility function of an agent depends on the allocation of all commodities for that agent, then more complicated mechanisms are needed, with higher time requirements [3], [4], [5], [6].

We want to create an iterative market mechanism that finds a good solution to this kind of allocation problem, with multi-dimensional utility functions. Since it is unrealistic to believe that agents will submit their complete utility functions to the mechanism, we want to use an auction where agents put in bids in an iterative fashion. The problem of selecting which bids to implement is in general $\mathcal{NP}$-hard [7], so we want to create an approximate optimum in polynomial time. Note that it is impossible to find an approximation algorithm that in polynomial time guarantees a certain upper bound on the approximation error [5]. Our aim is, however, to find an auction mechanism and corresponding algorithm that will be useful in practice.

When can a problem like this occur in practice? Consider an energy market, where different amounts of energy are to be distributed among a set of agents. The agents could be any energy consumers. For simplicity, in this example we assume them to be households. The utility functions of the agents should reveal different households' different preferences of energy consumption, e.g. some households may want a high indoor temperature while others prefer a cooler climate. Furthermore, some households may be prepared to pay very much to keep indoor temperature exactly at the preferred level, while it would be okay for others to deviate from their preferred level if this will decrease their energy costs. When the utilities of each agent are established, energy can be distributed in such a way that the utility over all agents is maximized.

Now, assume that different resources consist of energy at different hours. If agents (households) always could set up their utility functions as utilities for each hour, then the problem would be one-dimensional. If, on the other hand, their need for energy one hour is dependent on how much energy they get in another hour this would no longer be true. This situation could occur if for example an agent wants to run a washing machine that would take 1.5 hours. Then there is no need at all for the agent to have energy to run the washing machine only for one hour, to be able to accomplish anything he has to know that he will get energy for a longer period. If each resource consists of energy during one hour, the utility function of this agent would now be multi-dimensional. Furthermore, the agent may not want to run the washing machine if it would not also be able to run the tumble drier, which will create dependencies for energy for even more hours.

2 Market oriented programming and combinatorial auctions

The market mechanism presented here is well suited for *market oriented programming*. The concept of market oriented programming was introduced by Wellman [8], as a general approach to deriving solutions to distributed resource allocation problems by computational economy. By introducing the concept of money and modelling the problem as a market a number of advantages can be achieved. A more thorough discussion of these advantages can be found elsewhere [9], [1]; here we just mention a few for completeness:

- Previously developed theories from mathematical economics can be used as guidelines for modelling computational solutions to resource allocation problems.
- Markets and auction mechanisms are easy to understand and based on everyday experience. In this way, the mechanisms of the algorithm should be easily explainable to anybody.
- Market approaches enable a natural decomposition of the problem, which is well suited for distributed environments.

When using market oriented programming we require a well-defined *market mechanism*. The market mechanism is the rules for how the negotiations and trades are performed among the agents. The market mechanism is supposed to include some notion of money and prices. Another requirement is that we expect *realistic agent behavior* in the market. Given the market mechanism the agents are supposed to act in their own best interest, i.e. the agents should not bother about finding a global optimum. If each agent acts in its own best interest, the market mechanism should be constructed in such a way that this behavior still leads to the global optimum. Realistic agent behavior is a basic condition if the market is supposed to be intuitive and easy to understand. In addition, we want the market to be general enough to be able to handle the case where the computational agents are replaced by human agents, interacting with the computational market system. If humans are trading on the market we could expect *any* behavior, the only limitations being the rules set by the market mechanism.

When creating an auction mechanism for agents with multi-dimensional utility functions, we need the mechanism to be able to handle the combinatorial structure of the preferences. A *combinatorial auction* will be receiving bids with content "I want x_1 of resource k_1 only if I also get x_2 of resource k_2 and x_3 of resource k_3". The aim of the mechanism is to select which bids to implement, a process that in general is $\mathcal{NP}$-hard [7]. This could be done in just one step or the mechanism could allow the agents to rebid if they are not satisfied with the outcome of the first reallocation. In the latter case we have an *iterative auction*. These auctions are expected to run until an equilibrium is found where supply equals demand and every agent is satisfied with its allocation, given the current prices.

In many cases the auction consists of some amounts of the resources to be sold to the agents. In that case the auctions is *one-sided*. In other cases the agents are supposed to trade with each other, creating a market where agents could sell as well as buy the resources. We call this kind of market *two-sided*.

3 Problem formulation

The basic problem is to divide some resources among a number of agents via a sound market mechanism. In order for the mechanism to be economically efficient, the total utility of all agents is maximized when agents bid rationally.

Each agent has a utility function telling how much it is worth for the agent to have a certain amount of the resources. If we view the problem as a resource allocation problem that should be solved via a market mechanism (or via market oriented programming), we have the following optimization problem:

Definition 1. *The maximization problem MP is given by*

$$\begin{array}{ll} \max_{\mathbf{x}} & \sum_{i=1}^{n} f_i(\mathbf{x}) \\ s.t. & \sum_{i=1}^{n} \mathbf{x}_i = \mathbf{X}, \end{array}$$

$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{ik}]$, $x_{ij} \in S_{ij}$, $\mathbf{S}_i = [S_{i1}, S_{i2}, \ldots, S_{ik}]$, $\mathbf{x} = [\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n]$, $\mathbf{S} = [\mathbf{S}_1, \mathbf{S}_2, \ldots, \mathbf{S}_n]$, $\mathbf{X} = [X_1, X_2, \ldots, X_k]$, $f_i : \mathbf{S} \mapsto \Re$, S_{ik} is any set of reals, X_j is the total available amount of commodity j and $\mathbf{x}_i$ is the amounts of the commodities currently allocated to agent i. The problem is interpreted as a resource allocation problem with n agents and k commodities and $f_i(\mathbf{x})$ as agent i's valuation of the allocation $\mathbf{x}$.

Next we give a proper market definition corresponding to the optimization problem of definition 1.

Definition 2. *With the notation from MP, the market M is a market of n agents and k commodities. Introduce a new resource m, which can be considered to be "money". All agents want to maximize their utility function defined by*

$$u_i(\mathbf{x}, m_i) = f_i(\mathbf{x}) + m_i,$$

where $m_i \in \Re$ is agent i's current amount of resource m, i.e. $u_i(\mathbf{x}, m_i)$ is quasi-linear with respect to m_i.

$\mathbf{X}$ is distributed as endowment among the agents. Let the endowment be $\mathbf{e} = [\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_n]$, $\mathbf{e}_i = [e_{i1}, e_{i2}, \ldots, e_{ik}]$ and let the endowment of m be $\mathbf{m}^\circ = [m_1^\circ, m_2^\circ, \ldots, m_n^\circ]$. Let the market price of m be 1, i.e. let

$$m_i = m_i^\circ - \sum_{j=1}^{k} (x_{ij} - e_{ij}) p_j$$

where $p_j \in \Re$ denotes the respective market price for the k commodities.

The utility functions in our case are multi-dimensional i.e. the agent's valuation of the resources is dependent on its allocation of all resources and can not be seen as the sum of the valuations of the single resources. In our experimental evaluation, we will use concave utility functions, which are natural for practical applications.

If possible, we would like the agents to put in bids once and then use some algorithm to compute the optimum. In our case this would imply that agents would have to submit their complete utility functions to the mechanism. First, this would require that the agents are able to express their preferences in some simple notation. Secondly, the mechanism must be able to handle this information in some effective way; the number of different possible combinations of amounts of resources to bid for is very large. And, third, it is not reasonable to assume that real bidders would be willing to reveal their complete utility functions in a closed form. Therefore we have chosen an iterative scheme in order to successively create new allocations and price vectors and allow the agents to put in new bids each time a new price vector is made public by the mechanism, i.e. we are creating a two-sided, iterative combinatorial auction.

4 Market mechanism

Given a market according to definition 2, the suggested market mechanism works as follows.

1: Each agent has an endowment.
2: Each agent submits initial bids as a demand function for each commodity. Hence, these bids are non-combinatorial. The agent's bid on one commodity is based on the expected clearing prices for the other commodities.
3: Compute an initial price vector at equilibrium.
4: For each agent, define the agent's *initial bid* as the reallocation and payment that comes out of the initial price vector.
5: **repeat**
6: Given the price vector, an agent might want some other allocation than his initial bid. Therefore, the agents are now allowed to put in new bids. A bid consists of a wanted allocation and a value telling how much the agent is prepared to pay to get that very allocation (multi-dimensional, combinatorial bids). The bids indicate how much the agent wants to buy/ sell relative to its initial endowment.
7: Find a new price vector and a new allocation such that surplus is maximized. For details, see below.
8: **until** no more surplus increment can be found.

The surplus of a bid is defined as the difference between the value of the bid and the cost of the bid at the current prices. The total surplus is the sum of the surpluses of all implemented bids.

The main design issue is how to perform step 7. This, and some other details, is discussed below.

The task of Step 7 is to choose bids to implement in such a way that bids are matched against each other and so that supply equals demand. Bids do not have to be implemented completely but can be implemented by a factor α, where $0 \leq \alpha \leq 1$. However, we use the following constraints:

1. It is possible for an agent to have parts of different bids, from different rounds, implemented at the same time. However, no agent will be allocated more than one complete bid, i.e. the sum of all α for one agent cannot be higher than 1.
2. Since the initial bids are based on non-combinatorial bids, they may not reflect true synergies. Therefore, in order for an agent to get rid of his initial bid, we ensure that if this bid is allocated to a factor x and $x < 1$, this bid can not be allocated to a factor greater than x in any subsequent round. In this way, as the agent manages to get other bids implemented, the initial bid will be removed.

It should be noted that the one-dimensional bids are no longer a part of the bidding process. The only remainder of these bids are each agent's initial bid, which is an artificial combinatorial bid.

Since we allow bids to be implemented in parts, the problem of finding a matching set of bids that maximizes surplus can be solved by linear programming (see appendix A). Hence, this part can be done in polynomial time. (In our implementation, we use a commercially available lp-solver to accomplish this task.[1])

When the linear programming has decided which bids to implement (and their corresponding α:s), new prices must be created. We formulate also this part in such a way that linear programming can be used (see appendix B). Based on the bids of the latest round, we set intervals in which the prices are allowed to deviate. The problem of finding prices in these intervals such that the cost requirements of the bids are fulfilled is now sent to the lp-solver.

If new prices cannot be found in the previous step, one of the (partly) implemented bids is temporally removed. The decision of which bid to remove is based on its assumed ability to raise the total utility of the system. When a bid has been removed, the new instance of the problem is sent to the lp-solver. If no prices have been found when all bids have been removed the price intervals are updated and the process is repeated with all bids present.

The bidding iteration stops if no new bids are received or if no allocation or no new prices could be found during one round or if no increase in total surplus can be found.

4.1 Comments on the market mechanism

Bidding All bids and payments are relative to the endowment. We are using what we call *absolute bids*, i.e. the bids are only dependent on the current prices in

[1] Since this lp-solver is based on the Simplex algorithm it is only polynomial in the average case. Its worst case is exponential.

the sense that the prices decide which point in the space of all possible allocations the agent will bid for. The value of the bid will be acceptable to the agent through the whole bidding process. In this way, once a bid is submitted to the mechanism it will remain implementable throughout the process and a bidder could never withdraw a bid. This is important from a conceptual point of view.

When bids are chosen for implementation, the allocation of an agent may consist of parts of different bids. Therefore the mechanism has to make sure that the sum of all α:s for an agent never exceed 1. If for example an agent has put in two bids with almost identical allocations this will assure that the agent would not get an allocation which is approximately twice of what it really wants.

The first set of bids, which are based on the equilibrium of the one-dimensional utility functions (called *derived utility functions*), are the only bids that are not based on the real utility functions. Their ability to get the system closer to the optimum are probable but not certain. It is important that these bids do not have too much impact on the allocations and we would like them to be replaced by other bids as early as possible in the bidding process. Therefore the mechanism makes sure that the factor by which the bid is implemented cannot increase from one round to the next. In this way, some of the initial bids will be implemented to smaller and smaller parts until they eventually disappear.

It is however important that the first allocation gives rise to bids in this way. First, it is important that the agents are forced to take the consequences of their bids based on the derived utility functions. This requirement is partly fulfilled by this procedure. Secondly, this gives agents that are satisfied with their current allocation an opportunity to keep this allocation. Without this functionality, the agent would have been forced to put in a new bid, identical to its current allocation, which seems to be a conceptual flaw.

The notion of surplus As mentioned above, the surplus of a bid is the difference between the value of the bid and the sum of the costs of buying/selling resources according to the bid at the current prices (multiplied by α). The total surplus is the sum of the surpluses of all implemented bids. When the mechanism is choosing which bids to implement (using the lp-solver), it is maximizing the total surplus while maintaining the equilibrium (i.e. demand shall equal supply for each resource).

The surplus can be used to rank the bids, according to how much they are supposed to increase the total utility, which the mechanism does not know. Assuming rational agent behavior and that the utility functions are quasi-linear, we can conclude that the bidding agent would not risk a decrease in utility by putting a value on the bid that is higher than its gain in utility. This means that its is reasonable to assume that if we implement a bid from an agent who claims to be willing to pay more per unit (or sell for less), this agent will contribute to the total utility more than an agent willing to pay less.

Creating prices Each implemented allocation needs to have an associated set of prices. The main consideration when selecting the prices is to fulfill the cost

requirements of the bids creating the allocation. In theory, this may be the only requirement on the set of prices. In practice however, we want the prices to be clearly directed towards the equilibrium prices. Therefore, we make restrictions on how the prices are allowed to deviate from the current prices. In our mechanism, the restrictions are dependent on whether the current prices give rise to excess in demand or supply for a resource. This information is accessible to the mechanism through the bids of the current round. The size of the excess/deficit shall in some sense be proportional to how much the prises are allowed to increase/decrease.

If the bids of the latest round (if all of them had been implemented) would have given rise to greater supply than demand of one resource, then the prices of that resource should be reduced. If, on the other hand, demand is greater than supply, the price of the resource should increase. Depending on how far from equality the bids are, the mechanism computes an interval for each resource telling how much prices are allowed to deviate from the current prices (see appendix C). The mechanism now tells the lp-solver to find prices in these intervals that corresponds to the new allocation. (Each agent has to afford to buy/sell according to its bid, without paying more than the value of the bid.) While remaining inside the intervals set by the mechanism, the lp-solver now tries to find upper and lower bounds in such a way that all prices between the bounds fulfill the cost requirements. The lp-solver computes the maximum of the sum of the differences between the upper and lower bounds and the new price is set to the mean of these two values.

Bid removal If the mechanism is unable to find a set of prices corresponding to a given allocation the reason is that the cost requirements of some bid could not be satisfied (given the restrictions of the allowed deviations from the current prices). In this case we remove one of the implemented bids. When choosing a bid for removal, we should consider the bid that is most likely to cause trouble in this sense. Therefore we rank the bids by surplus. The bid with the lowest surplus is removed. When a bid has been (temporally) removed, a new allocation is created (if possible) and the mechanism tries to find a corresponding set of prices.

If all bids are removed in this process, it may be caused by too high restrictions on the deviation of the prices. In this case the mechanism increases the allowed interval and restarts the search for prices with all bids present.

Criteria for termination The mechanism terminates if in an iteration, no agent wants to put in a bid. This means that all agents are satisfied with their allocation, given the current prices. In this case we are certainly very close to the optimal solution.

The mechanism also terminates if an allocation with a corresponding set of prices cannot be found. In this case there is no way for the mechanism to proceed and termination is unconditional.

A third reason for the mechanism to terminate is if the total surplus is decreasing from one round to the next. Small deviations from this criterion could be allowed in order to enhance flexibility.[2] Since surplus cannot grow forever, this guarantees that the mechanism will terminate.

4.2 Time complexity

The mechanism can be implemented to run in time polynomial in the number of agents and the number of resources.

Assume that the number of iterations is polynomial.[3] Each linear programming problem can be solved in time polynomial in the number of current bids. The number of current bids is no more than the number of agents times the number of iterations so far. To find an allocation with a corresponding set of prices at least two linear programming problems must be solved. When the lp-solver is unable to find a solution, i.e. when there does not exist any set of matching bids with prices in the given intervals, a bid is removed and thus we have new lp-problems as potential solutions. Hence, even if all bids are removed this process could be achieved in polynomial time.[4] If all bids have been removed without success, the price interval is updated. The process of searching for allocations and prices, and possibly bid removals, is then restarted with all bids present. If the price intervals could be updated at most a constant number of times, all this implies that the number of calls to the lp-solver will be polynomial. Since all other functions also are polynomial, one iteration will run in polynomial time, and so the complete mechanism will run in polynomial time.

It is well known that linear programming problems could be solved in polynomial time [10]. In our implementation however, we are using a lp-solver based on the Simplex algorithm, which takes exponential time in the worst case. On average it takes polynomial time and in practice linear programming problems can be solved very quickly.

5 Experiments

In our experiments, we assume the utility functions to be concave, i.e. the partial derivatives of all variables are decreasing.[5]

---

[2] In our implementation 5 rounds that do not increase the current best value are allowed. If the total surplus has not increased since the equilibrium based on the derived utility functions, 15 nonincreasing rounds are permitted.

[3] The tests of section 5.2 indicates that this assumption could be reasonable.

[4] In our implementation for a given price interval the number of bids that can be removed is maximized to twice the number of agents.

[5] The exact form of the utility functions in our experiments is $(\sqrt{c_1 x_1} + \sqrt{c_2 x_2} + \ldots + \sqrt{c_k x_k})^{c_{k+1}}$ where $1 \leq c_i < 5, \forall i, 1 \leq i \leq k, 1.5 < c_{k+1} \leq 1.8$.

5.1 Implementation specific issues

To be able to start a bidding process, we need a set of initial prices. One way to create this is to let the agents predict the equilibrium prices. It is reasonable to believe that agents have some idea of what the final prices will look like. Since the predictions give rise to corresponding allocations, given those allocations for $m-1$ resources a one-dimensional utility function for resource m can be computed. In this way, a function for each of the resources is created and the functions are sent to the mechanism as a kind of (one-dimensional) bids. To compute a competitive equilibrium from these bids is a straightforward task. The equilibrium gives rise to an allocation and a set of prices, which can be used as a starting point for the subsequent bidding. The quality of the allocation and the set of prices (compared to the optimum based on the real utility functions) are dependent on the quality of the predictions of the agents. If the allocation is very close to the optimum the algorithm may find it hard to find a better solution.

The derived utility functions will not fully correspond to the real utility functions. Therefore this first allocation will leave some agents dissatisfied. To compensate for this, they are now allowed to put in bids. The bids consist of how much of the resources the agent wants at the current prices and how much it is willing to pay to get it.

Bidding strategy When simulating the behavior of an agent in the algorithm, the agent bids for its optimal allocation at the current prices and the value of the bid is the difference between the utility of this allocation and the utility at endowment, i.e. the maximal amount of money it can pay without risking to get a decrease in its utility value. An agent only puts in a bid if its wanted allocation at the current prices differs at least ϵ from its current allocation, for some resource. This means that the agent may be unlucky and remain at the same utility value as before the reallocation, i.e. the agent is neither better nor worse off. This situation will occur if one of the agent's bids is implemented by factor 1 and at the same time the prices are high enough to demand all of the value of the bid. This is, however, just one of a number of potential outcomes. It is possible that the prices are low enough to allow the agent to keep a part of the money it was prepared to pay and thus it will increase its utility. Furthermore, the agent is aware of the possibility that its bid may be implemented in part. If the bid is implemented to factor α, the maximal payment for the agent is α multiplied by the value of the bid. Then, because of the concave shape of the utility function, the agent is guaranteed to increase its utility. If the bid is implemented in part, the allocation of the agent can be seen as a point on the line between endowment and the point corresponding to the complete bid. The cost of these allocations corresponds to a straight line between 0 and the value of the bid while the agent's valuation $f(x)$ is concave and increases with the value of the bid for the same allocations. Therefore the gain in $f(x)$ for these allocations will always be greater than the loss in m, i.e. the utility $u(x)$ will increase.

In our simulations of the system we assume that agents do not speculate about the behavior of the other agents. We also assume that the agents act like price-takers, i.e. they will not speculate about how their own behavior will affect the prices. Given these assumptions, the simulated behavior of the agents is rational. Apparently, the agent should not put a higher value on the bid; then it could loose more money than it would gain in utility. If, on the other hand, the agent would put a lower value on the bid, it would face the risk of not getting an allocation which would give it an increase in utility. It is extremely important for the agents to get as close as possible to their optimal allocation at each iteration, since they cannot ever be sure that another reallocation will occur. Thus, it is rational to put a value on the bid that corresponds to the real utility for the allocation of the bid. The amount of resources in the bid is chosen because of the fact that this very allocation is the optimal allocation for the agent, given current prices. To bid for this allocation would therefore be reasonable if the agent expected only minor changes in the set of prices from this iteration to the next. For each iteration of the algorithm, the agents are facing a bigger risk that this reallocation might be the last and at the same time they should expect smaller changes in prices. This indicates that the above agent behavior will be more and more reasonable the longer the run of the market algorithm.

In a more general setting, however, it is possible that agents do speculate about how the actions by themselves and other agents will affect the market. In this case, the market mechanism will still work, even though there might be a loss in efficiency.

5.2 Tests

To test the performance of our program, problems of 4 different sizes, with 10–15 instances each, have been randomly generated according to the set up restrictions. The results are shown in table 1– 4. The columns denote in order:

Ex the number of the instance.
Nr of rounds the number of iterations executed.
First round the total real utility after the first round relative to the optimal value. The result after the first round is the equilibrium according to the one-dimensional utility functions.
Last round the found value in relation to the optimal value.
Increment the increment from the first round to the last round in relation to the difference between the optimal value and the result of the first round, i.e. the improvement of the combinatorial auction in relation to the biggest possible improvement.

The tests show that the equilibrium according to the one-dimensional utility functions gives a good approximation of the real maximum, which is further improved by the combinatorial auction. The found optimal solutions, for the tests of table 1– 3, are always at least 98% of the real optimum, the mean values are always $> 99.6\%$. This could of cause not be generalized — we can

never guarantee an upper bound on the approximation error of an algorithm running in polynomial time for this problem [5]. The improvement made by the combinatorial auction is on average $> 79.3\%$ of the possible improvement.

Not surprisingly, it seems like problems where the number of agents is small relative to the number of resources are more troublesome for the mechanism. An example of this can be seen in table 4.

We can also see that the number of iterations is relatively small. This suggests that the algorithm may be very useful in practice, but more and bigger tests are needed to justify this claim.

Ex	Nr of rounds	First round	Last round	Increment
1	16	0.9974	0.9982	0.3081
2	11	0.9856	0.9979	0.8525
3	19	0.9842	0.9997	0.9795
4	6	0.9992	0.9992	0.0145
5	9	0.9867	0.9990	0.9244
6	38	0.9842	1.0004	1.0246
7	15	0.9919	0.9997	0.9677
8	37	0.9930	0.9997	0.9520
9	18	0.9050	1.0001	1.0014
10	41	0.9772	0.9794	0.0941
11	4	0.9795	0.9858	0.3062
12	15	0.9395	1.0002	1.0028
13	24	0.9794	1.0000	0.9979
14	23	0.9732	0.9992	0.9715
15	11	0.9828	1.0000	1.0009

Table 1. Results of 15 different instances with 10 agents and 3 resources. The mean of the results after the first round is 0.9773, the mean of the results after the last round is 0.9980 and finally the mean of the increments is 0.7958. Due to inexact computations the found value sometimes slightly exceeds the optimal value and therefore the increment can exceed 1.

6 Conclusions and future work

A combinatorial auction in general is $\mathcal{NP}$-hard. Our mechanism finds an approximate solution to this version of the problem in polynomial time. The output of the algorithm is on average very close to the real optimum. The created model and the needed simplifications are intuitive and easy to understand. The time complexity is reduced while the simplicity of the market mechanism has been maintained and a big part of the behavior is based on everyday experience. The mechanism can be explained in a way that makes it possible to understand the principles and the main functionality even for a non computer scientist.

Ex	Nr of rounds	First round	Last round	Increment
1	19	0.9271	0.9963	0.9492
2	18	0.9625	0.9974	0.9305
3	12	0.9436	0.9986	0.9758
4	31	0.9709	0.9968	0.8897
5	24	0.9629	0.9965	0.9055
6	19	0.9163	0.9948	0.9383
7	23	0.9769	0.9964	0.8445
8	25	0.9870	0.9973	0.7941
9	13	0.9805	0.9961	0.7982
10	46	0.9450	0.9964	0.9348

Table 2. Results of 10 different instances with 50 agents and 5 resources. The mean of the results after the first round is 0.9573, the mean of the found values is 0.9967 and finally the mean of the increments is 0.8961.

Ex	Nr of rounds	First round	Last round	Increment
1	9	0.9939	0.9965	0.4315
2	50	0.9449	0.9991	0.9828
3	43	0.9768	0.9971	0.8751
4	43	0.9716	0.9996	0.9860
5	24	0.9867	0.9985	0.8905
6	35	0.9747	0.9986	0.9455
7	35	0.9822	0.9889	0.3754
8	17	0.9617	0.9983	0.9551
9	41	0.9632	0.9975	0.9317
10	20	0.9812	0.9917	0.5570

Table 3. Results of 10 different instances with 25 agents and 10 resources. The mean of the results after the first round is 0.9737, the mean of the found values is 0.9966 and finally the mean of the increments is 0.7931.

Ex	Nr of rounds	First round	Last round	Increment
1	16	0.9898	0.9918	0.1934
2	12	0.9674	0.9889	0.6586
3	9	0.9753	0.9786	0.1321
4	15	0.9929	0.9929	0.0000
5	15	0.9915	0.9915	0.0000
6	18	0.9828	0.9866	0.2201
7	15	0.9890	0.9890	0.0000
8	33	0.9786	0.9873	0.4069
9	15	0.9829	0.9829	0.0000
10	22	0.9663	0.9833	0.5049

Table 4. Results of 10 different instances with 10 agents and 10 resources. The mean of the results after the first round is 0.9816, the mean of the found values is 0.9873 and finally the mean of the increments is 0.2116.

One limitation of the mechanism seems to be that it works best when the number of resources is small in relation to the number of agents. This is not surprising — since bids are supposed to be matched against each other, fewer bids and more resources make this process more complicated.

To our knowledge there is currently no other mechanism solving this kind of problem in polynomial time. Our goal has been to create simple mechanisms to find an algorithm that could be used in practice, while maintaining the easy-to-understand structure of the solution process. In this way a very complicated problem can be solved by a relatively simple method.

We presuppose that the real utility of the agent only is known by the agent and not by the system. The utility function can be described analytically and the partial derivatives of the functions are all decreasing. (Of cause, we are only testing a subclass of these kinds of functions.) The mechanism should be working for a broader class of functions; a problem that might arise is how the agents would compute which allocation to bid for at the current prices.

There are a number of ways in which this mechanism could be tested for generality. First, we need to perform more tests while maintaining the current constraints. Secondly, more complicated utility functions could be considered. Another issue that we would like to look further into is to start the bidding process with the endowment allocation and random prices.

References

1. F. Ygge. *Market-Oriented Programming and its Application to Power Load Management*. PhD thesis, Department of Computer Science, Lund University, 1998. ISBN 91-628-3055-4, CODEN LUNFD6/(NFCS-1012)/1-224/(1998) (Available from http://www.enersearch.se/ygge).
2. S. Bikhchandani and J. W. Mamer. Competitive equilibrium in an exchange economy with indivisibilities. *Journal of Economic Theory*, 74:385–413, 1997.
3. D. Parkes. iBundle: An efficient ascending price bundle auction. In *Proceedings of the First International Conference on Electronic Commerce*, pages 148–157. ACM Press, Box 11405, New York, NY, November 1999. (Available from www.eesc.harvard.edu/~parkes).
4. Y. Fujishima, K. Leyton-Brown, and Y. Shoham. Taming the computational complexity of combinatorial auctions: Optimal and approximate approaches. In *Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI'99*, pages 548–553, August 1999. (Available from robotics.stanford.edu/~kevinlb).
5. T. W. Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceeding of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI'99*, pages 542–547, August 1999.
6. A. Andersson, M. Tenhunen, and F. Ygge. Integer programming for combinatorial auction winner determination. In *Proceedings of the Fourth International Conference on Multi-Agent Systems ICMAS 00*, volume VII. IEEE Computer Society, Los Alamos, July 2000.
7. M. H. Rothkopf, A. Pekeč, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1995.

8. M. P. Wellman. A market-oriented programming environment and its application to distributed multicommodity flow problems. *Journal of Artificial Intelligence Research (www.jair.org/)*, (1):1–23, 1993.
9. M. Karlsson, F. Ygge, and A. Andersson. Market-based approaches to optimization. (Submitted for publication), 2002.
10. H. Karloff. *Linear Programming*. Birkhäuser, 1991.

A Lp-problem: Selecting bids

Parameters

NrAgents : the number of agents
NrBids : the number of bids
NrResources : the number of resources
x (1 .. NrBids, 1 .. NrResources) : the allocations of the bids
agent (1 .. NrBids) : agent$(k) = i$ iff agent i put in bid k
BidVal (1 .. NrBids) : the values of the bids
prices (1 .. NrResources) : the current prices for each resource
AlphaUpperBound (1 .. NrBids) : the upper bounds for the alpha-values, usually set to 1.

Variables

alpha (1 .. NrBids) : the implemented part of the bid

Maximize

$$\sum_{k=1}^{\mathsf{NrBids}} \mathsf{alpha}(k)\Big(\mathsf{BidVal}(k) - \sum_{j=1}^{\mathsf{NrGoods}} \mathsf{prices}(j)\mathsf{x}(k,j)\Big)$$

subject to

$$\sum_{k=1}^{\mathsf{NrBids}} \mathsf{alpha}(k)\mathsf{x}(k,j) = 0, \qquad \forall j, 1 \le j \le \mathsf{NrResources}$$

$$0 \le \mathsf{alpha}(k) \le \mathsf{AlphaUpperBound}(k), \qquad \forall k, 1 \le k \le \mathsf{NrBids}$$

$$\sum_{\substack{k=1 \\ \mathsf{agent}(k)=i}}^{\mathsf{NrBids}} \mathsf{alpha}(k) \le 1, \qquad \forall i, 1 \le i \le \mathsf{NrAgents}$$

B Lp-problem: Creating prices

Parameters

NrBids : the number of bids
NrResources : the number of resources
x (1 .. NrBids, 1 .. NrResources) : the allocations of the bids
BidVal (1 .. NrBids) : the values of the bids
prices (1 .. NrResources) : the current prices for each resource
up (1 .. NrResources) : the upper constraints on the prices
down (1 .. NrResources) : the lower constraints on the prices

Variables

ub (1 .. NrResources) : the upper bound on the prices that fulfills the requirements
lb (1 .. NrResources) : the lower bound on the prices that fulfills the requirement

Maximize

$$\sum_{j=1}^{\mathsf{NrResources}} \Big(\mathsf{ub}(j) - \mathsf{lb}(j)\Big)$$

subject to

$$\sum_{j=1}^{\mathsf{NrResources}} \mathsf{x}(k,j)\mathsf{ub}(j) \leq \mathsf{BidVal}(k), \qquad \forall k, 1 \leq k \leq \mathsf{NrBids}$$

$$\sum_{j=1}^{\mathsf{NrResources}} \mathsf{x}(k,j)\mathsf{lb}(j) \leq \mathsf{BidVal}(k), \qquad \forall k, 1 \leq k \leq \mathsf{NrBids}$$

$$\mathsf{down}(j) * \mathsf{prices}(j) \leq \mathsf{ub}(j) \leq \mathsf{up}(j) * \mathsf{prices}(j), \qquad \forall j, 1 \leq j \leq \mathsf{NrResources}$$

$$\mathsf{down}(j) * \mathsf{prices}(j) \leq \mathsf{lb}(j) \leq \mathsf{up}(j) * \mathsf{prices}(j), \qquad \forall j, 1 \leq j \leq \mathsf{NrResources}$$

$$\mathsf{lb}(j) \leq \mathsf{ub}(j), \qquad \forall j, 1 \leq j \leq \mathsf{NrResources}$$

C Creating price intervals

When creating the price intervals that are sent as input to the lp-solver we consider the total demand of the bids of the current round (a negative demand corresponds to an excess in supply). The lowest new value allowed is down times the current price and the highest new price is up times the current value. We are using the following heuristic:

Total demand	down	up
> 500	1.2	1.4
$100 - 500$	1.1	1.3
$50 - 100$	1.0	1.2
$0 - 50$	0.95	1.15
$-50 - 0$	0.85	1.05
$-100 - -50$	0.8	1.0
$-500 - -100$	0.7	0.9
< -500	0.6	0.8

Recent licentiate theses from the Department of Information Technology

2002-003 Emad Abd-Elrady: *Harmonic Signal Modeling Based on the Wiener Model Structure*

2002-004 Martin Nilsson: *Iterative Solution of Maxwell's Equations in Frequency Domain*

2002-005 Kaushik Mahata: *Identification of Dynamic Errors-in-Variables Models*

2002-006 Kalyani Munasinghe: *On Using Mobile Agents for Load Balancing in High Performance Computing*

2002-007 Samuel Sundberg: *Semi-Toeplitz Preconditioning for Linearized Boundary Layer Problems*

2002-008 Henrik Lundgren: *Implementation and Real-world Evaluation of Routing Protocols for Wireless Ad hoc Networks*

2003-001 Per Sundqvist: *Preconditioners and Fundamental Solutions*

2003-002 Jenny Persson: *Basic Values in Software Development and Organizational Change*

2003-003 Inger Boivie: *Usability and Users' Health Issues in Systems Development*

2003-004 Malin Ljungberg: *Handling of Curvilinear Coordinates in a PDE Solver Framework*

2003-005 Mats Ekman: *Urban Water Management - Modelling, Simulation and Control of the Activated Sludge Process*

2003-006 Tomas Olsson: *Bootstrapping and Decentralizing Recommender Systems*

2003-007 Maria Karlsson: *Market Based Programming and Resource Allocation*

UPPSALA
UNIVERSITET

Department of Information Technology, Uppsala University, Sweden