Acknowledgements

Ich bedanke mich herzlich bei Dipl.-Inform.Wirt Jochen Stößer und Dr. Dirk Neumann vom Institut für Informationswirtschaft und -management (IISM) in Karlsruhe für die ausgezeichnete Betreuung meiner Diplomarbeit.

Mes meilleurs remerciements vont à Ruby Krishnaswamy, Adam Ouorou, Eric Gourdin et David Savourey de France Télécom R&D à Issy les Moulineaux (France) pour l'excellent tutorat et leur soutien, leur patience, leur disponibilité et leur collaboration enthousiaste pendant mon stage et au-déla. Un grand merci également à l'équipe de Grid4All et toutes les personnes que j'ai pu rencontrées au cours de ma mission et qui m'ont permis de travailler dans des conditions optimales.

Ein ganz besonderer Dank gebührt meiner Familie, die mir dieses Studium überhaupt erst ermöglichte und mich während meiner Studienzeit in Karlsruhe und im Ausland stets unterstützte.

Darüber hinaus danke ich allen meinen Freunden für die schöne Zeit, die wir während des Studiums zusammen verbracht haben.

Abstract

The Grid is a promising concept to solve the dilemma of increasingly complex and demanding applications being confronted with the need for a more efficient and flexible use of existing computer resources. Even though Grid technologies have made progress within the context of large enterprises and academic projects, there has not yet been a widespread adoption by public institutions and small enterprises. One barrier to this adoption is the lack of economic paradigms which support the dynamic and efficient sharing of Grid resources by balancing resource scarcity and idle capacities. Economic algorithms promise to provide a good fit to the Grid's inherent strategic dimension by enabling users to express their valuation for computer resources. At the same time they provide incentives to contribute idle resources to the Grid in return for the market price.

This survey presents a market-based approach for allocating complex computational Grid services based on economic properties. The implemented combinatorial exchange aims at maximizing the social welfare of users. At its core, it provides a rich bidding language which is able to represent complex Grid services and simple workflows. The allocation mechanism is evaluated by means of a numerical experiment in order to gain detailed insights into the computational complexity of the underlying allocation problem. It provides input for market configuration.

Contents

List of Abbreviations

CPLEX	Named after the simplex method and the C programming language
CPU	Central Processing Unit
CRP	Cumulated Reservation Price
CV	Cumulated Value
FLOP	Floating Point Operation
FLOPS	Floating Point Operations per Second
GB	Gigabyte
GHz	Gigahertz
GLPK	GNU Linear Programming Kit
MACE	Multi-attribute Combinatorial Exchange
MB	Megabyte
MDGC	Market Design for Grid Computing
MIP	Mixed Integer Programming
NP	Non-deterministic Polynomial Time
OCEAN	Open Computation Exchange and Arbitration Network
OGSA	Open Grid Service Architecture
RAM	Random Access Memory
VCG	Vickrey-Clarke-Groves
VO	Virtual Organization
WDP	Winner Determination Problem

List of Figures

List of Tables

1 Introduction

Complex applications and experiments in science and business such as simulations of new airplane design in the aerospace industry and simulations of tornadoes in climate modeling create a tremendous demand for computer resources. Many of these applications exhibit fluctuating utilization patterns with few peak loads, which lead to long idle times and low resource utilization on average. This trend is confronted with limited budgets and the need for a more efficient use of scarce resources. Grid technology offers a promising way out of this dilemma by enabling the dynamic and efficient sharing of heterogeneous computational resources within organizations as well as across administrative domains (Foster et. al. 2001). Thus accessibility of computer resources such as processing power, storage devices and applications is possible for any market participant. Resulting benefits such as higher utilization and reliability may increase competitiveness on the one hand and the enterprise's scope of activity on the other. However, despite its promises, there has not yet been a widespread adoption of Grid technologies; only few enterprises and large academic projects benefit from the Grid's computing power. This deficit is addressed by the European project *Grid4All*. Its goal is the development of a market platform for institutions of the public sector and small enterprises in contrast to other projects that target utility centers with supercomputers. Grid4All enables those users to access Grid resources by providing Grid-compatible market mechanisms.

The lack of economic paradigms has been identified as one of the major inhibitors of Grid technology (Fellows and Wallage 2007). If computer resources are scarce, a scheduling strategy is needed that decides which resource should be allocated to whom at what time. Classic technical scheduling algorithms are solely built on system-centric metrics. Economic algorithms promise to provide a better fit to the Grid's inter-organizational and thus strategic nature by explicitly taking into account valuations (Gomoluch and Schroeder 2003). By enabling resource requesters and providers to report valuations in addition to technical metrics, the scheduling mechanism can consider user preferences for resource allocation. These preferences are used to collect and disseminate information about the system's status and help to control the dynamic demand and supply. At the same time they provide incentives to contribute idle resources to the Grid.

The objective of this work is the development of a market-based optimization mechanism for resource allocation and the consideration of different pricing schemes. The mechanism is based on a specific application scenario within the context of the Grid4All project. It challenges the allocation of resources while meeting the demands of market participants. The way of demanding resources decides about the underlying market mechanism; hence, a combinatorial auction is developed for the application scenario. The mechanism provides a rich bidding specification that includes multiple resource types with attributes associated with them, time characteristics and takes into account scenario-specific allocation properties. It may be implemented for resource management in distributed environments, such as open Grid markets that have no restriction on who can buy and sell. As the mechanism's accent is on resource allocation, several features of the market framework such as protocols and security aspects are excluded from consideration. The following central questions are focused on:

- What do market participants demand and how may they express their demands?
- What are the consequences from users' demands on market design?
- Which properties should the resource allocation mechanism provide to satisfy users?
- What conclusion can be drawn from the mechanism's performance with regard to the market configuration?

This work is structured as follows. Chapter 2 introduces the Grid concept (Section 2.1) and presents Grid-compatible market mechanisms including economic properties (Section 2.2). The sample application scenario on which the resource allocation mechanism is based, is introduced in Chapter 3. The essential scenario requirements are pointed out and a suitable market mechanism is derived. Related work on market mechanisms for resource allocation is discussed in Chapter 4. At the core of this work Chapter 5 presents the combinatorial auction mechanism and its scenario-compatible bidding specification (Section 5.1) as well as the mathematical model formulation (Section 5.2). Additionally to the allocation mechanism, dominant subsets and pricing considerations are addressed in Sections 5.3 and 5.4

respectively. In Chapter 6, the numerical experiment evaluates the mechanism with regard to the computational performance and the complexity of the underlying allocation problem and provides information for market configuration. In Section 6.3, the mechanism's limitations are identified and general as well as specific recommendations for extensions and further examinations proposed. Chapter 7 summarizes the work and points to future research directions.

2 Grid Resource Allocation

The increasing interconnection between computer stations has pushed the development of Grid technology. Resource sharing in the Grid has become a promising future concept. Old-fashioned approaches of clusters and supercomputers are confronted with challenges of growing application domains and new markets for which the Grid provides solutions. Thereby, market-based resource allocation mechanisms take into account economic properties.

2.1 Grid Concept

The increasing demand for computer resources that is evoked by complex applications and simulations in the public and business sector challenges the development of high-performance computing. The combination of fluctuating resource utilization with occasional peak loads, which leads to long idle times and the pressure to cut down computational expenses necessitates a more efficient use of resources. A possibility is the outsourcing of computing infrastructure (Foster and Tuecke 2005). This, however, may result in resource scarcity and thus more overlapping in resource demand. Overlapping use consumes resources in such a way that some users may be unable to run simulations and test beds accurately or at all. In such an environment, resource scarcity is unacceptable and provokes the need for supplementary resources. But the supply of resources from the outside is subject to restrictions for political, financial and geographic reasons (Shneidman et. al. 2005).

Grid technology offers a promising way out of this dilemma. Inspired by the electrical power Grid, computer scientists developed the computational Grid in the mid-1990s as a successor of metacomputing. The evolution of the World Wide Web allowed to aggregate vast collections of computers into large-scaled computational platforms beyond institutional boundaries. The motivation for computational Grids was initially driven by resource intensive applications in science, engineering and commerce whose need of resources cannot be satisfied by a single computer (Personal computer, workstation, supercomputer or cluster) (Buyya et. al. 2005). As a form of distributed computing, Grid computing links computer resources from across a business, a company or an academic institution. The network of computers is then used as a single, unified resource that allows sharing, selection and aggregation

of a wide variety of heterogeneous and geographic distributed resources including supercomputers, storage systems, data sources and specialized devices owned by different organizations (CERN 2007, Buyya et. al. 2005). Due to this geographic distribution of supporting nodes, resource delivery is done over the internet. In contrast to conventional distributed computing, Grids focus on large-scaled resource sharing and innovative applications. The example of complex climate modeling brings out the performance of Grids. The issue is to launch many similar calculations to investigate how different parameters affect the model. Each calculation is a small-grained parallel calculation that needs to run on a single cluster or supercomputer. Using the Grid, these many independent calculations can be distributed over many different clusters on the Grid. This saves a lot of time (CERN 2007).

Principally, Grids could give access to every form of computing, thus they are the most generalized and global form of distributed computing (Foster et. al. 2001).

According to Foster and Kesselmann (2003) a Grid is a system that differs from other forms of computing as it

1. *"coordinates involved resources that are not subject to centralized administrative control,*

2. *uses standard, open, general-purpose protocols and interfaces, and*

3. *delivers nontrivial quality of service."*

An example of a Grid in the scientific community is presented by *TeraGrid* that links major U.S. academic sites. According to Wolski et. al. (2001), the overall goal of computational Grids is to allow applications to treat distributed resources including computational, network and storage resources as individual and interchangeable commodities, and not specific machines, networks, and disk or tape systems. The fact that the Grid gives users direct access to resources that belong to someone else, requires unquestionable confidence in the security regulations of the Grid, such as authentication and authorization methods.

Grid technology enables participating parties to perform computational applications spontaneously through provided resources in the Grid that are not under the control of the user (Foster 2002). Thereby the most challenging aspect that underlies the Grid concept is the coordination of resource sharing to meet end-user

requirements including resource management and scheduling in dynamic, multi-institutional environments. A *virtual organization (VO)* refers to a set of individuals and/or institutions that share rules, pool resources and collaborate in order to achieve a common goal (Buyya et. al. 2005, Foster and Kesselmann 2003). VOs define and manage resources for participants, rules for accessing and using resources and conditions under which resources can be used.

With regard to the Grid architecture, computing resources are integrated in the Grid through middleware and infrastructure software. Besides the underlying network and resource control, Grid projects focus research on middleware software that provides seamless access to applications for any potential participant. The architecture addresses the task of identifying fundamental system components, specifying their purposes and functions and indicating how they interact with one another. Furthermore, it allows to automatically negotiate resource-sharing arrangement among a set of participating parties (Foster et. al. 2001). Common protocols are used to represent these sharing relationships. During the last few years, the evolution of Grid technology has generated several application programming interfaces, layers and access protocols for services. As the nature of the Grid requires a standard, open, general-purpose protocol structure, the open source software toolkit (*Globus Toolkit*) has been developed as reference implementation by the Globus Alliance and many others all over the world (Globus-Alliance 2007). All in all, it provides a set of software tools required to construct a Grid including security measures, resource location, resource management and communications. The *Open Grid Service Architecture (OGSA)* specifies fundamental middleware components for the Grid. It extends, adapts and evolves Globus Toolkit protocols. The layered structure of the generic Grid architecture is presented in Figure 1. Each layer fulfills a specific task. The highest layer is the application layer and visible to users. Its function is to provide applications for different domains as well as to develop toolkits for application support. In contrast to this user-centric layer, lower layers are more focused on hardware. The middleware layer enables the participation of various Grid resources in the Grid by dint of tools and software. Computers, storage systems, electronic data catalogs, sensors and telescopes are resident in the underlying resource layer. The lowest layer is the network that establishes the physical connection of Grid

resources. In this context, it is pointed out that a Grid is not evaluated by its architecture but its applications, business value and scientific results that it delivers (Foster 2002).

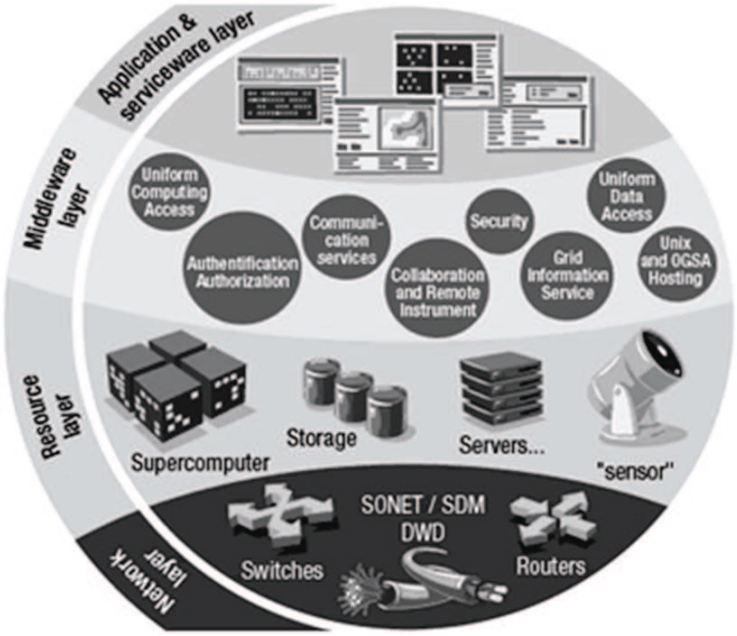


Figure 1: The layered structure of the Grid architecture (according to CERN 2007).

In recent times, application domains for Grid structures are expanding. While commerce is mainly concentrating on the implementation of Grids for single enterprises, research is setting up, testing and deploying large collaborative Grid infrastructures that span several countries and many institutions. Thus, computational Grids may be seen as future concept that solves the problem of resource scarcity and idle capacities by delivering resources wherever needed. They open up new internet markets by giving the possibility to acquire supplementary computational resources in open markets, where providers may be commercial utility providers or individual end users. In this context, the idea of VOs that grow or shrink on demand based on evolving requirement for resources is promoted. On the one hand the resource accessibility enables organizations to outsource nonessential computing infrastructure and purchase supplementary resources on the Grid if required. On the other hand little incentives are offered to contribute idle resources to the Grid. Some companies harness this lack of resource offers and provide paid Grid services (Bapna et. al. 2005). However, the systems are poorly conceived and hardly successful.

This implies the importance of market mechanisms that include economic pro-

perties in the coordination of resources.

2.2 Applying Markets to Grid

In recent times, the idea of incorporating market mechanisms into Grid technology has increasingly gained attention (e.g. Shneidman et. al. 2005, Lai 2005).

A *market* denotes in a narrower sense the location in which goods are traded or exchanged periodically. In a wider sense, a market is an economic system in which two parties, sellers and buyers, voluntarily exchange goods and services for money (Beutel 2006, Britannica-Concise-Encyclopedia 2006). Applied to internet markets that trade Grid resources, participants acquire the right to use a certain amount of system resources. As any situation where demand exceeds supply leads to unhappy users, the market has to define policies. These are a set of rules for resource allocation (Shneidman et. al. 2005). Respecting these policies, the key challenge today is the development of Grid-compatible market mechanisms. Such market mechanisms are composed of two basic components: the technic-oriented resource allocation and the economic-oriented pricing scheme.

The question arises why market mechanisms are needed when there are simpler allocation mechanisms, such as first-come-first-serve, reservation systems, automated voting schemes, randomized allocation, best effort and proportional share (Shneidman et. al. 2005, Lai 2005). These classic scheduling algorithms are solely based on technical system properties. They focus on satisfying functionalities and good performances in terms of scalability and runtime and aims at optimizing the performance of a system. So, the basis for decision making is predefined and static policies and priorities, and not economic paradigms such as the consideration of user preferences. Such an allocation, which maximizes resource utilization and balances system load is highly inefficient in an economic sense. Reactions on the dynamic nature of Grids are impossible and thus market requirements cannot be met. In this context, Buyya et. al. (2005) notes

> *"The support for economy-based resource management within Grid computing environments is essential for pushing Grids into mainstream computing"*

Market mechanisms address the deficiency of technical schedulers. They consider

desirable *economic properties* (see Table 1 on page 11) in resource allocation as the determining factor for solution performance. The goal of distributed resource allocation is no longer to maximize utilization but the efficient provision of resources to satisfy demand. Within the Grid environment, variable and unpredictable changes in demand and supply are assumed. This challenges the market to customize permanently to demand and supply. The performance of market-based systems is widespread. Providing a simple and efficient platform for offering several services for different applications at different times, they trade a large range of resources including computation, storage, network and data service. Supply and demand curves on the market are based on individual user preferences combined with external factors. Thereby, market participants are considered as self-interested parties. Market mechanisms provide a common basis to compare conflicting needs by allowing users to explicitly express requirements and objectives in form of valuations for services in addition to technical metrics.

In literature, the maximization of overall happiness by satisfying complex needs is seen as natural goal (e.g. Shneidman et. al. 2005). Based on individual utilities of users for goods, market mechanisms aim at finding the solution that maximizes the *social welfare*, that is the global utility. Social welfare is termed the overall welfare of users. It is assumed that utility is transferable among all participants.

In general, the *outcome* of a market-based resource allocation mechanism respects more or less of the economic properties listed in Table 1. As outcome is denoted the combination of winning bidders including the allocated services associated with them and their payments (Krishna 2002). *Allocative efficiency* comes to the fore as it meets the overall design goal of economic resource allocation mechanisms. An efficient allocation is a distribution of goods that can put nobody in a better situation without putting someone else in a worse situation. It is also termed *Pareto-Optimum*[1] (Beutel 2006). Due to the different individual utilities, each participant tries to find his optimal individual strategy that maximizes his utility. In a system without suitable market mechanism or significant social pressure, users are assumed to act strategically by not revealing their true valuations for goods (e.g. Lai 2005). This manipulation leads to suboptimal global utility. Economic proper-

[1]Named after V. Pareto (1848-1923)

ties in mechanism design may avoid manipulation and guarantee allocative efficiency. Thus, a mechanism has to offer incentives to accurately report information about valuations. Assuming *individual rational* users, the report of true valuations, called *truthtelling*, is the dominant strategy in equilibrium (Parkes et. al. 2001). The optimal strategy is the only that is always optimal. Thus, *incentive compatibility* enforces global truthtelling and avoids suboptimal results from strategic manipulation in terms of the mechanism's outcome and the welfare distribution in pricing.

Furthermore, valuations are used to collect and disseminate information about the system's status. They help to control the dynamic demand and supply while at the same time providing incentives to participate in the market by contributing idle resources to the Grid. Previously unknown prices may be established based on reported valuations which reflect the scarcity of respective resources (Stößer et. al. 2007).

Allocative efficiency	The general design goal of allocative efficiency is attained if the sum of individual utilities is maximized.
Incentive compatibility	The utility maximizing strategy in equilibrium for all participants is to report truthfully their valuations.
Individual rationality	Users only enter the market if their utility in participating is greater than or equal to the utility in not participating. E.g. Auctions with high prices will not attract potential buyers.
Budget balance	A mechanism achieves budget balance when the total payments of consumers are equal to the received payments of suppliers. Otherwise the system becomes deficient and depends on subsidies.
Computational tractability	Computational tractability refers to the computational complexity of obtaining the mechanism's outcome. Some optimization problems are solvable in theory, but not in practice, thus intractable.

Fairness	Fairness in allocation and pricing increases the market transparency and ensures the existence of prices (Bapna et. al. 2005). Thereby, fairness can be referred to several issues such as the division of bids in winners and losers according to a given scheme, the handling of ties of the optimal solution (see *ties occurring* in Section 5.3), the avoidance that consumers starve out and the preference of small applications.
Low transaction costs	Participating in an auction market is associated with bid submitting and getting information about allocated resources or bid rejection. Users appreciate transaction costs to be as low as possible.
Market transparency	For practical use, market transparency needs special attention. The market mechanism is no longer a 'black box' but a tractable transparent mechanism that allows verification from the outside.

Table 1: Economic properties of marked-based resource allocation mechanisms.

The indication of valuations provides economic incentives for users to reduce their priority in favor of incurring a fewer expense. This encourages the solution of time critical problems first and leads to higher global utility. So, users benefit from higher resource utilization, increased throughput, lower delay and higher reliability. Valuations are reported in currency terms to force users to put money where their valuations are. Truthful valuations in combination with a well defined currency system provide a promising basis for welfare maximization. The introduction of an open currency (e.g. US$) especially motivates participation, as it provides lower barriers for market entry (Buyya et. al. 2005, Lai 2005).

While individual rationality, low transaction costs and market transparency are basic economic principles that can either be assumed or easily be provided, others

such as allocative efficiency and fairness have to be explicitly taken into account and are not attainable simultaneously. A particular challenge in mechanism design is doubtlessly the trade-off between allocative efficiency and *scalability*. Scalability is termed the ability to maintain the required quality of service as the system load increases without changing the system (Macromedia 2001). With rising complexity, scalability may increase drastically and exceed the limit of computational tractability. Consequently, mechanism designers have to define priorities and limitations on economic properties in consideration of the underlying market requirements.

3 Application Scenario: Collaborative Learning

This chapter introduces the sample application scenario *Collaborative learning*. Firstly, the underlying idea of learning environments is presented and the specific process is described. Subsequently, the scenario is analyzed with regard to its domain- and process-specific character and requirements for the development of a suitable market mechanism are derived.

3.1 Scenario Description

3.1.1 Scenario Environment

The importance of education in future development is pointed out as follows in (Grid4All 2006):

> *"Education is considered to be one of the most important applications of grid computing in the near future. In this way, some efforts have already been made in order to identify the potential uses of grid infrastructures within the context of education in general and collaborative learning in particular."*

The need for further training is increasing in technology and research domains which see rapid evolutions in technology such as networking and telecommunications. The trend in recent years indicates that more and more research organizations and academic institutions are interested in further training for target audiences. In this context, computer supported virtual campuses and collaborative learning have established themselves as a strong complement to classic universities. They enable participants to engage in activities that have not been possible due to constraints in time, location etc. The evolution of the way, in which further training is organized shows the requirements of today. In the past, interested parties registered in advance in announced training offers. By contrast, the current trend is the organization of further training on demand and tailored to the specific needs of participants. Virtual and open universities are entering in this space to address the increasing needs from industry for further training.

Launched by *France Télécom R & D* in 2006, the European project Grid4All aims at developing a market platform on which Grid computer resources are traded, such

as computational, storage and application services. The objective is to deliver those services to requesters that are spread over several countries and connected by the internet. It focuses on providing dynamic access to Grid resources for educational institutions and small enterprises in contrast to utility centers and other European projects that are related to the business sector e.g. *Self-Organizing ICT Resource Management (SORMA)*. Unlike utility centers, actors on these markets are small-sized VOs.

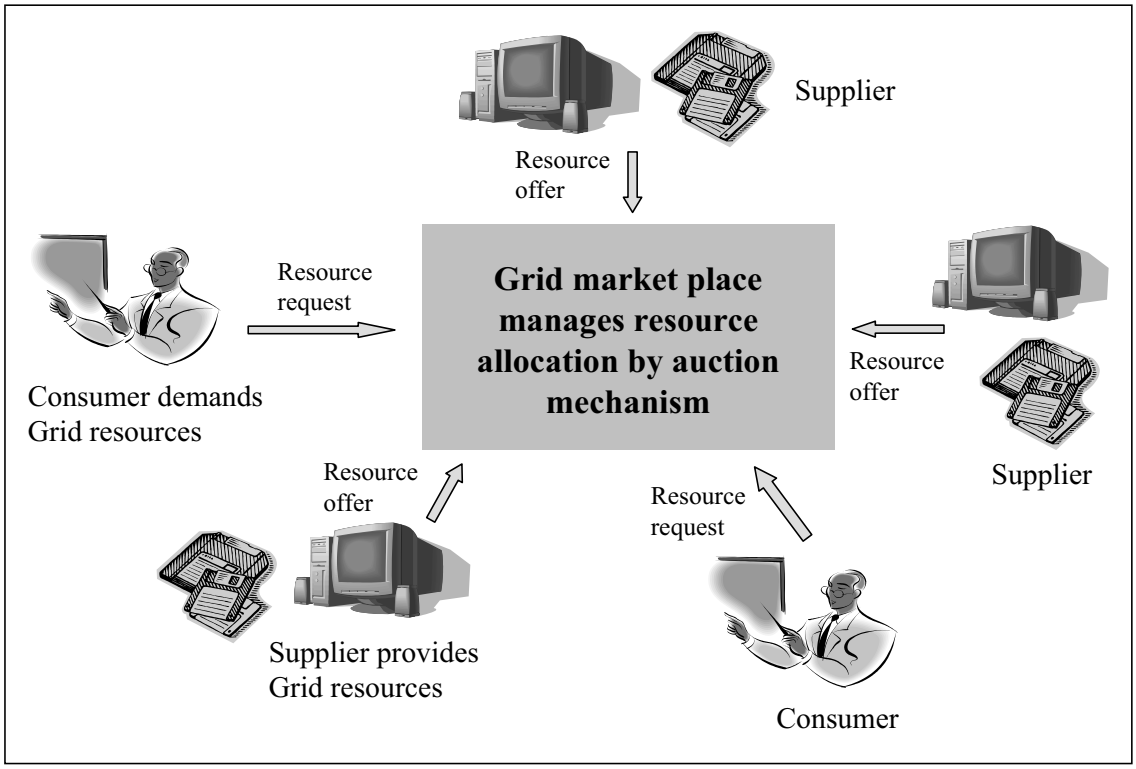


Figure 2: Application scenario. Consumers demand computer resources and providers offer resources on the Grid marketplace. The market mechanism manages the allocation of resources.

Grid4All proposes a list of scenarios representing applications of Grid computing. One of these scenarios is collaborative learning. In general, learning environments involve users, roles, computer resources (software, hardware and tools), information produced and consumed during the learning activity and applied methods.

A suitable environment for collaborative learning is characterized by a VO that is created with the goal to provide basic and further training for money to interested parties. Training may take place periodically or on specific demand in form of e.g. learning classrooms or workshops. A general overview of the market is illustrated

in Figure 2. The 'black box' in the middle of the figure manages the allocation of resources that are provided by suppliers and asked by consumers. The target audience of collaborative learning are e.g. working engineers and students. As hands-on experience is important in the networking and telecommunication domain, simulation software is more and more used to enable participants to design networks and analyze network configurations, routing and traffic engineering. Such hands-on training implies the need for computational and storage resources to enable the execution of simulation software.

In the past, training providers invested in their own capacities. This was unfavorable as it led to high expenses and the resource provision was not guaranteed anyhow. Scarce resource capacities limit the number of potential participants and thus the ability to meet the fluctuating demand. Grid computing in learning environments starts off at this point by enabling the acquisition of resources on demand.

3.1.2 The Process

Figure 3 shows an exemplary flowchart of the scenario. The amount of supplementary required resources is calculated by consumers and is not part of the process. Thereby, Broberg et. al. (2007) point out the difficulty for users to forecast resource requirements. It is assumed that users are able to determine their needs.

A workshop is considered, in which groups of students generate input files to execute network simulations. In collaborative learning, students may use simulation software to simulate a developed network design that is analyzed later on in terms of e.g. quality of service, traffic and security. Another example could be simulations of molecular processes in biology and tornado simulations in climate modeling may be instanced.

Network simulations are structured in form of simple workflows. A simulation consists of the execution of network simulation software in combination with data storing and the execution of a data analyzer. Each group executes such a sequence of steps that requires processing and storage capacity. The amount of additional resources from the Grid marketplace is calculated. The workshop represents the consumer that submits a resource request by a user interface. It contains several parameters that qualify the simulations and delivers information for the market.

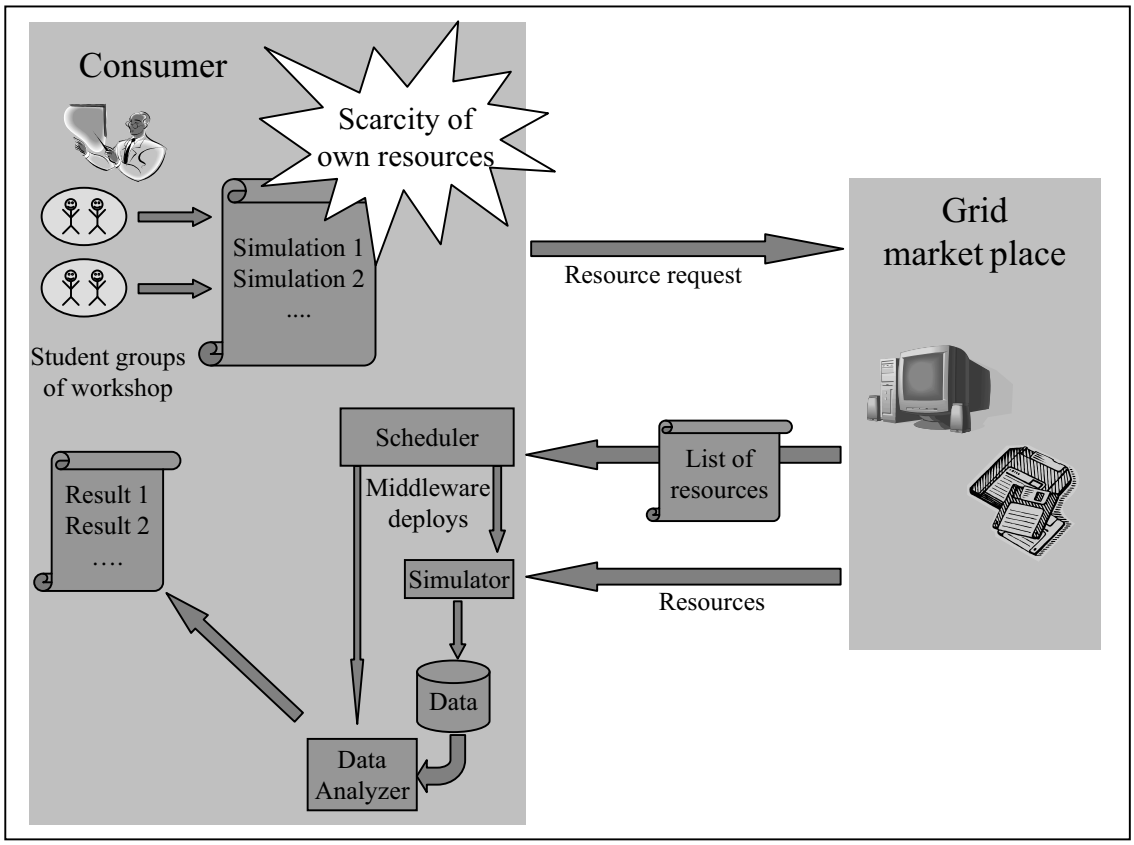


Figure 3: Scenario flowchart. The workshop represents a consumer that requests resources for collaborative learning. The Grid market place provides the required services for simulation execution.

That is where the resource allocation mechanism gets involved. It determines the optimal allocation of overall demanded and provided resources and hence the winning consumers and suppliers. If the request is successful, a list of allocated resources is returned to the requester. As the Grid market does not define machine scheduling, an offline scheduler generates independently from the market mechanism the flowchart of simulation execution on simulator and data analyzer. This information is sent to the middleware that deploys the simulator and data analyzer. The generated results are delivered to the students.

3.1.3 Scenario Characteristics

The focus of Grid4All is on different types of off-the-shelf applications and not on rare and exceptive experiments and simulations. Within this scope, the goal is the development of a market mechanism that is ideally applicable to a widespread range of scenarios. Primarily it has to be tailored to collaborative learning but also incorporate flexibilities to extend its range of application.

It is distinguished between essential and supplementary features. Essential features of collaborative learning are imperatively required to guarantee a suitable allocation that satisfies basic needs of market participants. Supplementary features customize the allocation mechanism and facilitate the application to other scenarios that consider those supplementary features as essential. One of the essential features is planning in advance. Assuming that workshop participants have to execute simulations within a fixed time frame, the resources for execution have to be available in time. Due to the risky resource providing in real time, the workshop's success is unpredictable without the possibility of purchasing resources in advance. Hence, planning in advance has to be supported by the mechanism.

In the scenario, traded commodities are computer *resources*[2] such as computation and storage service that are specified by quantity and quality attributes. The disposal of both resources at the same time is an indispensable scenario characteristic. Hence, it is assumed that agents always request and provide resource combinations. As no inventory is kept within the system to balance an excess demand on the

[2]The term *resource* is used equally to the meaning of *service*. Thus, resources may include more than a **single** hardware resource.

market, resource scarcity may emerge. For such a situation, collaborative learning provides the flexibility to request the optimal required resource quantities but also minimal required quantities. The workshop is considered to be successful when a sufficient amount of resources is obtained, i.e. at least the minimum quantity.

Main players in the market are consumers and suppliers that are combined under the term *agents*. On the demand side of the market, *consumers* ask for services in form of *jobs*. The terminology job is used to indicate the total set of resources with among others quantity and quality attributes that are required to execute a given number of network simulations. The simulation structure represents a simple scientific *workflow* that requires the execution of simulator and data analyzer as well as storage devices at predefined times. In general, 'workflow' refers to the automation of a sequence of job steps, called *tasks*, that are structured based on their control and data dependencies. The workflow structure indicates the temporal relationship between tasks in terms of parallel or sequential allocation, iterations and execution when its associated conditions are true, such as the achievement of a defined processing status (Yu and Buyya 2005, Yu and Buyya 2007).

On the supply side of the market, resource providers, called *suppliers*, offer computer services in form of *bundles*. A bundle refers to a combination of resources including among others quantity and quality attributes. Suppliers are supposed to be small providers with little resource capacities. Requests and offers are available within a given time frame that is defined by users and meets the time horizon of the market.

3.2 The Market Mechanism

Resource trading takes place in a virtual market environment that is solely determined by the type of scenario and hence beyond the designer's control. It includes potential participants, possible outcome and specifications of participants such as capabilities and preferences (Milgrom 2004). The nature of the traded resources and the way in which resources may be purchased and provided establishs a set of rules to govern the interactions of parties. The term 'mechanism' denotes this set of rules that is under the control of the designer e.g. it may specify the rules of an auction.

First of all, a scenario-compatible market model has to be selected. There are

numerous models for resource management, such as commodity markets, posted price models, bargaining models, tendering, contractnet models, auctions, bid-based proportional resource sharing models, cooperative bartering models as well as monopolies and oligopolies (Buyya et. al. 2005). In this work, the focus is on markets with exchange character, which means that several participants on each side of the market do not only demand but also provide resources. The two most relevant types of exchanges that are considered in literature are *commodity markets* and *auctions*.

In general, the trading goods on commodity markets are standardized products, such as raw or primary goods with the characteristic that goods of the same type are interchangeable. Mostly, future contracts on resources are bargained. These are agreements to deliver goods at a given time in the future for the established price at the time of agreement. This market form presumes the existence of ex-ante known clearing prices based on cumulated supply and demand. In the context of Grid resource allocation, the approach of Subramoniam et. al. (2002) applies the *Tâtonnement* (*eng.* groping) procedure of L. Walras for which real-world markets come to a price equilibrium. The auctioneer determines the price in equilibrium by first giving randomly a price and collecting the corresponding demanded and supplied quantities. This process is repeated and thus the market price is 'groped' (Wied-Nebbeling and Schott 2005).

By contrast, an auction is the process of trading resources by offering them up for bid and selling them to the winning bidder. In economic terms, it is a method for determining the value of a resource whose price is ex-ante unknown. Revenue and efficiency are considered as criteria for comparing auctions. While revenue is the expected selling price from the seller's perspective, a mechanism is efficient when the traded goods end up in the hands of agents who value them most ex-post. All types of auctions have in common that information is passed in form of bids which solely build the basis for outcome determination (Krishna 2002). In the context of computational resource allocation, consumers bid for the right to use resources for a fixed period of time. As one of the oldest types of markets, auctions attracted attention of economists in connection with the increase of oil prices by the cartel of the Organization of the Petroleum Exporting Countries (OPEC) in the early seventies. Auctions are widely studied as they provide advantageous properties, such as

little requirements for global price determination and simplicity of implementation. Besides the most common objective of auctions to maximize the social welfare, there are a range of further possible goals. In the past, when economic principles were not commonly used, the goal was to maximize utilization i.e. to allocate as many resources as possible regardless of any user preference. By contrast, Shneidman et. al. (2005) propose social policies such as preferences for small simulations or underrepresented stakeholders to meet the complex goal. Targets that concentrate on the profit of one side of the market are taken into consideration that privilege either suppliers or consumers. Varian (2007) even declares the profit maximization of suppliers as possible natural goal of auctions.

Combinatorial auctions represent a special type of auctions that focus on the trade of various resources at the same time. The term 'combinatorial' describes this auction type because the running of an auction can involve solving a combinatorial optimization problem. The distinctive feature of combinatorial auctions is the bidding on packages (several items) rather than single items. Market participants express demands and offers in form of package bids. Therefore combinatorial auctions are also called *package auctions* (Milgrom 2004). The first combinatorial auction took place in 1982. Proposed by Rassenti, Bulfin and Smith (1982), the pioneering package auction solved the problem of airport time slot allocation. Combinatorial auctions are of topical interest in the public sector and business-to-business (B2B) community. Started in the early mid-1990s, the first internet business-to-business exchange used combinatorial bidding in the trade of air emission credits. Afterwards, further research was effected on combinatorial auctions among others by de Vries and Vohra (2003), AuYoung et. al. (2004) and Bapna et. al. (2005). In recent times, combinatorial auctions have solved problems in a large range of domains, e.g. in transportations, bus routing and industrial procurement. Without focusing on a set of economic properties, combinatorial auctions could not solve the resource allocation problem efficiently.

Grid4All bounds the general choice of market models for the scenario by predefining the implementation of an exchange. Commodity markets and auctions meet this requirement. The underlying idea of commodity markets – to determine ex-ante

market prices for traded goods and to subsequently give agents the possibility to set trade quantities – is not feasible in multiple-item markets with a wide range of different item combinations. Auctions overcome this difficulty. The mechanism's main objective is the maximization of social welfare that is calculated as difference between revenue of all winning jobs and revenue of allocated bundles. To determine the optimal resource allocation, the auction mechanism matches provided and demanded services to identify winning jobs and bundles, and determines the welfare. Payments for consumers and suppliers are calculated depending on the applied pricing scheme. From the point of mechanism design, the challenge is the development of a suitable mechanism that meets the specific requirements of collaborative learning.

On a virtual open marketplace, agents may launch auctions that consist of three entities: consumers, suppliers and a third party mediator. The auction is characterized by a *two-sided* market structure in which both suppliers and consumers may offer and demand resources. In real world settings, users do not submit manually requests but utilize software agents. Within the bidding period, participants submit their bids *sealed* and *anonymous*. The identity of bidders is irrelevant for determining the outcome. Sealed bidding means that no bidder knows about the bid of any other participant. *Auction clearing* that means the start of the resource allocation by the auction, starts at a predefined time and no further bids can be submitted. This process character is called *single-shot* as no iterations exist. The periodical clearing of single-shot auctions can be minutes or hours ahead of the allocation period and processes successively resource allocation and pricing. By contrast, *continuous auctions* may clear at any time to assign arriving bids (Waldspurger et. al. 1992). Continuous clearing is essential for spontaneous resource demands. In comparison to periodic clearing, the allocation possibilities only depend on demand and supply on the market at a fixed time. Regarding a longer period of time, the efficiency lost is considerable.

As the disposal of a combination of computational and storage service is necessary for collaborative learning, consumer valuations on resource combinations are greater than the sum of valuations for individual resources in the combination. Such resources are termed *complementary goods* in micro-economics (Krishna 2002). In

auction terminology, this is expressed by superadditive valuations e.g.

$$v(\text{Computing power, Storage}) \geq v(\text{Computing power}) + v(\text{Storage})$$

The correlated demand functions of consumers for the two resources dictate a *combinatorial auction*. This auction type avoids the *exposure problem*. It means that consumers may end up with one resource and not the other. In single-item auctions, the exposure problem may happen as two items A, B are offered in separate auctions. Assuming that a consumer values the combination AB at 10 however A or B on their own are valueless. If the consumer bids 5 on each item A and B, he risks 'exposing' by winning only a subset meaning one item without the other. Thus, the consumer has lost 5 and won an item that is valueless for him without the other. Due to their need of resource combinations, consumers are generally not able to determine valuations for single resources and value combinations only.

Alternatively, *iterative auctions* (Parkes 2006, de Vries and Vohra 2003) may be applied for markets with ex-ante unknown resource prices. Their advantage is that they give agents the possibility to incrementally adjust offers and demands.

To guarantee a successful working auction, agents are supposed to comply with regulations. They behave faithfully by paying for allocated resources and granting access to conferred resources. Also, once a supplier bid is submitted, the resources are assigned to the Grid and agents have no influence on the allocation. Secure authentication methods and access control have to be provided for a successful trade (Wolski et. al. 2001). In this context, a collusion-free market is assumed so that distortion of competition is excluded and decisions of bidders are made independently. For instance, *bidding rings* are a form of collusion in which bidders agree not to outbid each other with the goal to obtain lower prices (Krishna 2002, Epiq-Technologies 2006). Such situations may lead to unintentional allocations in which utilities of manipulating agents are artificially raised. Furthermore, bidders are assumed to be risk neutral by seeking to maximize their expected profits.

3.3 Scenario-specific Requirements

In this chapter, the characteristics of the scenario environment and of the process are analyzed and the resulting requirements for the design of the allocation mechanism

determined.

3.3.1 Traded Commodities

In general, an arbitrary number of independent resources may be traded in a combinatorial auction. For computer services however, interactions between components influence the resource definition in the way that they may be considered as entities e.g. random access memory without processor is useless. A resource is characterized by at least one attribute. From the consumer's perspective, an attribute indicates the minimal required resource quantity or quality. The more attributes a resource consists, the more specifically may consumers indicate their demands. But the more complex becomes the handling of provided information and bid expression. Allocation is only possible whether the bundle's quality and quantity attributes match at least the demanded performance; otherwise the job-bundle combination cannot be realized. E.g. if a job demands processor speed of 2 GHz, a bundle with at least 2 GHz has to be allocated. The scenario-specific combinatorial auction trades the two commodities computation and storage service consisting of three attributes each. The duration of task execution depends on the performance of the CPU (Central Processing Unit) and their purchased number. As CPU is worthless without RAM (Random Access Memory), the combination is traded as an entity, whereas CPU is the resource and RAM is one of its attributes.

The question arises in which form CPU performance is modeled. As quantity and quality are related, they may be expressed either by specifying the number of CPUs in combination with the desired CPU speed in GHz or in an aggregated way by using the work rate, called FLOPS (Floating Point Operations per Second), or even the total number of task instructions, called FLOP (Floating Point Operation). The number of FLOPs is the multiplication of FLOPS and task duration. The drawback of the first proposal is that the real performance cannot be identified without supplementary information. A chip's clock speed is almost irrelevant in determining the overall performance of a computer. The value indicates how many times per second the processor operates on the fundamental set of instructions that makes a computer function. But it is no longer that simple. Depending on the design, some types of chips can currently process more instructions in each cycle

which makes them 'faster' compared to other chips with higher clock speed. In reality, clock speed is just one factor in a computer's performance, besides other figures e.g. the number of transistors, data width, cache size and bus speed. FLOPS may help to overcome this difficulty as they give more information about CPU performance without differentiating between the types of CPU (e.g. Intel, IBM). However, the system is dependent on consumers to give acceptable segmentations of FLOPS to avoid unfavorable allocations e.g. 998 FLOPS of job 1 to supplier 1, 2 FLOPS of job 1 to supplier 2. The number of acceptable segmentation may be very numerous. In this 'segmentation problem', consumers may end up with two allocated bundles of which one provides so little resources that the consumer cannot execute any application. The expression in FLOP is more exact since it specifies the total capacity of floating point instructions needed by the task. However this implies that the mechanism has to manage two interdependent variables: work rate and duration. Even in this case, it may be necessary that consumers specify the range of acceptable performances. Due to this segmentation problem, the concept of fixed CPU quantity and speed is used.

Storage service is simpler to classify in quantity and quality characteristics. Agents provide information about the number of hard disks, disk sizes and throughput which is characterized as the speed of writing on a hard disk. Thereby, network capacities and the transfer time to pass data from CPU to a storage device are ignored and the writing time is considered as relevant throughput. Figure 4 shows both resources and attributes associated with them.

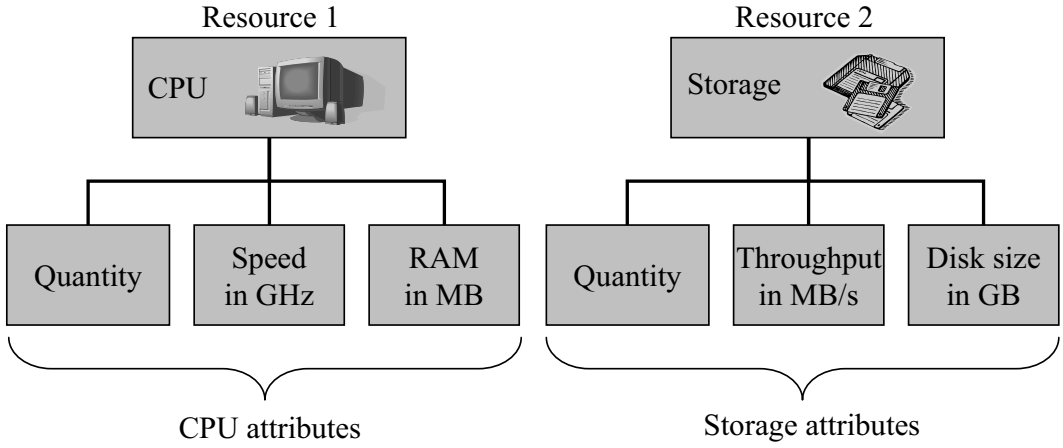


Figure 4: Traded commodities and attributes: Computation service as resource 1 and storage service as resource 2.

Before jobs are launched on purchased resources, the data input files are transferred via network. As this transmission time depends on the distance between the agents, it cannot be considered as a self-contained resource. Thus, further resource attributes are required to incorporate network distances. In consideration of complexity, network distances and connectivity are excluded of examination (see Section 6.3).

3.3.2 Bid Configuration

Agents purchase and offer services by submitting bids to the market platform. Although both supplier and consumer bids contain resource combinations, their structures differ.

Supplier bids consist of a combination of bundles of which an arbitrary number may be allocated. Thereby, the sum of all resources in the set of bundles corresponds to the total trading capacity of a supplier. The provided bundles are assumed to be compatible with applications. If a bundle is composed of resources of several physical machines, the supplier is responsible for any application that is executed. Either each application is run on a separate physical machine or on a virtual machine with shared memory layers. They are generated to avoid the security problem in case that several jobs are executed on the same physical machine. In general, each virtual machine or combination of virtual machines may be offered as a bundle. Due to the technical progress, the auction mechanism could assign multiple jobs to one physical machine and suppliers divide it into the required amount of virtual machines respecting the given allocation times. Due to the variable bundle divisibility, the allocation possibilities of jobs increase. This approach is considered by reason of the supplementary complexity and the difficulty of price determination for divided bundles.

Unlike suppliers, consumers express their substitute preferences by submitting bids consisting of multiple exclusive jobs. In micro-economics, this characteristic is expressed by the subadditive function $\widehat{x}(\text{Job 1}) + \widehat{x}(\text{Job 2}) \geq \widehat{x}(\text{Job 1, Job 2})$ with $\widehat{x}$ as consumer preference (Krishna 2002). From the market perspective, this conjunction of jobs increases the allocation possibility and adds value to consumers. On the other hand, the more expressiveness is allowed, the more complex become the bids.

A possible restriction could be the trade of predefined combinations of resource attributes exclusively from which agents may choose (Pekeč and Rothkopf 2003). Consumers indicate preferences among jobs through valuations that represent the maximum amount each bidder is willing to pay. Bidders are assumed to have *private values* that only depend on their individual valuation of the bid and are known to themselves at the time of bidding (Krishna 2002). In a workshop for instance, bids are submitted with the intention of purchasing resources for 10 simulations in the ideal case but acceptable may also be resources for 9 simulations. Preferences are piloted by setting different valuations to acceptable jobs: (Resources for 10 simulations, Valuation = 5), (Resources for 9 simulations, Valuation = 4). As the benefit from the package of resources for 10 simulations is greater than the benefit from the other package, consumers value it higher. Jobs may not only differ in valuations but also in terms of resource configurations, time and allocation characteristics.

As mentioned in Section 3.1.3, the structure of a simulation process may be represented as workflow. Is is assumed that all simulations submitted in a job are homogeneous i.e. they need the same resources, and have equal duration. Consequently, workflows are homogeneous as well. In the following, the workflow structure is simplified. Whenever a resource is demanded, joint allocation of both resources is effected at the same time. This determination still meets the scenario's basic needs. However, it implies the extension of allocation duration for a resource. For instance, a job that needs CPU in interval [12:00, 12:20] and storage in interval [12:00, 12:10], is handled as if it demanded storage in interval [12:10, 12:20] as well. Consumers express those simplified workflows as *job parts* within a job. As identical job parts feature the same specifications, a simulation may be an instance of a job part. To satisfy a job, all of its job parts have to be entirely allocated for the given duration as otherwise the partially assigned job would be of no need for the consumer. The possibility to indicate several job parts is one of the central scenario requirements and a first step toward 'real' workflow representation as described in Yu and Buyya (2005).

Although the simplification in workflow expression (see Figure 5) meets the process requirements, consumers obtain an abundance of resources. This results in overpayment at the expense of consumers. This can be exemplified by the idle CPU

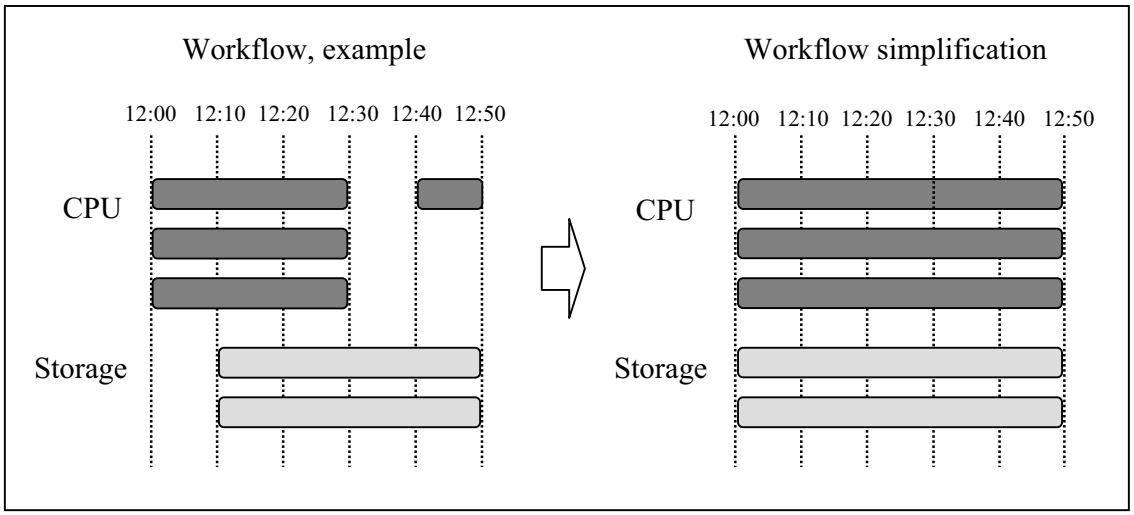


Figure 5: Sample workflow of the scenario and its simplification.

time between the end of simulator execution and the execution start on the data analyzer. Although computation service is not needed, but storage service, both resources are assigned all along. Thus, the consumer pays for the time of allocation of both resources.

In general, jobs and bundles may demand or offer a subset of resources. In the scenario however, simulations require both resources for successful execution. It is assumed that agents only demand and provide resource combination. Hence, resources are considered as indivisible within a job part.

Figure 6 illustrates a possible allocation of jobs to bundles whereas the size of the boxes indicates the amount of demanded or supplied resources. The number of jobs and bundles in bids is variable within a market defined range from agent to agent. In the example allocation, both consumers are satisfied and only one bundle remains idle.

3.3.3 Time Characteristics

The market defines an *allocation horizon* with constant length that is the total period of time slots starting at auction clearing until a defined date. Agents express *time frames* of availability by indicating earliest and latest possible allocation times of jobs and earliest and latest availability times of bundle. Allocation is only possible within the interval in which job and bundle availability times overlap. Additionally, consumers state the duration of jobs depending on the auction's time concept either

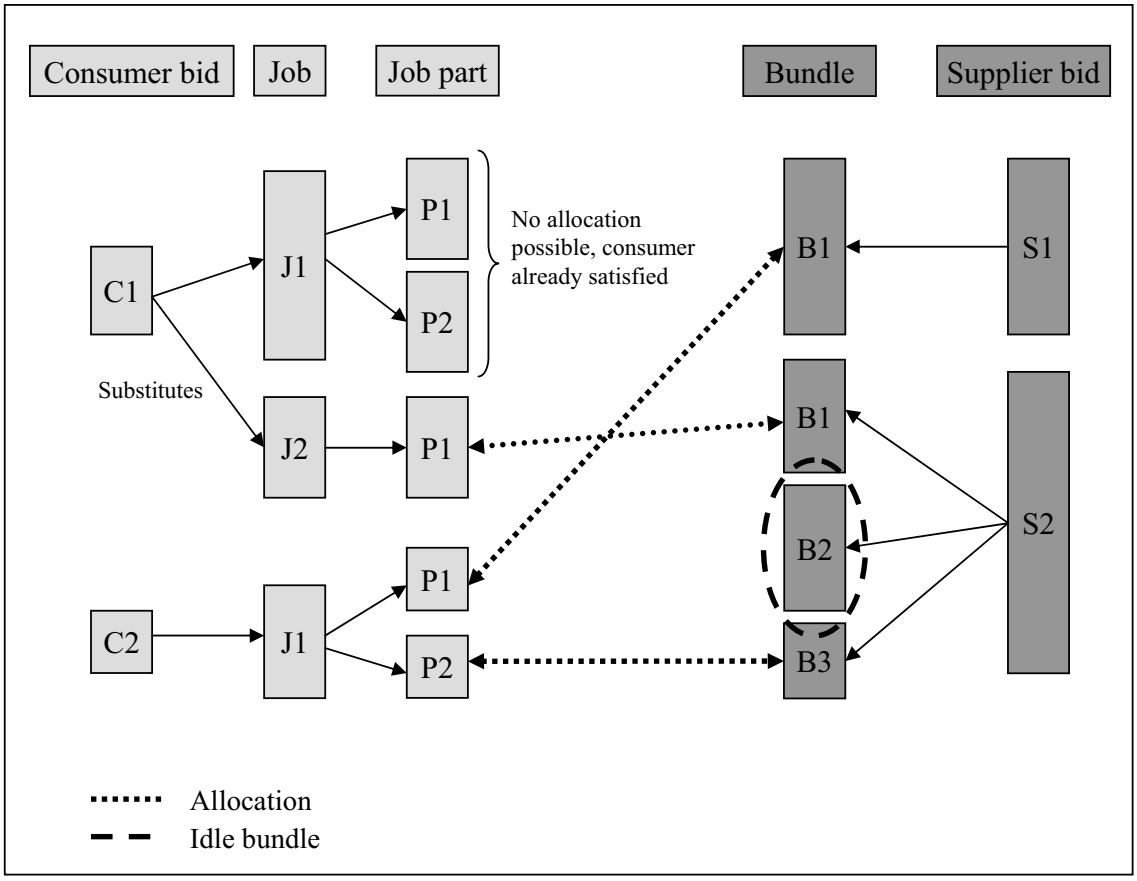


Figure 6: Allocation structure. A job may be allocated to multiple bundles and a job part to only one bundle.

as a numerical time period or in units of *time slots*[3]. The requirement that a bundle can be allocated is that it is available for the entire requested duration of the job. A challenge in determining time attributes is the calculation of job duration. Due to significant progress in the ex ante estimation of workload size, Smith et. al. (1998) provide techniques to predict the runtime of applications. Thereby, applications are compared to similar applications whose runtime is known.

Two main concepts of time representation are considered in literature: continuous and discrete time intervals within the allocation horizon. Discrete intervals may be divided into time slots of variable and fixed length.

To provide more flexibility in bid expression, jobs are characterized by the combination of the fixed duration and the time frame of allocation. Consumers may specify different earliest start times and latest end times for jobs but all job parts have the same time characteristics. Suppliers set the earliest and latest availability time for bundles that may vary from bundle to bundle. Time slots are supposed to be of uniform length predefined by the market. They are considered as units that are indivisible among jobs within a time slot.

In the concept of continuous time intervals, both suppliers and consumers indicate start and end times corresponding to their *effective* availability times. This is the time interval in which agents have idle resources that they offer in the Grid. Unlike the concept of discrete time intervals, agents do not have to take into account start and end time restrictions. Within this time frame, jobs may be assigned to bundles at any time. However, this concept has a disadvantage. For instance, a bundle is considered that is available within interval [12:05, 12:30] and job 1 with a duration of 10 minutes and an allocation time frame of [12:00, 12:20]. In this case, the job may be allocated for 10 minutes at any time within [12:05, 12:20]. By assuming jobs of reasonable duration, the allocation of job 1 in [12:07, 12:17] leads to declining likelihood of further use of this bundle in [12:05, 12:07]. The allocation of the job in e.g. [12:10, 12:20] would be more favorable. Thus the arbitrary allocation of time intervals may end up in inconvenient allocations and idle times. The striped box in Figure 7 points up the idle time of 2 minutes.

The next concept proposes discrete time slots of uniform length within the avai-

[3]Time slots are abbreviated as 'T' in figures.

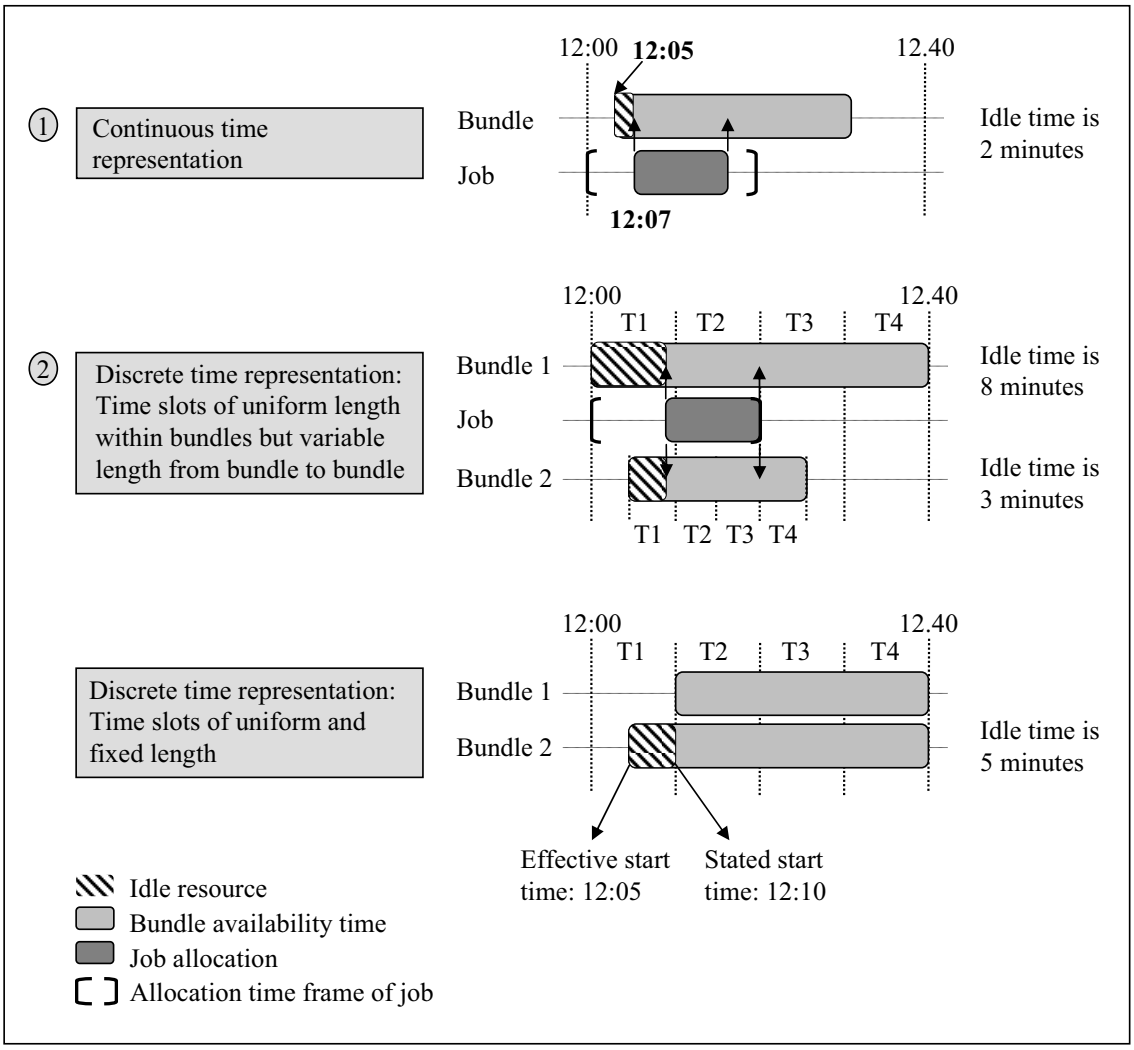


Figure 7: Continuous time representation (1) compared to discrete time slots with variable and fixed length (2). The specified 'idle resource' refers to the special difficulty of allocation in the respective time concept.

lability time of bundles but arbitrary duration from bundle to bundle. The difficulty for consumers is that they cannot foresee the suppliers' divisions of time intervals. Thus, their demands are not adjusted and they risk paying for unneeded resources. For example, bundle 1 is available in [12:00, 12:40] in time slots of 10 minutes (T1 [12:00, 12:10], T2 [12:10, 12:20] and so on), bundle 2 is available in [12:05, 12:25] in time slots of 5 minutes (T1 [12:05, 12:10] and so on) and a job of 12 minutes has to be allocated in [12:00, 12:20]. Although the job's time property matches those of both bundles, no exact allocation is feasible and the consumer has to pay for idle times. The allocation to bundle 1 or bundle 2 leads to idle time of 8 minutes or 3 minutes respectively (Figure 7). As bundle pricing per time slot is assumed, consumers have to pay for the entire duration of resource allocation. So the allocation to bundle 1 costs the consumer the price of 2 time slots that corresponds to 20 minutes instead of 12 minutes.

Contrary to the concepts specified above, it is the market that dictates the interval length for all bundles in this concept. The discrete time slots have a fixed and uniform length. This determination affects start and end times of jobs and bundles in such a way that *stated* availability times in bids do not necessarily correspond to effective times. Stated time is denoted the time of possible allocation respecting the configuration of the time concept. Although resources could have been provided earlier, effective start and end times need to be adjusted to the market configuration. Assuming 4 time slots of 10 minutes within the horizon [12:00, 12:40], this configuration is unfavorable for a bundle, whose effective availability time starts or ends within a time slot, such as a start time at 12:05 (see Figure 7). As the earliest possible allocation is at 12:10, the resources are idle for 5 minutes.

A general issue in modeling time concepts is the performance expression of CPU in combination with the job duration. Resource attributes of bundles only have to meet the demanded minimal requirements of jobs. The job duration however is calculated according to the minimal CPU performance. This may lead to unfavorable situations. For instance, a bundle's CPU speed is 2 GHz in a time slot of 10 minutes and the allocated job needs at least CPU speed of 1 GHz within this time slot. The job's duration of 10 minutes is calculated for a 1 GHz machine. As the job is allocated to a more powerful machine, the job finishes in less than 10 minutes.

However the remaining period cannot be allocated otherwise.

Due to its favorable handling, the concept of discrete and fixed time slots of uniform length is applied and may be upgraded in a further step (see Section 6.3).

3.3.4 Allocation Characteristics

Besides the matching of resource attributes and availability times, bundle prices and job valuations have to match for successful allocation. Consumer preferences in form of valuations indicate the maximum amount each consumer is willing to pay for a job. Generally, it is assumed that agents are able to calculate their valuations and have no budget constraints so that they have the ability to pay up to their respective valuations. As consumers pay not more than their indicated valuations, these values serve as upper bounds for payment determination. Consumers value entire jobs in contrast to suppliers that account for bundles and time slots. Suppliers define *reservation prices* that represent the minimal acceptable revenue for a bundle per time slot. Thus, reservation prices determine lower bounds for payments. Thereby, the sum of reservation prices of all winning bundles that are assigned to a job, do not have to exceed the job's valuation.

The description of agents, traded commodities, time and price restrictions constitutes the basic components of the allocation mechanism. Furthermore, specific constraints based on the scenario's environment are required to guarantee a suitable outcome.

Trading on the internet implies a large number of providers. In collaborative learning, each of them offers a small number of resources. Thus single bundles may not contain sufficient resources to satisfy the relatively large jobs compared to bundle size. So bundles may need to be combined to satisfy a job. As suppliers determine individually bundle sizes and the resource splitting among bundles, the allocation mechanism does not divide bundles within time slots. However, it is necessary to allocate jobs to different bundles. In general, the splitting of jobs and their allocation to multiple bundles may lead to unfavorable constellations for consumers (e.g. the 'segmentation problem' on page 24). Therefore, the allocation mechanism does not support the arbitrary split of jobs. As jobs consist of various job parts, each part may be allocated to another bundle.

Time plays an important role with regard to capacity utilization. Even allocated resources are idle in times of data transfer before and after job execution and in case that jobs are handed over between sites. As it is assumed that data transmission between bundles during execution is not possible, these idle capacities are avoided. The transfer of input files to the executing resources and the return transfer of output files to consumers are imperative and a question of network throughput and geographic proximity. Simulation execution yields big amounts of data output. It is assumed that the capacities of small providers are insufficient to buffer output files with the intention to continue execution later and launch another simulation in the meantime. This leads to the implementation of two conditions: *coupling* and *time consistency*. 'Coupling' expresses that each job part is allocated to a single bundle over time even though an entire job can be assigned to several bundles. The cases 2 and 4 in Figure 8 meet the coupling property in contrast to the cases 1 and 3 in which job part 1 is spread on 2 bundles. The term 'time consistency' is used equally to the expression 'continuously in time' and denotes that job parts are allocated in succession to time slots during their entire duration i.e. there is no unassigned time slot ('gap') within the allocation period of the job part on this bundle as show cases 3 and 4 in Figure 8.

Planning in advance is important to successfully schedule collaborative learning lessons. Besides the disposal of the right resource quantities and qualities, the availability times are the decisive factor. In some cases, consumers require resources for job parts simultaneously i.e. with the same duration, start and end times within a given time frame. The auction mechanism supports this allocation condition that is called *parallelization*. It can be 'activated' individually by consumers within a job and applies to all job parts. Figure 9 shows allocations with and without activated parallelization. A final examination in network simulations may serve as an example for the need of parallelization. Two conditions have to be fulfilled: The examination date is determined depending on the time interval in which resources are allocated and the examinees have to launch the simulations at the same time. This implies the need for resources in parallel within a given time frame.

Particularly, parallelization gains in importance when simulations need to communicate with one another during execution. Referring to the example on page 5,

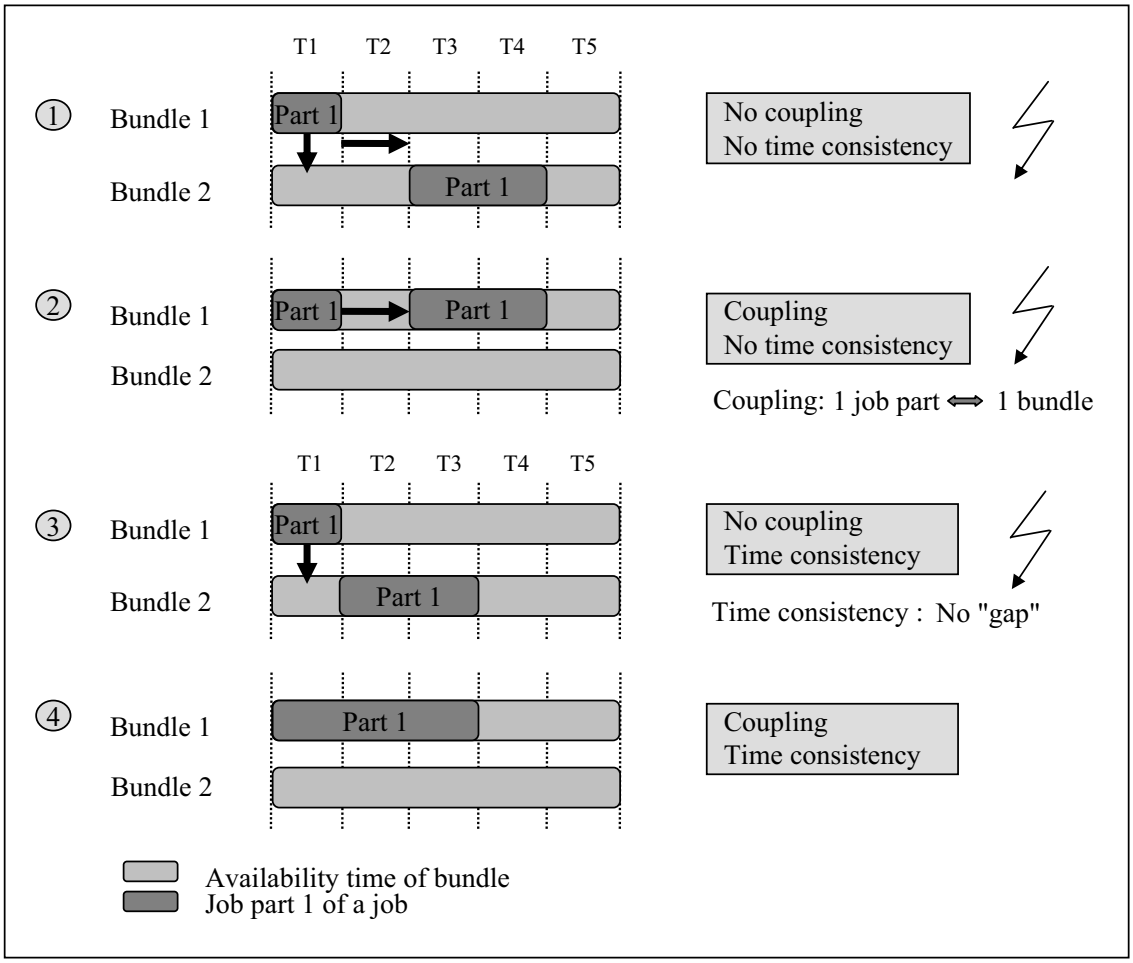


Figure 8: Coupling and time consistency. Cases 1-3 demonstrate allocations that are not allowed. Both conditions have to be fulfilled to assign job part 1 to a bundle (case 4).

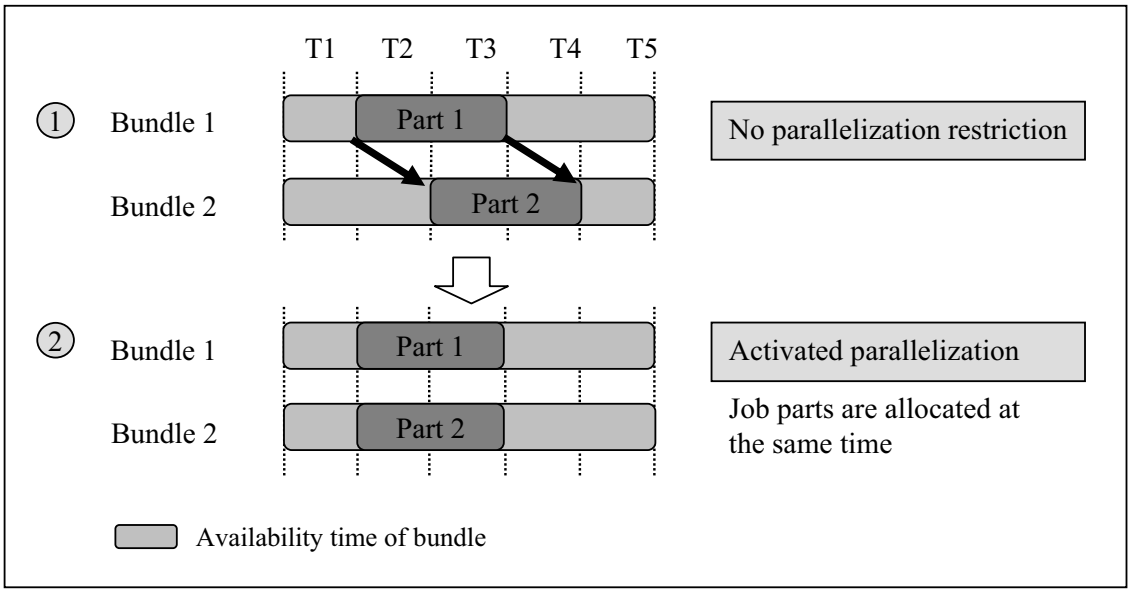


Figure 9: Parallelization. Case 1 shows an allocation of job parts without parallelization restrictions. In case 2, parallelization is activated and thus the job parts are allocated simultaneously.

each fine-grained calculation depends on intermediate results of other calculations. When simulating the weather, each calculation in one volume of atmosphere is affected by surrounding volumes. Those fine-grained calculations require parallel execution so that the right information is available to processors at the right time. Message passing interfaces may be used to run these applications. However for this purpose, knowledge about network distances is required.

Figure 10 illustrate a potential allocation with activated parallelization that meets the two central allocation conditions: coupling and time consistency.

3.3.5 Preliminary Pricing Considerations

Resource allocation and pricing are the two components of a market mechanism. The resource allocation algorithm represents the technical part that determines the winning jobs and bundles based on the matching of resources, time and allocation properties as well as valuations and reservation prices. Pricing is considered in a second step. It aims at yielding *payments* for agents that fulfill given economic properties. Thereby, valuations and reservation prices exclusively provide the basis for payment determination. The term 'payment' describes the effective price that consumers pay and suppliers gain. Payments of consumers are assumed to be nei-

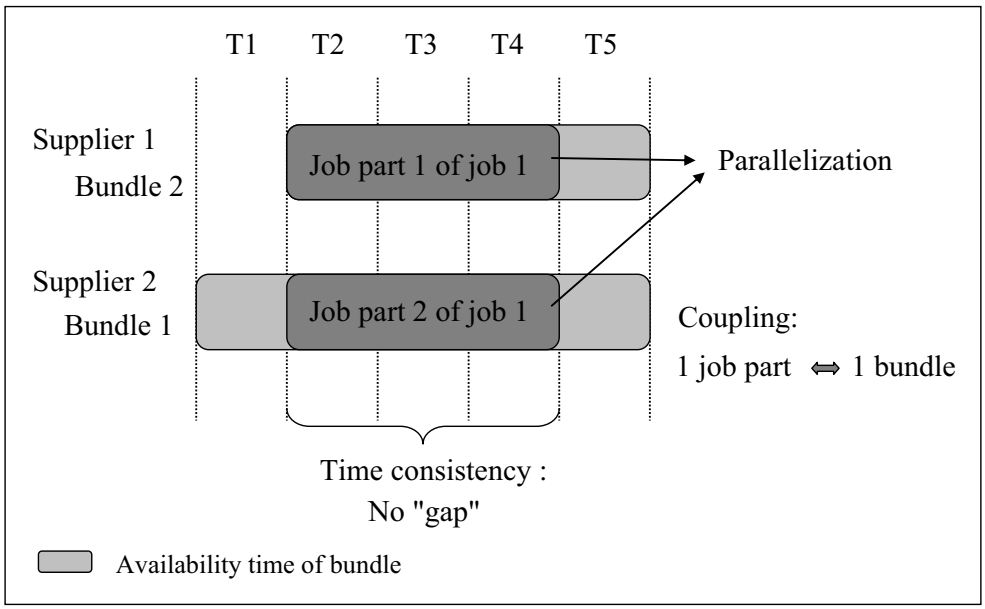


Figure 10: Potential allocation with activated parallelization.

ther greater than the job's valuation nor less than the sum of reservation prices of allocated bundles. The weighting of economic properties determines the choice of pricing scheme. Each pricing scheme leads to a different payment structure that privileges a group of agents e.g. the consumer or supplier side of the market. As agents are assumed to be individual rational, the payment determination of a pricing scheme decides among others about the participations of agents in the auction.

The central question in pricing is how the social welfare may be distributed among agents. As there is no market mechanism that fulfills all economic properties, pricing concentrates on the most relevant properties for the given application. In this way the guarantee of incentive compatible payments is beneficial, as it motivates agents to report their true valuations. In the case that agents deviate from their true valuations, the allocation mechanism may yield another resource allocation that is less efficient. Thus, misreporting leads to the deterioration of user's utility.

For markets that cannot be subsidized from the outside, the property of budget balance is the decisive factor for the choice of the pricing scheme. The market platform in the scenario environment represents such a case. As it is established for the public sector, no subsidization can be transferred from the outside and thus budget balance is an imperative requirement.

4 Related Work

In the research of auction-based scheduling of Grid resources, two main streams are outstanding. Whereas some mechanisms only consider the trade of one type of resource (single-item mechanisms), others take dependencies between various Grid resources into account (multi-item mechanisms).

Within each of those streams, several further distinguishing factors make it difficult to transfer developed mechanisms to other application areas, such as domain-tailored specifications or different priority orders of economic principles. Mechanisms can also be classified by their type of implemented algorithm for problem solving. The performance of *exact optimization algorithms* is the delivery of optimal solutions to a problem in contrast to *heuristics* that generate fast solutions. The complex *winner determination problem (WDP)* of combinatorial auctions addresses the difficulty of how to efficiently determine the allocation of resources after completed bid submitting (Lehmann et. al. 2006). Depending on the problem size, the optimal solution or even a feasible solution can rarely be found in adequate runtime[4]. As time is an important competitive factor in on demand delivery of computing resources, time-saving solutions for resource allocation have to be found.

Heuristics overcome this difficulty of exact mechanisms. Implemented as separate algorithms or in combination with exact mechanisms, they are not geared to finding either the optimal solution or a solution in good runtime. Heuristics yield feasible but mostly suboptimal solutions regarding welfare as they approximate the optimal solution. Generally, the solution is found in shorter runtime with less computing resources than by using exact mechanisms. Although heuristics may yield the only feasible solution to very difficult optimization problems, the significance of solution is questionable in terms of usability (Wikipedia.org 2007, TechTerms.com 2007).

4.1 Single-item Mechanisms

Single-item mechanisms have been proposed in Waldspurger et. al. (1992), Chun and Culler (1999), Regev and Nisan (2000), Lai (2005), Padala et. al. (2003) and several other papers. The main advantage of these mechanisms is their simplicity. Users do not have to specify complex technical requirements of jobs. Due to their

[4]The WDP is NP-hard, see Section 6.2.1 on page 73

calculating speed, these mechanisms support real-time applications which require immediate allocation of computing time.

The first implementation of a distributed computational economy is a continuous auction, called *SPAWN* (Waldspurger et. al. 1992). It examines price dynamics and uses monetary funding as a form of priority. The most prominent mechanism is *market-based proportional share* where resource requester i with a reported valuation of v_i gets assigned computing time of $v_i/\sum v_j$, i.e. proportional to i's share in the total reported valuation (Chun and Culler 1999).

4.1.1 OCEAN

The work of Padala et. al. (2003) is a further example of a single-item mechanism. The *Open Computation Exchange and Arbitration Network (OCEAN)* aims to provide a marketplace for resource trading in metacomputing environments in contrast to the scenario which turns its attention to small-sized VOs. As security is one of the major issues in distributed systems, OCEAN focuses especially on secure signature modules. A distributed scalable peer-to-peer matching network forms the underlying technology. The matching layer adopts the functionality of resource localization and the negotiation layer starts the contracting process with potential users. The negotiation process of this decentralized market-based system can lead to one-to-one negotiations between consumers and suppliers. Therefore, time attributes are not included in OCEAN but required by the scenario to realize the time-phased exchange with fixed clearing periods. Since the market model implemented by OCEAN is a continuous double auction trading single items, it does not respond to the scenario demand for resource combinations as users risk being 'exposing'.

4.2 Multi-item Mechanisms

This disadvantage of single-item mechanisms gave rise to mechanisms which trade combinations of heterogeneous resources such as computing power, memory, bandwidth and even software licenses. They approach Grid settings by means of combinatorial exchanges that trade multiple items, e.g. AuYoung et. al. (2004), Bapna et. al. (2005), Pekeč and Rothkopf (2003), Englmaier et. al. (2006), de Vries and

Vohra (2003) and Schnizler et. al. (2006). Referring to de Vries and Vohra (2003), economic efficiency is enhanced when bidders are allowed to express preferences over combinations of different items.

4.2.1 Bellagio System

As the first system which supports allocation of heterogeneous resource combinations, the *Bellagio System* (AuYoung et. al. 2004) meets both specific functional requirements (e.g. usability) and economic properties (e.g. allocative efficiency and fairness). It supports resource discovery as well as resource allocation whereas the resources are spread over different administrative domains. The Bellagio System aims at maximizing the cumulated end user utility and defines a bidding language that respects substitutes. It implements *SHARE* (Chun et. al. 2004) to clear periodically the combinatorial auction. SHARE utilizes approximation algorithms to determine the winners. The pricing scheme relies on the approximation of the truthful *Vickrey-Clarke-Groves (VCG)*[5] prices proposed by Parkes et. al. (2001). Nevertheless, the Bellagio system is not designed for small-sized VO's as it allows, for example, the split of bundles within single time slots.

Due to their benefit for resource allocation, economic principles are decisive factors in the mechanism design of e.g. *G-commerce* (Wolski et. al. 2001) and *Gridbus* projects (Buyya et. al. 2005). G-commerce examines supply and demand in commodity markets as well as in auctions and introduces currency to control the purchasing power in form of *Grid bucks ($G)* which are refreshed in periodic intervals.

4.2.2 A Market Design for Grid Computing

The combinatorial auction in *A Market Design for Grid Computing (MDGC)* (Bapna et. al. 2005) focuses on the integration of economic principles in Grid resource allocation. In a first step, it is based on an exact allocation mechanism that yields an efficient solution and incentive compatible VCG payments. Although the bidding specification allows the expression of resource combinations and availabi-

[5]Named after Vickrey (1961), Clarke (1971), Grove (1973), see page 63 in Section 5.4 for closer description of this pricing scheme

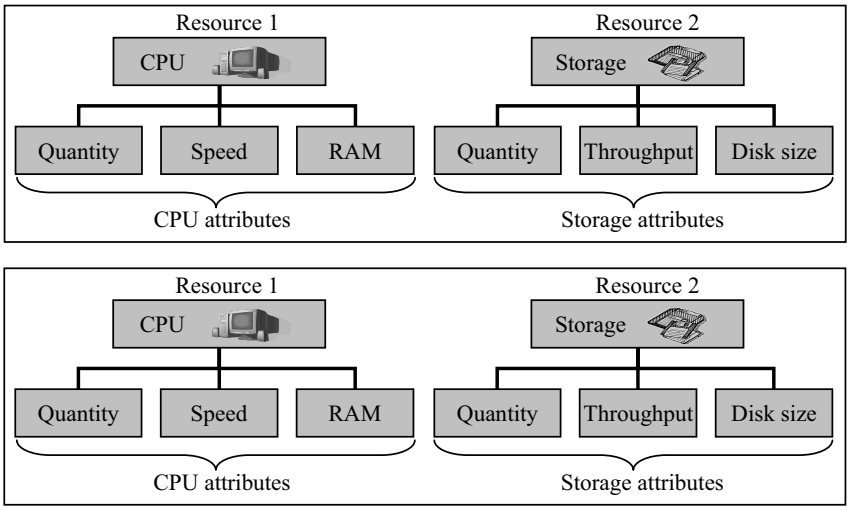
lity times, allocation-specific characteristics, such as substitute resources are not supported. As suppliers are seen as price takers, they only participate in the market whether their reservation prices exceed the exogenous reservation price that is defined by the market. This implies only little incentives for participation, as competitors cannot differ from each other with respect to prices. In a second step, Bapna et. al. (2005) turn their attention on the fairness principle while still maintaining the objective of welfare maximization. The idea is to provide a fair mechanism that yields temporal clearing prices for resources. Therefore, constraints in form of *fair allocation rules* are added to the WDP that avoid jobs being rejected, while other jobs with lower valuations are accepted. The yielded resource prices give bidders important information about rejection and acceptance of bids and thus increase the transparency in the market. Furthermore, this approach reduces the runtime and preserves incentive compatibility. In a third step, Bapna et. al. (2005) present the *time sensitive fair Grid heuristic* that is developed for markets that require fast solutions such as real-time or online implementations. Relaxing the maximum allocation property and neglecting incentive compatibility, the heuristic yields prices within a short runtime. However, it does not provide an efficient allocation.

4.2.3 MACE

The mechanism presented in *A Multi-Attribute Combinatorial Exchange (MACE)* (Schnizler et. al. 2006) is most relevant to this work. The rich bidding language of MACE enables users to request and offer arbitrary combinations of Grid resources specified by quality and quantity attributes as well as by time characteristics, co-allocation and coupling restrictions. Provided bundles may be split into parts to satisfy several consumer jobs. Due to the implementation of an exact mechanism, MACE can apply the VCG pricing scheme. As the economic principle of budget balance is not achieved, MACE deploys the so called k-pricing[6] which guarantees budget balance but neglects incentive compatibility in return. To what extent is MACE applicable to the scenario's design requirements? The answer lies in modeling the scenario in the bidding language of MACE and screening its adaptability and limits.

[6]See page 63 in Section 5.4 for closer description of this pricing scheme

Initial representation of job parts:



Possible representation in the bidding specification of MACE:

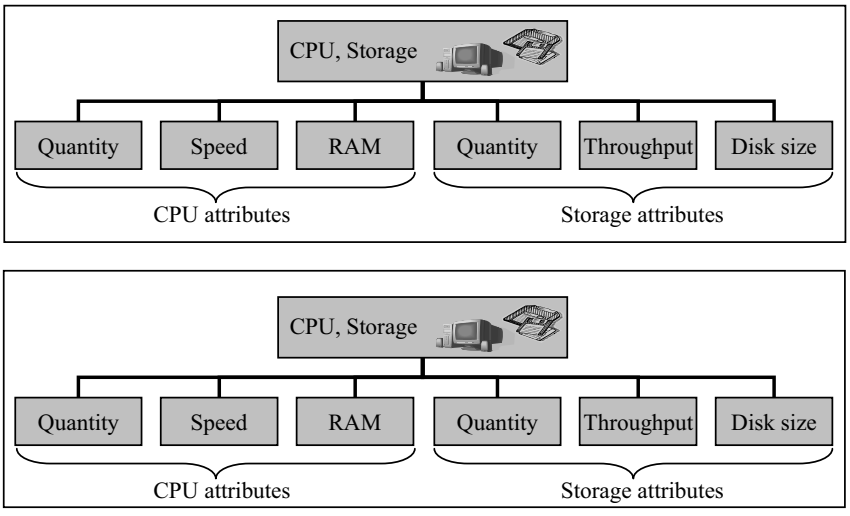


Figure 11: Job parts are modeled in form of resources in the bidding specification of MACE.

- **Job parts:** MACE does not explicitly support the split of jobs in uniform parts as it does not provide the *AND* operator. However, each job part can be represented as a *MACE-resource* that contains six attributes; three for computation service and three for storage service. Hence, modeled jobs consist of variable quantities of uniform MACE-resources (see Figure 11). For example, the expression of a job turns from

  > {(Resource 1: CPU with 3 attributes, Resource 2: Storage with 3 attributes, Time characteristics), Number of job parts = 2, ...}

  to

  > {(Resource 1 with 6 attributes, Resource 2 with 6 attributes, Time characteristics), ...}

  in the bidding specification of MACE. An example is the following job with the two resources CPU and storage

  > {({3, 2 GHz, 512 MB}, {2, 100 MB/s, 5 GB}, time attributes), Job parts = 2, ...}

  that can be formulated in the bidding language of MACE as

  > {({3, 2 GHz, 512 MB, 2, 100 MB/s, 5 GB}, {3, 2 GHz, 512 MB, 2, 100 MB/s, 5 GB}), time attributes ...}.

  As job parts cannot be allocated in partially to several bundles, neither can MACE-resources.

- **Coupling:** The coupling requirement of job parts corresponds to activating the co-allocation restriction in MACE all the time i.e. the split of MACE-resources and the allocation of these MACE-resource parts to different bundles is not allowed.

- **Small providers:** As there are small providers in the scenario environment, the type of coupling of resource that is proposed in MACE can not be applied. It enables consumers to define that a combination of resources is allocated to the same bundle of a supplier. Applied to the modeled MACE-resources,

it means the allocation of multiple MACE-resources to one supplier. This is not possible as one MACE-resource demands as much CPU and storage as a job part yet and suppliers have limited resource capacities. In the worst case, the activated coupling restriction of MACE could force the allocation of the entire job to one bundle. Consequently, there has to be no restriction for the allocation of MACE-resources to several bundles.

- **Time consistency:** MACE does not support the allocation of job parts i.e. MACE-resources continuously. To meet this requirement, many jobs have to be generated in MACE. The length of availability time frames in each generated job has to correspond to the stated job duration. The longer the availability time frame is the more jobs have to be generated. For example, the availability time frame [1, 5] provides 4 possibilities to allocate a job part with a duration of 2 time slots continuously in time. Due to this division, the flexibility of allocation within a time frame is implicitly maintained but the expression of time frames cannot be used anymore. Otherwise the constraint of time consistency may be violated. For instance, a job part demands resources for two time slots within the time frame [1, 3]. Corresponding to the time consistency condition, the job part is allocated in time slots 1 and 2 or in time slots 2 and 3. In MACE, this condition has to be expressed by the separate jobs: the first job with start time in time slot 1 and end time in slot 2 and the second job with start time in slot 2 and end time in slot 3.

- **Parallelization:** For parallel allocation of MACE-resources, no further determination is required. In contrast, the condition of time consistency effects parallelization of all MACE-resources within a job. Due to the time consistency constraint, the allocation of MACE-resources of a job is merely possible within the fixed time frame whose length corresponds to the job's duration. Thus, all MACE-resources of that job are always allocated in the same time slots, i.e. in parallel to each other.

As follows from the listing, job parts can be represented as uniform MACE-resources. However, the bidding language of MACE is merely able to model the scenario's special case of parallel allocation of all job parts of a job. The general case, in

which each job part can be allocated arbitrarily and independently but contiguously within the given time frame, cannot be expressed. Responsible is the added time consistency condition. The second restriction of MACE is the reduction of time attributes to start time, end time and duration. Availability time frames cannot be expressed anymore. A further differentiating factor is the formulation of the WDP. In MACE, it is formulated as mixed integer problem and in the model for the scenario as integer problem.

Although the mechanism of MACE is applicable to the scenario in the most parts, several specific requirements cannot be modeled in the bidding specification of MACE. Chapter 5 presents a suitable allocation model for the scenario.

An overview of the central distinctive features between the required mechanism for the scenario and the two most relevant exact mechanisms for this work, MDGC and MACE, are shown in Table 2.

4.2.4 GreedEx

Unlike the preceding models, *GreedEx - A Scalable Clearing Mechanism for Utility Computing* (Stößer and Neumann 2007) proposes an exchange for clearing utility computing markets, which is based on a greedy heuristic. This group of heuristics makes a local optimum choice at each stage with the intention of finding the global optimum without reconsidering the choices made so far. Due to the performance of heuristics, GreedEx yields near-optimal allocation schedules within a very brief period. Hence, GreedEx may be deployed as a mediatory platform that links several auction markets in order to decide on which market it is favorable to submit bids. GreedEx implements the critical-value-based pricing scheme that is similar to the Vickrey principle to determine prices for resource requests. Afterwards, proportional pricing is used to distribute the revenue to the agents.

	Scenario	**MDGC**	**MACE**
Combinatorics in bids	Substitutes, Job parts	No	Substitutes, Job parts[a] in parallel
Resource Attributes	Yes	No	Yes
Time attributes	Time frame, Job duration in time slots	Time frame, Job duration in CPU cycles	Time frame, Job duration in time slots
Coupling	Yes, implicit	No	Yes, if desired
Parallelization (Co-allocation)	Yes, for resources implicit, for job parts if desired	No	Yes, for resources if desired
Continuously in time	Yes, implicit	Yes	Limited, only if job duration covers the entire length of time frame.
Splitting of bundles within one time slot	No	Yes	Yes
Reservation price	Set by suppliers	Exogenously set	Set by suppliers

[a] see paragraph *job parts* on page 42

Table 2: Comparison of MDGC, MACE and the scenario-compatible mechanism.

5 Designing a Combinatorial Exchange

In this chapter, the optimization model for resource allocation is developed according to the scenario requirements. On the basis of the bidding specification that is introduced in the first section, the WDP and dominant subsets are formulated. In Section 5.4, the most relevant pricing schemes are discussed and applied to the mechanism.

5.1 Bidding Specification

The bidding specification defines the content of communication exchanged between the agents and the auction. The more information the bidding specification can contain the more exact agents may express bids. Hence the mechanism responds more accurately to supply and demand. A full expressive bidding language together with the optimal winner determination are needed to motivate truthful bidding and thus to guarantee economic efficiency (Sandholm 2006). However, the more flexibility in bid expression is provided, e.g. by workflow representation, the more complex become the problem. The trade-off between expressiveness in bidding specification and the constitution of these complex bids represents the main difficulty in designing bidding languages.

Let N be the set of consumer bids[7] consisting each of a set of jobs $J^n = \{1, \ldots, \bar{j}\}$ and $\bar{j} \in \mathbb{N}$ as maximum job quantity that is predefined by the market. A consumer $n \in N$ may ask an arbitrary number of jobs. As jobs are connected by the XOR[8] operator, one job is accepted at most. This bid structure

$$\{\text{Job } 1 \veebar \text{ Job } 2 \veebar \ldots \veebar \text{ Job } \bar{j}\}$$

is called *XOR bidding language* WDP_{XOR} according to Lehmann et. al. (2006). A consumer that submits various bids is considered as several independent consumers. The job specification is extended by the parameter $\bar{\alpha} \in \mathbb{N}$ that indicates of how many parts the job may consist. A set of job parts $Q^{nj} = \{1, \ldots, \bar{\alpha}\}$ is determined for each job. Parameter $\bar{\alpha}$ is predefined in the market configuration and serves as an upper limit for the number of job parts in all jobs. In other words, it is the maximum

[7] The notations *consumer* and *consumer bid* are used equally in this context.

[8] X XOR Y ($X \veebar Y$) means either $\emptyset$, X or Y, but not XY.

number of similar job allocations required to satisfy the job. The expression of job parts using AND[9] connections can be considered as a basic component of workflow representation and hence a first step in this direction. In this specification, job parts represent simulations.

The same applies for the set of supplier bids[10] M that contains the set of bundles $B^m = \{1, \ldots, \bar{b}\}$ with $\bar{b} \in \mathbb{N}$ as maximum bundle quantity predefined by the market. Thereby, $m \in M$ represents an arbitrary supplier. Bids are linked by the OR[11] operator and are structured as follows:

$$\{\text{Bundle } 1 \vee \text{Bundle } 2 \vee \cdots \vee \text{Bundle } \bar{b}\}$$

that is called *OR bidding language* WDP_{OR}. Consequently, the total number of market participants and bids is defined by $(\bar{n} + \bar{m})$ whereas $\bar{n}$ and $\bar{m}$ represent the maximum number of consumers and suppliers respectively. Figure 12 shows an overview of the bid structures.

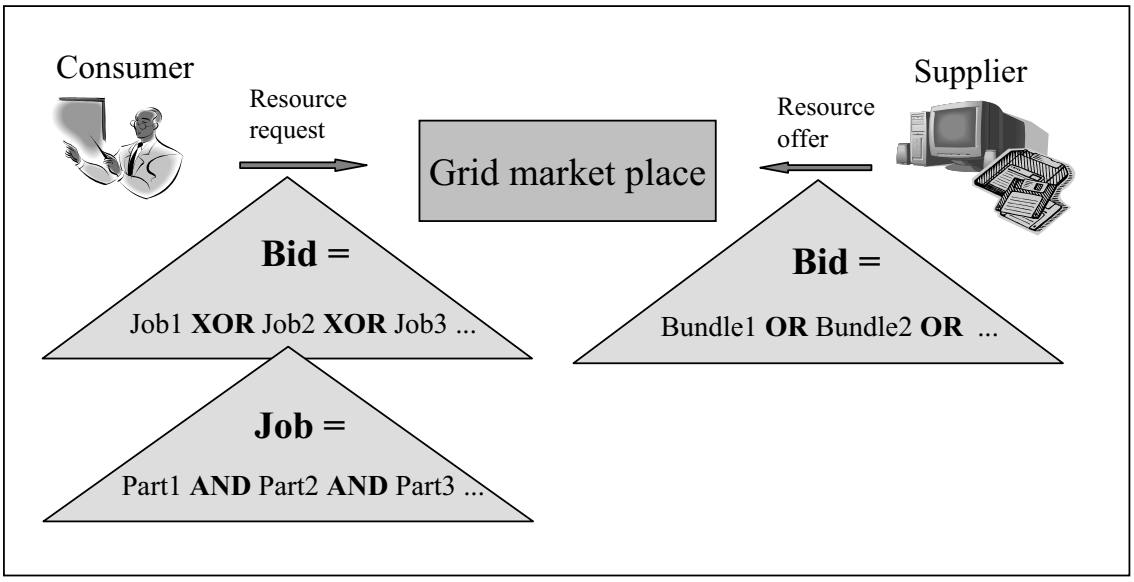


Figure 12: Bid structures of agents

Let $R = \{1, \ldots, \bar{r}\}$ be the set of resources and $A_r = \{1, \ldots, \bar{a}_r\}$ the set of resource attributes with $\bar{r}$ and $\bar{a}$ as maximum quantities. To distinguish between demand and supply side, parameter $a_{ri}^{nj} \in A_r$ applies to resource attributes of jobs and parameter $d_{ri}^{mb} \in A_r$ to bundles. Regarding the scenario, two resources $R = \{1, 2\}$ are traded

[9] X *AND* Y $(X \wedge Y)$ means XY but not $\emptyset$, X or Y.

[10] The notations *supplier* and *supplier bid* are used equally in this context.

[11] X *OR* Y $(X \vee Y)$ means $\emptyset$, X, Y or XY.

containing three attributes per resource $A_r = \{1, 2, 3\}$. They indicate the minimal required quantity or quality of service. It is assumed that jobs and bundles are always composed of both resources. Table 3 specifies how resources and attributes are modeled to meet the scenario requirements.

Computation service is described by resource 1. Attribute a_{11}^{nj} indicates the required quantity of CPUs, a_{12}^{nj} the CPU speed and a_{13}^{nj} the RAM size. Storage service, resource 2, contains the attributes a_{21}^{nj} that represents the number of hard disks, a_{22}^{nj} that indicates the throughput and a_{23}^{nj} that gives the size of disks. For simplification, the value ranges of the attributes 'CPU speed' and 'storage throughput' are split in three discrete intervals and can thus take the values $\{1, 2, 3\}$. For 'CPU speed', the intervals signify:

- Interval 1 (slow) to CPU speed < 1.5 GHz
- Interval 2 (medium) to CPU speed between 1.5 and 3 GHz
- Interval 3 (fast) to CPU speed > 3 GHz

Analogously, the intervals of 'storage throughput' signify:

- Interval 1 (slow) to throughput < 10 MB/s
- Interval 2 (medium) to throughput between 10 MB/s and 500 MB/s
- Interval 3 (fast) to throughput > 500 MB/s

In the present concept of discrete time slots, the length of the allocation horizon is defined by $T = \{1, \ldots, \bar{t}\}$. The maximum number of time slots $\bar{t}$ is determined exogenously in the market configuration. Each $t \in T$ determines a time slot with a fixed and uniform length. Agents indicate time frames of availability and consumers additionally the number of required time slots. To enable allocation, this job duration q^{nj} - which is limited by the execution time frame $[e^{nj}, l^{nj}]$ - has to meet the bundle's time frame $[c^{mb}, f^{mb}]$. Thereby it is assumed that the job duration does not exceed to the time frame's length, so $q^{nj} \leq l^{nj} - e^{nj} + 1$.

Job preferences are indicated as valuations $v^{nj} \in \mathbb{R}^+$ and reservation prices for bundles per time slot by $p^{mb} \in \mathbb{R}_0^+$. Parallelization of job parts within a job is denoted $\gamma^{nj} \in \{0, 1\}$. In case of activated parallelization ($\gamma^{nj} = 1$), the job parts are allocated in the same time slots and $\gamma^{nj} = 0$ otherwise.

Parameter	Range	Signification
R	$\{1, 2\}$	Set of resources
$\bar{r}$	$\bar{r} = 2$	Number of resources
A_r	$\{1, 2, 3\}$	Set of resource attributes
$\bar{a}_r$	$\bar{a}_r = 3$	Number of attributes
a_{11}^{nj}, d_{11}^{mb}	$a_{11}^{nj}, d_{11}^{mb} \in \mathbb{N}$	CPU quantity
a_{12}^{nj}, d_{12}^{mb}	$\{1, 2, 3\}$	CPU speed in GHz
a_{13}^{nj}, d_{13}^{mb}	$a_{13}^{nj}, d_{13}^{mb} \in \mathbb{N}$	RAM size in MB
a_{21}^{nj}, d_{21}^{mb}	$a_{21}^{nj}, d_{21}^{mb} \in \mathbb{N}$	Disk quantity
a_{22}^{nj}, d_{22}^{mb}	$\{1, 2, 3\}$	Storage throughput in MB/s
a_{23}^{nj}, d_{23}^{mb}	$a_{23}^{nj}, d_{23}^{mb} \in \mathbb{N}$	Disk size in GB

Table 3: Signification and ranges of scenario-specific parameters.

As some parameters appear only in jobs or in bundles and others are expressed differently, Table 4 shows the associated parameters with jobs and bundles.

Parameter	Job	Bundle
Earliest available start time slot	e^{nj}	c^{mb}
Latest available end time slot	l^{nj}	f^{mb}
Duration	q^{nj}	-
Valuation	v^{nj}	-
Reservation price	-	p^{mb}
Parallelization	γ^{nj}	-
Number of job parts	α^{nj}	-
Resource attributes	a_{ri}^{nj}	d_{ri}^{mb}

Table 4: Comparison of specific parameters of jobs and bundles.

In conclusion, the combinatorial bids of agents are specified in the following way:

Consumer bid = Job 1 XOR Job 2 XOR ...

Supplier bid = Bundle 1 OR Bundle 2 OR ...

Job = $\{(\{a_{11}^{nj}, a_{12}^{nj}, a_{13}^{nj}\}, \{a_{21}^{nj}, a_{22}^{nj}, a_{23}^{nj}\}, e^{nj}, l^{nj}, q^{nj}), \alpha^{nj}, v^{nj}, \gamma^{nj}\}$

Bundle = $\{(\{d_{11}^{mb}, d_{12}^{mb}, d_{13}^{mb}\}, \{d_{21}^{mb}, d_{22}^{mb}, d_{23}^{mb}\}, c^{mb}, f^{mb}), p^{mb}\}$

Job example: {(CPU: {3, 2 GHz, 512 MB}, Storage: {2, 100 MB/s, 5 GB}, 2, 5, 2), 2, 8, 0}
Bundle example: {(CPU: {2, 2 GHz, 1 GB}, Storage: {3, 700 MB/s, 10 GB}, 1, 10), 4}

Figure 13 visualizes the sample job which has a valuation of 8 and consists of 2 job parts. For the duration of 2 time slots within time slots 2 and 5, each job part require the following performance:

- A computation service consisting of 3 CPUs with at least 2 GHz speed (interval 2) and at least 512 MB RAM and
- a storage service consisting of 2 hard disks with a throughput of at least 100 MB/s throughput (interval 2) and a disk size of at least 5 GB.

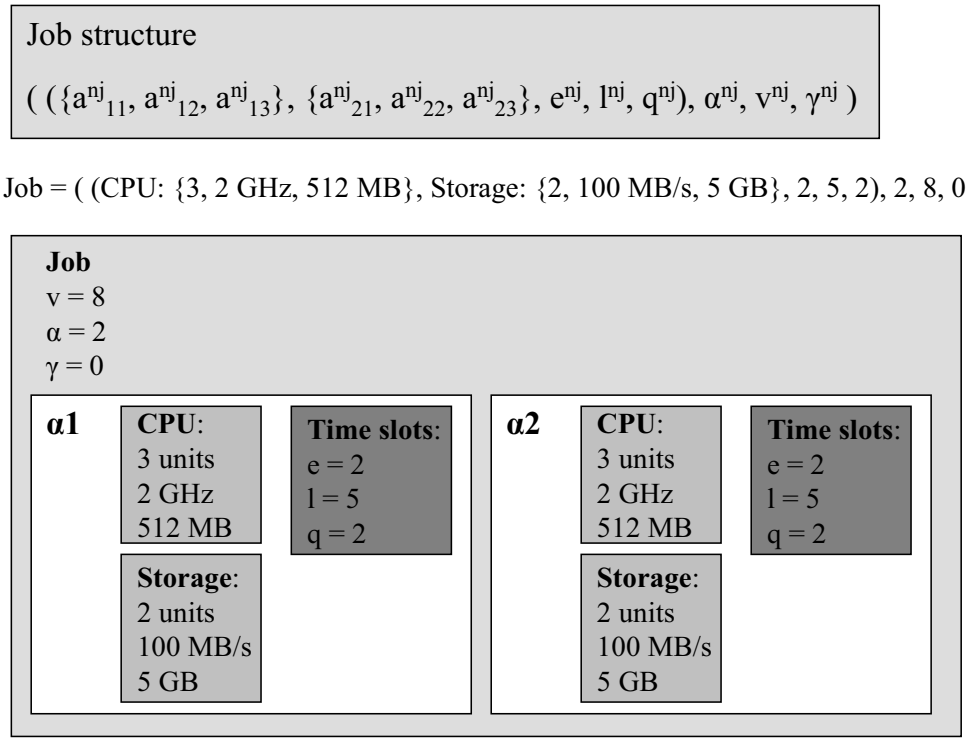


Figure 13: Example of a job with two job parts.

Figure 14 shows the sample bundle which has a reservation price of 4 and provides the following performance between time slot 1 and 10:

- Computation service consisting of 2 CPUs with at least 2 GHz speed (interval 2) and 1 GB RAM

- Storage service consisting of 3 hard disks with at least 700 MB/s throughput (interval 3) and 10 GB disk size

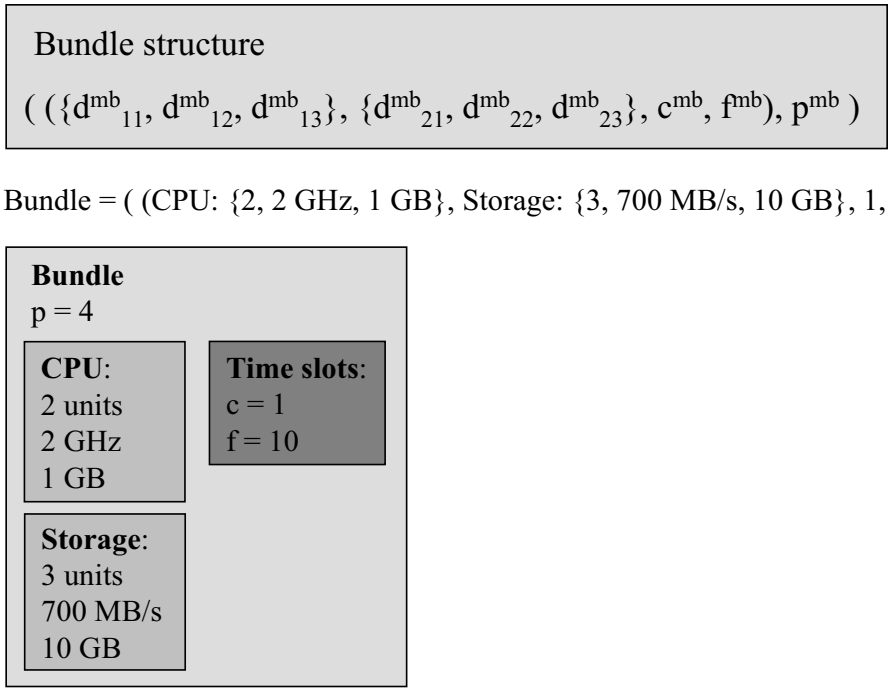


Figure 14: Example of a bundle.

5.2 The Allocation Mechanism

The WDP is formulated in form of a linear integer program. It is started from a first model whose performance is improved during the process of development by paraphrasing some time-consuming features. The result is the final model formulation. Firstly, two decision variables are introduced. Binary variable z specifies if job $j \in J^n$ of consumer bid n is allocated ($z^{nj} = 1$) or not ($z^{nj} = 0$). The second binary variable y indicates whether job part $\alpha \in Q^{nj}$ of job j of consumer bid n is allocated to bundle $b \in B^m$ of supplier bid m in time slot t ($y^{nj\alpha}_{mb,t} = 1$) or not ($y^{nj\alpha}_{mb,t} = 0$).

One of the major improvements is the change in meaning of variable y. In the first model, $y^{nj\alpha}_{mb,t} = 1$ means that job part α of job j is allocated to bundle b in time slot t. The value 1 is assigned to all allocated time slots in contrast to the final model. Once job part α is allocated to bundle b, the variable takes this value 1 only in the first time slot of allocation (see Figure 15). In other words, if the duration of the job part is longer than one time slot, $y^{nj\alpha}_{mb,t}$ takes the value 1 in the first time slot

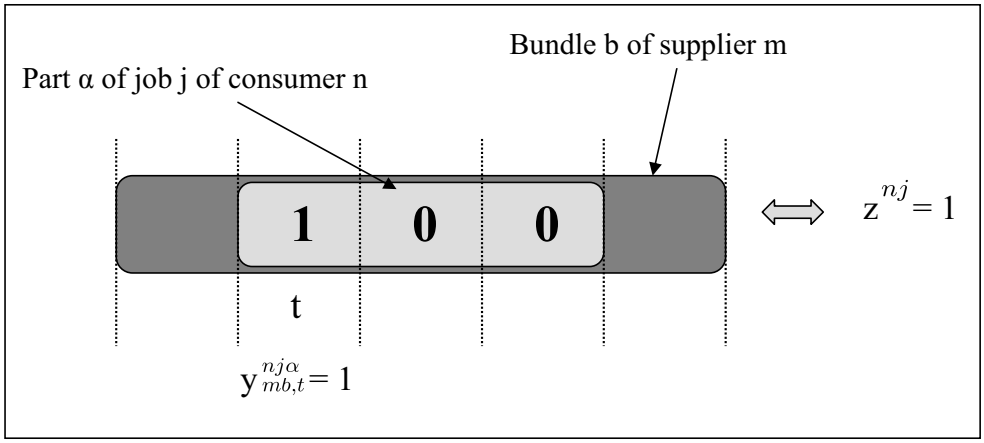


Figure 15: Signification of decision variable y in the final model formulation.

and the value 0 in the successive time slots. This change in signification simplifies the expression of time consistency drastically. In the following, the final model is presented and the improvement process is highlighted.

5.2.1 Mathematical Formulation

Objective is the maximization of welfare V:

$$\max V = \sum_{n\in N}\sum_{j\in J^n} z^{nj}\, v^{nj} \;- \sum_{m\in M}\sum_{b\in B^m}\sum_{n\in N}\sum_{j\in J^n}\sum_{\alpha\in Q^{nj}}\sum_{t\in T} y^{nj\alpha}_{mb,t}\; p^{mb}\, q^{nj} \qquad (1)$$

The objective function aims at maximizing the social welfare of participants. This is represented by the difference of the sum of job valuations v^{nj} for allocated jobs and the sum of reservation prices p^{mb} of allocated bundles in the associated time slots.

subject to

$$\sum_{j\in J^n} z^{nj} \leq 1, \quad \forall\, n \in N \qquad (2)$$

Constraints (2) ensure the representation substitute in job allocation (XOR). Only one job j of consumer bid n can be allocated.

$$\sum_{m\in M}\sum_{b\in B^m}\sum_{t\in T} y^{nj\alpha}_{mb,t} - z^{nj} = 0, \quad \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj} \qquad (3)$$

Constraints (3) force variables y and z to be coherent. Any job part α of an allocated consumer bid ($z^{nj} = 1$) is assigned only once over all bundles and time slots.

This guarantees coupling. Job part α is considered that is allocated to two different bundles b_1 and b_2. As always the first time slot of allocation is marked by $y^{nj\alpha}_{mb,t} = 1$, for this allocated job part is $y^{nj\alpha}_{mb,t} = 2$ but $z^{nj} = 1$. One can see that the constraints are violated in case that a job part is allocated to more than one bundle.

The first model describes the condition of job part allocation contiguously in time in the following way:

$$y^{nj\alpha}_{mb,t'} + y^{nj\alpha}_{mb,t-1} - y^{nj\alpha}_{mb,t} \geq 0, \quad 2 \leq t \leq t' \leq t + q^{nj} - 1, \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t, t' \in T \tag{4}$$

$$y^{nj\alpha}_{mb,t'} - y^{nj\alpha}_{mb,1} \geq 0, \quad 1 \leq t' \leq q^{nj}, \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t' \in T \tag{5}$$

Starting in time slot $t = 2$, constraints (4) are operated for all consumers, jobs, job parts, suppliers and bundles as well as the duration of jobs. As soon as a job part is allocated, each time slot within the time frame has to be allocated to that job part as well. Thus the allocation of each job part in ensured for exactly q^{nj} time slots. Constraints (5) consider time consistency in case that job part allocation starts in $t = 1$.

These constraints are extremely time-consuming. Due to the change in meaning of variable y, the conditions are represented in the final model as follows:

$$\sum_{n \in N} \sum_{j \in J^n} \sum_{\alpha \in Q^{nj}} \sum_{t' = t - q^{nj} + 1}^{t} y^{nj\alpha}_{mb,t'} \leq 1, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, t \in [c^{mb}, f^{mb}] \tag{6}$$

Consistency in time and avoidance of overlapping jobs are expressed by constraints (6). For each supplier and bundle are merely considered time slots within the availability time frame $t \in [c^{mf}, f^{mb}]$ of the bundle. For each job part α, the first time slot of allocation is marked by setting $y^{nj\alpha}_{mb,t} = 1$. This applies only to this single time slot. Due to check of time slot $t' = t - q^{nj} + 1$, in the following $q^{nj} - 1$ time slots no other job part can be allocated and the overlapping of jobs on a single bundle is avoided. Considered are two jobs j_1 and j_2 that overlap in time slot t on bundle b. The two jobs start operating after time slots $t - q^{n_1 j_1} + 1$ and $t - q^{n_2 j_2} + 1$, otherwise they would not be in process in time slot t. One can see that $\sum_{t' = t - q^{nj} + 1}^{t} y^{nj\alpha}_{mb,t'} \geq 2$. In this case the constraints are violated. So the guarantee of no overlapping jobs is always

given, if these conditions are fulfilled for all bundles and time slots. At the same time, the allocation of $q^{nj} - 1$ time slots in succession ensures consistency in time.

$$\gamma^{nj} \sum_{m \in M} \sum_{b \in B^m} y_{mb,t}^{nj\alpha} - \gamma^{nj} \sum_{m \in M} \sum_{b \in B^m} y_{mb,t}^{nj\alpha'} = 0, \quad \forall\, n \in N, \forall\, j \in J^n, \forall\, t \in [e^{nj}, l^{nj}], \\ \forall\, \alpha \neq \alpha' \in Q^{nj} \tag{7}$$

Constraints (7) guarantee the allocation of job parts of a given job in exactly the same time slots within the job's time frame, i.e. in parallel. Indeed, these constraints are a tautology whether $\gamma^{nj} = 0$ and activate parallelization in the case of $\gamma^{nj} = 1$.

In the first model, the conformity checks of availability time frames and resource attributes are expressed by

$$e^{nj}\, y_{mb,t}^{nj\alpha} \leq t\, y_{mb,t}^{nj\alpha} \leq l^{nj}\, y_{mb,t}^{nj\alpha}, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \\ \forall\, \alpha \in Q^{nj}, \forall\, t \in T \tag{8}$$

$$c^{mb}\, y_{mb,t}^{nj\alpha} \leq t\, y_{mb,t}^{nj\alpha} \leq f^{mb}\, y_{mb,t}^{nj\alpha}, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \\ \forall\, \alpha \in Q^{nj}, \forall\, t \in T \tag{9}$$

The collection of constraints (8) and (9) expresses the matching of availability time frames. Allocation can only take effect within the interval in which the job's time frame $[e^{nj},\, l^{nj}]$ and the bundle's time frame $[c^{mb},\, f^{mb}]$ overlap.

$$a_{ri}^{nj}\, y_{mb,t}^{nj\alpha} - d_{ri}^{mb}\, y_{mb,t}^{nj\alpha} \leq 0, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \\ \forall\, \alpha \in Q^{nj}, \forall\, t \in T, \forall\, r \in R, \forall\, i \in A_r \tag{10}$$

Constraints (10) check quality and quantity attributes. Matching can only be effected if the bundle's attributes d_{ri}^{mb} are in keeping with the job's attributes a_{ri}^{nj}.

These constraints are formulated exclusively in the final model so that right from the beginning non-feasible allocations are not taken into consideration.

$$y_{mb,t}^{nj\alpha} = 0, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \\ \forall\, t \in T \setminus [\max(e^{nj}, c^{mb}), \min(l^{nj}, f^{mb}) - q^{nj} + 1] \tag{11}$$

$$y_{mb,t}^{nj\alpha} = 0, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \\ \forall\, t \in T, \forall\, r \in R, \forall\, i \in A_r \setminus \{a_{ri}^{nj} \leq d_{ri}^{mb}\} \tag{12}$$

Time constraints (11) ensure allocation within the period, in which job and bundle time frames overlap. As $y^{nj\alpha}_{mb,t} = 1$ indicates that job part α starts in time slot t, the constraints force variable y to be zero if time intervals do not match. Constraints (12) consider the quality and quantity attributes. Allocation of job part α to bundle b is impossible if the value of the job's attribute a^{nj}_{ri} is greater than the value of the respective attribute d^{mb}_{ri} of bundle b.

$$z^{nj} \in \{0,1\}, \quad \forall\, n \in N, \forall\, j \in J^n \tag{13}$$

$$y^{nj\alpha}_{mb,t} \in \{0,1\}, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t \in T \tag{14}$$

Constraints (13) indicate whether job j of consumer n is allocated ($z^{nj} = 1$) or not ($z^{nj} = 0$). Constraints (14) specify if job part α of job j of consumer n starts on bundle b of supplier m in time slot t ($y^{nj\alpha}_{mb,t} = 1$) or not ($y^{nj\alpha}_{mb,t} = 0$).

In Appendix A, the two model formulations are listed separately in their complete form.

5.2.2 Sample Schedule

The schedule of a sample allocation that meets the scenario requirements is demonstrated by considering 5 time slots, 3 consumer bids and 2 supplier bids.

Cons. bid n	Job j^n	v^{nj}	α^{nj}	e^{nj}	l^{nj}	q^{nj}	γ^{nj}	a^{nj}_{11}	a^{nj}_{12}	a^{nj}_{13}	a^{nj}_{21}	a^{nj}_{22}	a^{nj}_{23}
1	1	60	2	1	4	3	1	2	2	2048	1	3	100
	2	10	1	3	5	1	0	2	2	3096	2	1	50
2	1	70	1	1	4	3	0	4	1	512	5	1	80
	2	40	2	2	4	3	1	1	2	2048	1	3	80
3	1	90	3	1	5	4	0	2	2	2048	2	2	50

Table 5: Consumer bids with XOR connection.

Table 5 gives the consumer bids with associated jobs, job parts, time and allocation characteristics. For instance, job 1 of consumer bid 1 is valued with 60 and consists of 2 job parts. The job with the duration of 3 time slots has to be allocated between time slot 1 and 4. The job demands a computation service consisting of

2 CPUs with speed between 1.5 and 4 GHz (interval 2) and 2 GB RAM as well as a storage service consisting of 1 hard disk with a throughput of 1 GB/s (interval 3) and a disk size of 100 GB. As parallelization is activated, the job parts have to be allocated in the same time slots.

Supplier bid m	Bundle b^m	p^{mb}	c^{mb}	f^{mb}	d_{11}^{mb}	d_{12}^{mb}	d_{13}^{mb}	d_{21}^{mb}	d_{22}^{mb}	d_{23}^{mb}
1	1	3	1	5	5	2	3096	6	3	80
	2	4	1	5	5	2	3096	2	3	80
2	1	6	2	5	3	3	3096	7	2	100
	2	5	1	3	6	1	1024	8	3	80

Table 6: Supplier bids with OR connection.

Correspondingly, Table 6 gives the specification of supplier bids. For example, bundle 1 of supplier bid 1 indicates the reservation price of 3 per time slot and offers the following resources between time slot 1 and 5: a computation service consisting of 5 CPUs that have a speed between 1.5 and 3 GHz (interval 2) and 3 GB RAM as well as a storage service that consists of 6 hard disks with 1 GB/s throughput (interval 3) and 80 GB disk size.

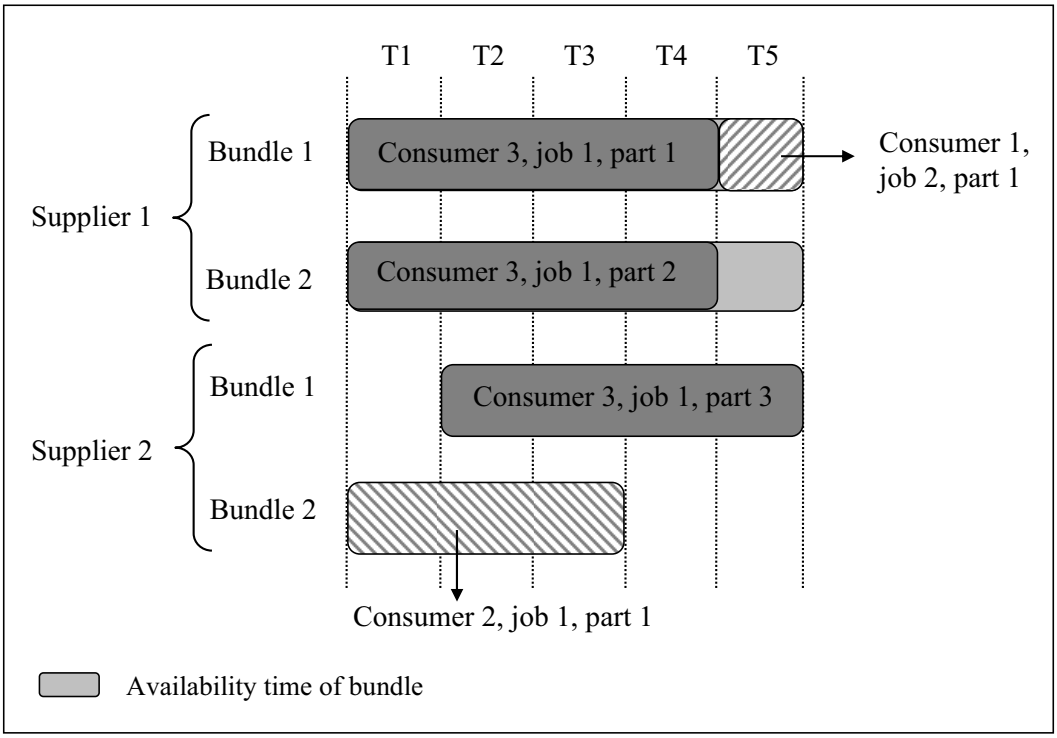


Figure 16: Allocation schedule of the example.

The generated test data is run with *GNU Linear Programming Kit (GLPK)* which is a tool for solving linear programming problems, mixed integer programming problems and other related problems. The first model yields an optimal allocation in 0.7 seconds and the final model in less than 0.1 seconds. As shown in Table 7 and Figure 16, job 1 of consumer bid 3 which consists of 3 job parts is allocated to bundles 1 and 2 of supplier bid 1 in time slots 1 to 4 and to bundle 1 of supplier bid 2 in time slots 2 to 5. Job 2 of consumer 1 is allocated to bundle 1 of supplier 1 in time slot 5. So, bundle 1 of supplier 1 is reserved in its complete availability time in contrast to bundle 2 which is idle in time slot 5. As there are no more unsatisfied consumers, this time slot stays idle. In a market with exceeding demand, time slots may also stay idle in case that unallocated jobs have unsuitable characteristics. For instance, the requirement for matching job 1 of consumer bid 3 to bundle 2 of supplier bid 1 is that the bundle's resource attributes are equal or exceed those of the job. The job would not match up bundle 2 of supplier 2 as e.g. the bundle's attribute 'CPU speed' does not meet the job's performance demand ($a_{12}^{31} > d_{12}^{22}$). To determine the total welfare, the partial welfare of each allocation is calculated and summed up. Thereby, the job valuations are divided by the number of job parts contained in the job. Thus, the welfare of the optimal solution achieves 100 and is calculated by $(10 + 90 + 70) - (3 \cdot 4 + 3 \cdot 1 + 4 \cdot 4 + 6 \cdot 4 + 5 \cdot 3)$.

Supplier			Consumer		Time slot	Partial welfare
m	b^m	p^{mb}	$n/j^n/\alpha^{nj}$	v^{nj} per α^{nj}	t	
1	1	3	3/1/1	30	1-4	18
			1/2/1	10	5	7
	2	4	3/1/2	30	1-4	14
2	1	6	3/1/3	30	2-5	6
	2	5	2/1/1	70	1-3	55
Total welfare is						100

Table 7: The sample allocation scheme shows the optimal allocation.

Note that the partial welfare of bundle 2 of supplier 2 is the greatest because job 1 of consumer bid 2 values the execution with 70. The job does not need all provided resources of the bundle and there are other jobs that use even fewer resources. For

instance, job 2 of consumer bid 1 only uses 2 of the 5 supplied CPUs and 2 of 6 disks. This job drastically overpays the bundle as it would be satisfied with fewer resources.

5.3 Dominant Subsets

With increasing problem complexity, it becomes more and more important to find tools that limit the time needed for problem solving. In this context, the idea of preferences in form of dominant subsets is promoted.

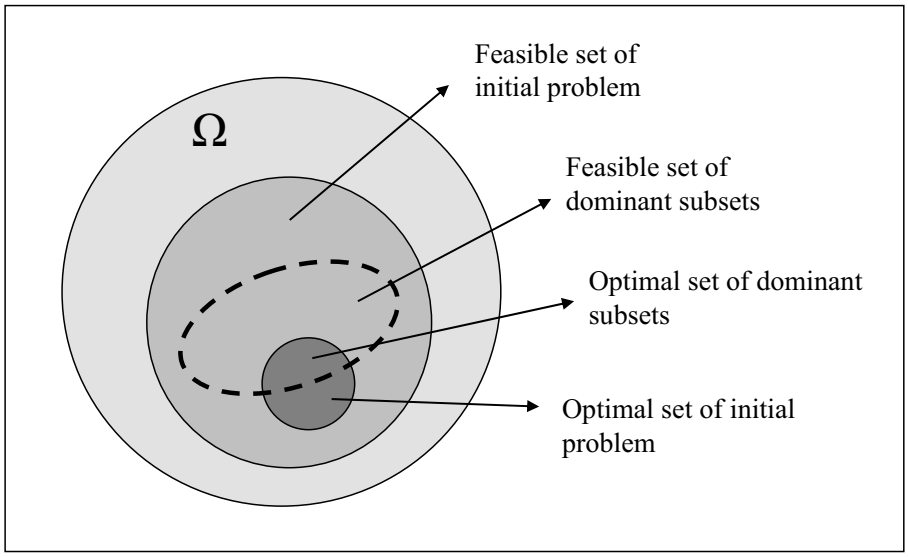


Figure 17: Solution sets with and without dominant subsets.

In this section, two specifications of *dominant subsets* are considered. A subset of solutions is said to be dominant whether it contains a least one of the optimal solutions. Figure 17 shows the different solution sets. In practice, constraints are added to problems that exclude unimportant solutions in advance. Thus the set of feasible solutions as well as the set of optimal solutions are restrained. The algorithm takes advantage of the smaller range of feasible solutions. In spite of this benefit, the optimization problem still contains multiple solutions that yield equal welfare. Pekeč and Rothkopf (2003) describe this situation as *ties occurring*. The probability of ties has to be minimized in a system, in which transparency and fairness matter. An expressive bidding language contributes to avoid ties. As larger the range of biddable combinations is, the less frequently ties occur. Two ways are proposed to break ties: Either a winning allocation is selected at random among

all optimal solutions or a preference ordering on allocations is defined. Referring to preference ordering, dominant subsets are implemented that are able to overcome ties. As test runs have confirmed, the optimal overall welfare can be achieved by multiple allocations. For instance, the mechanism delivers several equal solutions for each allocated job that consists of more than one job part. As they have the same characteristics, the optimal solution can be found without considering all allocation possibilities. Hence, it is sufficient to look for solutions in the dominant subset.

5.3.1 Sequential Allocation of Job Parts

Based on the homogeneity of job parts within a job, any part can be equally allocated to a bundle. This means that they are interchangeable. For instance, a feasible solution is considered in which job j composed of two parts α_1 and α_2 is respectively allocated to bundles b_1 and b_2 of supplier m. The inverse solution in which α_1 is allocated to bundle b_2 and α_2 to bundle b_1 is equally feasible and yields the same welfare.

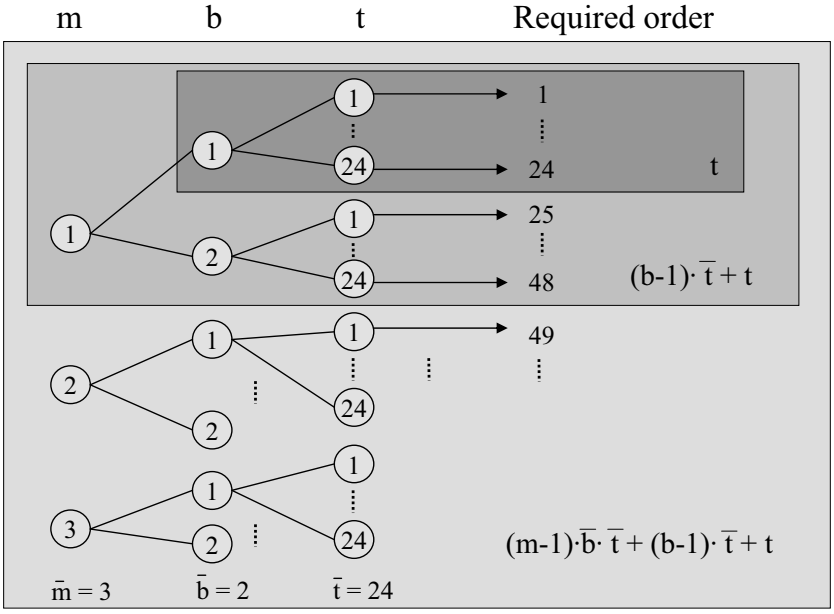


Figure 18: Tree structure to determine the index order of supplier-bundle-time slot combinations.

From this establishment follows that there are multiple equivalent solutions that are obtained by swapping job parts of the same job within any given feasible solution. Consequently, the check of all such equivalent solutions may be avoided by the

constraints

$$\sum_{m\in M}\sum_{b\in B^m}\sum_{t\in T} y_{mb,t}^{nj\alpha}\; s(m,b,t) - \sum_{m\in M}\sum_{b\in B^m}\sum_{t\in T} y_{mb,t}^{nj\alpha+1}\, s(m,b,t) \leq 0,$$
$$\forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj} \qquad (15)$$

The integer $s(m,b,t)$ and the function s enable to define a numerical order of suppliers, bundles and time slots in form of a tree (Figure 18). Starting with suppliers in the first row of nodes and bundles in the second row of nodes, the time slots form the leaves. Each branch of the tree represents a possible supplier-bundle-time slot combination and gets a separate number beginning from 1. Thus, all branches can be accessed by this index number $s(m,b,t)$. Function s assigns to each branch the corresponding index number. Thus these constraints imply that the k lowest α of job j has to be matched with the k lowest branch (according to function s) associated to job j. More precisely, s is defined by

$$s(m,b,t) = (m-1)\cdot\bar{b}\cdot\bar{t} + (b-1)\cdot\bar{t} + t. \qquad (16)$$

Figure 19 shows equivalent allocations in term of welfare without restrictions of dominant subsets on the left and an allocation that meets the requirements of dominant subset 1 on the right.

5.3.2 Time Displacement of Jobs

The second dominant subset can be seen as an application of *active schedules* in scheduling theory. An active schedule is a particular subset of all possible schedules and contains a subset of the optimal schedules (Giffler and Thompson 1960). The goal is to avoid the random allocation of a job within its time frame. Starting from a feasible allocation, all remaining alternative allocations of a job are examined that yield the same welfare and that are generated only from varying the job's starting time. The idea of this dominant subset is to shift jobs as far as possible to the left, i.e. toward their start time so that they start as soon as possible. Parameter $st(j,b)$ denotes the start time of job j that is allocated to bundle b. Job j is shifted as far as possible to the left on bundle b if one of these three conditions is satisfied:

- $st(j,b) = \max(e^{nj}, c^{mb})$;

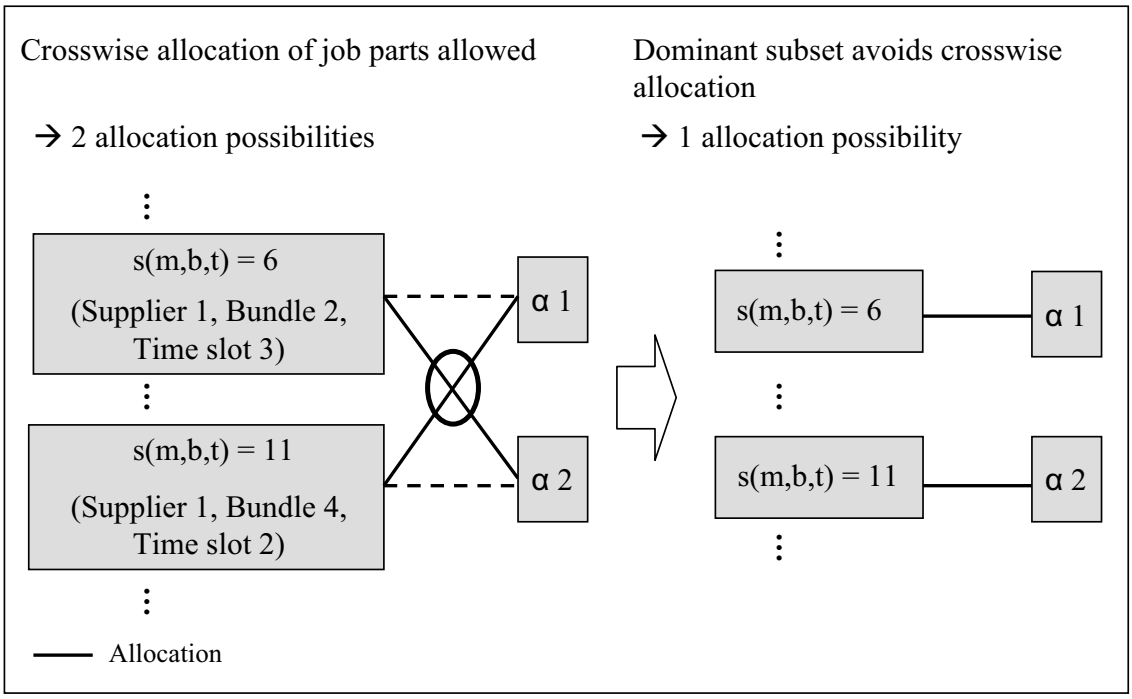


Figure 19: Dominant subset to swap job parts according to constraints (15): Sequential Allocation of Job Parts.

- $\exists\, n' \in N$, $j' \in J$, such that $\sum_{\alpha \in Q^{n'j'}} y^{n'j'\alpha}_{mb,st(j,b)-q^{n'j'}} = 1$;
- $\gamma^{nj} = 1$.

The first condition means that job j cannot be allocated earlier due to either its earliest start time, that is $e^{nj} = t$, or the earliest availability time of bundle b ($c^{mb} = t$) to which job j is assigned. The second condition implies the existence of another job j' that is executed on the same bundle as job j so that there is no idle time between the execution of job j and j'. The third condition means that jobs with activated parallelization restriction are excluded of consideration as the allocation time of job parts interdependent. Figure 20 illustrates the idea that is formulated by constraints (17).

$$\sum_{n \in N} \sum_{\substack{j \in J^n \\ t > e^{nj}}} (1 - \gamma^{nj}) \sum_{\alpha \in Q^{nj}} y^{nj\alpha}_{mb,t} - \sum_{n \in N} \sum_{\substack{j \in J^n \\ t > q^{nj}}} \sum_{\alpha \in Q^{nj}} y^{nj\alpha}_{mb,t-q^{nj}} \leq 0,$$

$$\forall\, m \in M, \forall\, b \in B^m, \forall\, t \in]c^{mb}, f^{mb}] \qquad (17)$$

The left of constraints (17) specifies if a job is allocated in time slot t and could have been allocated earlier, so in $[e^{nj}, t]$ under the condition that $t > e^{nj}$ and inactive

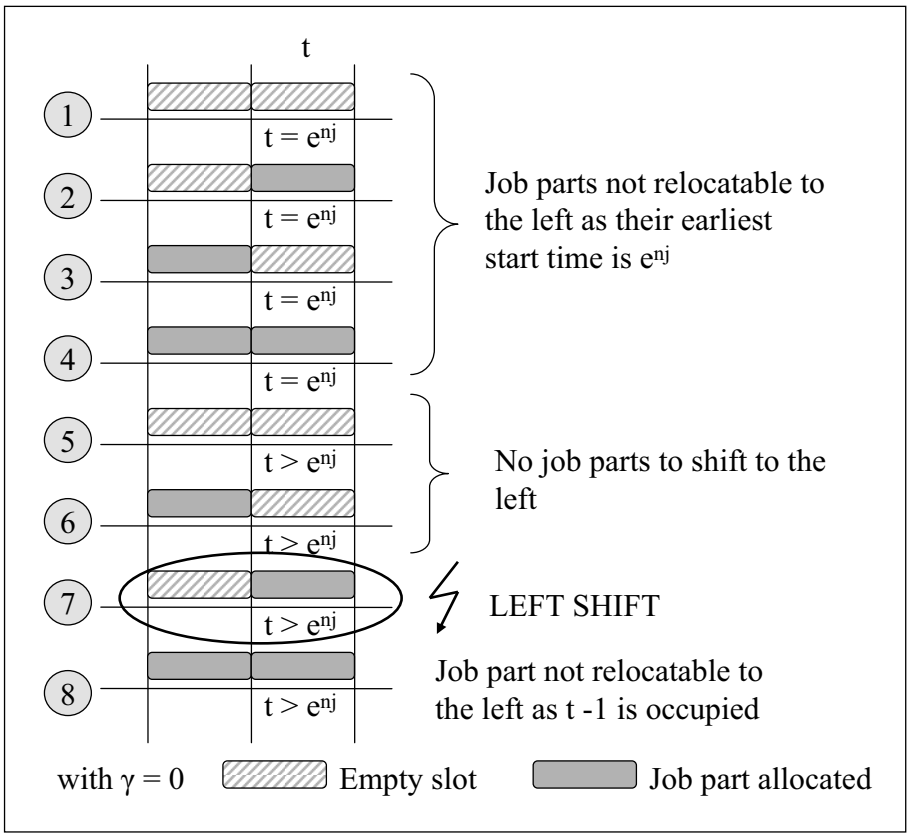


Figure 20: Potential situation of allocation of a job part. Case 7 shows the case that the job part is shifted to the left according to dominant subset 2.

parallelization ($\gamma^{nj} = 0$). Once a job part fulfills this part of the equation, cases 1 to 6 in Figure 20 can be excluded. The right of the constraints specifies whether any job ends in $t-1$. If not, case 8 is also excluded and the job has to be shifted to the left as shown in case 7. Thereby, such jobs are considered that meet the condition $t > q^{nj}$. Otherwise, jobs with longer duration would be shifted beyond the beginning of the allocation horizon. Thus, these constraints exclude alternative allocation and force job parts to be shifted as far as possible away from their deadlines. In case that applications are time-sensitive, the resulting allocation adds value to consumers.

5.4 The Pricing

The question of how to calculate payments for winning bids after the end of the auction that satisfy economic properties is addressed in this section. The well-known impossibility theorem of Myerson and Satterthwaite (1981) for bilateral trades demonstrates that no exchange can be allocation-efficient, budget-balanced and individually rational at the same time. Thus, the mechanism designer has to prioritize economic properties dependent on the desired outcome.

5.4.1 VCG Pricing

Various pricing mechanisms have been suggested according to the implemented market mechanism and the intent pursued. In periodic single-shot combinatorial auctions, payments are neither calculated ex-ante nor approximated in an iterative process. Approaches in pricing schemes for combinatorial auctions are based on the distribution of overall welfare involving valuations and reservation prices, such as the Vickrey-Clarke-Groves pricing scheme. VCG mechanisms considers the utility gain of the participation of agents and presents the only mechanism combining incentive compatibility and efficiency in allocation while guarantying individual rationality (Schnizler et. al. 2006). The idea is that each agent i obtains a Vickrey discount $\Delta_{vick,i}$ corresponding to the gain in welfare which results from his participation. More precisely, the granted discounts for participants are the difference in welfare achieved with and without participation of each agent. Let V^* denote the surplus of the optimal allocation and $(V_{-i})^*$ the surplus without the participation of agent i. The Vickrey discount to agent i is calculated by $\Delta_{vick,i} = V^* - (V_{-i})^*$ and expresses

the social gain of including agent i in the allocation. Thus, the payment of a consumer is the result of the discount subtracted from the job's valuation. A supplier gains the discount in addition to the reservation prices. The payment calculation of the VCG pricing scheme does not balance the budget.

If the WDP is solved to optimum, the VCG pricing rule yields incentive compatible payments and motivates agents to report their true valuations. This leads to efficient allocations. Although nothing bars them from misreporting, truthful bidding is a weakly dominant strategy (de Vries and Vohra 2003). Requiring long runtime by solving the WDP $n-1$ times, the VCG pricing may be used for batch mechanisms but not for interactive applications that demand fast market clearings. To guarantee incentive compatibility, the VCG pricing requires the exact solution of the allocation problem. Therefore, it is not applicable for others than exact mechanisms. Due to the permanently needed subsidization from the outside, other pricing mechanisms such as *approximate VCG pricing* and *k-pricing* have been developed meeting the requirement of budget balance.

5.4.2 Approximate VCG Pricing

The approach of Parkes et. al. (2001) is based on the VCG mechanism and adds payment rules that minimize the distance to Vickrey discounts for some metrics. Approximate VCG pricing meets the economic properties of individual rationality and budget balance, yields efficient allocations and provides incentives for revealing truthful valuations. However, the mechanism is not incentive compatible and removes more or less easy opportunities for manipulation. The optimization problem is formulated as linear program. Let $I^* \subseteq \Phi$ denote the set of trading agents. Discounts $\Delta = (\Delta_1, \ldots, \Delta_I)$ are computed to minimize the distance $\mathbf{L}(\Delta, \Delta_{vick})$ to Vickrey discounts Δ_{vick} for a suitable distance function $\mathbf{L}$.

$$\min_{\Delta} \mathbf{L}(\Delta, \Delta_{vick}) \tag{18}$$

$$\text{s.t.} \quad \sum_{i \in I^*} \Delta_i \leq V^* \tag{19}$$

$$\Delta_i \leq \Delta_{vick,i}, \quad \forall\, i \in I^* \tag{20}$$

$$\Delta_i \geq 0, \quad \forall\, i \in I^* \tag{21}$$

One of this payment rules is the *threshold rule* that is formulated as

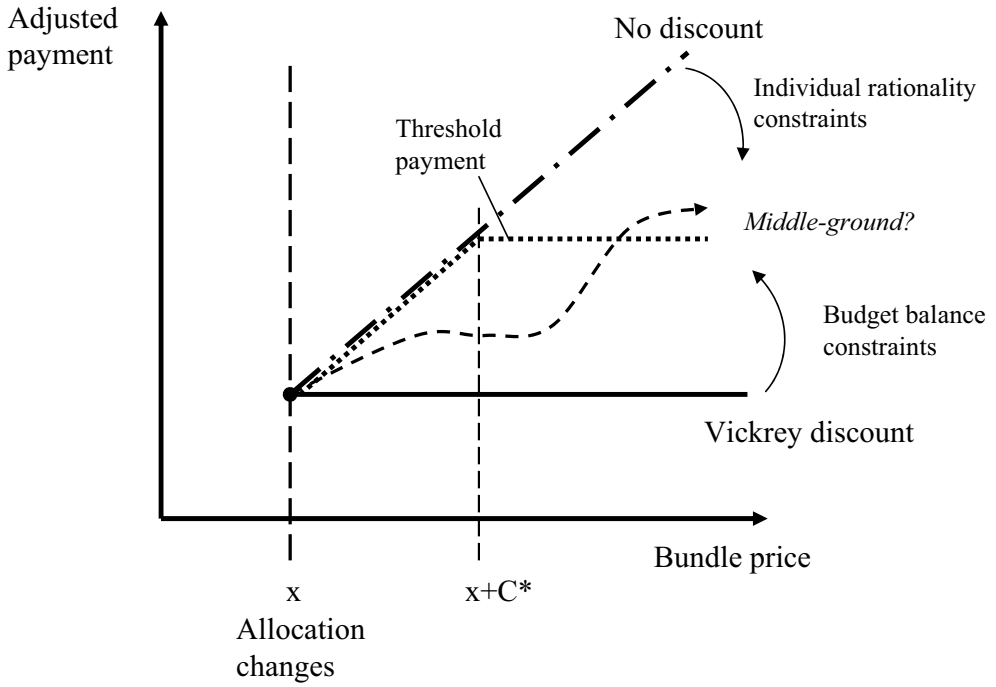


Figure 21: Comparison of payment structures: Approximate VCG pricing with threshold rule, VCG pricing with VCG discounts and 'no discounts' (according to Parkes et. al. 2001).

$$\Delta_i^*(C) = \max(0, \Delta_{vick,i} - C)$$

whereas Δ_i^* is the optimal allocation of discount to agent i and C the threshold parameter. In the threshold scheme, agents whose payments have a greater distance than a certain threshold value from their Vickrey payments, receive a surplus. The goal is to retrieve the optimal threshold parameter C^* that represents the smallest value for which the total sum of payments maintains budget balance. Agents whose Vickrey discount exceeds the threshold, receive a discount in the amount of the threshold payment on their Vickrey payments ($\Delta_{vick,i} - C^*$), while the other agents receive no discount.

Figure 21 illustrates the adjusted payment[12] to agent i under VCG pricing and in combination with the threshold rule as well as the curve of 'no discount'. The curve of VCG discounts runs independently of bundle prices[13] and provides a flat adjusted payment in contrast to the threshold payments. The change in bundle prices influences the threshold payments as long as bundle prices do not exceed the

[12]The term 'adjusted payment' denotes the effective selling price of the bundle.

[13]In this context, the term 'bundle price' denotes the value of a resource combination.

specific bundle price $x+C^*$ and are close to the critical price x which is still sufficient to maintain the same allocation. With rising bundle price, the curve of 'no discount' increases proportionally to the adjusted payments. Agents are tempted to approach the smallest price that still maintains the same allocation. The performance of the threshold payment rule is on the one hand the dynamic adoption to changes in bundle prices and on the other hand the supply of consistent payments beginning from the bundle price $x + C^*$.

5.4.3 K-Pricing

Due to the time-consuming calculation of payments, (approximate) VCG pricing is not applicable in practice. K-pricing relaxes the constraints of allocative efficiency and yields approximately truthful prices in polynomial runtime. The underlying idea is the distribution of local surplus according to parameter k. This factor $k \in [0, 1]$ determines the share of local surplus, denoted by π, between agents. In case of $k = 0.5$, the surplus is evenly distributed to consumers and suppliers. k and $(1 - k)$ determine the share of π that is dedicated to consumers and suppliers respectively. By varying the value of k, payments can easily be influenced in order to privilege either the demand ($k > 0.5$) or supply side ($k < 0.5$) of the market.

5.4.4 Sample Pricing

In this section, the payment structure under VCG pricing, approximate VCG pricing and k-pricing are exemplified. It is based on the sample allocation schedule of page 57. In this context, the term 'payment' refers to both consumers and suppliers. It means an expense for consumers and an income for suppliers.

Step (1) in the VCG pricing scheme is the calculation of the optimal welfare without participation of each agent i, $(V_{-i})^*$, based on the overall optimal welfare $V^* = 100$ and the resulting Vickrey discounts $\Delta_{vick,i}$ (Table 8). In step (2) and (3), the discounts are credited against valuations and *cumulated reservation prices (CRP)*. The valuation of consumer i is discounted by $\Delta_{vick,i}$ to obtain his payment. As reservation prices are given per time slot, the cumulated reservation price of one supplier is determined by the sum of the product of reservation prices of allocated bundles multiplied by the number of time slots over all allocated bundles of this

supplier, i.e.

$$CRP(m) = \sum_{b \in B^m} \sum_{n \in N} \sum_{j \in J^n} \sum_{\alpha \in Q^{nj}} \sum_{t \in T} y_{mb,t}^{nj\alpha} \, p^{mb} \, q^{nj}. \tag{22}$$

In other words, the cumulated reservation price is the value of allocated resources of a supplier assessed based on reservation prices. Thus, the $CRP(m)$ of supplier 1 is $CRP(1) = 3 \cdot 1 + 3 \cdot 4 + 4 \cdot 4 = 31$. Vickrey discount and cumulated reservation price add up to the supplier's payment.

Step		n_1	n_2	n_3	m_1		m_2	
(1)	$(V_{-i})^*$	93	45	68	59		68	
	$\Delta_{vick,i}$	7	55	32	41		32	
(2)	v^{nj}	10	70	90				
	Payment of consumer $(v^{nj} - \Delta_{vick,i})$	-3	-15	-58				
(3)					b_1	b_2	b_1	b_2
	p^{mb}				3	4	6	5
	$\sum_{n \in N} \sum_{j \in J^n} \sum_{\alpha \in Q^{nj}} \sum_{t \in T} p^{mb} \, q^{nj}$				15	16	24	15
	$CRP(m)$				31		39	
	Payment of supplier $(CRP(m) + \Delta_{vick,i})$				+72		+71	

Table 8: VCG pricing.

The calculation of prices under the approximate VCG pricing scheme is based on Vickrey discounts. In order to achieve budget balance, Parkes et. al. (2001) propose several payment rules. The threshold rule is constructed from distance metric $L_2 = \sum_i (\Delta_{vick,i} - \Delta_i)^2$ (Table 9). The method of Lagrange multipliers is used to find Δ_i^*, the optimal allocation of discount to agent i, for all agents with $\Delta_i^* > 0$. Parameter $C \geq 0$ denotes the difference between Vickrey discounts and calculated discounts Δ_i^* across all agents and takes the value 13.4. An index is assigned to each agent in decreasing order according to its Vickrey discount. This order of agents is searched until index K is such that

$$\Delta_{vick,K+1} \leq C^* \leq \Delta_{vick,K}. \tag{23}$$

Budget balance is given for $C^* = 15$ calculated by

$$C^* = (\sum_{i=1}^{K} \Delta_{vick,i} - V^*)/K. \tag{24}$$

The final discount for each agent whose Vickrey discount exceeds 15 (so consumers 2 and 3 as well as suppliers 1 and 2) is determined by

$$\Delta_i^* = \Delta_{vick,i} - C^*. \tag{25}$$

The payment of consumers (suppliers) results from the difference of valuation (addition of cumulated reservation price) and discount Δ_i^*.

	n_1	n_2	n_3	m_1	m_2
$\Delta_{vick,i}$	7	55	32	41	32
v^{nj}, $CRP(m)$	10	70	90	31	39
Discount Δ_i^*	-	40	17	26	17
Payment	-10	-30	-73	+57	+56

Table 9: Approximate VCG pricing.

K-pricing provides the flexibility to determine the distribution of welfare between the two sides of the market from the outside. For instance, 0.4 is assigned to parameter k. Based on valuations and reservation prices, the local surplus π is calculated as difference between job valuation and *cumulated value (CV)* for each allocated job, i.e.

$$\pi(n,j) = v^{nj}\, z^{nj} - CV(n,j). \tag{26}$$

Thereby, cumulated value denotes the sum of the product of reservation prices from allocated bundles to the actual job and the associated number of allocated time slots over all allocated bundles. Algebraically, the CV of a single job is

$$CV(n,j) = \sum_{m\in M} \sum_{b\in B^m} \sum_{\alpha\in Q^{nj}} \sum_{t\in T} y_{mb,t}^{nj\alpha}\; p^{mb}\, q^{nj}. \tag{27}$$

Consequently, the local surplus of e.g. job 2 of consumer 1 is $\pi(1,2) = 10{\cdot}1 - 3{\cdot}1 = 7$. With this follows that consumer n has to pay an amount of

$$\bar{p}\,(n,k) = \sum_{j\in J^n} \left(v^{nj}\, z^{nj} \; - k \;\cdot\; \pi(n,j)\right), \tag{28}$$

i.e. $\bar{p}\,(1,\, k=0.4) = 10 \;-\; 0.4 \cdot 7 = 7.2$. Supplier m gets a payment in the amount of

$$\bar{p}\,(m,k) = \; CRP(m) + (1-k) \sum_{b\in B^m} \sum_{n\in N} \sum_{j\in J^n} \sum_{\alpha\in Q^{nj}} \sum_{t\in T} \frac{\pi(n,j)}{\omega}\; y_{mb,t}^{nj\alpha} \tag{29}$$

with

$$\omega = \sum_{\tilde{m} \in M} \sum_{\tilde{b} \in B^m} \sum_{\tilde{\alpha} \in Q^{nj}} \sum_{\tilde{t} \in T} y^{nj\tilde{\alpha}}_{\tilde{m}\tilde{b},\tilde{t}}. \tag{30}$$

As the number of job parts may differ from the maximum number that is allowed in a job, ω calculates the effective number of parts in a job. From this follows that supplier 1 receives $\bar{p}\,(1,\, k = 0.6) = 31 + 0.6 \cdot (7 + \frac{38}{3} + \frac{38}{3}) = 50.4$. The payment structure is pointed out in Table 10.

To / From	m_1	m_2	Payment $\bar{p}\,(n,\, k = 0.4)$
n_1	7.2	-	-7.2
n_2	-	48	-48
n_3	43.2	31.6	-74.8
Payment $\bar{p}\,(m,\, k = 0.6)$	+50.4	+79.6	

Table 10: K-pricing with $k = 0.4$.

For concluding the example, Table 11 illustrates the payments yielded under VCG pricing, approximate VCG pricing and k-pricing with $k = 0.4$, $k = 0.5$ and $k = 0.6$. The economic property of budget balance is outstanding. While VCG pricing yields a negative budget of -67, k-pricing and approximate VCG pricing achieve budget equilibrium. As in the scenario subsidization from the outside is impossible, VCG pricing is not applicable.

Comparing approximate VCG pricing and k-pricing, the payment difference for agents with high valuations and reservation prices is to highlight. In approximate VCG pricing, consumer 2 has costs of 30 that correspond to 43% of his stated valuation while consumers 1 and 3 have to effect payments of 100% and 81% respectively of their valuations. In this scheme, the suppliers' payments are almost equal in contrast to k-pricing that privileges supplier 2 who has higher reservation prices and fewer allocated time slots than supplier 1. Although these two pricing schemes achieve budget balance, they do not distribute the same total value among suppliers and consumers. Participants on each side of the market share an amount of 113 in the approximate VCG scheme and an amount of 120 (130, 110) with $k = 0.5$

($k = 0.4$, $k = 0.6$) in the k-pricing scheme. Transferred to k-pricing, the amount of 113 corresponds to payments that are yielded if k is in a range of [0.5, 0.6]. That means the preferential treatment of the consumer side of the market.

Pricing Scheme	VCG	Approximate VCG with threshold rule	k-Pricing		
			$k = 0.4$	$k = 0.5$	$k = 0.6$
n_1	-3	-10	-7.2	-6.5	-5.8
n_2	-15	-30	-48	-42.5	-37
n_3	-58	-73	-74.8	-71	-67.2
m_1	+72	+57	+50.4	+47.2	+44
m_2	+71	+56	+79.6	+72.8	+66
Budget	-67	0	0	0	0

Table 11: The payment structure depends on the applied pricing scheme: Comparison of VCG pricing, approximate VCG pricing and k-pricing.

6 Evaluation

The outcome of the mechanism is an efficient allocation of computation and storage service. In this section, the mechanism's performance is examined with respect to influences of instance size and parameter choice on complexity, scalability and runtime. In the verification phase, the model is implemented in GLPK to check its correctness. In a second phase, the numerical experiment is processed with the standard solver for linear optimization problems *ILOG CPLEX 10.0* on a machine composed of a 3 GHz processor and 1.5 GB RAM. The tests serve to analyze the delivered allocation under different circumstances. They aim at identifying specific behaviors of parameters so that conclusions for market configuration can be drawn.

6.1 Data Generation

As the evaluation cannot bear on reported user data, the input data is based on artificial problem instances. For such test cases with unknown data characteristics the uniform distribution U is particularly well applicable. Reservation prices are supposed to be distributed around a certain value, so that the normal distribution N can be deployed. The way of combining resource attributes is not predefined by the market. However, to simplify the data generation, the amount of combination possibilities is reduced to eight reasonable attribute constellations of CPU speed' and 'CPU RAM' as well as eight combinations of 'storage throughput' and 'storage size' are proposed. The combinations of CPU attributes a_{12}^{nj}, a_{13}^{nj} and d_{12}^{mb}, d_{13}^{mb} respectively are composed of two values. The first value represents the attribute 'CPU speed' taking a value of interval (1, 2, 3) and the second value represents 'CPU RAM' taking a value of 512 MB, 1 GB, 2 GB or 3 GB. The same applies for the combination of storage attributes a_{22}^{nj}, a_{23}^{nj} and d_{22}^{mb}, d_{23}^{mb} respectively. A combination contains a value of 'storage throughput' of interval (1, 2, 3) and a value for 'storage size' of 30 GB, 50 GB, 80 GB or 100 GB. Quantity attributes are arbitrary and independent. Table 12 specifies the combination possibilities for quality expression.

The maximum number of jobs and bundles in bids is predefined in the market configuration. Some parameters are fixed for the test settings. It is determined that $\bar{j} = 5$, $\bar{b} = 3$ and $\bar{t} = 8$ or $\bar{t} = 12$ depending on the test.

Resource Attributes	CPU $(a_{12}^{nj}, a_{13}^{nj}), (d_{12}^{mb}, d_{13}^{mb})$	Storage $(a_{22}^{nj}, a_{23}^{nj}), (d_{22}^{mb}, d_{23}^{mb})$
1	(1, 512 MB)	(1, 30 GB)
2	(1, 1 GB)	(1, 50 GB)
3	(1, 2 GB)	(1, 80 GB)
4	(2, 1 GB)	(2, 50 GB)
5	(2, 2 GB)	(2, 80 GB)
6	(2, 3 GB)	(2, 100 GB)
7	(3, 2 GB)	(3, 80 GB)
8	(3, 3 GB)	(3, 100 GB)

Table 12: Default attribute combinations of CPU (speed, RAM) and storage (throughput, size) for the experiment.

The generation of valuations and reservation prices bears on these package definitions as well as on the attributes that indicate the resource quantity. To calculate valuations and reservation prices, the eight packages for each resource type are sorted as shown in Table 12 and weighted. This weighting poses a challenge as in general the importance of packages may differ between applications (and users). With regard to the scenario, uniform preferences of agents are assumed. A larger amount of resources and packages of high performance quality attributes are valued more than fewer resources and combinations of lower performance attributes. In order to represent that computation and storage service are complementary goods for consumers, valuations are generated for single resources and the value $\widetilde{v}$ is added. So, the test data meets the property of resources being complements of each other, i.e.

$$v^{nj}(\text{Computation+Storage}) = v^{nj}(\text{Computation}) + v^{nj}(\text{Storage}) + \widetilde{v}.$$

with $\widetilde{v} \in [0, \frac{1}{10} (v^{nj} (\text{Computation}) + v^{nj} (\text{Storage}))]$ that is uniformly distributed.

The evaluation procedure starts with the generation of bid streams, called *settings*, that frame a set of 30 test instances differing in terms of randomly generated quality parameters. The mean value is calculated over the test instances of a setting to soften random outcomes and to avoid the influence of outliers. The initial parameter constellation consists of 130 agents divided into 60 consumers and 70

Parameter	Ranges and distribution
Agents	130 divided into 70 suppliers and 60 consumers
Bundles	$U(1,3)$
Jobs	$U(1,5)$
Job parts	$U(1,6)$
CPU and storage quantity	$U(1,8)$ for suppliers and $U(1,5)$ for consumers
Time slots	8 or 12
Reservation price	$N(7,1)$
Valuation	$U(7,12)$
Parallelization	Binomial with $p = 0.3$
Others	Uniform

Table 13: Ranges of parameter distribution and initial parameter setting. Not listed parameters are uniformly distributed. The generated values of 'CPU quantity' and 'storage quantity' are distributed among the two resource attributes.

suppliers, 8 or 12 time slots and parallelization in 30% of the jobs within an instance. The parameter distributions and initial ranges of parameters form the basis input data for the evaluation (Table 13).

6.2 Analysis

The model formulation suggests characteristic behaviors concerning scalability and complexity and hence offers interesting approaches for numerical experiments.

6.2.1 Complexity

The data analysis highlights the model's performance on the one hand as well as its limitations of computational tractability on the other hand. The complexity of the WDP is the decisive factor. Considered is the case of

- one supplier with only one bundle,
- one job per consumer,
- one part per job,

- the allocation time frames of jobs are equal to availability time frames of bundles,
- the bundle's resource attributes are greater than all resource requirements of jobs,
- no parallelization is possible and
- the reservation price of bundles is zero.

This particular case can be seen as a reduction of the WDP to the one-dimension *Knapsack Problem* (Garey and Johnson 1979) that is known to be NP[14]-hard. The simplest approach to solve such a problem is to enumerate all possibilities. However, even for small problems, NP-hard means that problem solving may result in a 'combinatorial explosion' (Hoffman and Padberg 2000). Consequently, only the smallest test instances could be solved by such an approach and there is a need for high performance solvers, such as CPLEX.

Besides its theoretical complexity, the practical complexity depends on the choice of parameter values and supposes characteristic behaviors concerning e.g. runtime. The required runtime does not only depend on the size of the instance with regard to the number of bids, bundles, jobs, job parts and time slots but also on the specific composition of time characteristics, resource attributes and allocation parameters (Sandholm 2006).

Parameters of the model can be divided into two groups - *dimension* and *quality* parameters - that have different influences on the complexity of the problem. The first group are responsible for array sizes. The dimension of an array is determined by the maximum value that the corresponding parameter can take. The market configuration determines the maximum values of those parameters (consumers, suppliers, jobs and bundles per bid, parts per job, resources, resource attributes and time slots). Quality parameters fill these arrays with values, such as the number of start and end time slots, resource attributes and allocation parameters. Although the array size of the problem can be calculated by multiplication of dimension parameters, the product only gives an indicator of instance size. It does not suggest the difficulty of the problem that also depends on the specific composition of quality parameters.

[14]NP means *Non-deterministic Polynomial Time*

As the choice of quality parameters is in the hands of market participants, they implicitly decide on the problem complexity.

6.2.2 Numerical Experiment

The characteristic of NP-hard problems is the computational intractability within a meaningful time frame even for small instances. As the scenario takes place in an offline market, simulation planning may be done in advance and so there is no need for an auction clearing in real-time. It is acceptable to find the optimal solution within a reasonable time frame. However, even within such a time frame, it cannot be guaranteed to obtain the welfare maximizing allocation. For some instances of the NP-hard problem, CPLEX yields the optimal solution in a few seconds. Others, however, are not solvable in hours. Even if the optimal solution cannot be found within reasonable runtime, CPLEX provides algorithms that return the best retrieved solution under user-defined conditions, such as within a predefined time frame. In periodic time intervals, CPLEX yields the value of the 'gap'. It represents the size of the interval bounded by upper and lower welfare values in which the optimal solution is located. Thus, the optimal solution can be approximated by setting a threshold of a defined percentage. Once the implemented termination condition is achieved e.g. as soon as the time limit is exceeded, the algorithm cuts execution and returns the best allocation found. The threshold indicates the proximity to the optimal welfare in form of the maximum gap.

Two stop conditions are defined for the test runs: a threshold value of 5% as well as a time limit of 20 minutes. Based on earlier test runs, these critical values seem to be reasonable. The solution of each test is measured by performance indicators. The first of them is the average runtime over all instances to attain a gap less than the threshold value to the optimal welfare within at most 20 minutes. This runtime is denoted by μ. The second indicator is the ratio of 'feasible' instances to *hard* instances in a setting. An instance is defined as hard when it does not achieve a 'gap' of less than the threshold value within a runtime of 20 minutes. This ratio is denoted by $\lambda = Number\,of\,hard\,instances\,/\,Total\,number\,of\,instances$. The runtime of hard instances is included in μ with a duration of 20 minutes.

In order to distinguish between the effects of single parameters, parameter groups

and supplementary restrictions, the tests examine independently the following situations:

Dimension parameters:

Test 1 Increasing number of time slots within the auction's time horizon

Test 2 Increasing number of agents

Quality parameters:

Test 3 Influence of the parallelization parameter

Test 4 Increasing number of job parts

Dominant subsets:

Test 5 Influence of dominant subsets on the performance

The results of these test runs are presented in the following text. In the first four tests, dominant subsets are not implemented and the maximum number of time slots is fixed at $\bar{t} = 8$. The test of dominant subsets is based on $\bar{t} = 12$ time slots.

Test 1 - Increasing Number of Time Slots

The objective of this test is to provide a statement on the mechanism's scalability with an increasing number of time slots. Thereby, it is not considered the redistribution of exactly the same jobs in terms of time attributes over an extending allocation horizon. The impact of an uniformly extended length of each time slot within the allocation horizon of constant length is examined.

Based on the initial parameter setting, the number of time slots escalates in steps of two within the four test settings, starting from $\bar{t} = 10$ and until it reaches $\bar{t} = 16$. This implies a decrease in the length of each time slot as the length of the allocation horizon keeps the same. The settings differ in the number of time slots and consequently in parameters that depend on these numbers, such as the time parameters $e^{nj}, l^{nj}, q^{nj}, c^{mb}, f^{mb}$, valuations v^{nj} and reservation prices p^{mb}. Not only the jobs are distributed to a larger number of time slots, e.g. to 16 instead of 10. But also the time frames of availability $[e^{nj}, l^{nj}]$ and $[c^{mb}, f^{mb}]$ as well as

the job duration q^{nj} are spread to longer time units and have to be adjusted from setting to setting. Although jobs require more time slots on average, valuations remain constant and do not increase in contrast to reservation prices. As valuations are coupled to the 'real' job duration that is calculated exogenously, the number of required time slots for job execution may change but not the monetary valuation of the job. Reservation prices are indicated per time slot and thus they have to be adjusted to the modified slot length. It should be noted that the test data for valuations is calculated on the basis of the randomly generated number of time slots for a job. Thus, the valuation for the same job increases with rising number of time slots. Consequently, the average welfare increases too and provides no information for this analysis.

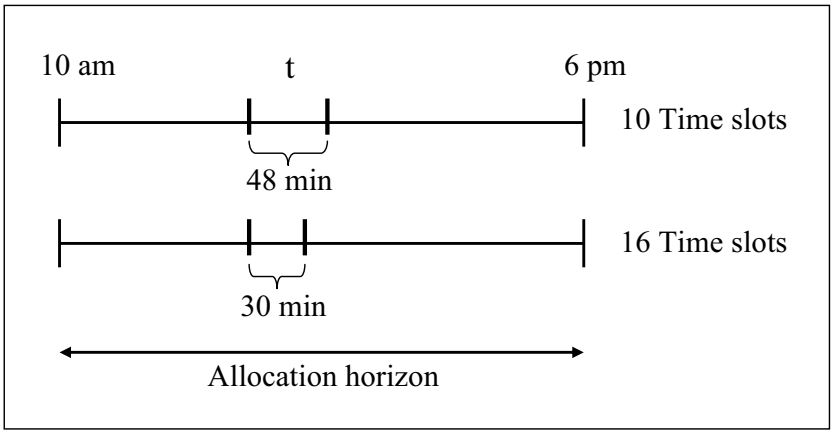
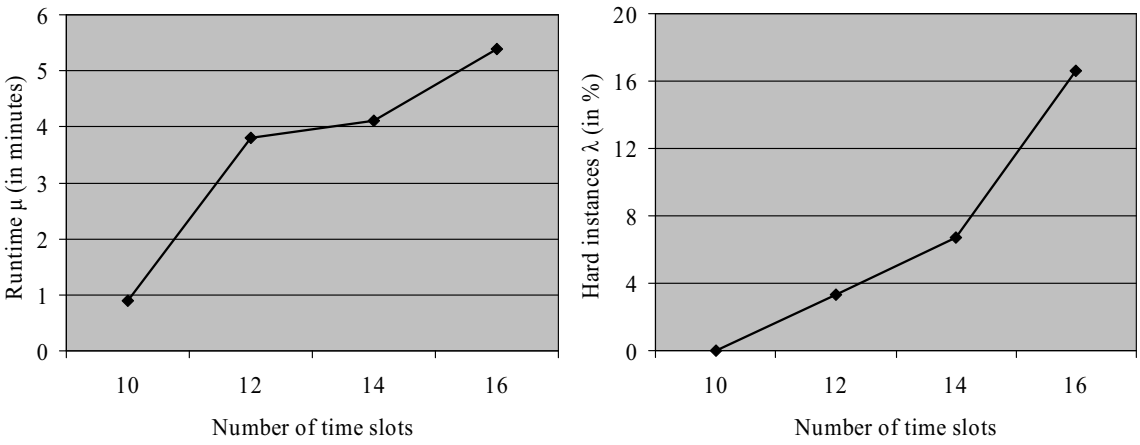


Figure 22: Influence of an increasing number of time slots on their length within the allocation horizon of constant length.

Figure 22 exemplifies the allocation horizon with 10 and 16 time slots. Considered is a time horizon of 8 hours from 10am to 6pm, 10 time slots with a length of 48 minutes and 16 time slots with a length of 30 minutes. A job with duration of 3 hours needs for execution 4 time slots in the concept of 10 time slots and 6 time slots in the concept of 16 time slots. μ and λ are low in the concept of 10 time slots and the expressiveness for consumers rises with an increasing number of time slots.

Figure 23 shows the run of the curves for the performance indicators μ and λ. The rising curves indicate the increasing complexity of problem solving with an increasing number of time slots. In case of 10 time slots, there is a 100% probability that the mechanism yields an optimal solution with a threshold value less than 5%



(a) Runtime μ (b) Hard instances λ

Figure 23: Development of runtime μ and hard instances λ with an increasing number of time slots.

within 20 minutes. This ratio drops slowly to 93% for 14 time slots and shows then a sharp reduction to 83% for 16 time slots. Analogously, the runtime increases from less than one minute to 5.4 minutes for 16 time slots. As all other dimension parameters keep constant and quality parameters adjust proportionally, there are no further influences. Thus the test shows that an increase of time slots leads to rising complexity and runtime. The curve sharps give rise to supposition that the system scales. The steady and proportional rise of μ and λ with an increasing number of time slots provides information for mechanism designers on how to configure the length of allocation horizon and time slots.

Consequently, the trade-off between increasing runtime and flexibility in expression of time attributes has to be considered to configure the time concept of the market.

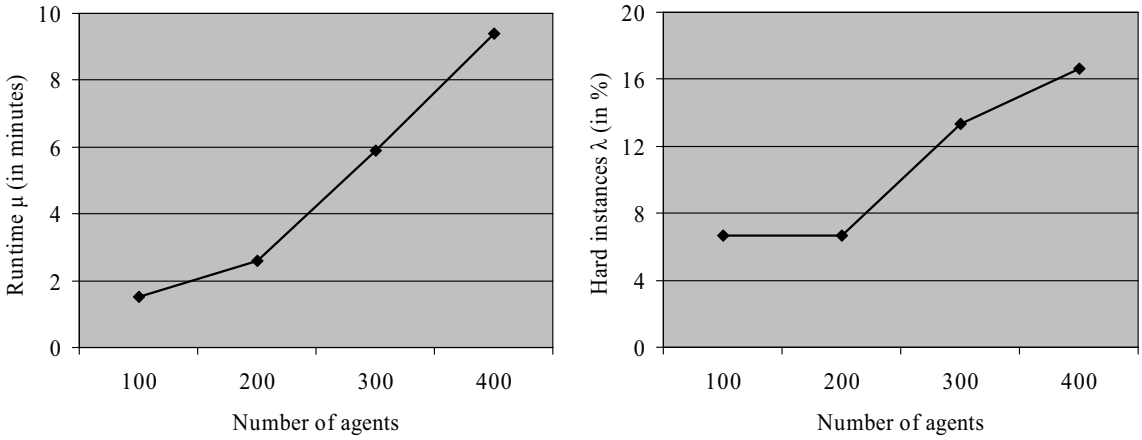
Test 2 - Increasing Number of Agents

In this classic test, the scalability of the model, runtime and its limitation of feasibility are examined. Therefore, the number of agents is raised in steps of 100 starting from $\bar{n} + \bar{m} = 100$ through to 400 by keeping the ratio of consumers to suppliers of $2 : 3$ constant. As the number of jobs $\bar{j} = 5$ and bundles $\bar{b} = 3$ is fixed, the ratio of supply to demand increases proportionally. The setting of 100 (200) agents, for instance, contains at most 200 (400) jobs and 180 (360) bundles.

The scope of the problem develops depending on the number of agents partici-

pating in the market. The change in supplied and demanded quantity influences the number of allocation possibilities and thus the time to find the optimal solution. In case of 40 consumers and 60 suppliers, the algorithm needs 1.5 minutes on average to fall below the threshold and to yield the best solution found. With an increasing number of agents, the curve of μ rises slowly until 200 agents are reached and soars in the range of 200 to 400 agents. For 400 agents the runtime μ goes up to 9.4 minutes. At the same time, the curve of λ increases too (Figure 24). The setting of 100 agents contains only one hard instance. This instance, however, represents an outlier as it merely achieves a gap of 11.4% within 20 minutes. Even in the setting with 400 agents, the instances do not show a gap that exceeds the value of 6.7%. This instance that seems to be as simple to solve as the other instances of the same size, shows the unpredictable nature of the WDP.

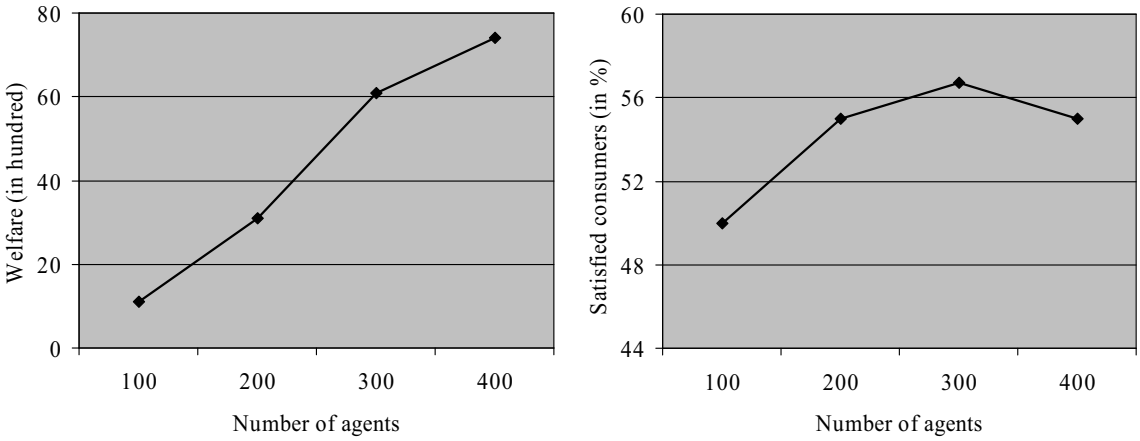
Although the ratio λ does not change between 300 and 400 agents, the average runtime μ almost doubles from 5.4 minutes to 9.4 minutes.



(a) Runtime μ (b) Hard instances λ

Figure 24: Development of runtime μ and hard instances λ with an increasing number of agents. The ratio of consumers to suppliers keeps constantly 2 : 3.

The increasing number of agents within a constant time horizon effects a steady rise on average welfare that behaves proportionally to the number of agents (Figure 25). As more agents imply a greater supply of bundles and demand for resources, the algorithm selects the winning bids from a larger range. It is remarkable that the curve progression of satisfied consumers rises first from 50% to 56.7% and falls slightly starting from 300 agents. As the algorithm selects the winning bids



(a) Average welfare

(b) Satisfied consumers

Figure 25: Development of welfare and satisfied consumers on average with an increasing number of agents.

from a larger range of possibilities, the percentage of satisfied consumers is expected to stay firm. The observed trend gives rise to the suspicion that the algorithm scales well up to a specific number of bids. Subsequently the increase of hard instances interferes the solution finding. For instance, the increasing number of jobs raises the probability of small time-consuming jobs and jobs with few job parts. Thus, the probability rises that idle times of bundles are avoided by placing these kinds of jobs.

Approaching 500 agents, the problem size exceeds the limit of computational tractability (CPLEX runs out of memory). Not only the optimal solution at any-time is of interest but also a solution time that remains within a reasonable time frame. This implies the need for an upper bound to limit the number of agents in the market, for which the problem is computationally tractable within a reasonable time frame. As the curve of welfare flattens between 300 and 400 agents, the interval of about 300 agents could be an optimal bound of the number of participants.

The second test series deals with the issue of how far parameter values influence the runtime. Therefore the impacts of parallelization and an increasing number of job parts on computational complexity and runtime are analyzed.

Test 3 - Influence of Parallelization

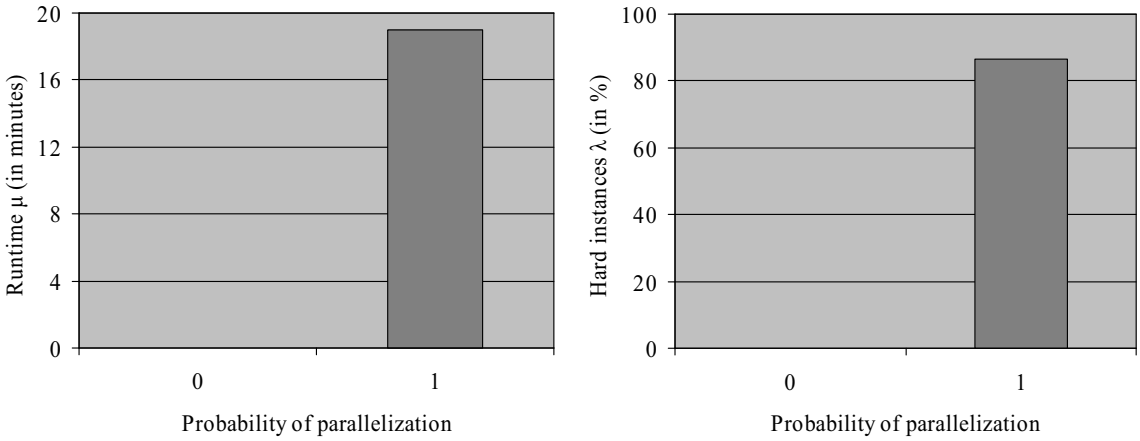
The model formulation gives rise to the supposition that the quality parameter γ^{nj} has a strong impact on the mechanism's performance in terms of complexity and scalability. Thus the idea of this test is the comparison of the two extreme cases: activated parallelization in all jobs ($\gamma^{nj} = 1$) and the same setting without parallelization restrictions ($\gamma^{nj} = 0$).

The effect of parallelization restrictions is the drastic reduction of the feasible set of allocations in the problem. The determining factor are constraints (7) that only generate restrictions in case of activated parallelization. Their quantity depends on the number of consumers, jobs, time slots within the time frame of availability and job parts. For instance, a consumer is considered that demands two jobs with 4 job parts each, a time frame of availability of 3 time slots and parallelization. Even in this small example are generated 72 constraints to guarantee parallelization. This is calculated by: $Jobs \cdot Time\,slots \cdot \frac{\bar{\alpha}!}{(\bar{\alpha}-2)!}\, Combinations\, of\, job\, parts$, so $2 \cdot 3 \cdot 12 = 72$. Evidently, the parallelization restrictions influence the complexity of the problem.

The comparison of the two extreme cases is shown in Figure 26. The activated parallelization in 100% of the jobs nearly avoids achieving a non-hard instance in the setting; 87% are hard instances in figure 26(b). Hence the runtime μ is long too and takes 19 minutes. On the contrary, there are no hard instances ($\lambda = 0$) in the setting without parallelization restrictions. The runtime μ is negligible and reaches a value close to zero (0.03 minutes). The result is that the simultaneous allocation of job parts is out of all proportion to complexity. The increased user utility of activated parallelization has to be opposed to its downgrading effect on scalability.

Test 4 - Increasing Number of Job Parts

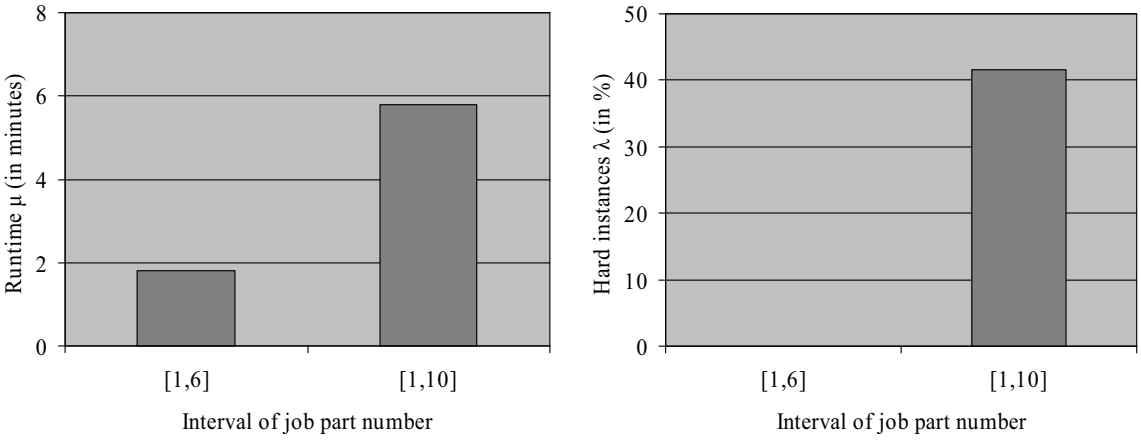
One of the central scenario requirements is the representation of numerous simulations. As job parts can be seen as simulations, the objective is to examine the performance of the mechanism with an increasing number of job parts. Starting with a number of job part of the interval [1, 6], the maximum quantity is increased in a step of 4 to $\bar{\alpha} = 10$. The determining factor that affects the problem complexity are constraints (7). They generate a corresponding α' to each α in case of activated parallelization. Additionally, most of the other constraints run through $\bar{\alpha}$



(a) Runtime μ (b) Hard instances λ

Figure 26: Influence of parallelization restrictions on the performance.

times. Thus the more job parts exist, the more restrictions are generated. Figure 27 illustrates the resulting curve shapes of μ and λ. In contrast to the parallelization parameter, the curves rise moderately but nevertheless they augment complexity. The number of hard instances increases from 0% for interval [1, 6] up to 13% for interval [1, 10] and at the same time, the runtime μ triples from 1.8 minutes to 5.8 minutes. However, the average number of satisfied consumers decreases (Figure 28). The more parts are contained in a job, the more challenging becomes the allocation. As larger jobs take more positions, fewer jobs may be allocated. The increase of job parts causes both more overlapping demand in time slots and more dependencies between job parts that have to be fulfilled for allocation.



(a) Runtime μ (b) Hard instances λ

Figure 27: Development of runtime μ and hard instances λ with an increasing number of agents.

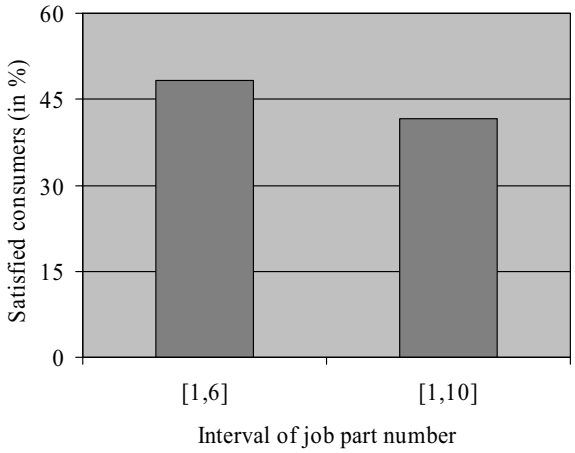


Figure 28: Development of number of satisfied consumer with an increasing number of job parts.

As the possibility to indicate numerous job parts is a crucial factor of the scenario, market configuration has to provide a sufficiently large maximum number of job parts. On the other hand, the decreasing number of satisfied consumers with rising $\bar{\alpha}$ has to be taken into account. It is essential to weigh up these two opposite trends.

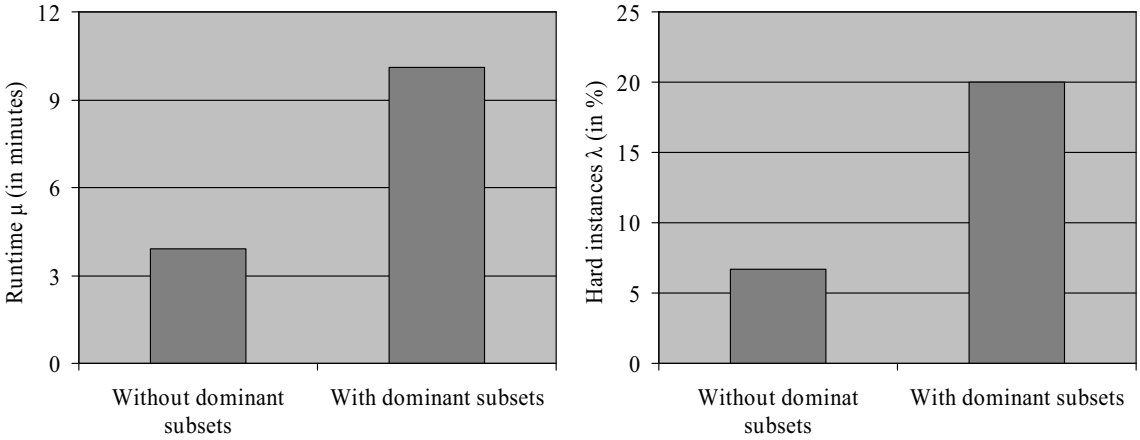
The analysis of quality parameters γ^{nj} and α^{nj} provide evidence for their impact on performance. In case of the combination of these parameters, i.e. many activated parallelizations and a great number of job parts in a setting, it is suspected that the problem becomes even more complex.

As the mechanism provides a complex bidding specification and complies with the extensive scenario requirements, it quickly reaches the limit of computability. Therefore, the performance of dominant subsets is examined in addition to the parameter considerations.

Test 5 - Dominant Subsets

The two dominant subsets are added to the problem with the objective of solving the problem within a shorter period of time. Thereby the initial welfare that is yielded without dominant constraints does not change as the dominant subsets solely rearrange jobs within the allocation. By comparison of the two cases, with and without dominant subsets, the trend in performance is analyzed in terms of hard instances λ

and runtime μ. Although it is supposed that the supplementary restrictions reduce the complexity of the problem, the opposite effect can be noted. This prove the curve shapes in Figure 29. By adding dominant subsets, λ triples from 6.7% to 20% and μ rises from 3.9 to 10.1 minutes. On closer inspection, however, one can notice that the two hard instances of the setting without dominant subsets are solvable within reasonable runtime by implementing the dominant subsets. The runtime of these hard instances is reduced to 12.2 and 7.1 minutes respectively. This test result suggests that these type of dominant subsets are not suitable for the developed mechanism.



(a) Runtime μ (b) Hard instances λ

Figure 29: Development of runtime μ and hard instances λ with and without dominant subsets.

According to Hoffman and Padberg (2000) however, dominance properties are efficient to implicitly eliminate allocation possibilities for enumerative approaches e.g. the method of branch and bound. Thus if CPLEX uses this method, dominances may lead to performance improvements in terms of the number of considered nodes and runtime. With regard to the proposed allocation mechanism, further investigations need to be done in order to adapt the dominant subsets mentioned above to the algorithm of CPLEX and to find more suitable dominance properties.

In conclusion, the experiment points out that the type of parameter determines the grade of influence on complexity and runtime. The decisive factor for dimension parameters is the increasing instance size because it affects directly the runtime. By contrast, varying quality parameters have no effect on instance size but on comp-

lexity, as the number of generated constraints changes. In this way the increase of problem complexity again leads to rising runtime. The results of evaluation are promising as possible inputs for market configuration. They help to determine maximum figures that dimension the market, such as a maximum limit of potential participants or a minimal length of time slots within a predefined allocation horizon. A special focus should be placed on parallelization that adds enormous complexity to the problem. Especially in combination with a high number of job parts, parallelization restrictions reduce the model's performance. For such complex problems, the implementation of well adapted dominant subsets could help to restrict complexity.

6.3 Limitations and Extensions

While developing the allocation model, several assumptions and simplifications have been made. Some aspects imply ideas for model extensions.

(1) Inclusion of network distances and capacities

Especially for large amounts of data, the time of transfer between agents is of great importance. Therefore, the idea of including network distances and capacities is proposed. Network distances as well as network capacities are required to calculate the potential throughput between participants. The term 'throughput' denotes in this context the amount of data that is transferred per time unit between two agents. However, this presumes that the corresponding network distances are determinable. Current *peer-to-peer (P2P)* solutions provide this opportunity. As the network capacity depends on both agents, it deserves closer attention. Supposing that consumer n that has the network capacity $N1$ transfers a job to supplier m that provides the network capacity $N2$. To meet a given quality of service, a capacity limit x has to be defined which has not to be fallen below ($\min(N1, N2) \geq x$). By dint of these determinations, the potential throughput is made computable.

(2) Representation of computation performance in FLOPS

As pointed out in Section 3.3.1, CPU quantity and quality may be described in several ways. The representation of computational performance in FLOPS could be

focused on in a second step. The segmentation problem (see page 24) has to be taken into account in order to avoid unfavorable resource allocations. This modification is an upgrade of the model as bundle splitting is enabled, thus the allocation can be customized more efficiently and overpayment of bundles can be avoided.

(3) Representation of real workflows

In general, a workflow refers to a sequence of tasks that may have different characteristics such as time attributes, location dependencies, allocation restrictions and dependencies in processing status. As mentioned, consumers demand resources that are allocated in form of workflows. For instance, a workflow in the scenario could be as follows: CPU 1 in time slot 1, 2 disks in time slot 1 to 6 and CPU 2 in time slot 5. Currently, consumer bids consist of XOR-combinations of jobs whose specifications apply to all parts of the job. Ideally, a job should be an AND-combination of job parts in which each part specifies its components separately. Hence, time dependencies between job parts or resources may be expressed and the example workflow could be represented. However, the use of AND-connections does not imply that every kind of workflow is presentable. The expression of other types of dependencies requires more complex bidding specifications, e.g. dependencies of processing status. For instance, the starting time of job part 2 on bundle 2 depends on the progress of job part 1 on bundle 1. This progress may be either the total completion or the partial completion of part 1. Initially, the starting time of job part 2 is unknown and the mechanism has to provide a tool which manages such demands.

(4) Experiment to provide information about the length of the allocation horizon

Following test 1, a further test objective could be the examination of the length of the allocation horizon to provide significant information for market configuration. With an increasing number of time slots, the length of time slots and the job duration remains unchanged and the allocation horizon expands. The length of availability time frames of jobs and bundles also remains unchanged but they are redistributed over the extended allocation period. Hence, the system load is much better spread within the allocation horizon. With regard to competition, the average

number of satisfied consumers and the average welfare are expected to rise. However, the effect of an increasing number of time slots on scalability has to be opposed.

(5) Examination of changing competition structure

In the context of test 2, a further test could be the examination of a changing competition structure in the market. The goal is to determine the impact of competition on scalability. Thereby competition is simulated by varying the proportion of consumers to suppliers. Starting from an oversupply, i.e. a ratio of $\bar{n} < \bar{m}$, the proportion is balanced, so that $\bar{n} = \bar{m}$, and then reversed by generating an excess demand ($\bar{n} > \bar{m}$). As suppliers offer many bundles on average, it is obvious that an increase of suppliers effects a multiplication of provided bundles. Theoretically, the increase in demand leads to more exact matchings of jobs and bundles in terms of resource attributes and valuations as there are more possibilities of job allocation. The 'waste' of capacities of allocated bundles and idle resources can be avoided by this better matching. The sample allocation schedule on page 57 in Section 5.2.2 shows a job-bundle allocation, in which the job uses less than half of the provided resources. If there was more competition in the market, another job could be allocated that needs more resources and hence has a higher valuation.

(6) Inclusion of the difference between peak times and slack periods

Workload in a system is not uniformly distributed within the allocation horizon; Peak times and slack periods alternate. The concept of discrete and fixed time slots of uniform length that is proposed in Section 3.3.3 may be upgraded in a further step. The difference between peak times and slack periods could be taken into account by lengths of time slots that are adjustable or by price discounts. Shorter duration of time slots in peak times enable a more precise resource allocation that may result in higher resource utilization and consumer satisfaction. Furthermore, price discounts for time slots in times of depression contribute to balance the workload. As they enable the shift of resource requests away from peak times, the satisfaction of consumers increases.

(7) Consumption-adapted prices

Acting on a market with small-sized VO's requires consumption-adapted prices. The precondition that one bundle can only be allocated to one job per time slot, obliges consumers to pay for the whole amount of resources of the bundle. This is likely to lead to overpayment as consumers do not need the provided resource amount. Hence, it is recommended that prices are fixed according to the demanded number of resource. For instance, a job demands 2 CPUs and 2 disks is allocated to a bundle that consists of 3 CPUs and 3 disks and has a reservation price of 2 per time slot. The job should only cost the price of 4 instead of 6. This concept would be fair with regard to consumers who demand small quantities. However, bundle pricing poses a problem for price formation. As prices for single resources are not known, the question has to be approached of how to fix the price for a bundle segment.

On the other hand, volume discounts could be offered to consumers, so that prices are reduced depending on requested resource quantities or time slots. This, however, privileges big consumers and puts small consumers out of business.

(8) Relaxation to improve the runtime

Besides dominant subsets, a further approach to control scalability provides *relaxation*. By dropping integer constraints, the method of linear programming relaxation transforms the initial problem in another problem which is simpler to solve. Hence, the range of feasible and optimal solutions can be expanded. Results are scaled up in a different optimization problem that includes the initial problem. The solution of this transformed problem has to be checked, so that it meets the requirement of integrality (Li 2007).

7 Conclusion and Outlook

The motivation for computational Grids was initially stimulated by resource intensive complex applications and simulations in the public and business sector. The increasing demand for computer resources challenges the establishment of open Grid markets where resources can be traded. To overcome the deficits of technical schedulers for resource allocation on such a market, economic properties were elucidated that are taken into account in mechanism design for resource allocation. As education is considered to be one of the most important applications of Grid computing in the near future, Grid4All was introduced. It addresses the needs of small-sized enterprises and institutions of the public sector by aiming to provide a market platform that enables those users to access Grid resources. Within this context, the specific application scenario 'Collaborative learning' was considered. It was analyzed and conclusions were drawn as to necessary requirements.

Related work was presented on market mechanisms and it was pointed out that current mechanisms do not fully meet the scenario requirements. Either they are not able to represent the required combinatorial connections in bids and allocation peculiarities required by the scenario, or consumers that ask for multiple items risk exposing.

A Grid-compatible market mechanism was developed based on specific scenario requirements, such as the representation of various simulations and the allocation of jobs 'continuously in time'. These requirements resulted from the environment and the specific process of collaborative learning. While satisfying the demands of market participants, the mechanism challenges the allocation of heterogeneous resources by implementing a combinatorial auction. The work deals with the central questions posed in the introduction: What demand market participants, in which way they express demands and the consequences of these demands on market design and which properties the resource allocation mechanism has to provide to satisfy users. The work highlighted the key issues that arise from designing an adequate market mechanism. Aiming at maximizing the social welfare, the auction mechanism is efficient in allocating combinatorial resources, while at the same time satisfying market participants. Special scenario characteristics were either included directly in the model design, such as coupling and allocation continuously in time, or could

be expressed individually by agents. Subsequently, alternative pricing schemes were presented to complement the allocation mechanism. VCG pricing, approximate VCG pricing and k-pricing were applied to an exemplary allocation schedule.

The model was evaluated with regard to the market configuration. To this end, four numerical tests were carried out to investigate different influences of dimension and quality parameters on average runtime and problem complexity. Although the model scaled well with an increasing number of time slots and agents, the scope of the problem exceeded the limit of computational tractability when 500 agents were approached. Quality parameters added enormous complexity to the problem. In particular, the combination of a large number of job parts and the parallelization restrictions reduce the model's performance. For such complex problems, the implementation of dominant subsets should help to reduce the complexity. A further test, in which settings with and without dominant subsets were investigated, highlighted that these additional restrictions downgrade the performance contrary to the trend that was expected. As the evaluation could not rely on reported input data, it would be of interest to run tests with real workload. A next step in model evaluation could be a numerical comparison with MACE. As MACE is formulated as mixed integer program and the proposed model in this work as integer program, a benchmark test would provide information about the influence of the problem formulation.

Based on the presented model, general and specific recommendations for mechanism design as well as further objectives for examinations were proposed. Future extensions could consider the involvement of network distances and capacities as well as the representation of complex workflows. Furthermore, the time concept could be upgraded in a way that it responds flexible to peak loads and slack periods.

Including the pricing considerations, the next step would be the design of a scenario-tailored pricing scheme. Consumption-adapted prices as well as the difference between peak loads and slack periods could be taken into account. Another approach on this could be the implementation of an iterative process to determine prices. Iterative combinatorial auctions provide this functionality by incrementally approximating prices depending on supply and demand.

With regard to the complexity of the problem, a further topic of interest could be to improve the mechanism's scalability by dint of well adapted dominant subsets

or a scenario-compatible heuristic.

A Appendix

Sets and Parameters of the Model

Parameter	Range	Explication
V	$V \in \mathbb{R}^+$	Welfare
N	$\{1, \ldots, \bar{n}\}$	Set of consumers/consumer bids
$\bar{n}$	$\bar{n} \in \mathbb{N}$	Maximal number of consumers/consumer bids
M	$\{1, \ldots, \bar{m}\}$	Set of supplier/supplier bids
$\bar{m}$	$\bar{m} \in \mathbb{N}$	Maximal number of supplier/supplier bids
J^n	$\{1, \ldots, \bar{j}\}$	Set of jobs
$\bar{j}$	$\bar{j} \in \mathbb{N}$	Maximal number of jobs in consumer bids
B^m	$\{1, \ldots, \bar{b}\}$	Set of bundles
$\bar{b}$	$\bar{b} \in \mathbb{N}$	Maximal number of bundles in supplier bids
R	$\{1, \ldots, \bar{r}\}$	Set of resources
$\bar{r}$	$\bar{r} \in \mathbb{N}$	Maximal number of resources
A_r	$\{1, \ldots, \bar{a}\}$	Set of resource attributes
$\bar{a}$	$\bar{a} \in \mathbb{N}$	Maximal number of resource attributes
a_{ri}^{nj}	$a_{ri}^{nj} \in A_r$	Arbitrary resource attribute of consumer
d_{ri}^{mb}	$d_{ri}^{mb} \in A_r$	Arbitrary resource attribute of supplier
T	$\{1, \ldots, \bar{t}\,\}$	Set of time slots
$\bar{t}$	$\bar{t} \in \mathbb{N}$	Maximal number of time slots
e^{nj}	$e^{nj} \in T$, $e^{nj} \in \mathbb{N}$	Earliest start time slot for job j
l^{nj}	$l^{nj} \in T$, $l^{nj} \in \mathbb{N}$	Latest start time slot for job j
q^{nj}	$q^{nj} \in \mathbb{N}$	Number of required slots for job j
c^{mb}	$c^{mb} \in T$, $c^{mb} \in \mathbb{N}$	Earliest availability of bundle b
f^{mb}	$f^{mb} \in T$, $f^{mb} \in \mathbb{N}$	Latest availability of bundle b
v^{nj}	$v^{nj} \in \mathbb{R}^+$	Valuation of job j
p^{mb}	$p^{mb} \in \mathbb{R}_0^+$	Reservation price of bundle b

Parameter	Range	Explication
γ^{nj}	$\gamma^{nj} \in \{0, 1\}$	Parallelization for job j
Q^{nj}	$\{1, \ldots, \bar{\alpha}\}$	Set of job parts
$\bar{\alpha}$	$\bar{\alpha} \in \mathbb{N}$	Maximal number of job parts of job j

Table 14: Sets and parameters of the model formulation

Variables of the Model

Variable	Range	Explication
$y_{mb,t}^{nj\alpha}$	$y_{mb,t}^{nj\alpha} \in \{0,1\}$	Part α of job j of n is allocated to bundle b of m in t or not
z^{nj}	$z^{nj} \in \{0,1\}$	Job j of n is allocated or not

Table 15: Variables that are used in the model formulation

First Formulation of the Optimization Model[15]

Maximization of welfare:

$$\max V = \sum_{n\in N}\sum_{j\in J^n} z^{nj}\, v^{nj} \; - \sum_{m\in M}\sum_{b\in B^m}\sum_{n\in N}\sum_{j\in J^n}\sum_{\alpha\in Q^{nj}}\sum_{t\in T} y_{mb,t}^{nj\alpha}\, p^{mb} \tag{31}$$

subject to

$$\sum_{j\in J^n} z^{nj} \le 1, \quad \forall\, n \in N \tag{32}$$

$$\sum_{m\in M}\sum_{b\in B^m}\sum_{\alpha\in Q^{nj}}\sum_{t\in T} y_{mb,t}^{nj\alpha} - z^{nj}\, q^{nj}\, \alpha^{nj} = 0, \quad \forall\, n \in N, \forall\, j \in J^n \tag{33}$$

$$\sum_{m\in M}\sum_{b\in B^m}\sum_{t\in T} y_{mb,t}^{nj\alpha} - z^{nj}\, q^{nj} = 0, \quad \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj} \tag{34}$$

$$\sum_{n\in N}\sum_{j\in J^n}\sum_{\alpha\in Q^{nj}} y_{mb,t}^{nj\alpha} \le 1, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, t \in T \tag{35}$$

$$e^{nj}\, y_{mb,t}^{nj\alpha} \le t\, y_{mb,t}^{nj\alpha} \le l^{nj}\, y_{mb,t}^{nj\alpha}, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t \in T \tag{36}$$

$$c^{mb}\, y_{mb,t}^{nj\alpha} \le t\, y_{mb,t}^{nj\alpha} \le f^{mb}\, y_{mb,t}^{nj\alpha}, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t \in T \tag{37}$$

$$y_{mb,t'}^{nj\alpha} + y_{mb,t-1}^{nj\alpha} - y_{mb,t}^{nj\alpha} \ge 0, \quad 2 \le t \le t' \le t + q^{nj} - 1, \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t, t' \in T \tag{38}$$

$$y_{mb,t'}^{nj\alpha} - y_{mb,1}^{nj\alpha} \ge 0, \quad 1 \le t' \le q^{nj}, \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t' \in T \tag{39}$$

$$\sum_{m\in M}\sum_{b\in B^m} y_{mb,t}^{nj\alpha} \le 1, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t \in T \tag{40}$$

$$a_{ri}^{nj}\, y_{mb,t}^{nj\alpha} - d_{ri}^{mb}\, y_{mb,t}^{nj\alpha} \le 0, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t \in T, \forall\, r \in R, \forall\, i \in A_r \tag{41}$$

[15]Note that the numbering is not identical to Chapter 5.2.1

$$z^{nj}\, v^{nj} - \sum_{m\in M}\sum_{b\in B^m}\sum_{\alpha\in Q^{nj}}\sum_{t\in T} y^{nj\alpha}_{mb,t}\, p^{mb} \geq 0, \quad \forall\, n \in N, \forall\, j \in J^n \tag{42}$$

$$\gamma^{nj}\sum_{m\in M}\sum_{b\in B^m} y^{nj\alpha}_{mb,t} - \gamma^{nj}\sum_{m\in M}\sum_{b\in B^m} y^{nj\alpha'}_{mb,t} = 0, \quad \forall\, n \in N, \forall\, j \in J^n,$$
$$\forall\, t \in [e^{nj}, l^{nj}], \forall\, \alpha, \alpha' \in Q^{nj} \tag{43}$$

$$z^{nj} \in \{0,1\}, \quad \forall\, n \in N, \forall\, j \in J^n \tag{44}$$

$$y^{nj\alpha}_{mb,t} \in \{0,1\}, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N,$$
$$\forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t \in T \tag{45}$$

Final Formulation of the Optimization Model[16]

Maximization of welfare:

$$\max V = \sum_{n\in N}\sum_{j\in J^n} z^{nj}\, v^{nj} \; - \sum_{m\in M}\sum_{b\in B^m}\sum_{n\in N}\sum_{j\in J^n}\sum_{\alpha\in Q^{nj}}\sum_{t\in T} y_{mb,t}^{nj\alpha}\, p^{mb}\, q^{nj} \tag{46}$$

subject to

$$\sum_{j\in J^n} z^{nj} \leq 1, \quad \forall\, n \in N \tag{47}$$

$$\sum_{m\in M}\sum_{b\in B^m}\sum_{t\in T} y_{mb,t}^{nj\alpha} - z^{nj} = 0, \quad \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj} \tag{48}$$

$$\sum_{n\in N}\sum_{j\in J^n}\sum_{\alpha\in Q^{nj}}\sum_{t'=t-q^{nj}+1}^{t} y_{mb,t'}^{nj\alpha} \leq 1, \quad \forall\, m \in M, \forall\, b \in B^m, \quad \forall\, t \in [c^{mb}, f^{mb}] \tag{49}$$

$$\gamma^{nj}\sum_{m\in M}\sum_{b\in B^m} y_{mb,t}^{nj\alpha} - \gamma^{nj}\sum_{m\in M}\sum_{b\in B^m} y_{mb,t}^{nj\alpha'} = 0, \quad \forall\, n \in N, \forall\, j \in J^n, \quad \forall\, t \in [e^{nj}, l^{nj}], \forall\, \alpha \neq \alpha' \in Q^{nj} \tag{50}$$

$$y_{mb,t}^{nj\alpha} = 0, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \quad \forall\, t \in T \setminus [\max(e^{nj}, c^{mb}),\, \min(l^{nj}, f^{mb}) - q^{nj} + 1] \tag{51}$$

$$y_{mb,t}^{nj\alpha} = 0, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \quad \forall\, t \in T, \forall\, r \in R, \forall\, i \in A_r \backslash \{a_{ri}^{nj} \leq d_{ri}^{mb}\} \tag{52}$$

$$z^{nj} \in \{0,1\}, \quad \forall\, n \in N, \forall\, j \in J^n \tag{53}$$

$$y_{mb,t}^{nj\alpha} \in \{0,1\}, \quad \forall\, m \in M, \forall\, b \in B^m, \forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}, \forall\, t \in T \tag{54}$$

[16] Note that the numbering is not identical to Chapter 5.2.1

Dominant Subsets[17]

$$\sum_{m\in M}\sum_{b\in B^m}\sum_{t\in T} y_{mb,t}^{nj\alpha}\, s(m,b,t) - \sum_{m\in M}\sum_{b\in B^m}\sum_{t\in T} y_{mb,t}^{nj\alpha+1}\, s(m,b,t) \le 0, \tag{55}$$

$$\forall\, n \in N, \forall\, j \in J^n, \forall\, \alpha \in Q^{nj}$$

$$\sum_{n\in N}\sum_{\substack{j\in J^n\\ t>e^{nj}}} (1-\gamma^{nj}) \sum_{\alpha\in Q^{nj}} y_{mb,t}^{nj\alpha} - \sum_{n\in N}\sum_{\substack{j\in J^n\\ t>q^{nj}}}\sum_{\alpha\in Q^{nj}} y_{mb,t-q^{nj}}^{nj\alpha} \le 0, \tag{56}$$

$$\forall\, m \in M, \forall\, b \in B^m, \forall\, t \in]c^{mb}, f^{mb}]$$

[17] Note that the numbering is not identical to Chapter 5.2.1

Outcome of GLPK for Sample Schedule

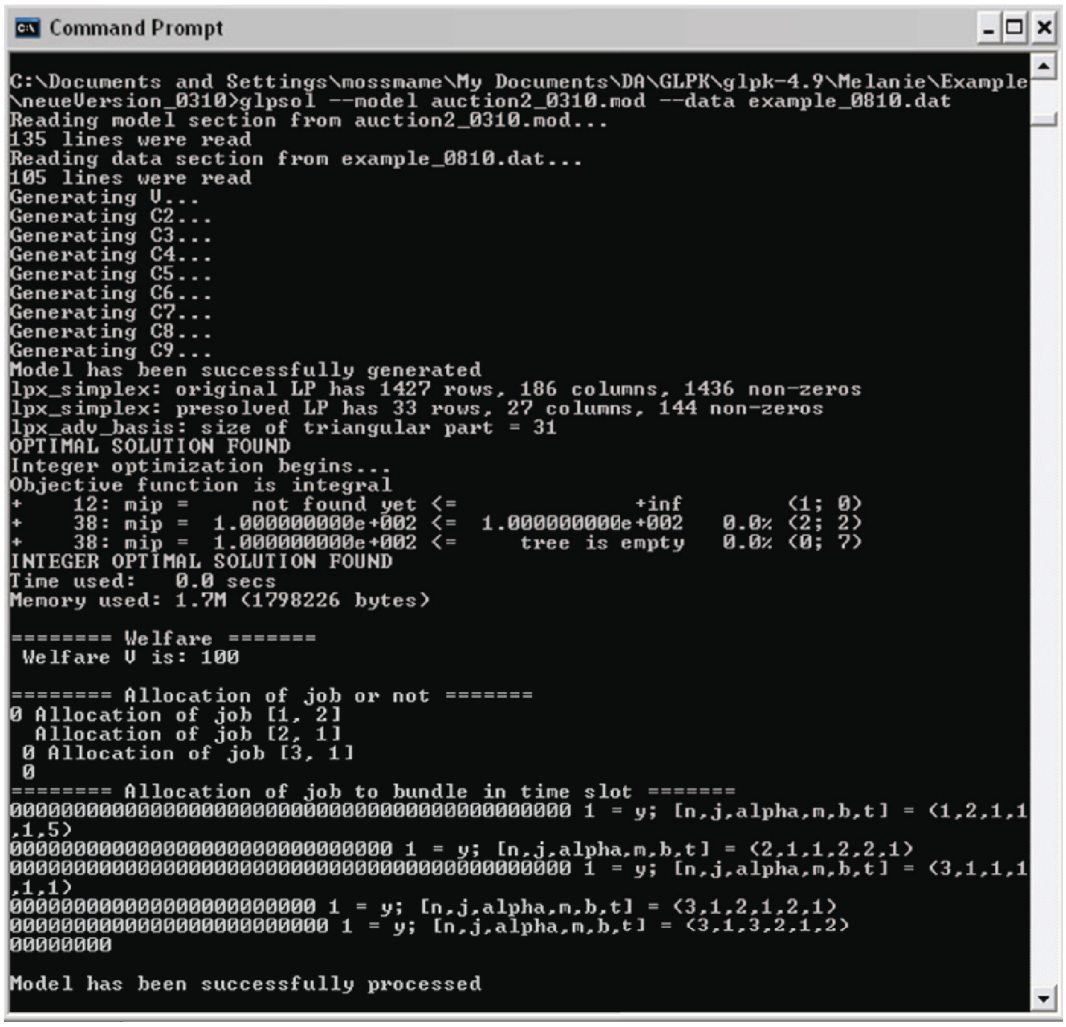


Figure 30: Screenshot of the GLPK solution of the sample schedule from page 55

Evaluation Settings for Numerical Experiment

Test	Setting	Agents	$\bar{n}$	$\bar{m}$	$\bar{t}$	$\bar{\alpha}$	Probability that $\gamma^{nj} = 1$	Dominant Subsets
1	1	130	60	70	10	[1, 6]	30%	No
	2				12			
	3				14			
	4				16			
2	1	100	40	60	8	[1, 6]	30%	No
	2	200	80	120				
	3	300	120	180				
	4	400	160	240				
3	1	130	60	70	8	[1, 6]	0%	No
	2						100%	
4	1	130	60	70	8	[1, 6]	30%	No
	2					[1, 10]		
5	1	130	60	70	12	[1, 6]	30%	No
	2							Yes

Table 16: Settings for numerical experiment

Numerical Results of the experiment

Test	Setting	μ (in sec)	λ (in %)	Ø Satisfied Consumers (in %)	Ø Welfare
1	1	54	0%	48.3%	(2265)*
	2	225	3.3%	48.3%	(2521)*
	3	245	6.7%	50%	(2821)*
	4	323	16.6%	51.7%	(3396)*
2	1	91	6.7%	50%	1097
	2	159	6.7%	55%	3139
	3	355	13.3%	56.7%	6056
	4	565	13.3%	55%	7438
4	1	2	0%	(50%)*	(2008)*
	2	1141	86.7%	(46.7%)*	(1792)*
5	1	109	0%	48.3%	(1802)*
	2	347	13.3%	41.7%	(2145)*
6	1	234	6.7%	(48.3%)*	(2520)*
	2	607	20%	(46.7%)*	(2478)*

Table 17: Numerical results for the tests

* Values in brackets allow no interpretation and are mentioned to get a complete picture

References

AuYoung, A., B. Chun, A. Snoeren and A. Vahdat (2004). Resource Allocation in Federated Distributed Computing Infrastructures. *Proceedings of the* 1^{st} *Workshop on Operating Systems and Architectural Support for the On-demand IT Infrastructure*.

Bapna, R., S. Das, R. Garfinkel and J. Stallaert (2005). A Market Design for Grid Computing. *Journal on Computing, Forthcoming available at SSRN: http://ssrn.com/abstract=927036*.

Beutel, J. (2006). *Mikroökonomie*. R. Oldenburg Verlag München Wien.

Britannica-Concise-Encyclopedia (2006). Britannica Concise Encyclopedia. *http://www.answers.com/topic/market, page 4, Retrieved November 2007*.

Broberg, J., S. Venugopal and R. Buyya (2007). Market-oriented Grids and Utility Computing: The state-of-the-art and future directions. *http://www.gridbus.org/reports/MarketGridUtilitySurvey2007.pdf, Retrieved November 2007* (GRIDS-TR-2007-16).

Buyya, R., D. Abramson and S. Venugopal (2005). The Grid Economy. *Preceedings of the IEEE 93*(3), p. 698–714.

CERN (2007). Gridcafe - The Place for Everybody to Learn About the Grid. *http://gridcafe.web.cern.ch/gridcafe, Retrieved June 2007*.

Chun, B. and D. Culler (1999). Market-based Proportional Resource Sharing for Clusters. *Millenium Project Research Report*.

Chun, B., C. Ng, J. Albrecht, D. Parkes and A. Vahdat (2004). Computational Resource Exchanges for Distributed Resource Allocation. *Unpublished manuscript, available at http://www.eecs.harvard.edu/~chaki/doc/share04.pdf, Retrieved December 2007*.

de Vries, S. and R. Vohra (2003). Combinatorial Auctions: A Survey. *Journal on Computing 15*(3), p. 284–309.

Englmaier, F., P. Guillén, L. Llorente, S. Onderstal and R. Sausgruber (2006). The Chopstick Auction: A Study of the Exposure Problem in Multi-Unit

Auctions. *CESifo Working Paper No. 1782*.

Epiq-Technologies (2006). Auction Glossary. *http://www.epiqtech.com/auction_-software-Glossary.htm, Retrieved November 2007*.

Fellows, W. and S. Wallage (2007). Grid Computing – The state of the market. *451 GARS - Report 14 Executive Overview*.

Foster, I. (2002). What is the Grid? A Three Point Checklist. *GRIDToday*.

Foster, I. and C. Kesselmann (2003). Concepts and Architecture. In: *Foster, I. and Kesselmann, C.: The Grid 2*, p. 37–64. Morgan Kaufmann Publishers Inc, US.

Foster, I., C. Kesselmann and S. Tuecke (2001). The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *The International Journal of High Performance Computing Applications 15*(3), p. 200–222.

Foster, I. and S. Tuecke (2005). Describing the Elephant: The Different Faces of IT as Service. *Queue 3*(6), p. 26–29.

Giffler, B. and G. Thompson (1960). Algorithms for Solving Production-Scheduling Problems. *Operations Research 8*(4), p. 487–503.

Globus-Alliance (2007). About the Globus Toolkit. *http://www.globus.org/toolkit/about.html, Retrieved November 2007*.

Gomoluch, J. and M. Schroeder (2003). Market-based Resource Allocation for Grid Computing: A Model and Simulation. *Proceedings of the 1st International Workshop on Middleware for Grid Computing, Rio de Janeiro, Brazil*.

Grid4All (2006). D4.1: Specification of scenarios, user requirements, and infrastructure requirements. *Specific Targeted Research Project (STREP), Thematic Priority 2: Information Society Technologies*.

Hoffman, K. and M. Padberg (2000). Combinatorial and Integer Optimization. In: *Encyclopedia of Operations Research*, p. 76–83.

Krishna, V. (2002). *Auction Theory*. Academic Press.

Lai, K. (2005). Markets are Dead, Long Live Markets. *Sigecom Exchanges 5*(4), p. 1–10.

Lehmann, D., R. Müller and T. Sandholm (2006). The Winner Determination Problem. In: *Cramton, P., Y. Shoham and R. Steinberg: Combinatorial Auctions*, p. 297–317. MIT Press.

Li, P. (2007). Optimale Steuerung 1 - Mixed-Integer Lineare Optimierung. *http://tu-ilmenau.de/fakia/fileadmin/template/startIA/simulation/Lehre/-Vorlesungsskripte/Opti1/OS1-kapitel-3.pdf, Retrieved December 2007*.

Macromedia (2001). Advanced ColdFusion Administration. *http://livedocs.adobe.com/coldfusion/5.0/Advanced_ColdFusion_Administration/overview2.htm, Retrieved December 2007* (Version 5.0).

Milgrom, P. (2004). *Putting Auction Theory to Work*. Cambridge University Press.

Myerson, R. and M. Satterthwaite (1981). Efficient Mechanisms for Bilateral Trading. *Journal of Economic Theory 29*(2), p. 265–281.

Padala, P., C. Harrison, N. Perfort, E. Jansen and M. Frank (2003). OCEAN: The Open Computation Exchange and Arbitration Network, A Market Approach to Meta Computing. *Proceedings of the* 2^{nd} *International Symposium on Parallel and Distributed Computing (ISPDC'03)*.

Parkes, D. (2006). Iterative Combinatorial Auctions. In: *Cramton, P., Y. Shoham and R. Steinberg: Combinatorial Auctions*, p. 41–78. MIT Press.

Parkes, D., J. Kalagnanam and M. Eso (2001). Achieving Budget-Balance with Vickrey-Based Payment Schemes in Combinatorial Exchanges. *IBM Research Report RC 22218 W0110-065*.

Pekeč, A. and M. Rothkopf (2003). Combinatorial Auction Design. *Management Science 49*(11), p. 1485–1503.

Regev, O. and N. Nisan (2000). The POPCORN Market - An Online Market for Computational Resources. p. 148–157.

Sandholm, T. (2006). Optimal Winner Determination Algorithms. In: *Cramton, P., Y. Shoham and R. Steinberg: Combinatorial Auctions*, p. 337–368. MIT Press.

Schnizler, B., D. Neumann, D. Veit and C. Weinhardt (2006). Trading Grid Services - A Multi-attribute Combinatorial Approach. *European Journal of Operational Research, forthcoming*.

Shneidman, J., C. Ng, D. Parkes, A. AuYoung, A. Snoeren, A. Vahdat and B. Chun (2005). Why Markets Could (But Dont Currently) Solve Resource Allocation Problems in Systems. *IEEE Technical Committee on Operating Systems (TCOS)*.

Smith, W., I. Foster and V. Taylor (1998). Predicting Application Run Times Using Historical Information. *Proceedings of the IPPS/SPDP 1998 Workshop on Job Scheduling Strategies for Parallel Processing*.

Stößer, J. and D. Neumann (2007). GreedEx – A Scalable Clearing Mechanism for Utility Computing. *Networking and Electronic Commerce Research Conference (NAEC) 2007*.

Stößer, J., D. Neumann and C. Weinhardt (2007). Market-Based Pricing in Grids: On Strategic Manipulation and Computational Cost. *Journal of AIS Sponsored Theory Development Workshop, Montreal*.

Subramoniam, K., M. Maheswaran and M. Toulouse (2002). Towards a Micro-Economic Model for Resource Allocation in Grid Computing Systems. *Proceedings of the 2002 IEEE Canadian Conference on Electrical & Computer Engineering*, p. 782–785.

TechTerms.com (2005-2007). Tech Terms Dictionary. *http://www.techterms.com/definition/heuristic, Retrieved November 2007*.

Varian, H. (2007). *Grundzüge der Mikroökonomie*, Volume 7. R. Oldenburg Verlag München Wien.

Waldspurger, C., T. Hogg, B. Huberman, J. Kephart and W. Stornetta (1992). Spawn: A Distributed Computational Economy. *IEEE Transactions on Software Engineering 18*(2), p. 103–117.

Wied-Nebbeling, S. and H. Schott (2005). *Grundlagen der Mikroökonomie*. Springer Verlag, Berlin, Heidelberg.

Wikipedia.org (2007). Heuristic (computer science).

http://encyclopedia.thefreedictionary.com/Heuristic+(computer+science), Retrieved November 2007.

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

Yu, J. and R. Buyya (2005). A Taxonomy of Workflow Management Systems for Grid Computing. *http://www.gridbus.org/reports/GridWorkflowTaxonomy.pdf, Retrieved November 2007* (GRIDS-TR-2005-1).

Yu, J. and R. Buyya (2007). Workflow Scheduling Algorithms for Grid Computing. *http://www.gridbus.org/reports/WorkflowSchedulingAlgs2007.pdf, Retrieved November 2007* (GRIDS-TR-2007-10).