Universität Karlsruhe (TH)
Forschungsuniversität • gegründet 1825

Market-Based Scheduling in Distributed Computing Systems

Zur Erlangung des akademischen Grades eines
Doktors der Wirtschaftswissenschaften

(Dr. rer. pol.)

von der Fakultät für
Wirtschaftswissenschaften
der Universität Karlsruhe (TH)

genehmigte
DISSERTATION

von

Dipl.-Inform.Wirt Jochen Stößer

Tag der mündlichen Prüfung:	20. Januar 2009
Referent:	Prof. Dr. Christof Weinhardt
Korreferent:	Prof. Dr.-Ing. Stefan Tai

Karlsruhe, 2009

Acknowledgements

I am indebted to many people for their support and collaboration in the course of my Ph.D. research. Firstly, I thank my supervisor Prof. Dr. Christof Weinhardt for his strong and continuous support, which already started well during my undergraduate studies. He ultimately convinced me to pursue this Ph.D. research. I would also like to thank Prof. Dr.-Ing. Stefan Tai for co-advising this thesis and for providing me with valuable and constructive insights and comments. Prof. Dr. Bruno Neibecker and Prof. Dr. Wolf Fichtner have been so kind to serve on the board of examiners. I would like to especially thank Dr. Dirk Neumann for his guidance and for pointing me – and constantly pushing and challenging me – in the right directions.

I would also like to thank the many people who have contributed feedback and discussions to my research, especially within the projects SORMA and Biz2Grid. I would like to particularly thank Prof. Dr. Amnon Barak, Lior Amar, and Dr. Ahuva Mu'alem from The Hebrew University of Jerusalem for hosting my research stay in 2007 and for the very fruitful and interesting collaboration. I would like to express my gratitude to my colleagues from the Information & Market Engineering group for providing a stimulating and fun environment in which to grow and learn. Special thanks go to Arun Anandasivam, Benjamin Blau, Carsten Block, and Thomas Meinl for proofreading parts of this thesis.

I dedicate this thesis to my family, my parents Ingrid and Karl-Heinz and my brother Florian. I have relied on their encouragement and support throughout my studies. Finally, I want to thank Nadine for her love and her patience with my restlessness and the distractions caused by this thesis.

Contents

List of Figures

List of Tables

Abbreviations

API	Application Programming Interface	14
ASP	Application Service Providing	160
BSD	Bounded Slowdown	111
CPU	Central Processing Unit	3
CV	Coefficient of Variation	61
DLGM	Decentralized Local Greedy Mechanism	100
EERM	Economically Enhanced Resource Management	24
EGEE	Enabling Grids for E-sciencE	14
FIFO	First-In-First-Out	110
GAP	Generalized Assignment Problem	48
GB	Gigabyte	19
GLUE	Grid Laboratory Uniform Environment	27
HDD	Hard Disk Drive	28
IT	Information Technology	1
JSDL	Job Submission Description Language	27
LCG	Worldwide LHC Computing Grid	2
MACE	Multi-Attribute Combinatorial Exchange	44
MB	Megabyte	47
MKP	Multiple Knapsack Problem	48

N1GE	N1 Grid Engine	71
NP	Non-Deterministic Polynomial Time	49
OGSA	Open Grid Services Architecture	15
OSG	Open Science Grid	2
OWL-S	Web Ontology Language for Web Services	27
P-DLGM	Preemptive Decentralized Local Greedy Mechanism	103
PC	Personal Computer	12
PS	Proportional Share	141
RAM	Random Access Memory	27
RES	Reservation Requests	144
SaaS	Software as a Service	159
SH	Sanghavi-Hajek	141
SLA	Service Level Agreement	24
SORMA	Self-Organizing ICT Resource MAnagament	21
UNICORE	Uniform Interface to Computing Resources	14
US	United States	14
USD	US Dollar	2
VCG	Vickrey-Clarke-Groves	55
WSDL	Web Service Description Language	27
WSMO	Web Service Modeling Ontology	27
WSPT	Weighted Shortest Processing Time	105
ZIC	Zero Intelligence Constrained	133

Chapter 1

Introduction

1.1 Motivation

In recent years, computer resources have become increasingly considered as being a utility that can be accessed dynamically in analogy to classic utilities such as electricity, water, and telecommunication services: "Utility computing is the on-demand delivery of infrastructure, applications, and business processes in a security-rich, shared, scalable, and standards-based computer environment over the Internet for a fee. Customers will tap into information technology resources – and pay for them – as easily as they now get their electricity or water" (Rappa, 2004). According to Rappa, utilities are characterized by necessity, reliability, ease of use, fluctuating utilization patterns, and economies of scale. Computer resources match this profile very well. Rather than being a strategic asset, information technology (IT) has become a basic necessity for almost all businesses, as the prominent and controversial IT critic Nicholas Carr points out (Carr, 2003). If IT systems fail, this can result in unsatisfied customers and severe contractual penalties. Moreover, IT systems must be easy to use for customers and employees and must adapt and connect to heterogeneous systems. Most importantly, especially small- and medium-size companies will not constantly require massive amounts of computer resources. Instead, these resources will be required in the design phase of a new product only or to create daily / monthly / yearly reports, leading to fairly dynamic and unpredictable (from the provider's point of view) demand. To be able to accommodate the resulting peak loads, enterprises must maintain large computing clusters that sit idle most of the time, thus incurring tremendous costs. The results of a meta-study show that corporate data centers are only using 10% to 35% of their available computing power and that 50% to 60% of enterprises' data storage capacity is being wasted (Carr, 2005).

The validity of the utility computing approach mainly stems from its immense economies of scale. Instead of having each company and research organization to maintain its own costly computing and data centers, resource providers can benefit from economies of scale and offer these resources on-demand. Setting up computing and data centers for the first customer takes tremendous fixed costs, but serving an incremental customer or request with these existing resources only requires (comparably) minor efforts. Carr (2005) compares this to the electricity production and predicts that dispersed, private IT "plants" will be displaced by large, centralized providers.

Grid computing (oftentimes also called "the Grid") and cluster computing offer a promising technological basis for this utility-like view. These technologies permit the creation of pools of resources, which can be shared both within as well as across administrative or even organizational boundaries, e.g. between enterprises or among business units and departments within one enterprise (Foster *et al.*, 2001). This has two main benefits. Firstly, the resource pool can be used to aggregate the resource demand across users and or business units within a company to generate a more stable overall demand, ultimately at least partially eliminating the dynamic peaks in resource demand. Secondly, by enabling the dynamic access to external resources, enterprises need only accommodate the basic load; peak loads are serviced externally on demand. Especially the latter benefit leads to high expectations. For instance, The Insight Research Corporation estimates an increase of worldwide grid spending from USD 1.84 billion in 2006 to approximately USD 24.52 billion in 2011 (Insight Research, 2006). It has been projected that grids can lower total IT costs by 30% (Minoli, 2005). According to the Gartner group, early adopters will source 40% of their IT as a service by 2011.[1]

Cluster computing has a long history for business applications. Advances in networking technologies made it possible to substitute clusters of standard computers for costly and complicated mainframe computers. In contrast, grid computing originally stems from the area of high performance computing (oftentimes also referred to as "e-Science") where users need easy access to massive processing and storage resources, as in the areas of high-energy physics, physical astronomy, engineering, and biology. Consequently, grid technologies have primarily been used as a capable middleware layer by researchers. In subsequent times, the focus evolved beyond accessibility towards "coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations" (Foster *et al.*, 2001). In academia, this sharing approach has led to the emergence of several virtual organizations (e.g. TeraGrid[2], Worldwide LHC Computing Grid (LCG)[3], Open Science Grid (OSG)[4]), which facilitate resource sharing among their members.

[1]http://www.gartner.com/it/page.jsp?id=593207
[2]http://www.teragrid.org/
[3]http://lcg.web.cern.ch/LCG/
[4]http://www.opensciencegrid.org/

For enterprises, this inter-organizational sharing model has not yet made it into practice. While technical advancements with respect to sensitive issues (such as security and quality-of-service management) have occurred, this slow adoption of grid technologies is not surprising as administrative barriers – even within enterprises – are too difficult to overcome to make the sharing approach work. For instance, the free-riding problem, meaning that members participate in the Grid without contributing to it (Adar and Huberman, 2000), strongly argues against the sharing approach. It has thus been suggested that the adoption of grid computing by enterprises has been slow due to the lack of viable business models and chargeable and reliable grid services. There is a deficit of mechanisms that enable users to discover, negotiate, and pay for the use of grid services. According to one of the leading grid research institutes, The451Group, the application of resource trading and allocation models is one of the crucial success factors for establishing commercial grids (The451Group, 2007).

Complying with the proposition of The451Group, Sun Microsystems has adopted the idea of selling resources within its utility computing initiative, believing that enterprises will eventually stop maintaining their own infrastructure and instead buy computing power from dedicated resource providers. To put weight on this idea, Sun Microsystems is currently establishing *network.com*[5], an electronic marketplace for trading resources and applications. Sun started out offering a fixed price for computing services of USD 1 per central processing unit (CPU) hour. Amazon has launched comparable efforts with its *Amazon Elastic Compute Cloud* (*Amazon EC2*)[6] and *Amazon Simple Storage Service* (*Amazon S3*)[7]. Like Sun Microsystems, Amazon is offering processing power and storage for a fixed price.

However, instead of selling computer resources for a fixed price, it has been proposed to determine resource allocations and prices *dynamically* by means of markets (e.g. Sutherland (1968) for cluster-like settings and Lai (2005) for grid settings, also see Neumann *et al.* (2008c) for a recent survey). Large computer hardware manufacturers like Hewlett-Packard have already worked on or at least pondered the options for such markets (cf. Lai *et al.* (2005)). Attaching fair prices to resources in a market-like fashion ensures that the resources are only invoked when needed; users have an incentive to shift their demand for computational resources to off-peak periods where prices are lower, thus performing some kind of user-driven load balancing. Moreover, markets provide the incentive to contribute idle resources to the system in return for the market price.

[5]`http://www.network.com/`
[6]`http://aws.amazon.com/ec2`
[7]`http://aws.amazon.com/s3`

1.2 Research Outline

The question that will be addressed in this work is how markets for computational resources should be designed in order to be at the same time applicable in practice and theoretically sound. It is a well-known result from Market Engineering that there is not one omnipotent market mechanism that can be used for any setting (Weinhardt *et al.*, 2003). Instead, a set of mechanisms is needed, where the design of the mechanisms depends on the properties of the users' applications (e.g. batch vs. interactive), the resources (e.g. dedicated or shared allocation), and the overall objective of the market designer (e.g. welfare vs. revenue maximization) (Wellman *et al.*, 2001; Roth, 2002; Neumann, 2004). Due to these dependencies, the set of mechanisms is inherently open and changing. Consequently, the overall objective of this work is to extend the existing body of research on the market-based scheduling of computational tasks and resources by developing and evaluating market mechanisms for specific application settings.

The first two research questions deal with markets for grid computing settings. Grid markets by definition involve the trading of heterogeneous computer resources that are owned by multiple providers. Moreover, the grid vision is about developing a large-scale distributed computing system (on a national, international, or even global level) that appears as one virtual supercomputer. Besides these technical and structural properties, the design of grid markets is exacerbated by the single users and providers pursuing their self-interest instead of working towards one common goal, such as economic efficiency. For instance, users might try to manipulate the system by misstating job properties and valuations in the hope for a lower price. Analogously, resource providers might misstate their costs in order to obtain a higher price. Hence, the first research question addressed in this work is:

Research Question 1:

> *How can a scalable, two-sided market mechanism for clearing heterogeneous grid settings be designed so that strategic behavior from users and providers is limited?*

This research question is addressed by (1) designing a market-based heuristic, (2) developing and presenting alternative pricing schemes that are designed to induce users and providers to report their true valuations and resource characteristics to the system, and (3) an in-depth analysis of the heuristic and these pricing schemes with respect to the complexity of the allocation problem, the incentives for users and providers to try to manipulate the mechanism, and the computational impact of the pricing schemes on the mechanism.

Compared to previous attempts, the proposed heuristic and pricing schemes offer a fundamental advancement in making grid markets truly applicable in practice. As opposed to exact mechanisms, which always optimally solve the allocation problem but are infeasible in practice, the proposed mechanism is highly scalable but still yields near-optimal allocations. Furthermore, compared to previous mechanisms, the mechanism allows strategic behavior on both sides of the market. Proportional critical-value pricing constitutes the main improvement over previous mechanisms. If complemented by this pricing scheme, the heuristic achieves truthfulness (with respect to the reporting of valuations) on the demand side of the market while limiting strategic behavior on the supply side.

On its downside, the proposed deterministic heuristic is vulnerable to specific worst cases. Randomization provides a promising remedy towards intercepting such worst cases. Random choices of the allocation algorithm can make these special cases improbable and thus mitigate the deficiencies of deterministic mechanisms (Nisan and Ronen, 2001; Goldberg *et al.*, 2006). From a strategic viewpoint, however, this also requires the design of new pricing schemes, as the prices used for deterministic mechanisms can no longer be determined given the random choices of the allocation algorithm. This gives rise to the second research question:

Research Question 2:

> *How can a randomization of the heuristic allocation algorithm improve efficiency?*
> *How can the pricing scheme be modified to limit strategic behavior from users?*

Randomization can easily be shown to avoid worst cases on average. The true benefit of randomization, however, lies in combining it with the inherently decentralized nature of markets. The idea is that the lightweight randomized heuristic can easily be computed by every market participant, for instance as kind of a mandatory "participation fee". A central auctioneer can then collect the local solutions from the participants and aggregate them so as to generate a desirable overall outcome.

While the theoretical benefits of randomization and distributed outcome determination are compelling, it remains to be shown whether this actually improves on the deterministic heuristic with respect to efficiency in the "average" case, i.e. given more realistic input than mere worst cases. Furthermore, randomization violates the monotonicity property of the allocation algorithm, which is crucial for a truthful (in dominant strategies) mechanism (Lavi *et al.*, 2003). Consequently, novel solution concepts are needed.

The contribution of this work is a randomization of the presented deterministic heuristic. A pay-as-bid pricing rule is proposed that is aligned with the allocation algorithm so as to incentivize users to bid "close" to their true valuation. Finally, the randomized heuristic is benchmarked against the deterministic heuristic to analyze the average case performance.

The first two research questions consider so-called batch or offline grid settings, in which the market mechanism has rather comprehensive information about the computational jobs of the users, in particular with respect to the release dates, i.e. the time when these jobs are submitted to the system. The third and fourth research question focus rather on cluster and utility computing settings, where users compete for a centralized and homogeneous pool of resources. Moreover, these questions consider so-called online settings. In online settings, the release dates of jobs are unknown to the mechanism until the jobs are released to the system. A typical example for such settings are interactive applications where the user steers the processing of the application. Online mechanisms – i.e. market mechanisms in online setting – thus have to make allocation and pricing decisions with less information and these decisions may prove unfortunate as new information (e.g. new jobs) is released. The advantage of online mechanisms compared to periodic, offline mechanisms (as considered in research questions one and two) is increased responsiveness.

In online settings, flexibility of the allocation mechanisms becomes crucial. Until recently, only few grid and cluster systems provided preemptive migration. Preemption can mitigate unfortunate earlier allocations decisions by allowing the allocation mechanism to suspend a running job in favor of some more desirable job and to possibly continue this suspended job later on the same compute node. Migration is a natural extension to preemption and denotes the ability of dynamically moving computational jobs *across* compute nodes. The importance of preemptive migration is stressed by the emerging technology of virtualization. Virtualization enables the efficient management of computing systems by abstracting from the physical resources. This technology provides off-the-shelf support for virtual machine migration, thus making the use of preemption and migration more accessible. The third research question centers around the use of preemption in economic online settings:

Research Question 3:

> *What is the benefit of performing preemptions in economic online settings with dedicated resources?*

The results of this work showcase how the performance of economic online mechanisms can be improved by performing preemptions, which has largely been neglected in the existing literature on market mechanisms. The analysis is done using an existing, non-preemptive mechanism by Heydenreich *et al.* (2006) as a baseline, extending this mechanism's allocation and pricing scheme so as to include preemptions, and analytically and empirically benchmarking the two mechanisms.

The previous research questions consider settings in which the resources are more or less dedicated, meaning the flexibility when allocating computational jobs to compute nodes is restricted by the indivisibility of the node's resources (e.g. CPU cores). Only a limited number of jobs can be allocated to the same node at the same time. With the recent rise of utility computing, there is a trend back towards sourcing computer resources from centralized providers. Moreover, virtualization technologies make the resources appear as a homogeneous and almost perfectly divisible *pool* of resources. The fourth research questions hence is:

Research Question 4:

How should shared resources be allocated and priced in online settings?

The currently most prominent market-based scheduling procedure for allocating and pricing such *divisible* resources to online requests is Proportional Share (e.g. Stoica *et al.* (1996)). Sanghavi and Hajek (2004) propose a market mechanism for the allocation of bandwidth that is also applicable to more general settings. This work discusses how this Sanghavi-Hajek mechanism can be applied to the allocation of more general types of computer resources. Conditions are derived under which the Sanghavi-Hajek mechanism outperforms Proportional Share regarding both the provider's revenue and the allocative efficiency for the case of two users. For larger settings, the mechanisms are compared by means of an agent-based simulation based on real-world workload traces. However, the basic Sanghavi-Hajek mechanism does not enable users to obtain service guarantees for critical applications. Therefore, an extension to the basic mechanism is proposed that allows users to choose between (1) a certain price and a dynamic (uncertain) service level or (2) a dynamic price and a guaranteed quality of service.

1.3 Structure of this Work

The previous research outline is reflected in the structure of this work (cf. Figure 1.1).

Chapter 2 introduces the concepts that build the basis of this work. First, distributed computing concepts are briefly discussed that support inter-organizational systems and thus *market-based* computing systems. Then a conceptual architecture for distributed computing markets is presented, which is followed by a discussion of the arising technical and economic challenges. The key result of this discussion is that there is not one single, omnipotent market, as pointed out above. Instead, there is a need for a suite of market mechanisms depending on the nature of the applications and resources that are to be traded. Finally, the research methodologies are briefly discussed that are subsequently used to answer the research questions underlying this work.

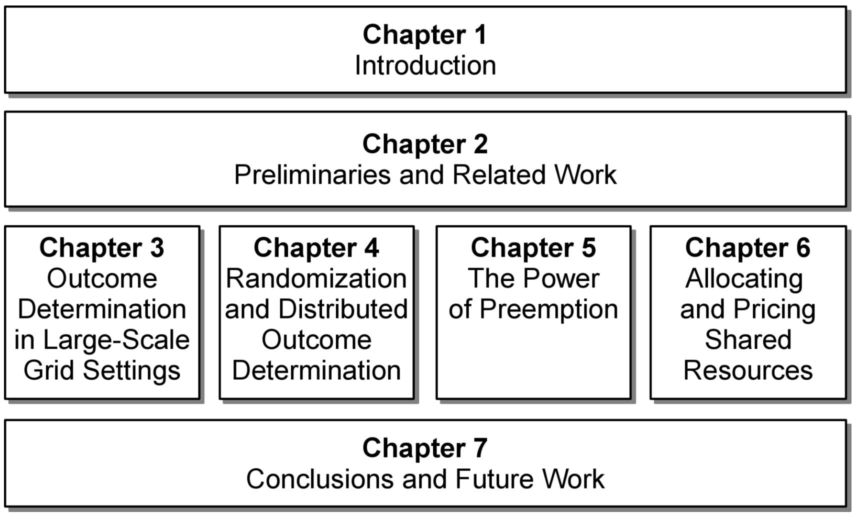

Chapter 1
Introduction
Chapter 2
Preliminaries and Related Work
Chapter 3
Outcome Determination in Large-Scale Grid Settings
Chapter 4
Randomization and Distributed Outcome Determination
Chapter 5
The Power of Preemption
Chapter 6
Allocating and Pricing Shared Resources
Chapter 7
Conclusions and Future Work

Figure 1.1: Structure of this work.

Chapters 3 to 6 constitute the core of this work and are structured along the four research questions and focus on four specific scenarios and market mechanisms. Whereas Chapter 2 introduces general related work, each subsequent chapter discusses more specific related work that is relevant to the scenarios at hand.

Finally, Chapter 7 summarizes the key contributions of this work and points to open questions for future research.

1.4 Related Publications

Parts of this thesis have been published and presented at various academic conferences, workshops, and in several journals.

With respect to Chapter 2, the presentation of a conceptual architecture for distributed computing markets, the discussion about economic and technical challenges as well as the resulting market structure have been published in two journal articles, Neumann *et al.* (2008c) in Electronic Markets and Neumann *et al.* (2008a) in the Journal of Grid Computing. A preliminary version (Neumann *et al.*, 2007a) has been presented at the 20^{th} Bled eConference, Bled, Slovenia.

The work on a deterministic, partially truthful heuristic mechanism in Chapter 3 has been presented at the 15^{th} European Conference on Information Systems, St. Gallen, Switzerland (Stößer *et al.*, 2007a), the Networking and Electronic Commerce Research Conference, Riva Del Garda, Italy (Stößer and Neumann, 2007), and the Journal of AIS Sponsored Theory Development Workshop, Montréal, Canada (Stößer *et al.*, 2007b). A consolidated and extended version has been published in a journal article (Stößer and Neumann, 2008).

Concerning Chapter 4, the results about the randomized extension of this heuristic have been presented at the IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services, Washington, D.C., USA (Stößer, 2008).

The work in Chapter 5 about the preemptive online scheduling of dedicated resources has been presented at the 5th International Workshop on Grid Economics and Business Models, Las Palmas, Spain (Amar *et al.*, 2008c).

The investigation of online mechanisms for shared resources in Chapter 6 has been presented at the 20th Bled eConference, Bled, Slovenia (Bodenbenner *et al.*, 2007) and the 41st Hawaii International Conference on System Sciences, Waikoloa, Hawaii, USA (Stößer *et al.*, 2008).

The intention of this work is to present a sound, complete and structured overview of this research, discuss the relationships between the research questions, and point to areas for future research.

Chapter 2

Preliminaries and Related Work

The aim of this chapter is to introduce the key technological and economic considerations underlying this work. First, distributed computing concepts are outlined that form the basis for inter-organizational systems and thus *market-based* computing systems. Following this brief technical introduction, the need for economic principles in such distributed systems is discussed. Since this work integrates both the technical and the economic points of view, the arising technical and economic challenges are subsequently highlighted. Most importantly, this chapter describes a market structure that frames the research questions in the design of distributed, market-based computing systems that build the core of this work. Finally, the underlying methodologies for approaching these questions are presented. While this chapter contains general related work on distributed computing systems and markets for such systems, the subsequent chapters contain more detailed discussions of the literature that is specific to the research questions at hand.

2.1 Distributed Computing Concepts

Distributed computing is defined as "a computer system in which several interconnected computers share the computing tasks assigned to the system" (Milojicic *et al.* (2002) citing IEEE (1990)). The focus of this work is on the market-based allocation of computational resources in settings where *selfish users* share *scarce resources* for the execution of *computing tasks*. Consequently, distributed computing systems can be seen as the technological basis underlying this work and enabling such markets. This section briefly introduces some of the key concepts and technologies for distributed computing systems and subsequently delineates the characteristics that distinguish these concepts for the purpose of this work.

2.1.1 Cluster Computing

In the 1980's and 1990's, cluster computing emerged as a viable alternative to mainframe computers (also simply called "mainframes"). In the past, mainframes had largely been computers that were specifically tailored towards certain applications, especially business-critical applications that processed large amounts of data. Mainframes were characterized by their high availability and performance. They oftentimes involved special computing and programming paradigms, e.g. vector computing (Dongarra *et al.*, 2005).

The ever-increasing performance and decreasing costs of standard personal computers (PCs) combined with advances in networking technologies and management software (e.g. for load-balancing and user management) allowed to create *networks of standard PCs*, so-called "clusters" (see Baker *et al.* (1995) for a review of cluster computing and cluster management systems). For Bell and Gray (2002), "a cluster is a single system comprised of interconnected computers that communicate with one another either via a message passing; or by direct, internode memory access using a single address space". Consequently, it became possible to improve the throughput of applications not only by the use of mainframes, but also by executing the applications on a distributed cluster of PCs at manageable costs. At the same time, traditional mainframe applications could be ported to standard PCs, which are generally easier to manage and program (Bell and Gray, 2002). *MOSIX Version 1*[1], *Condor*[2], and *Moab*[3] are some of the most prominent cluster management systems, which provide functionalities for managing the resources such as failover, user authentication, and distributing the workload on the resources.

In the past, cluster computing has been used for enabling the efficient sharing of computing resources between multiple users within the same organization. Recently, this scope has been extended to utility computing settings, where a centralized utility computing provider such as Amazon or Sun Microsystems hosts cluster systems that predominantly service *external* users. Note that the provisioning of computing resources to external users has recently also been subsumed under the term "cloud computing". However, cloud computing is more than the provisioning of low-level computing resources, as will be discussed in the outlook in Chapter 7.

2.1.2 Voluntary Computing

Computing power has not only become a scarce resource for business applications. In science, computer-based simulations and data analyses have become widely used, making costly computer resources a limiting factor in academia also. This gave rise to so-called "voluntary

[1] `http://www.mosix.org/`

[2] `http://www.cs.wisc.edu/condor/`

[3] `http://www.clusterresources.com/pages/products/moab-cluster-suite.php`

computing" projects that aim at connecting computer resources across research and computing centers, ultimately including voluntary, private contributors as well (Ong *et al.*, 2002; Cirne *et al.*, 2006). The thus far most prominent voluntary computing project was *Seti@Home* (now also called *Seti@Home Classic*)[4]. One result of this project has been the development of the *BOINC* platform[5], which now hosts a wide array of applications, e.g. a biological application for predicting the structure of proteins (*proteins@home*[6]) or a physical application for analyzing the data from gravitational wave detectors (*Einstein@Home*[7]). It is based on leveraging idle time of voluntarily participating desktop PCs and currently[8] comprises 319,691 volunteers and 568,288 computers.

One key advantage of voluntary computing systems typically is their lightweight middleware; the voluntary contributor only needs to install a rather small middleware (client) component (oftentimes called the "worker" or "slave") that then communicates with a central "master". The master process divides the computational task into smaller sub-tasks, distributes these tasks to the slaves, and aggregates the sub-results to the overall result (Shao *et al.*, 2000; Sarmenta, 2001).

2.1.3 Grid Computing

Grid computing denotes a computing model that distributes the processing of computational tasks across an administratively and geographically dispersed infrastructure (Foster *et al.*, 2001). By connecting many heterogeneous computing resources, virtual computer architectures are created, increasing the utilization of otherwise idle resources.

Due to the multitude of available technologies and computing paradigms it is oftentimes hard to distinguish grid computing from cluster computing and voluntary computing. Foster (2002) thus compiled a simplified "three point checklist" for assessing whether a distributed computing system might be regarded as being a grid. Due to the difficulty of giving a unique definition and thus the lack of such definitions, Foster's checklist has been widely adopted and referenced. According to Foster, a grid

- "coordinates resources that are not subject to centralized control [...],
- uses standard, open, general-purpose protocols and interfaces [...],

[4]`http://seticlassic.ssl.berkeley.edu/`

[5]`http://boinc.berkeley.edu/`

[6]`http://biology.polytechnique.fr/proteinsathome/`

[7]`http://einstein.phys.uwm.edu/`

[8]as of 14.11.2008

- delivers non-trivial qualities of service [...]".

A grid is based on grid middleware that provides services for authentication, resource discovery and access, file transfer, etc. It typically consists of a collection of cluster systems and constitutes an abstraction layer that provides unified access to the underlying cluster resources. The currently most prominent grid middlewares are *Globus Toolkit*[9], *UNICORE* (Uniform Interface to Computing Resources)[10], and *gLite*[11], which are all open source.

Globus Toolkit has been developed by US researchers and is currently available in version 4.2.1. Besides the core middleware services, *Globus Toolkit* offers an application programming interface (API) that allows the application developer to tailor the application towards the use in a grid environment (Snelling *et al.*, 2002). This underlying design principle is best described by the following quote from Foster and Kesselman (1997): "The Globus project is attacking the meta-computing software problem from the bottom up by developing basic mechanisms that can be used to implement a variety of higher level services." Consequently, *Globus Toolkit* can be used for a wide range of applications.

UNICORE has been the outcome of German and European grid efforts. It is currently available in version 6.1. In contrast to *Globus Toolkit*, it aims at a top-down approach, abstracting from the underlying grid environment. The idea is that the user ideally should not have to change the application. *UNICORE* allows the user to define a workflow of tasks to be executed in the Grid, such as dependencies between computational tasks and user interaction (Snelling *et al.*, 2002). The drawback of *UNICORE*'s abstraction, however, is the limited support for customization and application development.

One of the most prominent grid activities in academia is the EGEE III project, which is funded by the European Commission.[12] EGEE stands for "Enabling Grids for E-sciencE" and currently brings together researchers from 50 countries with the common aim of developing a grid infrastructure that is suited for almost any scientific research, especially where the time and resources needed for running the applications are considered impractical when using traditional IT infrastructures (e.g. weather forecasts, protein folding, etc). The established EGEE grid comprises over 80,000 CPU cores across about 300 sites. In total, EGEE serves more than 300,000 computational jobs a day from about 10,000 users.[13] The EGEE grid is built on *gLite*, currently in version 3.1. *gLite* consists of older *Globus Toolkit* components (from version 2.4.3), *Condor* components, as well as components that have been developed within the EGEE sub-projects themselves (Burke *et al.*, 2008). *gLite* is mainly being used in high energy physics.

[9]`http://www.globus.org/`

[10]`http://www.unicore.eu/`

[11]`http://glite.web.cern.ch/glite/`

[12]`http://www.eu-egee.org/`

[13]Statistics from `http://www.eu-egee.org/`, 24.10.2008

Globus Toolkit is oftentimes regarded as being the de-facto standard grid middleware. However, the co-existence of these various middlewares and different approaches makes it hard for users to develop and deploy grid applications, yet alone to realize the vision of "the Grid". To this end, the Open Grid Forum (formerly Global Grid Forum) is the most important standardization body for grid protocols and interfaces.[14] One of the outstanding results has been the specification of the Open Grid Services Architecture (OGSA) (Foster *et al.*, 2006), a reference architecture for grid middlewares. OGSA combines grid concepts with Web services principles and technologies and aims at standardizing the interfaces to common grid services such as job submission or security.[15] Up to now, all three middlewares introduced above have at least partially implemented OGSA standards.[16] This can be a first step towards fully standardized, easier-to-use grid middlewares and might thus increase the acceptance and adoption of grid technologies in the future.

2.1.4 A Classification

The various distributed computing concepts can be distinguished by many possible criteria. Based on the scenarios that will be considered in this work, the most useful and distinguishing criteria are listed in Table 2.1.

Criteria	Cluster computing	Voluntary computing	Grid computing
Sharing approach	intra-org. sharing, utility computing	CPU scavenging for scientific applications	inter-organizational resource sharing
Control over resources	centralized	decentralized	decentralized
Type of resources	homogeneous	heterogeneous	heterogeneous
Qualities of service	non-trivial	trivial	non-trivial
Mode of allocation	dedicated / shared	shared	dedicated / shared
Use of virtualization	widespread	n/a	in its beginnings
Support for preemptive migration	available	n/a	in its beginnings
Advance reservation	available	n/a	in its beginnings
Type of applications	batch & interactive	batch	batch

Table 2.1: Distinguishing distributed computing concepts.

[14] `http://www.ogf.org/`

[15] See `http://gdp.globus.org/gt4-tutorial/singlehtml/progtutorial_0.2.1.html#id2482481` for a brief introduction to these concepts.

[16] See e.g. *Java WS Core* for *Globus Toolkit* for building stateful Web services, `http://www.globus.org/toolkit/docs/4.2/4.2.1/common/javawscore/#javawscore`.

The resources in a cluster are under centralized control of the organization's computing department, such as a computing center of a utility computing provider such as Amazon or Sun Microsystems, an enterprise or a research center. Cluster systems are designed to operate at high qualities of service, such as reliability, throughput, response time, and security. The resources in a cluster are usually homogeneous to simplify the management of the cluster and are located in one single computing center. The resources – i.e., the computers within the cluster, in the remainder also called "compute nodes" or simply "nodes" – can be either allocated in a dedicated manner such that only one user task can be executed on a specific node at the same time, or in shared mode such that multiple users or tasks are assigned to the node in parallel. The latter is simplified by the use of virtualization technologies. Virtualization is used to efficiently utilize the resources, meet diverse user demands, e.g. with respect to the operating system, and to enhance security (Barham *et al.*, 2003). For instance, the user applications are isolated so that there are no side-effects between the applications running in different virtual machines, and the applications cannot tamper with the underlying hardware or operating system.

Cluster systems support a wide range of applications. So-called "batch applications" are characterized by a planned execution time and an expected termination time. Users send their computational tasks ("jobs") to the system, these tasks are executed without any further user interaction, and the results are finally reported back to the user. Oftentimes at least an estimate of the application's processing time and resource consumption is known a priori, e.g. by means of statistical analyses of past runs (Degermark *et al.*, 1997; Smith *et al.*, 1998). Examples are compute-intense applications like numerical simulations. For such batch applications, state-of-the-art cluster schedulers such as *Maui*[17] support the advance reservation of resources. In contrast, so-called "interactive applications" require responsive services or resources *on demand* depending on the interactions with users.[18] Different than batch applications, with interactive applications it is not possible to plan execution time far in advance, so there can be unpredictable peaks of requests occurring within a short time. Examples for interactive applications are interactive data analyses, where the user steers the analysis based on iterative input and output. Especially with interactive applications, it is important for the system to be able to preempt running jobs and possibly migrate jobs across nodes in order to react to system changes, such as the failure of a node or in case a high-priority job enters the system. Preemptive migration can be performed on several levels, such as the migration of single application processes (e.g. in the *MOSIX* system (Barak *et al.*, 2005)) or the migration of complete virtual machines (e.g.

[17]`http://www.clusterresources.com/pages/products/maui-cluster-scheduler.php`

[18]See e.g. `http://www.cisl.ucar.edu/docs/LSF/7.0.3/admin/interactive.html#wp16802` for a brief description of how to run interactive applications or tasks using the *Platform Load Sharing Facility*.

VMotion by *VMware*[19]).

In voluntary computing, in contrast to cluster computing, the resources are inherently under the decentralized control of the voluntary contributors. The execution of the application and the aggregation of the results is typically managed by the central master component (Bonorden *et al.*, 2006). However, this entity does not have ultimate control over the resources as such. The resources can only be accessed if this is permitted by the contributor and if there are idle resources available. The system must be able to integrate highly heterogeneous resources, e.g. operating systems, which are distributed across multiple locations. The applications are generally run in shared mode, i.e. in parallel to applications of the contributor. This model is oftentimes referred to as “CPU scavenging” or “cycle stealing” (Globus, 2001). While virtualization may be used by the individual contributor, it is not inherent to the voluntary computing system as such. The applications are typically small batch computations that the central master sends to the dispersed slave nodes, which then report the results back (Bonorden *et al.*, 2006). Preemptive migration is not supported, as there is no sophisticated middleware involved. The application domain of voluntary computing approaches are scientific applications, where private people have an altruistic interest in donating resources.

In summary, the basic difference between cluster computing and voluntary computing is the move from centrally managed, homogeneous resources to heterogeneous resources, which are contributed by private contributors and hence are under decentralized control. The resulting cost reductions, however, mainly come at the expense of reduced qualities of service.

A grid can roughly be considered as representing a collection of clusters. The grid middleware has no direct control over the underlying resources, but accesses these resources via the local (cluster) resource manager. As in a cluster, the compute nodes can be either allocated in dedicated or shared mode. The use of virtualization is only in its beginnings but is currently being pushed by the advances in the underlying cluster systems. Besides the benefits discussed above, virtualization technologies can be used to simplify the submission of computational jobs. Instead of having to specify the exact types of resources that are required for the computational job, the user can submit the application encapsulated inside a virtual machine (Freitag *et al.*, 2008). Essentially, most grid applications are batch applications since batch processing is easiest to manage. *Globus Toolkit*, for instance, currently only offers very limited support for the execution of interactive applications on grids.[20] The use of preemption is in its beginning. For instance, the *GridWay* meta-scheduler[21], which is part of the *Globus* project, supports migra-

[19]`http://www.vmware.com/products/vi/vc/vmotion.html`

[20]Interactive applications that are steered by means of a Web server, parameter files or similar workarounds are possible. However, it is currently not possible to run a graphical user interface on a remote compute node since in a grid the user cannot directly access this node.

[21]`http://www.gridway.org/`

tion. Due to the heterogeneity of the resources involved in a grid, migration is typically not yet support on a system level but only on a user level, meaning that the user has to implement checkpointing[22] and recovering strategies herself. A notable exception is the *MOSIX* system (Version 2) that – besides its long history in the cluster area – is moving towards grid settings and supports the migration of application processes (Barak *et al.*, 2005). While advance reservation is not fully supported by the grid middleware yet, *Globus Toolkit* will most likely support this feature in the near future to make use of the capabilities provided by the underlying cluster middleware.[23] As regards the sharing approach, by definition grids denote the sharing of resources across organizational domains.

Compared to cluster computing, grid computing has essentially the same distinguishing properties as voluntary computing does. However, in contrast to voluntary computing, resources are distributed on an organizational level and not on an individual level. Moreover, since grids are based on interconnecting clusters, they achieve qualities of service that are ideally comparable to the levels achieved by cluster systems.

While the predominant approach in these distributed computing concepts is the sharing of resources, this work proposes the use of markets for allocating and pricing the resources in such systems. The following section will present the rationale for introducing economic principles to these systems.

2.2 Why Markets for Scheduling in Distributed Computing Systems?

As pointed out above, the key assumption underlying this work is that computing resources are scarce. There are two main arguments that support this view. Firstly, while the processing power of computer systems has been increasing at an almost exponential rate (which has prominently been described by "Moore's law"), resource consumption has at least kept pace. There seem to be application domains in both science and industry, especially involving complex simulations and data analyses, that exhibit an almost infinite demand for computer resources. For example in the finance industry, companies need large amounts of computing power to calculate their risk exposure or to determine prices for financial options (Huang and Thulasiram, 2005; Marena *et al.*, 2008). Secondly, while the costs of producing computer hardware have been falling, the costs of operating and maintaining this hardware have steadily increased, which is exacerbated

[22]Checkpointing denotes the saving of the current state of the application in files that can later be used to restart the application from that state.

[23]`http://dev.globus.org/wiki/Incubator/GARS`

by the increase in electricity prices and the need to reduce carbon dioxide emissions.[24] For instance, the ratio of energy costs to hardware expenses (considered over a time period of three years) is rapidly approaching unity (Wang, 2007; Beladym, 2007). For each Watt that is spent on powering the servers, computing and data centers currently have to spend between one and two additional Watts on cooling, air conditioning etc. Consequently, resource providers will aim at reducing their pool of computing resources in order to minimize their operating costs.

Markets seem to be adequate for commercial computing systems as they contribute to business models that charge based on resource usage and scarcity. As a side-effect, markets have the ability to improve the efficiency of the resource allocation (Byde *et al.*, 2003; Shneidman *et al.*, 2005; Lai, 2005). Today's resource management systems in clusters and grids have recognized the need for expressing valuations by including user priorities, weighted proportional sharing, and service level agreements that set upper and lower bounds on the resources available to each user or user group (Irwin *et al.*, 2004; Neumann *et al.*, 2006b). Maximizing overall utility (i.e. the sum of the users' and providers' valuations), however, is only possible if the resource manager knows the attached valuations, meaning the exact relative weights at any point of time. Knowing the valuations at any time is a very demanding requirement, as users typically have no incentive to report decreases in their valuations because they loose priority and correspondingly value by not getting their computation completed (earlier).

Hence, valuation-oriented approaches are not sufficient per-se to achieve an efficient solution. Only if all participants are willing to report their priorities and valuations *honestly*, these algorithms (e.g. Proportional Share (Lai *et al.*, 2005)) will work well. This is where markets for the resource allocation in distributed computing systems enter the discussion. The pricing mechanisms of markets have the ability to set the right incentives for users to reveal their true valuation as well as for resource owners to provide the resources that are scarcest in the system. With the introduction of prices, incentives will be given to the users to substitute the scarce resource, say, memory, with a less scarce resource, say, storage. For instance, a fixed pricing scheme that requests \$10 for one Gigabyte (GB) of memory and \$1 for one GB of storage, sets incentives to swap out data to storage instead of using up the more expensive memory.[25]

A fixed pricing scheme, however, does generally not achieve an efficient allocation if demand is variable, which is illustrated by the following simplified example, based on Lai (2005). Suppose a resource provider is offering a fixed price and has negligible variable costs, as shown in Figure 2.1. There are two users, a user with a high valuation and a user with a low valuation. Demand (the users' willingness to pay) changes over time as depicted by the parabola. Consider

[24]These developments recently gave rise to so-called “Green IT” initiatives that aim at reducing the energy consumption of computing systems.

[25]In the remainder of this work, currency units (\$) will be omitted for the sake of simplicity when the meaning is clear from the context.

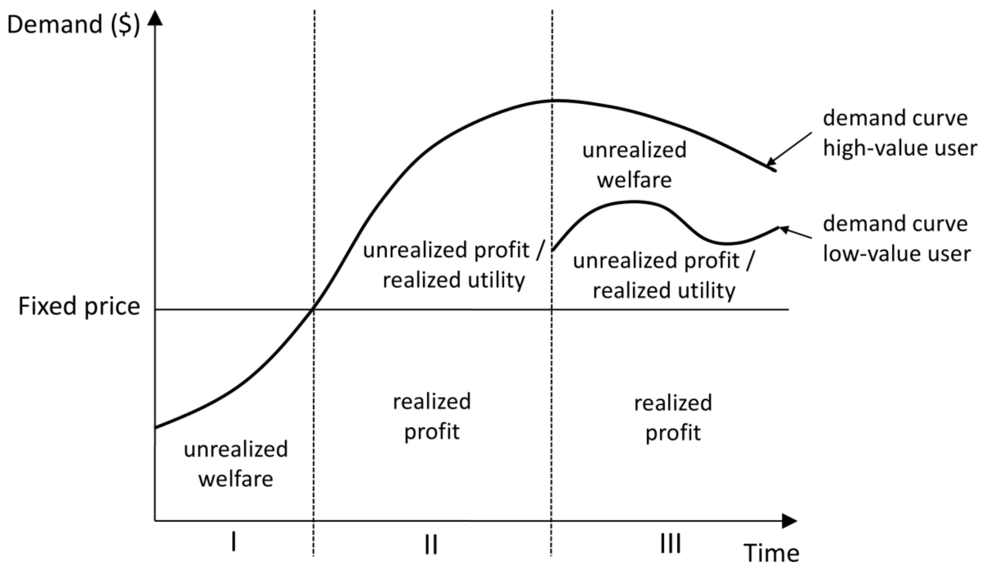

Demand ($)
demand curve high-value user
unrealized welfare
demand curve low-value user
unrealized profit / realized utility
unrealized profit / realized utility
Fixed price
realized profit
realized profit
unrealized welfare
I
II
III
Time

Figure 2.1: Fixed pricing, based on Lai (2005).

the following sample scenarios:

- **Scenario I** (left part of the graph): Only the user with the high valuation is present. If demand is below the fixed price, no resource will be allocated. This is because the valuation for the resource, represented by the demand curve, is below the price. As a consequence, there is unrealized welfare, which is depicted by the area below the demand curve, and that cannot be split between the user and the provider in any way.
- **Scenario II** (middle part of the graph): The willingness to pay of the high-value user exceeds the fixed price. There is unrealized profit for the provider, illustrated by the area between the demand cure and the fixed price. The provider could have charged the user a higher price and the user would still have accepted. This unrealized profit is instead realized by the user in the form of increased utility.
- **Scenario III** (right part of the graph): Assume both users are present but the provider is *not* able to determine the high-value user; the resource is allocated to the user with the low valuation. Consequently, there is unrealized welfare amounting to the difference in the users' valuations. Additionally, as in Scenario II, there is again unrealized profit, respectively.

Market mechanisms promise to work well for scheduling by taking into account the dynamic resource demand and supply when making their allocation and pricing decisions (Shneidman *et al.*, 2005). Appropriate market prices (appropriate in a way that will be established in the remainder of this work) cause the users to make efficient use of the resources (efficient coding, user-driven load balancing to off-peak periods) and to reveal their true valuations, which helps in avoiding unrealized utility and profit. At the same time, the prices provide incentives to resource owners to contribute their idle resources to the system. Overall, appropriate market mechanisms aggregate the dispersed and incomplete information about resource demand and supply to generate a desirable overall outcome.

It is important to note that in the remainder of this work it is assumed that the users and providers have decided to participate in the market. Hence, the aim of this work is not to elaborate about in which settings users and providers should participate in a market, and in which they should not. Further, it is assumed that market participants know their true characteristics, in particular with respect to their valuations for resources. To this end, Chapter 7 points to complementary research areas and questions for future research that can help to relax these assumptions.

2.3 Technical Challenges

Summarizing the discussion above, markets can play a crucial role for commercial computing systems as they set the right incentives for contributing resources while at the same time avoiding excessive resource consumption. However, the use of markets needs to be integrated with state-of-the-art distributed computing middleware – otherwise the concept will remain purely theoretic. In this section, a generic conceptual architecture for distributed computing markets is presented to allow for a discussion of the technical challenges that such markets raise.

Figure 2.2 illustrates the conceptual architecture in terms of the functional entities of a market for computational resources, their responsibilities and their dependencies. The boxes represent functional entities and the arrows indicate information flows between them, where an arrow from an entity *A* to an entity *B* means that *B* receives input from *A*. It should be noted that this architecture can be implemented in a decentralized manner. There can be multiple instances of the various components within the same system, each tailored towards specific functionalities or redundantly servicing parts of the overall system to achieve fault tolerance and robustness.

A first prototype based on this architecture has been implemented in the *SORMA* project (Self-Organizing ICT Resource MAnagement).[26] *SORMA* aims at implementing "methods and tools for establishing an efficient market-based allocation of IT resources in order to enable resource accessibility for all users and to increase user satisfaction, profit and productivity" (Neumann *et al.*, 2007b). The architecture and its description is based on SORMA Consortium (2007) and Neumann *et al.* (2008a).

[26]`http://www.sorma-project.eu/`

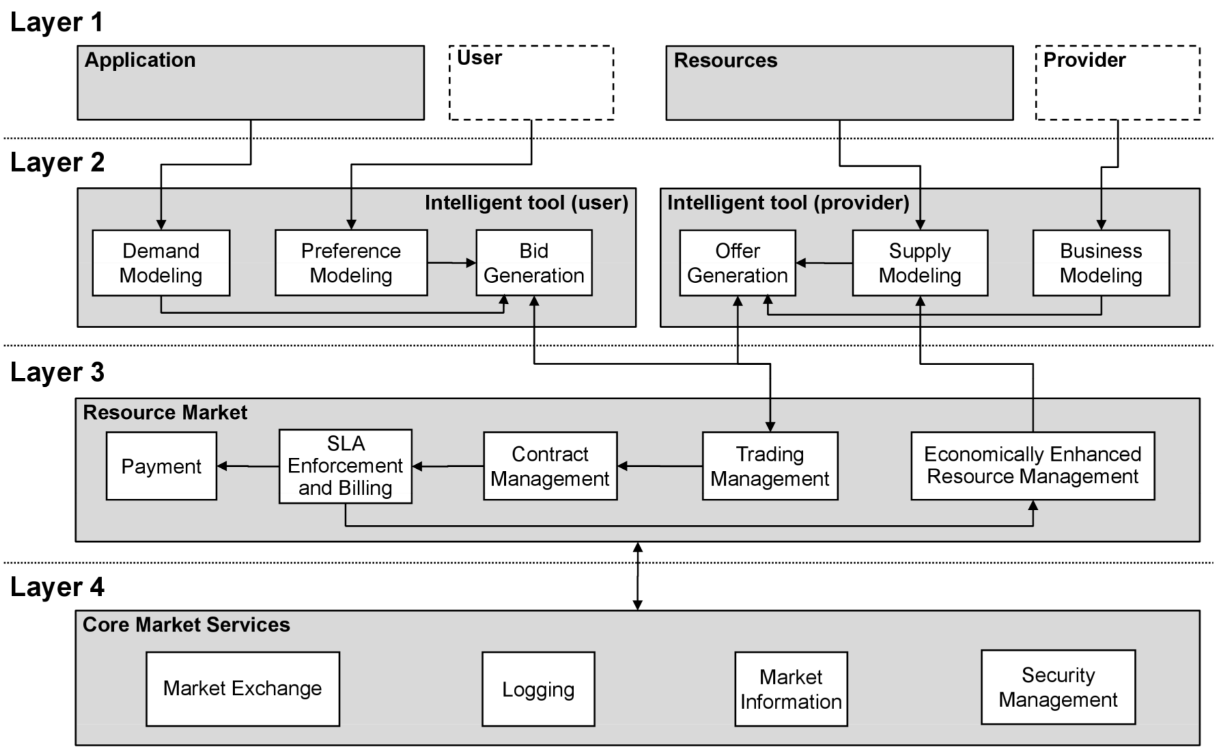


Figure 2.2: Conceptual architecture for a market for distributed computing systems.

2.3.1 Layer 1: Applications and Resources

Layer 1 comprises the end-users' applications as demand, the provided resources as supply and the corresponding market participants, i.e. the users and the resource providers. The resource provider makes use of the so-called intelligent tools of Layer 2 to model the business strategies and the offered resources. On the consumer side, either the user of the application or the application itself uses the intelligent tools to model the application's resource requirements based on the user's economic preferences. If an application is decomposed into several services, it is the application's task to realize a service deployment coherent to the allocations made on the market.

2.3.2 Layer 2: Intelligent Tools

For an easy access to the market, both the users and the resource providers must be supported by a set of intelligent tools (MacKie-Mason and Wellman, 2006; Neumann *et al.*, 2006a, 2008b).

- **Demand Modeling:** The user needs to be equipped with a tool to specify the technical requirements of her application, i.e. the properties of the resources. In case the requirements are specified in terms of aggregated services, demand modeling also assumes the task of decomposing a request into its constituent services.

- **Preference Modeling:** This component facilitates the description of the user's economic preferences that will determine her bidding strategies on the market, e.g. the user can define the maximal amount she is willing to pay or if she prefers cheap over reliable resources.
- **Bid Generation:** The bid generation denotes the component that generates and places the bids on the market on behalf of the user. For this purpose the bid generator retrieves the user preferences, the technical requirements and the current state of the resource market, and forms the bid or bid series accordingly.
- **Supply Modeling:** The supply modeling component is the provider's correspondent to the user's demand modeling component. This component allows the providers to specify the technical properties of the contributed resources.
- **Business Modeling:** Analogously to the consumer preference modeling, the providers need to specify their business models to generate adequate offers. For example, one part of the description could be a pricing model that specifies if the users have to pay for booked timeslots or for the actual usage. Moreover, the provider might charge users from outside the organisation a premium compared to internal users.
- **Offer Generation:** The offers are derived from the technical supply descriptions and the business model of the respective provider via the offer generation component. As an automated agent, the offer generation component continuously observes the state of the market and places and updates offers.

2.3.3 Layer 3: Resource Market

The third layer is headlined Resource Market, as it accounts for the economic matchmaking (i.e., which user request is allocated to which resource, when and for what price?). Accordingly, it provides the necessary functionality along the economic matching process, starting from the interaction with the market over the determination of the actual allocations and prices to the subsequent billing and payment.

- **Trading Management:** The actual matchmaking process that assigns the users' requests to suitable offers is executed by the trading management. As a first step, the trading management matches the technical descriptions of the request (received from the bid generation component) to the technical descriptions of the offered resources (received from the offer generation component). In the second phase the trading management orchestrates the bidding process between the users and the providers following the protocol of the employed market mechanism. If the bidding process finishes successfully, the pairs of corresponding bids and offers are submitted to the contract management.

- **Contract Management:** The contract management transforms the pairs of bids and offers into mutually agreed contracts. One important part of these contracts are the service level agreements (SLAs), which define the agreed terms of usage of the resources and the pricing as well as penalties in case these terms are violated by either party. After stating the contract, the contract management initiates the SLA enforcement.

- **SLA Enforcement and Billing:** The enforcement of the SLAs triggers the submission of user tasks, keeps track of the resource usage, compares it to the SLA, and at the end initiates the billing according to the results of the comparison.

- **Economically Enhanced Resource Management (EERM):** The EERM provides a standardized interface to the resource providing middleware (e.g. *MOSIX* or *Globus Toolkit*). Comparable with a wrapper, the EERM isolates its clients from middleware specific issues and enhances and complements its functionality with market relevant features. The EERM's main duties include:

  – Resource fabric management: Standardized interfaces to create instances of resources and later make use of them from the application.

  – Resource management: The resource management aims at the satisfaction of the service level agreement. It also notifies users and providers of variations and additionally coordinates independent resources to allow for co-allocation in case this is not directly provided by the fabrics.

  – Resource monitoring: The resource monitoring subcomponent monitors the state of the resources in terms of its technical parameters and reports them to the SLA enforcement.

- **Payment:** Once the SLA enforcement and billing component has determined the degree of SLA fulfilment and associated payments, the payment component is invoked. The payment component offers a unified interface to commercial payment services such as *PayPal*[27].

[27]`http://www.paypal.com/`

2.3.4 Layer 4: Core Market Services

Standard middleware does not provide all the infrastructure services necessary for the Resource Market. Layer 4 thus extends the middleware by basic infrastructure services:

- **Market Exchange:** All communication among market participants (users, providers and services of the Resource Market layer) is mediated by the market exchange service, which assures that information is routed to the appropriate party in a secure and reliable way.

- **Logging:** All transactions executed on the market must be registered in a secure log for auditing purposes (to avoid the repudiation of contracts, for instance).

- **Market Information:** The market information service allows participants to gather information (e.g. prices, resource usage levels) from the market. Participants can query the market information service about the current and historical state of the market or subscribe to the service in order to be notified about certain events in the market (Brunner *et al.*, 2008). For example, a user might want to subscribe to the service in order to be notified once the price for a certain resource drops below a certain level.

- **Security Management:** The security management component serves as entry point for a single sign-on mechanism and is responsible for an identity management for the users, the providers and the constituent components of the market.

The conceptual architecture is sufficiently generic to be applicable to both cluster and grid settings. However, some components can be simplified for cluster systems. For example, in grid settings, part of the EERM component's task is to abstract from the underlying middleware and it thus has to provide interfaces to all prominent grid middlewares. If the conceptual architecture is applied to a cluster setting, only one interface is necessary. Besides the EERM component, especially the Core Market Services on Layer 4 can be simplified or even substituted by the cluster middleware's information system, logging, or security components. Moreover, there only needs to be one instance of the provider's intelligent tools.

The implementation of the conceptual architecture sketched above raises several technical challenges that have to be addressed to allow for the vision of a comprehensive market for computing resources. The challenges that are summarized in Table 2.2 concern the technologies and tools used to implement the market.

Challenge	Description
Composition of applications	Applications that consist of multiple services or components need to be able to dynamically acquire these components on a market. A promising approach to address this issue is the use of semantic technologies – in grid computing usually denoted as the "Semantic Grid" (`http://www.semanticgrid.org/`).
Standards and stability	The rapid development of current grid and Web service standards leads to continuous changes in the technologies and tools that build the basis for a market for computational resources. This lack of stability in underlying systems makes it hard to mature the market components over time.
SLA formulation and enforcement	The service level of a computer resource can be specified by a wide range of metrics. This is exacerbated for computer resoueces and services, where not only non-functional but also functional aspects become crucial. Obviously, it is very complex to negotiate SLAs with such a large number of dimensions and to enforce them during execution. Thus, the relevant parameters have to be identified to allow for an appropriate expressiveness of SLAs and to restrict the complexity of their technical management at the same time.
Economic awareness of resources	Current resources and middleware components are not aware that they are situated in an economic environment, for example they do not know that they have a certain price or that their malfunction implies financial compensation. Thus, resources need extensions to cope with their economic nature and especially to inform clients of their economic as well as technical state.
Transparency	For a good acceptance of markets – besides economic factors – it is also necessary that the developed technology can be used as transparently as possible. The required changes in the existing systems must not be too invasive and the users must not be bothered with too many additional time-consuming tasks. While the users' tasks – at least to some extent – can be supported by intelligent agents and wizards, virtualization seems to be a promising remedy against the technical complexity arising from heterogeneous physical resources.

Table 2.2: Technical challenges for distributed computing markets.

2.4 Economic Challenges

2.4.1 A Classification of the Trading Objects

This work centers around the design of the economic logic that is encapsulated within the trading management component. From Market Engineering it is known that the design of markets ultimately depends on the characteristics of the objects that are being traded via the market (Neumann, 2004). For distributed computing, three types of trading objects can be distinguished based on the level of functionality, the possibility to efficiently and compactly specify and describe the object, and the mode of deployment, i.e. how the EERM processes the user requests / applications:

- *Physical resources* can be CPUs, memory, sensors, other hardware or even aggregated resources such as clusters (e.g. a *Condor* or a *MOSIX* cluster). From a technical point of view, resources are simple to describe as there exists only a finite and established set of attributes. For instance, a resource may be defined by the operating system (e.g. Linux operating system), the number of CPUs (e.g. 4 * x86 CPU), memory (e.g. 128 MB RAM) etc. The *Grid Laboratory Uniform Environment* (*GLUE*) schema[28] and the *Job Submission Description Language* (*JSDL*)[29] provide standardized vocabularies for describing computing elements. The standardization of resources offers an easy way to uniquely describe them. This in turn alleviates resource discovery as the technical matchmaking is straightforward.

- *Raw application services* (or simply *raw services* in the remainder) are resource-near services that access resources via standardized interfaces. Examples comprise computing and storage Web services such as the *Amazon Elastic Compute Cloud* (*EC2*)[30] and *Amazon Simple Storage* (*S3*)[31]. Raw services also comprise simple application services that can be described by means of expressive description languages such as the *Web Service Description Language* (*WSDL*)[32], which describes the information required to invoke a Web service. Other developments such as the *Web Ontology Language for Web Services* (*OWL-S*)[33] and the *Web Service Modeling Ontology* (*WSMO*)[34] allow further semantic

[28]`http://forge.ogf.org/sf/projects/glue-wg`

[29]`http://forge.ogf.org/sf/projects/jsdl-wg/`

[30]See `http://s3.amazonaws.com/ec2-downloads/ec2.wsdl` for the WSDL of this service.

[31]See `http://s3.amazonaws.com/doc/2006-03-01/AmazonS3.wsdl` for the WSDL of this service.

[32]`http://www.w3.org/TR/wsdl20/`

[33]`http://www.w3.org/Submission/OWL-S/`

[34]`http://www.wsmo.org/`

descriptions of the services, such as response time, the costs of a service invocation, or security issues.

- *Complex application services*[35] (or simply *complex services*) are so diverse that they cannot reasonably be standardized, e.g. services that perform complex data consolidation and aggregation or that (partially) implement a business process. *Force.com* is a sample platform that hosts a wide range of such complex services, e.g. for data cleansing and data integration. Complex services can theoretically be modeled by using semantic languages such as *OWL-S* and *WSMO*. However, the characteristics of these services are so manifold and complex that service requests and offers cannot be described and matched by means of a reasonable (with respect to size and computationally complexity) set of attributes (Blau and Schnizler, 2008).

See Blau and Schnizler (2008) for a detailed discussion of description languages. Figure 2.3 illustrates these different aggregation levels of services and resources (similar to the Grid Protocol Architecture in Foster *et al.* (2001) and Joseph *et al.* (2004)). On top of this stack, the actual application is located, such as a demand forecasting tool for supply chain management or computer-aided engineering tools for simulations. While these applications can be run directly on physical resources, they may access several complex services, e.g., services for integrating, aggregating and statistically analyzing vast amounts of data, to simplify the development and deployment by shielding parts of the computing system's complexity from the application. These complex services might in turn access raw services that provide standardized interfaces for accessing various data sources and or computational services. When relying on the direct access to physical resources, executables and external libraries need to be transferred across the computing system.

Designing one market for all kinds of computing systems and resources, from physical resources to sophisticated virtual resources or services, seems inappropriate due to both technical and economic factors:

- **Technical factors**: From the technical perspective, differences in the monitoring and deployment of services and resources exist, such that it is very difficult to devise a generic system for the EERM and the trading management that allows the trading of all kinds of resources and services. Physical resources – either accessed directly or as raw services via standardized interfaces – are *application-independent* in the sense that the application is fully transferred to and deployed on the physical resources (given that these resources

[35]Note that here the term "service" is used in a wider sense than Web services. In this work, a service is a software application that is hosted on a distributed computing system. It does not necessarily have to be invoked by means of standardized Web service technologies.

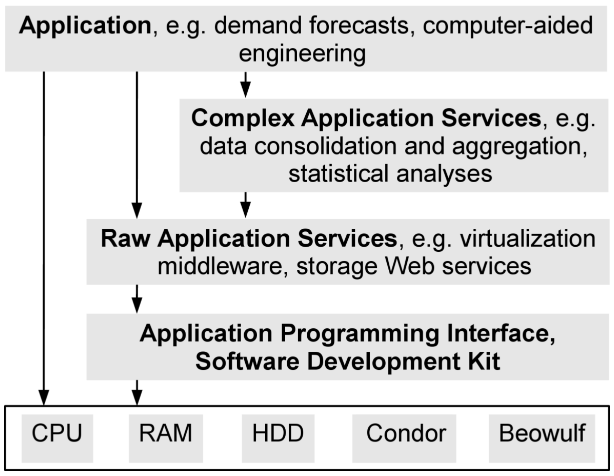

Application, e.g. demand forecasts, computer-aided engineering
Complex Application Services, e.g. data consolidation and aggregation, statistical analyses
Raw Application Services, e.g. virtualization middleware, storage Web services
Application Programming Interface, Software Development Kit
CPU
RAM
HDD
Condor
Beowulf

Figure 2.3: The different aggregation levels of services and resources.

match the application's technical requirements), but the resources are somewhat generic and can be used for a wide range of applications. Complex services on the other hand offer more specific functionality and can only be used by a limited number of applications. The interface (input and output, e.g. data format and accuracy) of the services needs to exactly match the needs of the calling applications and vice-versa, and complex services are therefore *application-dependent* or *application-driven*.

- **Economic factors**: The alternative ways of resource and service description, deployment and dependencies give rise to distinctively different requirements for potential markets. From an economic perspective, markets for physical resources and raw services are promising for automation via an organized electronic market. There are standardized items for sale that potentially attract many users and providers. Complex services have a disadvantage as demand and supply are highly specialized and distributed across niche markets such that only a relatively small number of potential users and providers is interested in the same or a related service. While physical resources and raw services are more or less a commodity for which auction mechanisms are expected to work well, complex services are inherently non-standardized making auction-like mechanisms hard to apply. From the economic perspective, market mechanisms need to achieve at least a subset of the standard objectives in mechanism design listed in Table 2.3 (see e.g. Mas-Colell *et al.* (1995); Parkes (2001); Nisan *et al.* (2007) and the latter chapters of this work for formal definitions).[36]

[36]It is important to note that there are well-known impossibility results regarding certain combinations of these objectives, cf. e.g. Parkes (2001). The objectives and some of these impossibility results will be formally introduced in the remainder of this work.

Objective	Description
Allocative efficiency	Allocative efficiency is typically the overall goal of market mechanisms for resource allocation. A mechanism is said to be allocatively efficient if it maximizes the utility across all participating users and providers (welfare or overall "happiness").
Budget-balance	Budget-balance is essentially one out of two feasibility constraints. A mechanism is budget-balanced if it does not need to be subsidized by outside payments. The payments coming from the users cover the payments made to the resource providers.
Individual rationality	Individual rationality is the second feasibility constraint. A mechanism is individually rational if users cannot suffer any loss in utility from participating in the mechanism, i.e. it is individually rational to participate.
Computational tractability	Due to the potentially large number of resource and service requests and offers, the complexity of the underlying allocation problem and the need for immediacy of the allocation decision, the mechanisms need to be solvable in polynomial runtime in the size of the input, i.e. the number of resource requests and offers.
Truthfulness	Truthfulness means that it is a (weakly) dominant strategy for users to reveal their true valuations to the mechanism. Truthfulness is a desirable feature since it tremendously simplifies the strategy space of the users; there is no need to reason about the bidding strategies (even the strategies of the other market participants), thereby essentially reducing the bidding problem to a preference elicitation problem.

Table 2.3: Economic objectives.

From this brief discussion, it can be seen that a one-size-fits-all market for computing systems is not feasible from both the technical as well as from the economic perspective.

2.4.2 A Two-Tiered Market Structure

Due to the heterogeneous properties of the trading objects, markets for computational resources and services can be divided into two different types of markets (cf. Eymann *et al.* (2006) and Schnizler (2007)): *resource-near markets* for physical resources and raw services on the one hand, and *markets for complex services* on the other hand, spanning out a "two-tiered market structure" as depicted in Figure 2.4.[37]

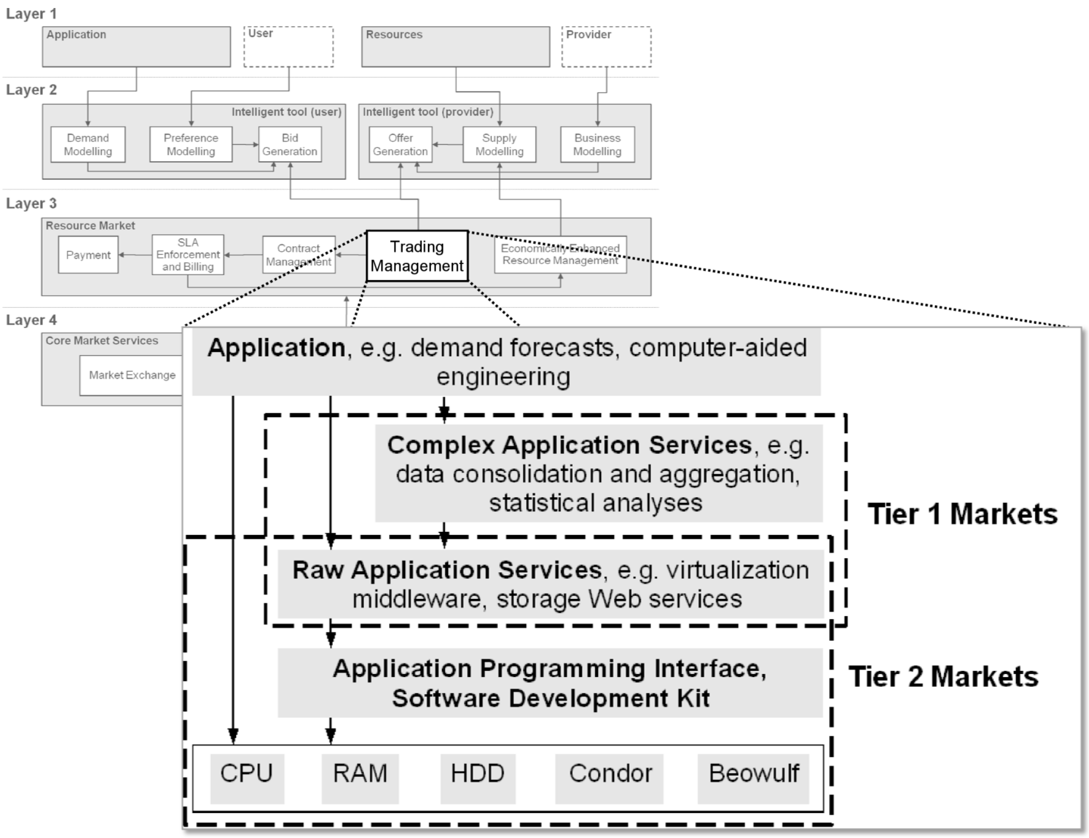

Layer 1
Application
User
Resources
Provider
Layer 2
Intelligent tool (user)
Demand Modelling
Preference Modelling
Bid Generation
Intelligent tool (provider)
Offer Generation
Supply Modelling
Business Modelling
Layer 3
Resource Market
Payment
SLA Enforcement and Billing
Contract Management
Trading Management
Economically Enhanced Resource Management
Layer 4
Core Market Services
Market Exchange
Application, e.g. demand forecasts, computer-aided engineering
Complex Application Services, e.g. data consolidation and aggregation, statistical analyses
Tier 1 Markets
Raw Application Services, e.g. virtualization middleware, storage Web services
Tier 2 Markets
Application Programming Interface, Software Development Kit
CPU
RAM
HDD
Condor
Beowulf

Figure 2.4: Two-tiered market structure.

In Tier 1 markets, applications demand the execution of their constituting complex services. Along the lines of the two-tiered market structure, complex services can be decomposed into smaller raw services that can in turn be translated into resources that are necessary for executing them. E.g., an application might require a complex service for cleansing and aggregating data.

[37]It should be noted that there could be multiple intermediate markets. For the sake of presentation only the extreme cases are considered.

The necessary computing power and the relevant data are available as raw services, which in turn need processing power, memory, storage, etc. Users in such a market request a complex service and the provider of this service, the service integrator, is responsible for obtaining the required raw services and physical resources in turn on the resource-near Tier 2 market, thus hiding parts of the complexity from the user. Such a hierarchical shading of complexity seems to be an appropriate approach since service users typically have no information about how much resources the complex service will consume (Eymann *et al.*, 2006).

The same conceptual architecture introduced above can be used for both types of markets, however with different instances of the components. In the forefront, the trading management component needs to implement a different market mechanism that supports the respective market. In the following, those two classes of markets are discussed in terms of their requirements on the market mechanisms.

Tier 1 – Markets for Complex Services

The requirements on market mechanisms for complex services are different than for resource-near markets. Complex services are used by a relatively small number of users – installing competition is hard. Here the difficulty rather stems from the matching, i.e. from having to find a counterpart that is exactly offering the capabilities to execute the application. The market mechanism is more search-oriented such as bilateral or multi-lateral negotiation protocols, giving rise to the following requirements:

- *Multi-attributes* (Schnizler *et al.*, 2008): Complex services have quality attributes defining the functional and non-functional characteristics of the service. Thus the mechanism needs to cope with multi-attributes.
- *Workflow support* (Emmerich *et al.*, 2005): To support complex applications and services, raw services and distributed resources such as computational devices, data, and applications need to be orchestrated while managing the application workflow. The market mechanism needs to account for this during design time and runtime of the workflow. This imposes extreme difficulties on the market mechanism, as the operations need to be performed in the defined manner, thus opening up lots of exposure risks: if one single task in the workflow fails, the complex service cannot be orchestrated. Thus the market mechanism needs to account for this problem in a rather quick manner.
- *Co-allocation* (Czajkowski *et al.*, 1999; Schnizler *et al.*, 2008): Capacity-demanding applications usually require the simultaneous allocation of several service instances from different providers. For example, a large-scale simulation may require several computa-

tion services to be completed at one time. This situation with the simultaneous allocation of multiple homogenous services is called co-allocation. A mechanism for the service market has to enable co-allocations and provide functionality to control it.

Since the focus of this work is on resource-near markets, a further discussion of markets for complex services is omitted at this point. Chapter 7 will discuss some of the characteristics of these markets and questions for future research.

Tier 2 – Resource-Near Markets

Similar to complex services, physical resources and raw services have quality attributes such as the speed of the CPU, the operating platform or bandwidth. Thus the mechanism needs to cope with *multi-attributes*. Other than that, the matching of physical resources and raw services is simpler than with markets for complex services, in particular as workflows and co-allocation usually do not need to be supported in the market mechanism as such, but must be implemented in the end-users' applications or the complex services.

Grids and clusters focus on the allocation of physical resources and are thus located on this market tier. For grids, which are inter-organizational and distributed by definition, markets are an almost straightforward resource allocation scheme to consider. From its definition, grid computing theoretically also comprises markets for complex services. Currently, however, grids are mostly used for the allocation of physical resources and raw services. But markets have been investigated for cluster-like settings as well in the 1960's already (cf. Sutherland (1968)).[38] There are three arguments for this:

- Markets can help in prioritizing not only external users, but also users within organizations.
- Markets can help in determining appropriate prices for billing resource access.
- Recent advances led to utility computing scenarios where also (or even predominantly) external users are serviced via cluster systems (see the discussion above).

Dividing the market structure into two tiers only is overly simplistic though. Resource-near markets need to be specifically tailored towards the type of the distributed computing setting (cluster vs. grid computing) and the users' application. The market mechanism needs to make full use of the underlying system's capabilities such as virtualization and preemptive migration and the information that is available about the users' applications, such as the timing of demand,

[38]Sutherland (1968) considers the use of simple auction protocols for allocating access to mainframe computers.

Criteria / Research Question	RQ 1 & RQ 2	RQ 3	RQ 4
Sharing approach	inter-organizational	intra-organizational / utility computing	intra-organizational / utility computing
Control over resources	decentralized	(de-)centralized	centralized
Type of resources	heterogeneous	homogeneous	homogeneous
Mode of allocation	shared	dedicated	shared
Use of virtualization	optional	optional	essential
Support for preemptive migration	essential	only preemption required	essential
Type of applications	batch	batch / interactive	batch / interactive
Advance reservation	essential	not used	not used
Most suitable concept	Grid	Cluster	Cluster

Table 2.4: Mapping of the scenarios considered in the research questions to the distributed computing concepts.

such as release dates (i.e., the time when the computational job is submitted to the system) and runtimes. This will be discussed in the following subsection in the light of the research questions that will be addressed in this work.

2.4.3 Scheduling in Resource-Near Markets

As introduced in Chapter 1, this work considers four specific scenarios. All of these scenarios are located on the tier for resource-near markets. This subsection further distinguishes market mechanisms for such resource-near scenarios based on the specific scenarios of this work and the characteristics of distributed computing systems introduced above (cf. Table 2.4).

Research questions one and two consider pure grid settings where multiple resource providers offer resources to multiple users. The resources can be allocated in shared mode. For instance, if a compute node offers four CPUs, it can be allocated up to four computational jobs. Virtualization is not assumed in this scenario but would only reduce the heterogeneity of resources. Advance reservation, however, is essential in this scenario. If a user's job is allocated, the user is guaranteed to have access to the requested amount of resources for the specified timeframe. The key assumption in this scenario is that users want to execute batch applications for which they have an (estimated) release date and runtime. From a scheduling point of view, the advantage when handling batch applications is that the scheduler has almost complete information about the relevant input and does not have to deal with an uncertain environment. Given honest reports of the users, the scheduler can thus determine the optimal allocation. This setting is commonly called "offline scheduling" (Arndt *et al.*, 2000). Offline scheduling, however, is computationally demanding. Since the scheduler has to consider the available information about resource consumption and supply, the timing of this demand and supply, and possibly further parameters, the scheduling problem easily becomes a combinatorial allocation problem (cf. Martello and Toth (1990) and Chekuri and Khanna (2006)). Finding the optimal solution hence becomes infeasible in practice for many settings, especially when large numbers of users and providers are involved. Consequently, heuristic mechanisms are employed that, however, can generally not guarantee a "close" bound with respect to the deviation from the optimal solution if based on a *deterministic* allocation algorithm. In order to intercept worst case scenarios of deterministic heuristic mechanisms, *randomizing* the algorithm might offer a loophole (e.g. Nisan and Ronen (2001)). These two alternative scheduling approaches distinguish research question one and research question two.

Research questions three and four consider settings that are closest to cluster and utility computing scenarios; the resources are homogeneous and need to be efficiently allocated to internal users and / or external users. Moreover, the focus is rather on interactive applications for which the release date is not known a priori. Schedulers in such scenarios are so-called "online schedulers". An online scheduling algorithm denotes an algorithm that has to make allocation decisions without complete information about the future input (Borodin and El-Yaniv, 1998; Arndt *et al.*, 2000). For the purpose of this work, online schedulers can be further distinguished by the mode of allocation, i.e. whether the scheduler assigns *dedicated resources* as in research question three or *shares of resources* as in research question four. Note that the latter denotes a much more fine-grained sharing as considered in the grid setting, where a compute node is shared on the level of CPUs and / or units of memory. In contrast, research question four considers settings in which the resource is (almost) perfectly divisible. Such settings have thus far mainly be considered for allocating divisible resources such as bandwidth (e.g. Sanghavi and Hajek (2004)). However, with the advent of virtualization techniques in mainstream computing and system management, it has become possible to treat whole computers as divisible resources (Barham *et al.*, 2003).

In summary, in the light of the research questions to be tackled in this work, market mechanisms for resource-near markets can first be distinguished by the assumptions the mechanism makes on the computational jobs of users, i.e. whether the mechanism has a priori knowledge of these jobs' release dates (offline scheduling) or whether these dates are unknown until the jobs actually enter the system (online scheduling). Additionally, in the course of this work the focus changes from the properties of the allocation algorithm (determinism vs. the randomization) to the properties of the resources (dedicated vs. shared resources). These distinctions are illustrated in Figure 2.5.

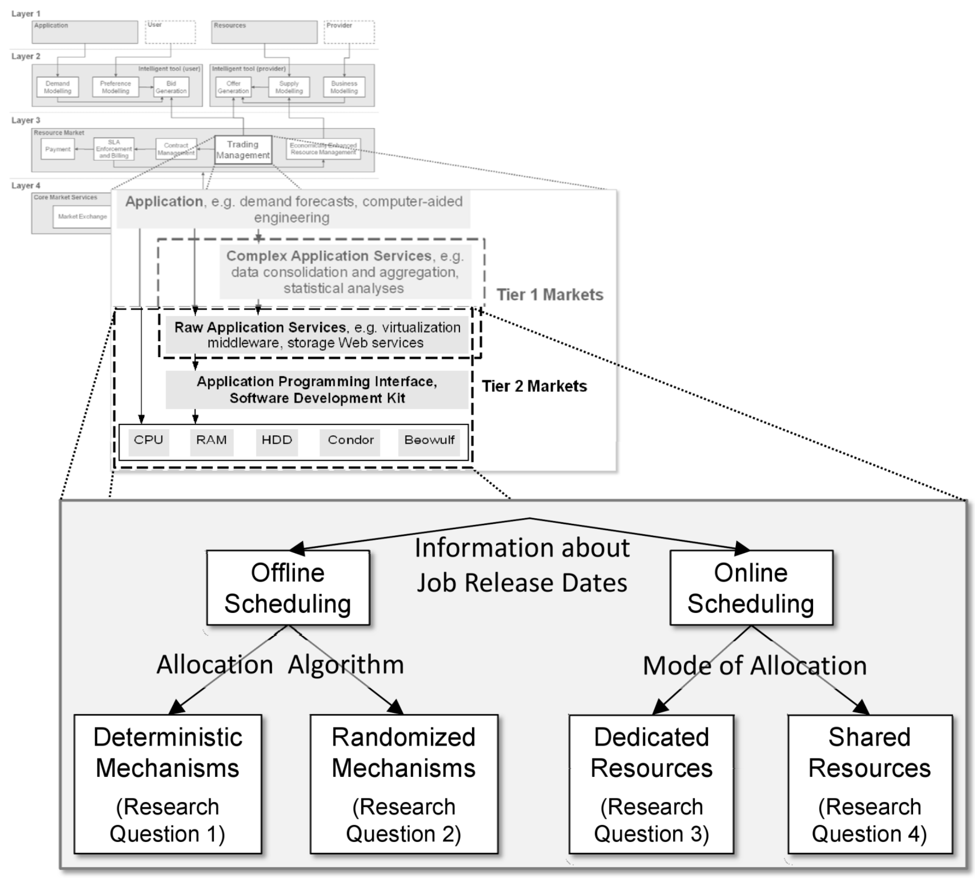

Layer 1
Application
User
Resources
Provider
Layer 2
Intelligent tool (user)
Intelligent tool (provider)
Demand Modelling
Preference Modelling
Bid Generation
Offer Generation
Supply Modelling
Business Modelling
Layer 3
Resource Market
Payment
SLA Enforcement and Billing
Contract Management
Trading Management
Economically Enhanced Resource Management
Layer 4
Core Market Services
Market Exchange
Application, e.g. demand forecasts, computer-aided engineering
Complex Application Services, e.g. data consolidation and aggregation, statistical analyses
Tier 1 Markets
Raw Application Services, e.g. virtualization middleware, storage Web services
Application Programming Interface, Software Development Kit
Tier 2 Markets
CPU
RAM
HDD
Condor
Beowulf
Information about Job Release Dates
Offline Scheduling
Online Scheduling
Allocation Algorithm
Mode of Allocation
Deterministic Mechanisms (Research Question 1)
Randomized Mechanisms (Research Question 2)
Dedicated Resources (Research Question 3)
Shared Resources (Research Question 4)

Figure 2.5: Taxonomy of resource-near (Tier 2) markets with respect to the research questions of this work.

2.5 Methodologies

In trying to answer the research questions presented in Chapter 1, two components of a market must be modeled and analyzed: *strategic behavior* from users and providers and the *performance of the allocation mechanism* (Smith, 1982). These two components are closely interconnected and mutually impact each other. Depending on the incentives set by the market mechanism, the users do not necessarily report their true characteristics to the mechanism but might benefit from misreporting. Thus, to be able to analyze the "realistic" performance of the mechanism (with respect to a subset of possible economic performance criteria, cf. Table 2.3), one must be able to describe "realistic" user behavior first. Vice versa, the user behavior inherently depends on the properties of the mechanism.

This work does not primarily aim at analyzing existing market (mechanisms), but at designing novel market mechanisms. As such, as pointed out by Roth (2002), an "engineering approach" is required. This work heavily relies on three methods to modeling and analyzing user behavior and market performance: theoretical analyses, empirical studies, and simulations. These approaches are briefly presented in the remainder of this section.

2.5.1 Theoretical Analyses

Strategic behavior from users and providers is studied by employing the concepts and methods of *game theory*. There exists a wide array of solution concepts that are used to model market participants, such as dominant strategies and Nash equilibria. Game theory involves making explicit assumptions about users and providers, e.g. with respect to utility functions and the information about the strategies of other market participants (Mas-Colell *et al.*, 1995).

The analytic evaluation of the performance of an allocation algorithm is oftentimes used to derive general impossibility and / or possibility results, e.g. the worst case behavior of an algorithm. For instance, in scheduling theory, *competitive analysis* is a common tool. It typically involves finding a bound that limits a specific online scheduling algorithm's deviation from the optimal offline mechanism (Borodin and El-Yaniv, 1998). Competitive analysis has also recently been applied to study the worst case performance of market mechanisms (Goldberg *et al.*, 2006; Heydenreich *et al.*, 2006).

Theoretic analysis provides the strongest results. On its downside, it typically relies on strong assumptions about strategic behavior and further parameters, which do not necessarily reflect realistic inputs.

2.5.2 Empirical Studies

While theory might produce possibility and impossibility results and give general insights, it is oftentimes more important and meaningful to analyze the "average" or "realistic" case by means of real-world data.

Empirical studies have been widely used in computer science to analyze computing systems (e.g. Harchol-Balter and Downey (1997)). Workload models of *clusters* have been derived from the characteristics of real-world workload traces to be able to generate accurate random input (cf. Feitelson (2008)). With the emergence of electronic markets and thus the comparably easy availability of real-world data, empirical studies have also become a common tool in the study of markets (e.g. Roth and Ockenfels (2002)).

With respect to the study of *grid* markets, empirical work is currently hard to perform since there is not much data available about grid workloads, for instance. Moreover, there is currently no grid market in operation that would generate data about user valuations.

2.5.3 Simulations

To this end, *numerical simulations* provide a useful tool to analyze markets by means of randomly generated workloads and / or user valuations (Böttcher *et al.*, 1999). Numerical simulations can be used to enrich purely technical workloads with user valuations up to generating all necessary attributes. While the former is closest to reality and thus relevance, the latter oftentimes helps in gaining insights into the general problem structure, i.e. the dependencies between market performance and the input parameters, such as the complexity of the allocation problem or the market's vulnerability to strategic misreporting dependent on the number of users.

In addition to numerical simulations, *agent-based simulations* are a useful tool for analyzing strategic behavior from users (Bonabeau, 2002). User behavior is simulated by means of collections of software agents. First, agent-based simulations allow to analyze complex settings with large numbers of users that preclude theoretical analyses. Moreover, agent-based simulations allow the analysis of strategies other than "traditional" game-theoretic constructs. On its downside, it is a crucial and complex task to model and incorporate strategic behavior into the software agents. However, there is an increasing body of research in this field and a whole set of strategies has been shown to work well in many settings (Phelps, 2007).

Chapter 3

Outcome Determination in Large-Scale Grid Settings

3.1 Introduction

With batch applications users oftentimes have information about the expected release dates and runtimes of jobs. State-of-the-art cluster schedulers, such as the *Maui* cluster scheduler[1], try to utilize this information and support the advance reservation of resources. With advance reservations, the access to a resource can be restricted to a certain computational job for a specific timeframe. Intuitively, this allows to manage resource allocations in a more efficient manner from both a technical and economic point of view, e.g. by avoiding "gaps" in the allocation schedule. As discussed in the previous chapter, in the near future grid middlewares can be expected to make use of the underlying clusters' capability to support advance reservation on the grid level as well.

While advance reservation promises increased efficiency, it also increases the computational complexity of the scheduling problem. Due to the additional information about jobs and resources, it is complex for the market mechanism to identify the feasible allocation schedules and try to find the optimal (in a sense that will be discussed below) schedule among these feasible schedules. This is exacerbated in settings with offline scheduling where the scheduler collects resource requests and offers for a period of time before making its allocation decisions. Another layer of complexity is added by the large-scale nature that is inherent to grids. Large grids combine the resources from many providers of heterogeneous resources. Moreover, the set of users and providers is highly dynamic, as grid users can become providers over time and vice versa.

[1] `http://www.clusterresources.com/products/maui/`

Hence, due to the large amount of available information and the large scale of many grid settings, the market mechanism needs to be scalable but still be able to find efficient allocation schedules. However, in finding these efficient schedules, the market mechanism inherently relies on the users and providers to truthfully report their private information about jobs and resources. Otherwise the mechanism tries to optimize over false input and will generally not be able to find the truly efficient allocation schedule.

Current market mechanisms, however, are not adequately able to contend with dependencies between multiple grid resources (e.g. the simultaneous need for computing power and memory) in large-scale settings with strategic users who try to benefit from manipulating the mechanism. First, this chapter presents a greedy, deterministic heuristic that can be applied to solve such large-scale scenarios for batch applications. The strategic dimension of scheduling in grids is addressed by presenting two pricing schemes. The first, proportional critical-value pricing, is a novel pricing scheme that successfully limits strategic behavior on both market sides on the expense of computational cost. The second pricing scheme, k-pricing, is highly scalable on the expense of strategic properties. It has been proposed and evaluated in previous research, however for a different allocation rule and setting (Schnizler *et al.*, 2008). The structure of the allocation problem, the efficiency of the heuristic, and the pricing schemes are evaluated analytically and by means of numerical simulations.

3.2 Requirements and Design Desiderata

There is a number of requirements, which a market mechanism must satisfy in order to be applicable to a grid environment for batch applications:

- **Double-sided market**: The mechanism must support the trading between multiple strategic users and multiple strategic resource providers that issue job requests and node offers. This requirement is inherent to grids, which consist by definition of resources that are under decentralized control.
- **Computational tractability**: The mechanism must compute allocations and prices in polynomial runtime in the size of its input, i.e. the number of job requests and node offers. For example, the Sun N1 Grid Engine, a state-of-the-art grid scheduler by Sun Microsystems, allocates every 15 seconds. The mechanism should therefore be able to compute the outcome for at least several hundred job requests and node offers within this timeframe.
- **Bundling**: Users must be able to express interdependencies between multiple resources and / or multiple job requests and node offers (Nisan, 2006). Supporting interdependen-

cies between multiple resources is important because a job requires computing power and memory on one node at the same time, for instance. Consequently, users must be able to specify technical constraints of their jobs that limit the feasible set of compute nodes to which these jobs can be allocated.

- **Time constraints**: In the batch setting considered in this chapter, users and providers are assumed to have a priori knowledge about the starting times and runtimes of jobs and the availability of nodes. For example, a provider might only be able or willing to temporarily contribute a node to the grid for a fixed and known period of time. The mechanism must therefore allow users and providers to express time constraints and consider these when making its allocation decisions.

The mechanism is intended to perform job scheduling in a distributed computing environment with heterogeneous and selfish users. This gives rise to a number of economic design desiderata (Mas-Colell *et al.*, 1995; Parkes, 2001; Nisan *et al.*, 2007) that a mechanism should ideally satisfy but that might be relaxed to a certain extent. In the following the focus will be on so-called *direct revelation mechanisms* where the users' and providers' strategies only consist of reporting their economic and technical attributes, i.e. their "types", to the market-based scheduler (Mas-Colell *et al.*, 1995). Let θ_i be the type of user or provider i and θ_{-i} the type of all other users and providers, where $\theta = (\theta_i, \theta_{-i})$. Then the following economic requirements can be stated:

Definition 1 (Allocative efficiency). *A mechanism is said to be* allocatively efficient *if, based on its input θ, it always determines the outcome o that maximizes the utility across all participating users and providers (i.e., welfare):*

$$\sum_{i \in J \cup N} u_i(o, \theta | \theta_i) \geq \sum_{i \in J \cup N} u_i(o', \theta | \theta_i),\ o' \in O$$

where $u_i(o, \theta | \theta_i)$ is the utility of user or provider i that she derives from the market outcome $o \in O$, O is the set of possible outcomes, J is the set of jobs, and N the set of nodes.

Definition 2 (Budget-balance). *A mechanism is said to be* weakly budget-balanced *if its payment scheme does not need to be subsidized (ex post) by outside payments. The payments coming from the users cover the payments made to the resource providers:*

$$\sum_{j \in J} p_j(\theta) - \sum_{n \in N} p_n(\theta) \geq 0$$

where p_j (p_n) denotes the payment of job j's user (node n's provider).

Definition 3 (Individual rationality). *A mechanism is said to provide the property of* individual rationality *(also called* voluntary participation*) if users and providers cannot suffer any loss in utility from participating in the mechanism:*

$$u_i(o,\theta|\theta_i) \geq \overline{u}_i(o,\theta|\theta_i),\ i \in J \cup N$$

where $\overline{u}_i$ denotes the utility that i could derive by withdrawing from the market.

Definition 4 (Truthfulness). *A mechanism is said to be* truthful *if it is a weakly dominant strategy for market participant i to reveal her true characteristics for arbitrary $\tilde{\theta}_{-i}$:*

$$u_i(o,(\theta_i,\tilde{\theta}_{-i})|\theta_i) \geq u_i(o,(\tilde{\theta}_i,\tilde{\theta}_{-i})|\theta_i),\ i \in J \cup N, \tilde{\theta}_i, \tilde{\theta}_{-i}.$$

Truthfulness is a desirable property since it tremendously simplifies the strategy space of the market participants; there is no need to reason about the strategies of other participants. In case of a periodic mechanism that continues for several rounds, it is assumed that the participants are myopic, i.e. they only consider the current round and do not reason about the future.

Users and providers are assumed to have quasi-linear utility functions. The market outcome o consists of an allocation schedule X and corresponding payments p, i.e. $o(\theta) = (X(\theta), p(\theta))$. Consequently, the user's ex post utility for job j is $u_j(o,\theta|\theta_j) = v_j(X(\theta)) - p_j(\theta)$, where $v_j(X(\theta))$ is the user's *true valuation* for allocation schedule X that the mechanism determined based on its input θ.[2] Analogously, the utility of the provider of node n is $u_n(o,\theta|\theta_n) = p_n(\theta) - v_n(X(\theta))$.

Throughout this chapter, it will be useful to refer to the framework for the design of periodic market mechanisms depicted in Figure 3.1. There are three basic components to be designed in a market mechanism (de Vries and Vohra, 2003; Schnizler *et al.*, 2008): *the bidding language*, which defines how job requests and node offers are specified (and which thus determines the strategy space of the selfish users and providers); *the allocation algorithm*, which decides which job is to be executed on which node at what time; and *the pricing scheme*, which translates the resulting allocation schedule into monetary transfers between the users and providers.

There exist several interdependencies between these components and the presented design desiderata. Budget-balance and individual rationality are hard constraints that the mechanism must satisfy. If these two requirements are not met, the mechanism will not be sustainable; participants will not voluntarily participate in the market if they incur losses and the market operator will not be willing to subsidize the mechanism in the long run. Moreover, there exist strong theoretic results that show that it is in fact impossible to achieve certain combinations of

[2]Note that this notation will be simplified to $u_i(\theta|\theta_i)$ if it is clear from the context that a specific and well-defined mechanism is considered, i.e. when the set of rules that determine o based on input θ has been defined.

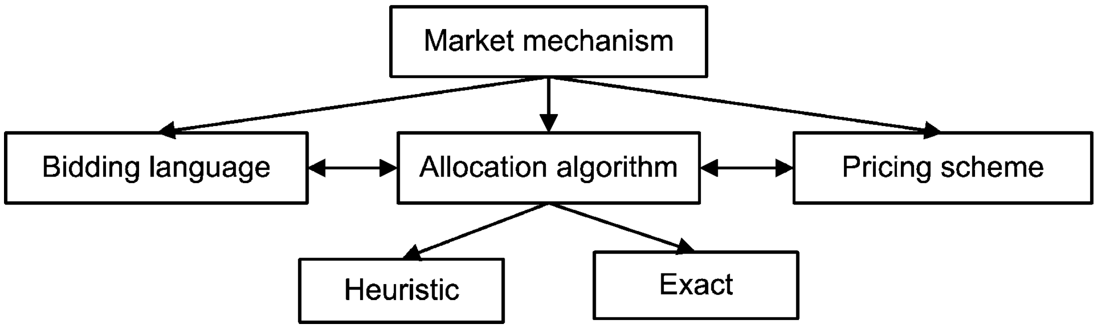


Figure 3.1: Design framework for periodic market mechanisms.

the design desiderata (cf. e.g. Parkes (2001)). Among the most prominent are the following two theorems:

Theorem 1 (Hurwicz Impossibility Theorem[3] (Green and Laffont, 1979; Walker, 1980; Hurwicz and Walker, 1990)). *There is no double-sided mechanism that is at the same time allocatively efficient, budget-balanced, and truthful in settings with quasi-linear preferences.*

Theorem 2 (Myerson-Satterthwaite Impossibility Theorem (Myerson and Satterthwaite, 1983)). *There is no double-sided mechanism that is at the same time allocatively efficient, budget-balanced, Bayesian-Nash incentive compatible, and (interim) individually rational, even in settings with quasi-linear preferences.*

The theorem of Myerson and Satterthwaite (1983) has stronger implications than the Hurwicz Impossibility Theorem, since it extends the latter to the solution concept of Bayesian-Nash incentive compatibility, where the users' / providers' aim is to maximize the expected utility instead of ex post utility. In doing so, an additional assumption of Myerson and Satterthwaite (1983) is that the mechanism must be individually rational.

The Myerson-Satterthwaite Impossibility Theorem implies that at most two desiderata out of allocative efficiency, individual rationality and budget-balance can be achieved when aiming at a truthful mechanism in settings with quasi-linear preferences. This in turn implies that either allocative efficiency or truthfulness must be sacrificed, at least to a certain extent, since budget-balance and individual rationality must be satisfied in the long run for the mechanism to be sustainable.

Moreover, for complex scheduling settings, allocative efficiency and computational tractability conflict because it takes a prohibitively long time to compute the optimal allocation schedule in

[3]This theorem is also oftentimes called Green-Laffont Impossibility Theorem due to Green and Laffont (1979). However, this work follows Parkes (2001) since the theorem is originally based on unpublished work by Hurwicz (1975)).

combinatorial settings. However, sacrificing efficiency generally comes at the expense of truthfulness, although there exist some special cases, one of which will be utilized below (Mu'alem and Nisan, 2008).

Consequently, when configuring the three components, there is an inherent trade-off involved between the various design desiderata. This will also become apparent in the following discussion of related works. The aim of this chapter is to find an "appropriate" trade-off: a mechanism that satisfies the applicability constraints of a double-sided market structure, computational tractability, bundling and time constraints, and that is individually rational, budget-balanced, truthful and at least approximately allocatively efficient. This trade-off will be tested both analytically and numerically.

3.3 Related Work

A conceptual view on the integration of market mechanisms into grid systems is provided by the G-commerce project (Wolski *et al.*, 2001) and GridBus (Buyya *et al.*, 2002). These projects propose and evaluate basic auction mechanisms in the grid context. But as Wellman *et al.* (2001) point out, scheduling problems often exhibit properties that are not captured by such basic auction protocols, e.g. the presence of multiple resource providers and time constraints. The authors point out that a suite of mechanisms is likely to emerge for a broader range of scheduling problems. Along the lines of the domain-specific applicability requirements, research in the market-based allocation of grid resources can best be separated into mechanisms that consider the trading of one type of resource only and mechanisms that account for *dependencies* between multiple grid resources.

Spawn (Waldspurger *et al.*, 1992), market-based Proportional Share (Chun and Culler, 2000; Lai *et al.*, 2005), and the Popcorn market (Regev and Nisan, 2000) allow the specification and trading of one resource only such as CPUs or CPU cycles. But, as discussed above, the users' jobs require a bundle of resources such as computing power, memory, and bandwidth. The approaches at hand thus lead to inefficient allocations since jobs with the same demand for computing power but different memory requirements, for instance, are treated the same. On the other hand, users are exposed to the risk of only being able to obtain one of the required resources in the bundle. Note that Spawn and Popcorn are efficient in the simplified setting with one resource only, but not in the light of the more complex setting considered in this work.

MACE (Multi-Attribute Combinatorial Exchange, Schnizler *et al.* (2008)) and the Bellagio system (AuYoung *et al.*, 2004) target these deficiencies by allowing users to request bundles of computer resources with quality attributes. Neither mechanism yields truthful prices and the scheduling problem in these combinatorial settings is computationally intractable.[4] The mechanisms are thus not applicable to large-scale settings in which users require the timely allocation of resources. The work of Bapna *et al.* (2008) is most relevant to the work presented in this chapter. In their model, multiple users and providers can trade both computing power and memory for a sequence of timeslots. However, the model does not allow for the strategic behavior of resource providers, but it imposes one common reserve price. First, an exact mechanism is introduced.[5] To mitigate the computational complexity of this exact formulation, a fast, greedy heuristic is proposed at the expense of both truthfulness and efficiency.

In summary, as comprised in Table 3.1, the mechanisms proposed for grids are not fully able to account for dependencies between multiple grid resources in large-scale settings with strategic users. In the following section, a scalable heuristic is proposed that allows the trading of both computing power and memory. Additionally, pricing schemes are presented that are designed to induce truthful behavior from strategic users.

3.4 The Mechanism

First, the bidding language will be introduced, which determines how job requests and node offers are expressed. As pointed out above, in order to support the fast allocation of resources, the market is implemented as a centralized mechanism where the only action of users and providers is to declare their economic and technical attributes. The mechanism clears periodically and offline, i.e. it collects job requests and node offers for a period of time in the so-called "order book" before making its allocation and pricing decisions.[6]

3.4.1 Expressing Job Requests and Node Offers

In the model at hand, computing power is the central scarce resource to be traded. Let N be the set of compute node offers, which the mechanism has collected over a period of time. When submitting a node offer $n \in N$, the resource provider reports the type $\theta_n = (v_n, C_n, M_n, R_n, E_n)$ to the system. $v_n \in \mathbb{R}_+$ denotes the provider's valuation (reserve price) *per unit of computing power and time* in some common monetary unit. $C_n \in \mathbb{N}$ and $M_n \in \mathbb{N}$ specify the maximum available amount of computing power (e.g. number of CPUs or CPU cycles) and memory (e.g.

[4]Note, however, that the architecture of the Bellagio system can also be applied to double-sided markets and to heuristic allocation algorithms. Here the exact, single-sided formulation with Parkes's VCG-based threshold pricing is considered (AuYoung *et al.*, 2004; Parkes *et al.*, 2002).

[5]The exact formulation does not solve the allocation problem to optimality due to what Bapna et al. call a "fairness constraint", which limits the mechanism's allocation decisions.

[6]Note that the clearing period can be as small as a couple of seconds.

Mechanism \ Requirement	Applicability Constraints				Economic Design Desiderata			
	Computational tractability	*Double-sided market*	*Bundling*	*Time constraints*	*Allocative efficiency*	*Budget-balance*	*Individual rationality*	*Truthfulness*
Spawn	✓	×	×	●	✓	✓	✓	✓
Proportional Share	✓	×	×	×	×	✓	×	×
Popcorn (Vickrey)	✓	×	×	×	✓	✓	✓	✓
Popcorn (double auctions)	✓	✓	×	×	×	✓	✓	×
MACE	×	✓	✓	✓	×	✓	✓	×
Bellagio	×	×	✓	✓	×	✓	✓	×
Bapna et al. (exact)	×	●	●	✓	×	✓	✓	✓
Bapna et al. (heuristic)	✓	●	●	✓	×	✓	✓	×
This work	✓	✓	●	✓	×	✓	✓	●

Table 3.1: Mapping of the existing mechanisms to the requirements. ✓ satisfied, ● partly satisfied, × not satisfied.

in Megabytes), respectively. $R_n \in \mathbb{N}$ and $E_n \in \mathbb{N}$ indicate the earliest and latest timeslots during which these resources are available. It is assumed that nodes can virtually execute multiple jobs in parallel, e.g. by viewing each node as a virtual machine, which makes this node almost perfectly divisible.

Let J be the set of job requests. Users wanting to submit a computational job $j \in J$ report the type $\theta_j = (v_j, c_j, m_j, r_j, e_j)$ to the system where $v_j \in \mathbb{R}_+$ expresses the user's valuation (maximum willingness to pay) *per unit of computing power and time*, $c_j \in \mathbb{N}$ and $m_j \in \mathbb{N}$ specify the minimum required amount of computing power and memory, respectively, and $r_j \in \mathbb{N}$ and $e_j \in \mathbb{N}$ mark the job's release date and end time. Consequently, the job's runtime (duration) is $d_j = e_j - r_j + 1$ timeslots. A job j can only be executed in its entirety, meaning it is only allocated if there are sufficient resources available in each timeslot $t \in [r_j, e_j]$. A job can only be run on one node at a time but can *migrate* across nodes over time without having to be restarted. Job migration is an important feature that distinguishes the scheduling in computational grids from other machine scheduling domains (see e.g. Harchol-Balter and Downey (1997)).

Example: The sample job requests and node offers listed in Table 3.2 have been submitted to the system. The user of job j_1 is requesting 35 units of computing power and 50 units of memory throughout timeslots 3 to 7. The user is willing to pay up to \$10 per unit of computing power and timeslot, i.e. $v_1 c_1 d_1 = \$1,750$ in total. The provider of node n_1 requires a payment of at least \$8 per unit of computing power and timeslot while she is offering up to 123 units of computing power and 98 units of memory throughout timeslots 1 to 8.

Job j	v_j	c_j	m_j	r_j	e_j	Node n	v_n	C_n	M_n	R_n	E_n
j_1	10	35	50	3	7	n_1	8	123	98	1	8
j_2	14	35	27	1	7	n_2	11	98	82	1	9
j_3	11	84	45	2	7						
j_4	16	71	48	2	7						
j_5	12	63	30	2	7						
j_6	17	55	20	2	7						

Table 3.2: Sample job requests and node offers.

3.4.2 Allocating Jobs to Nodes

Let T be the scheduling horizon for a problem instance, i.e. the set covering all timeslots specified in job requests and node offers. The binary decision variable x is defined as $x_{jnt} = 1$ if job j is allocated to node n in timeslot t and $x_{jnt} = 0$ else. Then the allocation problem, which determines an optimal allocation schedule $X = (x_{jnt})$ that assigns jobs to nodes so as to maximize allocative efficiency (welfare) W, can be formalized as a mathematical integer program:

$$\max_{X=(x_{jnt})} W = \sum_{j \in J} c_j \sum_{n \in N} \sum_{t \in T} x_{jnt}(v_j - v_n). \tag{3.1}$$

Since users and providers are assumed to have quasi-linear utility functions, efficiency is the overall difference between the users' job valuations and the providers' reserve prices (both expressed per CPU and timeslot). Hence, the aim is to allocate the most valuable jobs to the cheapest possible nodes.

Constraint 3.2 introduces the binary decision variable x and ensures that a job can only be allocated to a node that is accessible during the right timeslots and whose reserve price does not exceed the user's willingness to pay:

$$x_{jnt} \begin{cases} \in \{0,1\} & \text{if } r_j \leq t \leq e_j \wedge R_n \leq t \leq E_n \wedge v_n \leq v_j \\ = 0 & \text{else} \end{cases}, j \in J,\ n \in N,\ t \in T. \tag{3.2}$$

Furthermore, a job can only be allocated to a maximum of one node at a time (Constraint 3.3):

$$\sum_{n \in N} x_{jnt} \leq 1,\ j \in J,\ t \in T,\ r_j \leq t \leq e_j. \tag{3.3}$$

Constraints 3.4 and 3.5 specify that the jobs allocated to one node are not allowed to consume more resources than are available on this node:

$$\sum_{j \in J} x_{jnt} c_j \leq C_n,\ n \in N,\ t \in T,\ R_n \leq t \leq E_n, \tag{3.4}$$

$$\sum_{j \in J} x_{jnt} m_j \leq M_n,\ n \in N,\ t \in T,\ R_n \leq t \leq E_n. \tag{3.5}$$

Constraint 3.6 enforces atomicity, i.e. a job is either fully executed in all requested timeslots or it is not executed at all:

$$\sum_{u=r_j}^{e_j} \sum_{n \in N} x_{jnu} = d_j \sum_{n \in N} x_{jnt},\ j \in J,\ t \in T,\ r_j \leq t \leq e_j. \tag{3.6}$$

A compact presentation of this allocation problem is presented in Appendix A. This problem is a special case in between the Multiple Knapsack Problem (MKP) and the Generalized Assignment Problem (GAP).

In the notation of this work, MKP is defined as follows (Martello and Toth, 1990; Chekuri and

Khanna, 2006):

$$\text{[MKP]} \qquad \begin{aligned} \max_X W = & \sum_{j \in J} \sum_{n \in N} x_{jn} v_j \\ \text{subject to } & \sum_{j \in J} x_{jn} c_j \leq C_n, \ n \in N, \\ & \sum_{n \in N} x_{jn} \leq 1, \ j \in J, \\ & x_{jn} \in \{0,1\}, \ j \in J, n \in N, \end{aligned}$$

where v_j is interpreted as the "profit" of assigning job j to node n. It can easily be seen that the allocation problem at hand is more complex than MKP since the profit of allocating a specific job varies with the reserve price of the node it is allocated to. Moreover, the setting is constrained by two resources instead of only one. Most importantly, Constraint 3.6 essentially interconnects a separate allocation problem for each timeslot, where each of these separate allocation problems is already more complex than MKP.

GAP is defined as follows (Martello and Toth, 1990; Chekuri and Khanna, 2006):

$$\text{[GAP]} \qquad \begin{aligned} \max_X W = & \sum_{j \in J} \sum_{n \in N} x_{jn} v_{jn} \\ \text{subject to } & \sum_{j \in J} x_{jn} c_{jn} \leq C_n, \ n \in N, \\ & \sum_{n \in N} x_{jn} \leq 1, \ j \in J, \\ & x_{jn} \in \{0,1\}, \ j \in J, n \in N. \end{aligned}$$

It is not obvious if the allocation problem is less or more complex than GAP. On the one hand, GAP is more comprehensive as it allows for different capacity requirements of a job depending on the node it is allocated to. On the other hand, as MKP, it only supports one restricting resource and does not consider timeslots.

Since MKP is known to be NP-complete (Chekuri and Khanna, 2006) but still simpler than the allocation problem at hand, from a computational viewpoint, solving the model to optimality is clearly also NP-complete. Consequently, solving this combinatorial allocation problem to optimality is computationally intractable in practice. Users typically require the resources in a timely manner and the overhead for determining the allocations must be kept as low as possible.

This computational hardness forms the rationale for an allocation scheme based on a greedy heuristic whose basic form was originally proposed by Lehmann *et al.* (2002) and Mu'alem and Nisan (2008) for clearing single-sided single-attribute combinatorial auctions. This heuristic is extended to the setting of double-sided multi-attribute auctions. The greedy allocation scheme is specified in Algorithm 1.

Algorithm 1 Deterministic Allocation Heuristic.

```
Require: Set $J$ containing all job requests, sorted in non-increasing order of $v_j$.
Require: Set $N$ containing all node offers, sorted in non-decreasing order of $v_n$.
Ensure: Feasible allocation schedule $X = (x_{jnt})$.
 1: $X = (x_{jnt}) = O$, $X' = (x'_{jnt}) = O$ {Initialize allocation schedule $X$ and temporary schedule $X'$ to be zero matrices}
 2: for all $j \in J$ do
 3:   $X' = (x'_{jnt}) = O$ {Reset $X'$}
 4:   for all $t \in [r_j, e_j]$ do
 5:     for all $n \in N$ do
 6:       if $(R_n \le t \le E_n)$ && $(c_j \le C_n(t))$ && $(m_j \le M_n(t))$ && $(v_n \le v_j)$ then
 7:         $x'_{jnt} = 1$, break {Job $j$ can be allocated to node $n$ in timeslot $t$}
 8:   if $\sum_{t=r_j}^{e_j} \sum_{n \in N} x'_{jnt} == d_j$ then
 9:     $X = X + X'$ {Add the job to the allocation schedule if the job can be accommodated in all requested timeslots}
10:     for all $t \in [r_j, e_j]$ do
11:       for all $n \in N$ do
12:         if $x'_{jnt} == 1$ then
13:           $C_n(t) -= c_j$, $M_n(t) -= m_j$ {Update the node's remaining capacity in timeslot $t$}
14: return $X$
```

It essentially consists of two phases:

- **Step 1 (sorting phase)**: Sort jobs $j \in J$ in non-increasing order of their willingness to pay v_j, and nodes $n \in N$ in non-decreasing order of their reserve prices v_n.
- **Step 2 (allocation phase)**: Run sequentially through the job ranking, starting with the highest-ranked job. For each job j, try to allocate this job to the cheapest feasible node that still has idle capacity left.

The basic idea behind this heuristic is that, by sorting the job requests and node offers with respect to their reported valuation per unit of computing power and time, to try to greedily maximize the difference $v_j - v_n$ in the objective function (Equation 3.1) of the exact formalization of the allocation problem for each job assignment.

Example: Applying the greedy allocation scheme to the example above generates the allocation schedule depicted in Table 3.3 with welfare of $W^{greedy} = \$6,570$, while the optimum is \$6,858. In this example, the user of job j_6 is willing to pay up to \$17 per unit of computing power and timeslot, which is more than any other user is willing to pay. Consequently, j_6 is ranked first in the sorting phase, while j_4 and j_2 are ranked second and third, respectively. After having allocated these three jobs, the remaining jobs either do not fit on nodes n_1 and n_2 at the same time or the users are not willing to pay these nodes' reserve prices and their jobs are thus not allocated.

t	1	2	3	4	5	6	7	8	9
n_1	j_2	j_2, j_6	j_2, j_6	j_2, j_6	j_2, j_6	j_2, j_6	j_2, j_6	—	—
n_2	—	j_4	j_4	j_4	j_4	j_4	j_4	—	—

Table 3.3: Sample greedy allocation scheme.

This allocation scheme runs in polynomial time in the size of its input, i.e. the number of job requests and node offers. The sorting phase (Step 1) runs in $O(|J|\log|J|)$ and $O(|N|\log|N|)$[7] while the allocation phase (Step 2) runs in $O(|J||N|)$.

The computational speed of heuristics generally comes at the expense of efficiency. Let W^{OPT} be the (optimal) efficiency generated by the exact mechanism and W^{greedy} the efficiency generated by the greedy heuristic. Then the greedy heuristic exhibits the following worst case behavior:

Theorem 3 (Competitive ratio). *The greedy heuristic has a competitive ratio of* $W^{greedy}/W^{OPT} = 0$.

Proof. W.l.o.g., consider valuations and computing power only. Suppose one node offer with characteristics $(v_n, C_n) = (0, k)$, $k \in \mathbb{N}$, and two job requests $(v_{j_1}, c_{j_1}) = (2, 1)$ and $(v_{j_2}, c_{j_2}) = (1, k)$. The heuristic generates welfare of \$2 while the optimum is \$$k$. Clearly, $W^{greedy}/W^{OPT} \to 0$ for $k \to 0$. $\square$

While this is clearly a negative result, only a 2-approximation exists for GAP today (Chekuri and Khanna, 2006). Due to the multiple resources and time constraints (in particular considering Constraint 3.6), the structure of the allocation problem at hand is significantly more complex than the standard MKP and GAP, and these bounds thus probably do not hold for this special problem. Furthermore, the approximation techniques used to obtain these bounds generally do not allow for constructing truthful prices (Mu'alem and Nisan, 2008), which is a key feature of the mechanism to be developed in this chapter, as will be shown below. Those worst cases, however, are merely of theoretical value; the heuristic's efficiency needs to be evaluated in more realistic settings. Section 3.5 will thus report the results of an extensive numerical evaluation.

[7]Note that in practice this sorting can be done as jobs enter the system.

Besides its computational speed, the heuristic allocation scheme implements desirable strategic properties, such as limiting the potential gain to be had by both users and providers from manipulating the mechanism by reporting job and resource characteristics, i.e. computing and memory requirements and time constraints, that are untrue.

Truthfulness with respect to a *single* job's resource requirements is straightforward. If a job's resource requirements (CPU and memory) are understated, the job cannot be executed correctly[8]; the user has to pay for the used resources, but these are of no value. Overstating a job's requirements either increases the job's payment or results in the job not being scheduled at all due to insufficient available resources. The same holds for the job runtime d_j. There is, however, the possibility of users benefiting from shifting the job to later timeslots (i.e., $\tilde{r}_j > r_j$ while $\tilde{e}_j - \tilde{r}_j + 1 = e_j - r_j + 1$) since there might be less competition and thus lower prices. However, this strategic behavior is in fact desirable as it gives users an economic benefit to shift the system load to off-peak periods.

Essentially the same reasoning also applies when looking at resource providers. Providers clearly cannot gain from understating the amount of available resources, since it does not improve their nodes' position in the ranking phase of the heuristic but only reduces the set of possible allocations. Given sufficiently severe penalties (e.g. above market prices), providers cannot gain from overstating their available resources either, since they might then be unable to execute allocated jobs.

In the spirit of Moulin (2008), two further strategic properties of scheduling mechanisms can be introduced, which can be interpreted as a mechanism's robustness against *coalitions* of users or providers. These criteria are desirable as they promote fairness in the sense that low-volume users and providers are not at a disadvantage compared to high-volume users and providers.

Definition 5 (Job-merge-proofness). *An allocation algorithm is* job-merge-proof *when the user(s) of two job requests j_1 and j_2 cannot increase their combined ex post utility by merging $\theta_1 = (v_1, c_1, m_1, r_1, e_1)$ and $\theta_2 = (v_2, c_2, m_2, r_2, e_2)$ to one single job $\theta_M = (v_M, c_M = c_1 + c_2, m_M = m_1 + m_2, r_M = \min\{r_1,\ r_2\}, e_M = \max\{e_1,\ e_2\})$.*

The utility of users not only depends on the allocation scheme but also on the employed pricing scheme. Since two such pricing schemes will be presented later on, the investigation of this property is deferred at this point. Analogously to the demand side of the market, a merge-proofness property for resource providers can be defined:

Definition 6 (Node-merge-proofness). *An allocation algorithm is* node-merge-proof *when the provider(s) of two node offers n_1 and n_2 cannot increase their combined ex post utility by sub-*

[8]It is a common policy in grid systems to kill jobs requiring more resources than requested by the user.

mitting their offers $\theta_1 = (v_1, C_1, M_1, R_1, E_1)$ *and* $\theta_2 = (v_2, C_2, M_2, R_2, E_2)$ *as one single offer* $\theta_M = (v_M, C_M = C_1 + C_2, M_M = M_1 + M_2, R_M = \min\{R_1,\ R_2\}, E_M = \max\{E_1,\ E_2\})$.

In the setting at hand, this possibility can only be excluded due to technical constraints; it is currently not possible to define virtual machines that span multiple physical nodes. If this was possible, there are simple examples in which two nodes would benefit from merging since the mechanism could then allocate a large job to the merged node that would not fit on the single nodes.

In summary, the strategy space of a selfish user (provider) is essentially restricted to misstating the *valuation* for a job (node). In the following subsection, two pricing schemes are proposed that aim at aligning the *individual* participant's goal of utility maximization with the market designer's goal of allocative efficiency, i.e. *overall* welfare maximization.

3.4.3 Pricing the Outcome

As introduced above, besides the bidding language, (periodic) market mechanisms consist of two basic components, an allocation scheme and a pricing scheme (cf. Figure 3.1). While the allocation component generally accounts for the technical specifics of the grid setting, the objective of the pricing component is to induce the selfish participants in the market environment to act towards the general objective, in this case welfare maximization. To this end, in this section two pricing schemes are presented that can be used in conjunction with the greedy heuristic; both positive and negative results are derived analytically.

Proportional Critical-Value Pricing

The concept of critical-value pricing is based on Lehmann *et al.* (2002) and Mu'alem and Nisan (2008) and essentially implements the prominent Vickrey principle. Let $\phi_j(\theta) \in \mathbb{R}_+$ be the minimal valuation per unit of computing power and timeslot that job j would have needed to report to the mechanism in order to be allocated. Then, with critical-value pricing, the price of job j is set to

$$p_j(\theta) = \begin{cases} \phi_j(\theta) c_j d_j & \text{if } \sum\limits_{n \in N} \sum\limits_{t \in T} x_{jnt} > 0 \\ 0 & \text{else.} \end{cases}$$

It is easy to see that this pricing scheme is individually rational if the user of job j truthfully reports its type. Moreover, this pricing scheme implements the property of truthfulness with respect to job valuations:

Theorem 4 (Truthfulness of critical-value pricing). *The greedy heuristic with critical-value pricing is truthful with respect to job valuations:*

$$u_j(\theta_j, \tilde{\theta}_{-j}|\theta_j) \geq u_j((\tilde{v}_j, c_j, m_j, r_j, e_j), \tilde{\theta}_{-j}|\theta_j) = u_j(\tilde{\theta}_j, \tilde{\theta}_{-j}|\theta_j)$$

for all $j \in J$, θ_j, $\tilde{v}_j$, and $\tilde{\theta}_{-j}$.

Proof. Users are assumed to have quasi-linear utility functions:

$$u_j(\theta|\theta_j) = \begin{cases} v_j c_j d_j - p_j(\theta) & \text{if } j \text{ is allocated} \\ 0 & \text{else.} \end{cases}$$

Assume the user of j has truthfully reported her valuation, i.e. $\tilde{v}_j = v_j$. There are four cases to be considered (cf. Table 3.4).

State\Action	Decrease bid	Increase bid
Job j was allocated	Case 1	Case 2
Job j was not allocated	Case 3	Case 4

Table 3.4: Options for misreporting.

Suppose j was allocated, i.e. $\tilde{v}_j = v_j \geq \phi_j(\theta_j, \tilde{\theta}_{-j})$ and $p_j(\theta_j, \tilde{\theta}_{-j}) = \phi_j(\theta_j, \tilde{\theta}_{-j}) d_j c_j$. Reporting a lower valuation $\tilde{v}_j < v_j$ (**Case 1**) either leaves j in the allocation while it does not change ϕ_j ($\phi_j(\theta_j, \tilde{\theta}_{-j}) = \phi_j(\tilde{\theta})$) and consequently it does not change u_j, or it leads to j being rejected and thus $u_j(\tilde{\theta}|\theta_j) = 0 \leq (v_j - \phi_j(\theta_j, \tilde{\theta}_{-j})) c_j d_j = u_j(\theta_j, \tilde{\theta}_{-j}|\theta_j)$. Reporting a higher valuation $\tilde{v}_j > v_j$ (**Case 2**) leaves j being accepted while it does not change ϕ_j and consequently it does not change u_j.

Now suppose j was not allocated, i.e. $\tilde{v}_j = v_j < \phi_j(\theta_j, \tilde{\theta}_{-j})$ and $p_j(\theta_j, \tilde{\theta}_{-j}) = 0$. Reporting a lower valuation $\tilde{v}_j < v_j$ (**Case 3**) leaves j being rejected. Reporting a higher valuation $\tilde{v}_j > v_j$ (**Case 4**) either leaves j being rejected ($\tilde{v}_j < \phi_j(\tilde{\theta})$), or it leads to j being accepted ($\tilde{v}_j \geq \phi_j(\tilde{\theta}) > v_j$) and thus $u_j(\tilde{\theta}|\theta_j) = (v_j - \phi_j(\tilde{\theta})) c_j d_j < 0 = u_j(\theta_j, \tilde{\theta}_{-j}|\theta_j)$. □

The critical value of a job essentially hinges on the competition of other jobs for the same resources. If there is no competition, the job is only required to pay the cheapest possible reserve price. Otherwise, the job at least needs to outbid these competing jobs.

As pointed out above, the property of job-merge-proofness (cf. Definition 5) depends on the employed pricing scheme. With respect to the heuristic with critical-value pricing, the following result can be stated:

Theorem 5 (critical-value pricing is not job-merge-proof). *The greedy heuristic is not job-merge-proof if complemented by critical-value pricing.*

Proof. It is to be shown that the merger of two job requests results in a higher overall utility compared to submitting the two requests separately. This can be shown by a sample scenario.

W.l.o.g., assume that all requests and offers have the same runtimes and availability times, respectively. Consider the three job requests $(v_{j_1}, c_{j_1}, m_{j_1}) = (\$4, 1, 100)$, $(v_{j_2}, c_{j_2}, m_{j_2}) = (\$3, 1, 150)$, and $(v_{j_3}, c_{j_3}, m_{j_3}) = (\$2, 2, 100)$, and two resource offers $(v_{n_1}, c_{n_1}, m_{n_1}) = (\$0, 1, 150)$ and $(v_{n_2}, c_{n_2}, m_{n_2}) = (\$1, 3, 200)$.

When the jobs j_1 and j_3 are submitted separately, j_1 is allocated and has a critical value (price) of $\phi_1 = \$1$, j_3 is not allocated and has a critical value of $\phi_3 = \$3$. The utility of j_1 is $\$4 - \$1 = \$3$ and the utility of j_3 is \$0 since it is not allocated.

When j_1 and j_3 are merged to $(v_{j_M}, c_{j_M}, m_{j_M}) = (\$2\frac{2}{3}, 3, 300)$, j_M is allocated and has a critical value of $\phi_M = \$1$ (price per CPU unit). Consequently, the combined utility of j_1 and j_3 is \$5 and the users can increase their combined utility by merging their jobs. $\square$

The major drawback of critical-value pricing in this setting is the computational cost of determining the critical values. In order to calculate the critical value for a specific allocated job, it is not possible to simply take the valuation of the highest-ranking unexecuted job since this job might not have been executable in any case due to capacity constraints. Moreover, removing j from the allocation might change the allocation of other jobs within the allocation as well. Consequently, to determine the critical value for each allocated job j, the allocation *without* j needs to be determined: All other jobs from the ranking are successively allocated and, after having allocated a job, it is checked whether j can still be accommodated. Note that, while the mechanism seems to suffer from the same computational problems as the prominent VCG mechanism, the critical values can still be computed in polynomial time due to the heuristic allocation scheme.

If there is sufficient competition such that critical values are above reserve prices, critical-value pricing of job requests generates a surplus, i.e. the overall payments exceed the providers' reserve prices. The question then is how to distribute the surplus generated on the demand side of the market among resource providers in a way that will also produce truthful payments while preserving budget-balance. Unfortunately, critical-value pricing is not applicable to the pricing of resource offers. It would require a binary decision in the sense that a node is only allocated as a whole, or not at all. In the model at hand, however, resource offers are divisible. Worse, the following theorem holds:

Theorem 6 (Impossibility of truthful prices for node offers). *There is no payment scheme complementary to the greedy heuristic capable of generating truthful payments to resource providers.*

Proof. A key requirement for achieving truthfulness is designing a monotonically decreasing allocation scheme in the sense that providers cannot be allocated a higher load by overstating their reserve prices (Archer and Tardos, 2001). By means of a simple example it can be shown that with the heuristic providers can potentially *increase* their load by overstating their reserve prices.

W.l.o.g., consider valuations and computing power only. Assume two resource requests, $(v_{j_1}, c_{j_1}) = (\$10, 1)$ and $(v_{j_2}, c_{j_2}) = (\$9, 2)$, and two truthful resource offers, $(v_{n_1}, C_{n_1}) = (\$1, 2)$ and $(v_{n_2}, C_{n_2}) = (\$2, 2)$. Then the first offer will be allocated one unit of computing power. Now assume the first provider had reported $(\tilde{v}_{n_1}, C_{n_1}) = (\$3, 2)$ instead. Then she would have been allocated two units of computing power. Consequently the heuristic's allocation scheme is not monotonically decreasing. $\square$

This is clearly a strong negative result as it extends across any payment scheme for the greedy heuristic. As a consequence of Theorem 6, it is not possible to design a payment scheme that will preclude *any* strategic behavior on the part of resource providers; one can only try to limit these strategic possibilities. One possibility is to distribute any surplus according to the providers' contribution of computing power, the key resource in this setting. Let

$$S(\theta) = \sum_{j \in J} p_j(\theta) - \sum_{j \in J} \sum_{n \in N} \sum_{t \in T} x_{jnt} c_j v_n$$

be this surplus. The first term captures the total revenue collected by critical-value pricing, whereas the second term captures the providers' reserve prices for the allocation schedule at hand. Then, with *proportional payments*, provider n receives payments amounting to

$$p_n(\theta) = \sum_{j \in J} \sum_{t \in T} x_{jnt} c_j v_n + S(\theta) \cdot \frac{\sum_{j \in J} \sum_{t \in T} x_{jnt} c_j}{\sum_{j \in J} \sum_{m \in N} \sum_{t \in T} x_{jmt} c_j},$$

where the first summand captures n's reserve price for the allocation schedule and the second summand captures n's proportional payments. The rationale for proportional payments is that the share of the surplus that is allotted to n does not directly depend on n's reported reserve price. Instead, it only depends on the final allocation schedule, which can only be influenced by n to a rather limited extent depending on the competition on both sides of the market. The

Proportional critical-value pricing				k-pricing ($k = 0.5$)			
Job j	p_j	Node n	p_n	Job j	p_j	Node n	p_n
j_2	2,940	n_1	6,165.88	j_2	2,695	n_1	6,820
j_4	5,112	n_2	5,846.12	j_4	5,751	n_2	5,751
j_6	3,960			j_6	4,125		
j_1, j_3, j_5	0			j_1, j_3, j_5	0		
Σ	12,012	Σ	12,012	Σ	12,571	Σ	12,571

Table 3.5: Sample prices and payments.

prices and payments generated by proportional critical-value pricing for the sample job requests and node offers are listed in Table 3.5.

k-pricing

k-pricing is an alternative pricing scheme to proportional critical-value pricing. It was introduced in Schnizler *et al.* (2008) for double-sided combinatorial auctions. The basic idea is to distribute the welfare generated by the allocation algorithm between users and resource providers according to a factor $k \in [0,1]$. For instance, assume an allocation of resources from a specific provider to a specific user. The user values these resources at \$10 while the provider has a reserve price of \$5. Then the (local) welfare of this transaction is $\$10 - \$5 = \$5$, and $k \cdot \$5$ of the surplus is allotted to the user (who thus has to pay $\$10 - k \cdot \$5$) and $(1-k) \cdot \$5$ is allotted to the provider (who thus receives $\$5 + (1-k) \cdot \$5$).

Formally, the price of job j is

$$p_j(\theta) = \sum_{n \in N} \sum_{t \in T} x_{jnt} c_j \left(v_j - k \cdot \left(v_j - v_n \right) \right).$$

The payment to node n is

$$p_n(\theta) = \sum_{j \in J} \sum_{t \in T} x_{jnt} c_j \left(v_n + (1-k) \cdot \left(v_j - v_n \right) \right).$$

k-pricing has two main advantages. The distribution of welfare among users and providers can be flexibly pre-defined by setting the factor k accordingly, thus allowing for both fairness and revenue considerations. Prices can be determined in polynomial runtime. On the downside, it does not yield truthful prices on either side of the market and, as with critical-value pricing, users can benefit from merging jobs:

Theorem 7 (k-pricing is not job-merge-proof). *The greedy heuristic is not job-merge-proof if complemented by k-pricing.*

Proof. A simple scenario shows that two users can merge their jobs so as to make at least one of them better off while not reducing the utility of the other user.

W.l.o.g., consider valuations and computing power only. First, assume one node $(v_n, C_n) = (\$0, 2)$ and three job requests $(v_{j_1}, c_{j_1}) = (\$4, 1)$, $(v_{j_2}, c_{j_2}) = (\$2, 1)$, and $(v_{j_3}, c_{j_3}) = (\$1, 1)$. Then the jobs j_1 and j_2 are allocated while j_3 is not. j_1 has utility of $\$4 - (\$4 - k \cdot \$4) = k \cdot \$4$ while j_3 has zero utility.

Now assume a merger of j_1 and j_3 to $(v_{j_M}, c_{j_M}) = (\$2.5, 2)$. Then the merged job request is ranked higher than j_2 in the heuristic's sorting phase and is allocated. The utility derived by the merged job is $\$5 - (\$5 - k \cdot \$5) = k \cdot \$5 \geq k \cdot \$4$ for $k \in [0, 1]$. $\square$

The prices and payments generated by proportional critical-value pricing and k-pricing with $k = 0.5$ are listed in Table 3.5. By construction, both pricing schemes generate budget-balanced prices and payments. In this example, k-pricing produces higher revenue (\$12,571) than proportional critical-value pricing (\$12,012).

3.5 Numerical Evaluations

The analytic evaluation in the previous section gave some general insights into the strategic and computational properties of the presented allocation and pricing schemes. These general properties, however, provide rather limited guidance to the market operator, who has to choose an "adequate" mechanism for the setting at hand. Consequently, numerical simulations are employed in order to obtain more detailed insights into the properties of the presented allocation and pricing schemes, particularly with respect to

- the *complexity of the underlying allocation problem*,
- the *allocative efficiency of the greedy heuristic* compared to the optimum, and
- the *incentives* of selfish users to misreport their valuations to the mechanism.

Unfortunately, this evaluation cannot be based on real grid workload traces. The only publicly available traces are cluster workloads available in the Parallel Workload Archive[9]. However, this data exhibits various problems. The traces are often incomplete and do not contain all of the parameters needed for this specific setting. Another substantial limitation is that clusters mainly consist of homogeneous nodes, whereas this grid setting is heterogeneous. Consequently, artificial workloads are used for this evaluation. This will also permit testing the allocation problem

[9]`http://www.cs.huji.ac.il/labs/parallel/workload/`

Parameter	Job Requests	Node Offers
Computing power	1 + Binomial(5, 0.5)	1 + Binomial(10, 0.5)
Memory	Lognormal(4, 0.15)	Lognormal(5, 0.2)
Start time	Binomial(5, 0.5)	Binomial(4, 0.5)
Valuation	Uniform[10,20]	Uniform[7, 12]

Table 3.6: Simulation setting.

as well as the exact and the heuristic allocation scheme for their sensitivity to changes in the job and node characteristics as well as with respect to the competition in the market.

3.5.1 What Makes Instances Hard?

The computational tractability of the allocation problem mainly depends on the number of job requests and node offers along with the job runtimes, which determine the scheduling horizon. In addition to these parameters, the heuristic's approximation of the optimal solution also depends on the level of "competition" in the market. The level of competition can be distinguished along two dimensions: the size of the order book, i.e. the number of job requests and node offers, and the ratio of job requests to node offers. If the competition is very low, meaning that essentially all job requests are accommodated by the optimal solution, the heuristic will generally also be able to allocate most of the jobs, and the deviation from the optimal solution as regards allocative efficiency will be low. With increasing competition, the heuristic is more likely to take suboptimal allocation decisions.

Data Generation

Three parameters were varied: the number of job requests and node offers (20 nodes, 40 nodes, up to 200 nodes), the ratio of jobs to nodes (one job per node, two jobs per node, and three jobs per node), and the mean job runtime $\overline{d}$.

Based on the model and the corresponding bidding language, there is a set of further parameters upon which problem instances (henceforth "order books") need to be generated. Table 3.6 specifies the probability distributions based on which some of the job and node characteristics were generated and which were not changed across the several settings. Further information, e.g. about job runtimes and the availability of nodes, will be given in the description of the specific settings.

The computing requirements of the jobs were randomly drawn from a binomial distribution (adding one CPU to always obtain positive values) with n = 5 (i.e., five independent trials) and p = 0.5 (the probability of a success in each random trial). The binomial distribution with p = 0.5 is symmetric and bell-shaped like the normal distribution for continuous parameters. It was chosen so as to make the majority of the jobs have medium CPU demand, but to also have some outliers that have low or high CPU demand. For example, the distribution 1 + Binomial(5, 0.5) leads to a mean of 3.5 available CPUs; the minimum is 1, the maximum 6. Memory requirements (as integer values) were generated based on a lognormal distribution with a mean of ≈ 55.216 and a variance of ≈ 69.375 (mean 4 and variance 0.15 in log space). The resource characteristics of nodes were generated likewise with a mean of 5 and a variance of 0.2 in log space. The lognormal distribution is recommended by Feitelson (2002) for the creation of parameters such as file sizes. Representing a normal distribution in log space, it is positively skewed. The start times and end times were drawn from binomial distributions while the maximum willingness to pay and reserve prices were drawn from uniform distributions (as integer values). In the economic literature, the uniform distribution is a prominent choice to model valuations, cf. Maskin and Riley (2000) and the references therein.

The simulations were run on an Intel Pentium Xeon computer with 3.2 GHz and 2 GB memory. For each parameter combination, 30 order books were generated based on the aforementioned distributions. The means and coefficients of variation (which normalize the standard deviation by the mean) across all 30 runs will be reported.

Data Analysis

Impact of the Level of Competition on the Allocation Problem's Complexity

In order to obtain the exact solutions, the allocation problem was modeled and solved with CPLEX 9.1, a state-of-the-art commercial optimization engine. Due to the potentially large runtimes necessary to solve the integer problem, CPLEX was stopped in the event that it found a feasible solution within 0.1% of the projected optimum or after at most 30 minutes. CPLEX then returned the best solution found so far. This allocation algorithm is henceforth called "Anytime-CPLEX" in the spirit of Sandholm *et al.* (2005).

Table 3.7 shows the impact of varying the competition in the market. The runtimes of jobs were drawn from the Binomial distribution $1+B(5,0.5)$; node availability times were drawn from $1+B(8,0.5)$.

It can clearly be seen that the runtime of Anytime-CPLEX grows exponentially with increasing order book sizes. Anytime-CPLEX already takes about 9 minutes on average to optimally solve allocation problems with 200 job requests and 200 node offers. The effect of increasing the ratio $|J|/|N|$ of jobs to nodes is even more significant. With twice as many jobs as nodes, Anytime-CPLEX only finds feasible suboptimal solutions within 30 minutes in 9 out of 30 runs for 280

	$\lvert J\rvert/\lvert N\rvert = 1$			$\lvert J\rvert/\lvert N\rvert = 2$			$\lvert J\rvert/\lvert N\rvert = 3$		
Nodes	*time*	*CV*	*exec*	*time*	*CV*	*exec*	*time*	*CV*	*exec*
20	273.37	1.67	0.963	2,639.60	1.22	0.830	27,729.70	0.96	0.646
40	2,187.37	0.80	0.987	24,258.30	0.63	0.859	290,537.40	1.07	0.672
60	11,746.9	1.07	0.984	97,095.87	0.73	0.879	754,275.10	0.52	0.681
80	22,575.03	0.66	0.990	221,159.37	0.57	0.862	1,488,805.27	0.30	0.675
100	51,310.43	0.42	0.996	507,706.67	0.50	0.872	1,793,519.80	0.02	0.678
120	105,978.83	0.34	0.996	878,603.13	0.51	0.871	– (19)†	–	–
140	170,456.9	0.46	0.996	1,361,123.37 (9)‡	0.29	0.872	– (30)†	–	–
160	276,611.00	0.35	0.995	1,605,776.03 (18)‡	0.17	0.862	– (30)†	–	–
180	452,946.93	0.29	0.996	1,715,245.27 (24)‡	0.16	0.860	– (30)†	–	–
200	534,558.37	0.28	0.994	– (16)†	–	–	– (30)†	–	–

Table 3.7: Runtime of Anytime-CPLEX depending on the number of job requests and node offers and the ratio of job requests to node offers. *time* represents the mean runtime (in milliseconds), *CV* the coefficient of variation, and *exec* the ratio of allocated jobs to submitted job requests. "–" marks settings in which CPLEX ran out memory when trying to model the allocation problem, where "– $(n)^{\dagger}$" indicates that this happened in n out of 30 cases. "$(n)^{\ddagger}$" indicates that CPLEX only found a *suboptimal* solution within 30 minutes in n out of 30 cases.

jobs and 140 nodes. For a ratio of three, Anytime-CPLEX can only solve order books with 300 jobs and 100 nodes but nearly uses up its preset time limit of 30 minutes in all cases. For larger order books, Anytime-CPLEX is not able to internally model the allocation problem and runs out of memory.

Impact of Job Lengths on the Allocation Problem's Complexity

Table 3.8 shows the impact of increasing the mean job runtime (and thus the scheduling horizon) on the runtime of Anytime-CPLEX. This is an important issue as it essentially determines how "fine-grained" the timeslots can be set, which ultimately involves a trade-off between allocative efficiency (small timeslots) and computational tractability (large timeslots). Job runtimes were drawn from the Binomial distributions $B(5,0.5)$, $B(6,0.5)$, $B(7,0.5)$, $B(8,0.5)$, always adding one timeslot. The mean job runtimes thus are $\overline{d} = 3.5,\ 4,\ 4.5,\ 5$. Node availability times were drawn from $B(8,0.5)$ to $B(11,0.5)$, also always adding one timeslot, to obtain longer availability times.

As discussed above when comparing the allocation problem to MKP and GAP, the allocation problem essentially consists of several complex knapsack problems, one problem per timeslot. Constraint 3.6 of the allocation problem connects these problems by stipulating that a feasible overall solution must be feasible for all of these local problems at the same time. This complexity is reflected in the numerical results. While Anytime-CPLEX is still able to optimally solve all problem instances for $\overline{d} = 3.5$, its runtime more than doubles for almost all order book sizes when the mean job runtime is increased to $\overline{d} = 5$. This shows that strong consideration should be given to the granularity of the timeslots when designing the allocation scheme for a specific setting.

Table 3.8 lists the runtimes of the greedy allocation heuristic for the most complex setting with $\overline{d} = 5$. The results clearly showcase the computational speed of the greedy heuristic, which, if combined with k-pricing, only takes about 140 milliseconds to allocate 200 job requests and 200 node offers as opposed to more than 21 minutes for Anytime-CPLEX. The last column of Table 3.8 further shows the impact on the mechanism's runtime if switching to proportional critical-value pricing. While it adds considerably to the mechanism's runtime, the mechanism still takes less than three seconds on average to solve the allocation and pricing problem for the largest instances.

Impact of the Level of Competition on the Heuristic's Efficiency Ratio

The previous results show that the level of competition is one of the main reasons for the complexity of the underlying allocation problem. Moreover, as discussed above, with increasing competition the heuristic will increasingly take unfortunate allocation decisions and the deviation from Anytime-CPLEX (as a proxy for the optimal solution) with respect to allocative

	Anytime-CPLEX with k-pricing								Greedy heuristic ($\overline{d} = 5$)			
Jobs &	$\overline{d} = 3.5$		$\overline{d} = 4$		$\overline{d} = 4.5$		$\overline{d} = 5$		k-pricing		prop. critical-value	
nodes	*time*	*CV*	*time*	*CV*	*time*	*CV*	*time*	*CV*	*time*	*CV*	*time*	*CV*
20	179.70	1.04	239.00	1.33	174.43	0.65	263.93	1.63	2.10	2.55	7.30	1.07
40	2,713.50	1.20	4,101.53	1.07	4,349.60	0.84	5,083.50	0.79	10.87	1.45	37.53	0.49
60	10,978.53	1.08	16,624.03	0.84	17,859.83	0.47	19,309.47	0.64	20.77	0.74	93.30	0.20
80	22,851.07	0.53	37,607.70	0.62	62,458.30	0.61	74,331.23	0.62	34.67	0.48	194.73	0.12
100	54,460.93	0.36	75,485.03	0.40	85,488.53	0.48	141,974.37	0.42	41.17	0.21	365.97	0.10
120	98,768.17	0.46	137,179.13	0.35	196,347.33	0.38	206,411.00	0.24	53.97	0.18	636.40	0.11
140	187,001.47	0.44	280,872.87	0.40	313,291.60	0.48	379,043.30	0.36	90.57	0.32	942.67	0.08
160	257,522.77	0.35	356,705.77	0.30	459,101.00	0.22	594,920.23 $(1)^‡$	0.45	93.93	0.14	1,417.73	0.07
180	394,847.77	0.25	555,522.93	0.29	679,264.40	0.20	905,317.33	0.32	121.97	0.20	1,944.70	0.07
200	564,355.30	0.43	794,977.23 $(1)^§$	0.32	1,069,540.57 $(1)^‡(1)^§$	0.30	1,284,634.43 $(3)^‡(1)^§$	0.26	139.23	0.14	2,630.23	0.07

Table 3.8: Runtime of Anytime-CPLEX and the heuristic depending on the number of job requests and node offers and the (mean) job runtime. $\overline{d}$ is the mean job runtime (in timeslots), *time* represents the mean runtime (in milliseconds), *CV* is the coefficient of variation. "$(n)^‡$" indicates that CPLEX only found a *suboptimal* solution within 30 minutes in n out of 30 cases. The addition $(n)^§$ indicates that CPLEX could not find any *feasible* solution with positive welfare within 30 minutes in n out of 30 cases.

	$\|J\|/\|N\| = 1$			$\|J\|/\|N\| = 2$			$\|J\|/\|N\| = 3$		
Nodes	*ratio*	*CV*	*exec*	*ratio*	*CV*	*exec*	*ratio*	*CV*	*exec*
20	0.967	0.17	0.963	0.946	0.11	0.830	0.920	0.11	0.646
40	0.977	0.13	0.987	0.961	0.09	0.859	0.941	0.07	0.672
60	0.983	0.09	0.984	0.970	0.07	0.879	0.948	0.06	0.681
80	0.981	0.11	0.990	0.971	0.08	0.862	0.953	0.06	0.675
100	0.983	0.07	0.996	0.972	0.06	0.872	0.959	0.05	0.678
120	0.986	0.07	0.996	0.976	0.06	0.871	– $(19)^{\dagger}$	–	–
140	0.987	0.06	0.996	0.978 $(9)^{\ddagger}$	0.04	0.872	– $(30)^{\dagger}$	–	–
160	0.987	0.04	0.995	0.982 $(18)^{\ddagger}$	0.03	0.862	– $(30)^{\dagger}$	–	–
180	0.988	0.05	0.996	0.985 $(24)^{\ddagger}$	0.05	0.860	– $(30)^{\dagger}$	–	–
200	0.991	0.05	0.994	– $(16)^{\dagger}$	–	–	– $(30)^{\dagger}$	–	–

Table 3.9: Ratio of the heuristic's mean efficiency to the mean efficiency generated by Anytime-CPLEX depending on the number of job requests and node offers and the ratio of requests to offers. *ratio* denotes the ratio of the mean efficiencies, *CV* the coefficient of variation, and *exec* the ratio of allocated jobs to submitted job requests with Anytime-CPLEX."–" marks settings in which CPLEX ran out memory when trying to model the allocation problem, where "– $(n)^{\dagger}$" indicates that this happened in n out of 30 cases. "$(n)^{\ddagger}$" indicates that CPLEX only found a *suboptimal* solution within 30 minutes in n out of 30 cases.

efficiency will become larger. Consequently, to measure the inherent trade-off between computational complexity and allocative efficiency for the greedy heuristic, Table 3.9 shows the heuristic's deviation from the solutions returned by Anytime-CPLEX for increasing order book sizes and ratios of job requests to node offers. The results are based on the same order books as above, i.e. job runtimes were again drawn from $1+B(5,0.5)$ and node availability times were drawn from $1+B(8,0.5)$.

For a ratio of $|J|/|N| = 1$, Anytime-CPLEX and the heuristic can essentially accommodate all requests and the heuristic consequently approximates the near-optimal solution of Anytime-CPLEX by 96.7% for the smallest order book and by 99.1% for the largest order book. This approximation ratio will generally increase with increasing order book sizes as the heuristic receives additional degrees of freedom in making its allocation decisions. Interestingly, for increasing ratios of jobs to nodes, the deviation increases only slightly. Even for a ratio of $|J|/|N| = 3$, whereby only about 67% of the jobs can be accommodated, the heuristic still generates about 95% of the allocative efficiency of Anytime-CPLEX.

In summary, the heuristic strikingly outperforms Anytime-CPLEX with respect to computational tractability; the latter was also clearly shown to be infeasible in practice. At the same time, the heuristic closely approximates the optimal allocative efficiency in the average case, as opposed to the negative result for the worst case (Theorem 3).

3.5.2 Strategic Behavior

In this subsection, the effect of combining the greedy heuristic with the proposed pricing schemes is analyzed on the incentives of single users and providers to truthfully report their valuations. These incentives inherently depend on the level of competition in the market. Moreover, for k-pricing, it will be interesting to investigate the influence of the choice of the parameter k on the users' incentives.

Three parameters were varied: For k-pricing, the factor k was shifted ($k = 0.3, 0.5, 0.7$). Furthermore, the order book size was varied and the ratio of job requests to node offers was increased from one job per node to three jobs per node. For each such parameter combination, 200 order books were generated based on the simulation setting specified above in Table 3.6, with job runtimes being drawn from $1 + B(5, 0.5)$ and node availabilities from $1 + B(8, 0.5)$. Then each combination of the heuristic and the four pricing schemes (k-pricing with $k = 0.3, 0.5, 0.7$ and proportional critical-value pricing) was fed with the same set of order books, i.e. 200 runs per setting were performed. The large number of runs is due to the number of parameters and their interdependencies (e.g. recall that the total valuation of a job (node) equals its reported valuation times its required computing power times its time span); this might cause heavy statistical noise. Therefore the means across all 200 runs are reported. Since the users' and providers' utility is measured on a cardinal scale, the statistical significance was tested using one-tailed matched-pairs t-tests with the alternative hypothesis that the market participant benefits from misreporting, i.e. her mean difference in utility is greater than zero (Fahrmeir *et al.*, 2004).[10] The p-values of the tests are indicated in the tables.

Manipulation by a Single User

In the first setting, it was analyzed to what extent a single user can benefit from reporting an untrue valuation. The user of j reported 50%, 60%, up to 150% of her true valuation; $\tilde{v}_j$ is therefore a *percentage bid* of the true valuation.

The user of j has two possible strategies for deviating from truthful bidding:

- The user *understates* her true valuation for j. With k-pricing, there are two opposite implications for this strategy compared to truthful bidding: On the one hand (the *adverse*

[10] Due to the large sample size, the t-test is very robust to violations of the normality assumption underlying the test (Ramsey, 1980; Sawilowsky and Blair, 1992; Bridge and Sawilowsky, 1999).

effect), j risks receiving a lower ranking and may thus not be allocated at all, or it may be allocated to a node n with a higher reserve price and may thus obtain a k-fraction of the smaller resulting welfare spread $\tilde{v}_j - v_n$. On the other hand (the *beneficial effect*), j may remain in the allocation and may even be able to obtain a larger fraction of the welfare spread, e.g. if it is still allocated to the same node as with truthful bidding ($\tilde{v}_j - k \cdot (\tilde{v}_j - v_n) < v_j - k \cdot (v_j - v_n)$).

- The user *overstates* her true valuation for j. Equivalently, j may be allocated to a cheaper node and reap a k-fraction of the larger surplus (the *beneficial effect*). However, it could also remain allocated to the same or an only slightly cheaper node and obtain a smaller fraction of the surplus (the *adverse effect*).

The factor k and the competition in the market ultimately determine which effect will prevail. For example, the bigger k is, the bigger the beneficial effect and the smaller the adverse effect from overstating will generally be. Moreover, the greater the competition, the smaller the chances of being allocated to the same node as with truthful bidding when misreporting the valuation.

Table 3.10 lists the utility of a single user when misstating compared to truthful reporting for a low level of competition with 20 jobs and 20 nodes. Table 3.11 lists the utility for a more competitive setting with 60 jobs and 20 nodes.

For the low level of competition (one job per node), k-pricing is essentially robust to overstating; the adverse effect outweighs the beneficial effect on average. Moreover, the smaller k is, the less likely the user can gain from overstating. The point is that if k is small, the job must be allocated to a much cheaper node than would be the case with a larger k in order to benefit. For example, assume a job with a true valuation of \$100 is allocated to a node with a reserve price of \$50. Then, for $k = 0.5$, the job has to pay \$75. If the user overstates and reports a valuation of, say, \$120, the job must be allocated to a node with a reserve price of at most \$29 (if integer valuations are imposed) to benefit. Now assume $k = 0.3$. With truthful reporting, the job has to pay \$85. The user again decides to report a valuation of \$120 instead in order to achieve a higher ranking in the heuristic's sorting phase. The job must then be allocated to a node with a reserve price of at most \$3 to benefit.

The effect of understating is essentially inverse. The results show that the heuristic combined with k-pricing is significantly vulnerable to understating for smaller values of k ($k = 0.3, 0.5$). For example, for $k = 0.3$, a user can gain more than 80% in (expected) utility by reporting only 70% of her true valuation. Again, the effect inherently depends on the factor k. With a small k, it is easier for users to reap a higher fraction of the supposed surplus. For example, assume again that for truthful reporting, a job with a true valuation of \$100 is allocated to a node with a

Percentage bid	k-pricing						Critical-value pricing	
	$k = 0.3$		$k = 0.5$		$k = 0.7$			
	abs	*rel*	*abs*	*rel*	*abs*	*rel*	*abs*	*rel*
50%	17.61	0.75	18.05	0.46	18.50	0.34	6.93	0.13
60%	40.08	1.71***	42.31	1.09	44.54	0.82	33.34	0.62
70%	42.46	1.82***	47.30	1.21***	52.15	0.96	45.36	0.84
80%	38.86	1.66***	46.77	1.20***	54.67	1.00	50.64	0.93
90%	32.52	1.39***	43.94	1.13***	55.36	1.02	53.36	0.98
100%	23.37	1.00	38.95	1.00	54.53	1.00	54.20	1.00
110%	11.78	0.50	31.27	0.80	50.76	0.93	53.60	0.99
120%	-0.24	-0.01	23.15	0.59	46.54	0.85	53.06	0.98
130%	-12.30	-0.53	15.11	0.39	42.53	0.78	52.24	0.96
140%	-24.51	-1.05	6.65	0.17	37.81	0.69	52.24	0.96
150%	-36.68	-1.57	-1.77	-0.05	33.15	0.61	52.24	0.96

Table 3.10: Utility for a single misreporting user with 20 jobs and 20 nodes. *abs* denotes the mean absolute utility, *rel* the ratio of means of the utility when misreporting divided by the utility from truthful reporting. * denotes significance at the level of p = 0.1, ** at p = 0.05, and *** at p = 0.01.

Percentage bid	k-pricing						Critical-value pricing	
	$k = 0.3$		$k = 0.5$		$k = 0.7$			
	abs	*rel*	*abs*	*rel*	*abs*	*rel*	*abs*	*rel*
50%	0.22	0.01	0.23	0.01	0.24	0.01	0.92	0.04
60%	6.07	0.37	6.37	0.23	6.67	0.17	1.60	0.07
70%	16.61	1.01	18.22	0.67	19.84	0.52	10.85	0.46
80%	21.34	1.30***	25.21	0.92	29.08	0.76	18.60	0.78
90%	20.03	1.22***	27.01	0.99	33.98	0.89	22.99	0.96
100%	16.41	1.00	27.35	1.00	38.29	1.00	23.83	1.00
110%	8.05	0.49	22.10	0.81	36.15	0.94	23.08	0.97
120%	-1.74	-0.11	16.82	0.61	35.38	0.92	20.74	0.87
130%	-11.90	-0.72	10.81	0.40	33.52	0.88	17.27	0.72
140%	-23.40	-1.43	3.53	0.13	30.46	0.80	12.16	0.51
150%	-35.49	-2.16	-4.20	-0.15	27.08	0.71	9.26	0.39

Table 3.11: Utility for a single misreporting user with 60 jobs and 20 nodes. *abs* denotes the mean absolute utility, *rel* the ratio of means of the utility when misreporting divided by the utility from truthful reporting. * denotes significance at the level of p = 0.1, ** at p = 0.05, and *** at p = 0.01.

reserve price of \$50 and has to pay \$75 for $k = 0.5$ and \$85 for $k = 0.3$. If the user understates and reports a valuation of, say, \$80, for $k = 0.5$, it must be allocated to a node with a reserve price of at most \$74 to benefit. For $k = 0.3$, it must be allocated to a node with a reserve price of at most \$63 to benefit.

As stated in Theorem 4, with (proportional) critical-value pricing, users cannot gain from misstating their valuations in any case, so the same holds for the means across all runs. One interesting aspect is that critical-value pricing does not punish overstating as severely as k-pricing. If a job is allocated with truthful reporting, it is also allocated if the user overstates the valuation. But as the price does not directly depend on the reported valuation, the overall utility does not change. The only negative effect from overstating arises if a job only becomes allocated if the user overstates her valuation. Since in this case the critical value is larger than the true valuation, the user suffers a negative utility.

Interestingly, while k-pricing is vulnerable to understating, the potential gain from understating significantly diminishes as the competition in the market increases (cf. Table 3.11). The argument is that the risk of being allocated to a substantially more expensive node (and thus reaping a k-fraction of the smaller surplus) increases as the competition in the market increases. This is in line with the theoretic results of Hurwicz (1972) and Roberts and Postlewaite (1976) who show that, as the market size increases, users essentially become price takers and strategic considerations converge towards truthful behavior.

Manipulation by a Single Provider

In the second setting, which is symmetric to the setup above, it is analyzed to what extent a single *provider* can benefit from reporting an untrue reserve price. The implications from under- or overstating the true reserve price are essentially analogous to those on the demand side.

- As for users, there are two opposite effects of understating the reserve price. Recall that the payment to a resource provider consists of two components: her reserve price and her fraction of the surplus, i.e. total revenue less total reserve prices. On the one hand, by understating the reserve price, the provider inherently lowers the first component. On the other hand, she may achieve a higher ranking for her node and may thus be allocated more load and thus a higher fraction of the surplus.

- The implications of overstating the reserve price are essentially the reversion of the consequences of understating this valuation.

Percentage bid	k-pricing						Proportional critical-value pricing	
	$k = 0.3$		$k = 0.5$		$k = 0.7$			
	abs	*rel*	*abs*	*rel*	*abs*	*rel*	*abs*	*rel*
50%	70.10	0.97	17.81	0.34	-34.48	-1.11	-58.09	-2.15
60%	86.16	1.19***	35.92	0.69	-14.31	-0.46	-38.06	-1.41
70%	97.12	1.34***	50.95	0.98	4.78	0.15	-16.70	-0.62
80%	88.01	1.21***	52.20	1.01	16.39	0.53	4.05	0.15
90%	71.89	0.99	47.20	0.91	22.51	0.72	18.24	0.67
100%	72.60	1.00	51.86	1.00	31.12	1.00	27.05	1.00
110%	49.82	0.69	37.81	0.73	25.80	0.83	26.53	0.98
120%	29.21	0.40	23.34	0.45	17.48	0.56	19.44	0.72
130%	12.14	0.17	10.26	0.20	8.38	0.27	10.38	0.38
140%	11.90	0.16	10.26	0.20	8.62	0.28	10.19	0.38
150%	4.65	0.06	4.06	0.08	3.46	0.11	3.99	0.15

Table 3.12: Utility for a single misreporting provider with 20 jobs and 20 nodes. *abs* denotes the mean absolute utility, *rel* the ratio of means of the utility when misreporting divided by the utility from truthful reporting. * denotes significance at the level of p = 0.1, ** at p = 0.05, and *** at p = 0.01.

Table 3.12 lists the mean utility across all 200 runs for a single provider when misstating compared to truthful reporting in each of the four payment schemes and for a low level of competition (among users). Table 3.13 shows the results for the more competitive setting with 60 jobs and 20 nodes.

For a low level of competition, k-pricing with a small k $(= 0.3)$ is vulnerable to understating for the supply side of the market also. This vulnerability diminishes as k increases. These results can again be explained by means of the two opposite implications of understating or overstating. As discussed above, if provider n understates the reserve price v_n, then this component of the payment is automatically lowered, but she might be able to extract a fraction of the larger spread $v_j - \tilde{v}_n$. However, since the provider receives $(1-k)\cdot(v_j - \tilde{v}_n)$ of this spread, this beneficial effect of understating gradually diminishes as k converges to 1.

Moreover, k-pricing is robust to overbidding, as the loss in the surplus component of k-pricing's payment scheme outweighs the gain in the reserve price component.

An important result is that, while proportional critical-value pricing does not produce truthful payments (in dominant strategies), it is robust to both under- and overbidding on average. With critical-value pricing and a low level of competition, the surplus (payments of users exceeding reserve prices) to be distributed among providers with the proportional payment rule will gen-

Percentage bid	k-pricing						Proportional critical-value pricing	
	$k = 0.3$		$k = 0.5$		$k = 0.7$			
	abs	*rel*	*abs*	*rel*	*abs*	*rel*	*abs*	*rel*
50%	87.59	0.65	27.95	0.29	-31.69	-0.55	5.60	0.05
60%	109.55	0.82	50.88	0.53	-7.79	-0.14	27.86	0.27
70%	127.05	0.95	70.34	0.73	13.62	0.24	50.26	0.49
80%	131.47	0.98	80.64	0.84	29.81	0.52	71.33	0.70
90%	128.26	0.96	85.35	0.89	42.44	0.74	88.32	0.86
100%	134.02	1.00	95.73	1.00	57.44	1.00	102.36	1.00
110%	116.74	0.87	88.09	0.92	59.44	1.03**	101.74	0.99
120%	102.75	0.77	81.40	0.85	60.05	1.05**	101.75	0.99
130%	79.60	0.59	65.89	0.69	52.18	0.91	88.34	0.86
140%	66.71	0.50	56.49	0.59	46.27	0.81	74.54	0.73
150%	60.19	0.45	52.22	0.55	44.24	0.77	69.17	0.68

Table 3.13: Utility for a single misreporting provider with 60 jobs and 20 nodes. *abs* denotes the mean absolute utility, *rel* the ratio of means of the utility when misreporting divided by the utility from truthful reporting. * denotes significance at the level of p = 0.1, ** at p = 0.05, and *** at p = 0.01.

erally be low. Consequently, providers have little to gain by trying to strategically influence this component of the payment scheme.

One might think that increased competition on the demand side of the market would give providers more strategic leeway by making their load less sensitive to their reported reserve price. However, the results in Table 3.13 paint a somewhat different picture. k-pricing essentially becomes *less* sensitive to underbidding. k-pricing is vulnerable to underbidding because the gain in the payment's surplus component may outweigh the loss in the reserve price component as the provider's load may increase with a lower reserve price. But with high competition, the provider's load (and thus the payment's surplus component) is less sensitive to the reported reserve price. While k-pricing becomes more vulnerable to overbidding, even for a large k ($= 0.7$) the potential gain is insignificant.

For a higher level of competition, proportional critical-value pricing no longer punishes under- and overbidding that severely. However, on average it remains thoroughly robust.

In summary, in settings with a low level of competition, k-pricing is vulnerable to underbidding from both users and providers if only a small portion of the generated welfare (i.e. k is small) is allocated to users. As formally shown in Theorem 4, critical-value pricing is truthful on the demand side of the market. Even though this implies fairly low prices for resource requests in settings with a low level of competition (and thus a smaller portion of welfare for resource providers), the proposed proportional amendment to critical-value pricing is robust to both under- and overbidding by resource providers.

3.6 Implications

Thus far there has only been scant research about how grid markets should be implemented in complex (realistic) settings. As Lai (2005) points out, "the attention of system designers, especially those designing scalable systems, should also be balanced with equal concern given to strategic behavior". In this chapter, the design of grid markets was tackled from both ends: *scalability* and *strategic behavior*.

The analysis has shown that exact mechanisms, which always optimally solve the allocation problem, are infeasible in practice. In contrast, the greedy heuristic is highly scalable. If employed in interval scheduling mode, Sun Microsystems's *N1GE* scheduler is typically executed every 15 seconds. It was found that, if complemented with k-pricing, the heuristic can clear up to 2,500 orders per side on average across 200 order books within this timeframe (12.7 seconds). If compared to the size of PlanetLab, the largest testbed for networking and distributed computing, which currently comprises 927 nodes at 452 sites,[11] these results underline the heuristic's applicability to practical scenarios. While this scalability comes at the expense of efficiency, it was also shown that the heuristic generates near-optimal allocations on average.

With respect to the strategic behavior of grid market participants, two alternative pricing schemes have been investigated. From a welfare perspective, k-pricing offers the nice feature that, given truthful information from all users, welfare can be distributed among users and providers according to the parameter *k*, which can act as an adjusting screw. The welfare implications of proportional critical-value pricing are more involved, as the distribution of welfare hinges on the competition on both sides of the market. In contrast to k-pricing, there is no direct adjusting screw for the market operator. However, via the critical values of job requests, proportional critical-value pricing implements a desirable economic dynamicity in the sense that prices and payments adequately reflect resource scarcity. If resources are not scarce, critical values will be low, and resource providers might just receive their reserve prices. With increasing scarcity, critical values converge to the users' maximum willingness to pay, and resource providers extract a larger fraction of welfare. Considering strategic users, proportional critical-value pricing is clearly advantageous in that users cannot profit from misreporting in any case and resource providers cannot benefit from overstating their reserve prices on average. With k-pricing, the freedom in choosing *k* becomes somewhat restricted. On the one hand, if *k* – and

[11]as of 16.11.2008, `http://www.planet-lab.org/`

thus the users' surplus – is small, both users and providers have a significant incentive to understate their true valuations. On the other hand, if k is large, and hence the providers' surplus is low, this reduces the incentive for resource providers to contribute their idle resources to the grid. With respect to computational considerations, k-pricing clearly outperforms proportional critical-value pricing.

In large-scale settings, k-pricing will generally be preferable to proportional critical-value pricing due to its computational simplicity. Moreover, as discussed above, as the market size increases, the potential gains from misreporting become smaller. The increasing competition assumes the pricing scheme's role of inducing users to engage in truthful behavior. In small-scale settings, the importance of computational considerations naturally diminishes, and in line with the results for increasing market sizes, strategic considerations gain in importance as the market size decreases. Proportional critical-value pricing may thus prove superior to k-pricing in these settings, as it not only ensures truthful reporting of users on average, but in dominant strategies.

In conclusion, a market mechanism was presented that approaches the issues of scalability and strategic behavior in an integrated manner. The results presented in this work also suggest that centralized, direct revelation mechanisms can successfully be implemented even in complex settings, which refutes the contrary statement made by Guo *et al.* (2007).

3.7 Discussion

In Section 3.2, the need to cope with several domain-specific and economic requirements was discussed when designing a market mechanism for computational grids. As summarized in Table 3.1, up to now there has been no mechanism fully capable of accounting for dependencies between multiple grid resources in large-scale settings with strategic users. Unfortunately, as shown by the Myerson-Satterthwaite impossibility theorem, it is not possible to design such a perfect mechanism that satisfies all requirements. However, the proposed greedy heuristic and the pricing schemes present a significant step forward compared to previous work in this field.

The analysis suffers from two main limitations. Since there is currently no data available for real grid workloads, especially with respect to user valuations, the analysis built on artificial workloads. The results will thus have to be confirmed in the future as new data becomes available. Another limitation is the assumption that a job can be migrated between nodes at zero cost. It would be interesting to weaken this assumption and limit the heuristic's flexibility in order to avoid excessive migrations. Moreover, a further analyses of the presented pricing schemes would be interesting. The numerical simulation only considered individual misreporting by single market participants. By means of experiments and agent-based simulations, the effect of *multiple* misreporting users who possibly form coalitions might be investigated. Extensions of the heuristic allocation scheme, such as the use of more sophisticated norms in the sorting phase, as well as the study of their impact on the mechanism's strategic properties, might be further promising areas for future research.

Chapter 4

Randomization and Distributed Outcome Determination

4.1 Introduction

In the previous chapter, a highly scalable heuristic was designed and complemented with a pricing scheme that yields truthful prices for users. Unfortunately, the *deterministic* greediness and simplicity of this heuristic can lead to allocation decisions far from optimal in special cases (cf. Theorem 3 in the previous chapter). The focus of this chapter is thus on advancing the allocation algorithm. One possible means for intercepting worst cases as the one in the proof to Theorem 3 in the previous chapter is to *randomize* the allocation heuristic. Then, with positive probability, the desirable high-volume job (i.e., with large vc) will be considered instead of the small job with a higher density-based ranking (i.e., v). However, when introducing random choices to the mechanism, the prices of jobs cannot be based on critical values anymore, since the allocation algorithm is non-deterministic and it is thus impossible to determine the critical values of jobs. Consequently, there is also the need for new pricing rules.

The contributions of this chapter are as follows:

- The heuristic is randomized so as to account for worst cases. While the (one-sided) truthfulness of the pricing scheme cannot be maintained, a randomization and a payment scheme are presented, which together induce users to bid “close” to their true valuations.

- It is shown how this randomized mechanism and the deterministic mechanism can be combined to a distributed randomized mechanism that aims at capturing the benefits of both mechanisms.

- This distributed randomized mechanism is evaluated with respect to strategic users and the generated efficiency.

4.2 Related Work

Randomization has recently been the subject of much interest in mechanism design, the main reason being that randomized mechanisms can help in obtaining better performance ratios than deterministic mechanisms (Nisan and Ronen, 2001). For Nisan and Ronen (2001), "a randomized mechanism is a probability distribution over a family [...] of mechanisms." I.e., a randomized mechanism is constructed by randomizing over deterministic mechanisms. While the single deterministic mechanisms may still perform poorly in special cases, randomizing over these deterministic mechanisms makes these special cases improbable (Goldberg *et al.*, 2006). If a randomized mechanism consists of a random "coin flip" over truthful deterministic mechanisms, it is still truthful in dominant strategies and thus *truthful in the universal sense* (Nisan and Ronen, 2001; Dobzinski *et al.*, 2006).

Goldberg *et al.* (2006) present two mechanisms that are based on *random sampling*. The key idea is to randomly partition the bidders in combinatorial auctions into distinct groups. One group is used for market / preference analysis, and what is learned from this group is then applied to set parameters in the other group(s), such as reserve prices. The objective of the presented mechanisms is to maximize the revenue of the provider of a "digital good"[1].

Dobzinski *et al.* (2006) present a similar framework for constructing randomized mechanisms for combinatorial auctions. The objective of the framework is to enable the design of mechanisms that are truthful in the universal sense and that provide a good approximation to the optimal (welfare maximizing) allocation. The main idea of the framework is again to randomly assign the bidders to distinct groups. The bidders in one of the groups do not get any items but are only used to determine appropriate (reserve) prices for the second-price and fixed-price auctions, respectively, within the other groups. Two specific mechanisms are constructed, each for a specific type of bidder valuation functions, which attain a good performance ratio in expectation. The main problem of this framework is that there is a relatively high probability that the solution is *not* close to the expected value. For example, the specific mechanism for general bidders provides an expected performance ratio (i.e., deviation from the optimal solution) of $O(\frac{\sqrt{m}}{\varepsilon^3})$ with probability $1-\varepsilon$, where $0 < \varepsilon < 1$ is the parameter for randomly assigning the bidders to the different groups and m is the number of items that are to be sold. In consequence, a good expected performance ratio (large ε) implies having to compromise on the probability

[1] In the terminology of Goldberg *et al.* (2006), a digital good is a good with unlimited supply where all instances of the good are identical.

with which this ratio is achieved.

Dobzinski (2007) extends this framework to enable the design of universally truthful mechanisms that provide a good performance ratio *with high probability*. Again, two specific mechanisms are constructed, one for the case of general bidders and one for the case of sub-additive bidders[2]. The first mechanism provides an $O(\sqrt{m})$ performance ratio with probability of at least $1-O(\frac{\log m}{\sqrt{m}})$. The second mechanism has the performance ratio $O(\log m \log(\log m))$ with probability of at least $1-O(\frac{1}{\log m})$.

Awerbuch *et al.* (2003) consider online optimization problems where jobs arrive over time, such as in network admission control and routing problems. The objective in this setting is to maximize the provider's revenue. A framework is presented that essentially piggybacks on some existing, not necessarily truthful online approximation algorithm B, where B is ρ-competitive in that the revenue of an (optimal) offline, omniscient algorithm (that has complete information about the jobs) is not more than ρ times the expected revenue produced by B.[3] The framework then combines B with the randomized admission / rejection and pricing of jobs so as to design a truthful mechanism that is $O(\rho + \log \mu)$-competitive, where μ is the ratio of the maximum job valuation to the minimum job valuation.

The work of Archer *et al.* (2003) is closest to the approach presented in this chapter. Archer *et al.* (2003) design a mechanism for combinatorial auctions with single-minded bidders[4]. The mechanism is based on the *randomized rounding* of solutions to linear programming relaxations of the well-known set packing problem in combinatorial auctions. In contrast to the other works, the mechanism is not truthful in the universal sense. Instead, the weaker solution concept of *truthfulness in expectation* is introduced, where truthful behavior only maximizes the bidders *expected* utility. Users are induced to bid truthfully by giving them a discount amounting to their aggregated probability of being allocated when bidding less. But the approach leaves two difficulties (Archer *et al.*, 2003): The discount cannot be efficiently computed and the mechanism might frequently have to give large discounts to the users, which might distort budget-balance.

In summary, the recent theoretical results show that randomization is a promising complement to the toolbox of market designers. However, these theoretical results are practically infeasible in the more realistic and complex scenario at hand. But randomization is too promising to give up on it yet, and one might be willing to sacrifice the strong constraint of truthfulness in the hope for an increase in allocative efficiency. In the next section, a fundamentally different approach

[2]Informally, a bidder is sub-additive if the valuation for a combination of items is equal to or less than the sum of the valuations for the single items.

[3]B may hence be deterministic or randomized.

[4]A bidder is single-minded if she desires *exactly* one subset of the items that are for sale in a combinatorial auction.

to randomizing the allocation algorithm and determining the prices will thus be explored.

4.3 A Randomized Pay-as-Bid Mechanism

A simplified job type $\theta_j = (v_j, c_j, m_j)$ with $v_j > 0$ is used where the attributes correspond to the attributes used in the previous chapter, except that time constraints are not considered for the ease of presentation. Accordingly, node n has the type $\theta_n = (v_n, C_n, M_n)$.

The rationale for randomizing the heuristic proposed in the previous chapter is to intercept worst cases as in the proof to Theorem 3. Consequently, the approach will be to randomly choose the jobs in the allocation phase of the heuristic with the probability being dependent on the jobs' reported valuations. Let $\pi_j(\tilde{\theta}) = \pi_j(\tilde{\theta}_j, \tilde{\theta}_{-j})$ be the probability for job j of being allocated given its reported type $\tilde{\theta}_j$ and the (reported) types $\tilde{\theta}_{-j}$ of all other jobs *and* nodes. Users are assumed to truthfully report their jobs' resource constraints c_j and m_j but to possibly misreport about v_j, that is $\tilde{\theta}_j = (\tilde{v}_j, c_j, m_j)$. Resource providers are assumed to be obedient (i.e., $\tilde{\theta}_n = \theta_n$).

In this section, a so-called "pay-as-bid" mechanism is presented in which $p_j(\tilde{\theta}) = \tilde{v}_j c_j$ for each allocated job j, and $p_j(\tilde{\theta}) = 0$ else. Obviously, this payment scheme cannot be truthful. Regardless of how the allocation algorithm is randomized, users can only hope for a positive utility if they understate their true valuation. However, one can try to induce users to bid "close" to their true valuations. Similar to the well-known monotonicity condition for truthful deterministic mechanisms (cf. Lehmann *et al.* (2002); Mu'alem and Nisan (2008)), every truthful randomized mechanism must be monotone (Archer and Tardos, 2001; Archer *et al.*, 2003):

Definition 1 (Monotonicity). *A randomized mechanism is* monotone *if, for any j, c_j, m_j, and $\tilde{\theta}_{-j}$,*

$$\pi_j((\hat{v}_j, c_j, m_j), \tilde{\theta}_{-j}) \geq \pi_j((\tilde{v}_j, c_j, m_j), \tilde{\theta}_{-j}),\ \hat{v}_j \geq \tilde{v}_j.$$

The intuition behind this is that if the allocation mechanism was not monotone, the user could increase her job's chance of being allocated by reporting a lower valuation. Besides ensuring monotonicity, in order to induce users to report a valuation close to their true valuation, the underlying idea when randomizing the allocation algorithm is to try to give high bidders a "discount" such that a high bidder has a disproportionately higher probability of her job getting allocated than a user with a low reported valuation.

4.3.1 The Mechanism

The allocation algorithm of the heuristic presented in the previous chapter is randomized as follows. Instead of having a strict ordering when considering the jobs in the allocation phase, in each step the job that is considered for allocation is determined randomly. Let $\Delta_k = (j_1, j_2, \dots, j_k)$, $j_l \neq j_m$ for $l \neq m$, be the *sequence* of jobs that have been considered in the first k allocation decisions of the heuristic, where job j_l has been considered in l-th place. Let $\mathbb{1}_j(\tilde{\theta}, \Delta_k) = 1$ if job j can be allocated after the job sequence Δ_k has been allocated, and $\mathbb{1}_j(\tilde{\theta}, \Delta_k) = 0$ else. This indicator function contains all restrictions on the possible allocations that are due to capacity constraints and reserve prices. Then, in step k and given the prior allocation decisions Δ_{k-1} of the heuristic, job j is allocated with probability

$$\varphi_{j,\Delta_{k-1}}(\tilde{\theta}) = \begin{cases} \frac{\mathbb{1}_j(\tilde{\theta},\Delta_{k-1})\cdot\tilde{v}_j^{\alpha}}{\sum_{i\in J\setminus\Delta_{k-1}} \mathbb{1}_j(\tilde{\theta},\Delta_{k-1})\cdot\tilde{v}_i^{\alpha}}, \; \alpha \geq 1 & \text{if } \mathbb{1}_j(\tilde{\theta},\Delta_{k-1}) = 1 \\ 0 & \text{else.} \end{cases} \tag{4.1}$$

The pseudo-code of this allocation algorithm is given in Appendix B.1. The intuition behind this randomization is to determine the probability of choosing a specific job based on this job's reported valuation relative to the reported valuations of the other jobs. α is an adjusting screw that has to be set by the market designer to determine to what extent a high valuation job should be preferred over a job a with low valuation. The idea of a large α is to induce users to bid close to their true valuation by boosting their allocation probability. This will be analyzed in detail below.

The probability $\pi_{j1}(\tilde{\theta})$ that job j is allocated in step 1 is $\pi_{j1}(\tilde{\theta}) = \varphi_{j,\emptyset}(\tilde{\theta})$.[5] For step k, $1 < k \leq |J|$, the probability is

$$\pi_{jk}(\tilde{\theta}) = \sum_{\substack{\Delta_{k-1}=(j_1,\dots,j_{k-1}) \\ j_m\neq j,\, m=1,\dots,k-1}} \left(\prod_{l=1}^{k-1} \varphi_{j_l,(j_1,\dots,j_{l-1})}(\tilde{\theta}) \right) \cdot \varphi_{j,\Delta_{k-1}}(\tilde{\theta}).$$

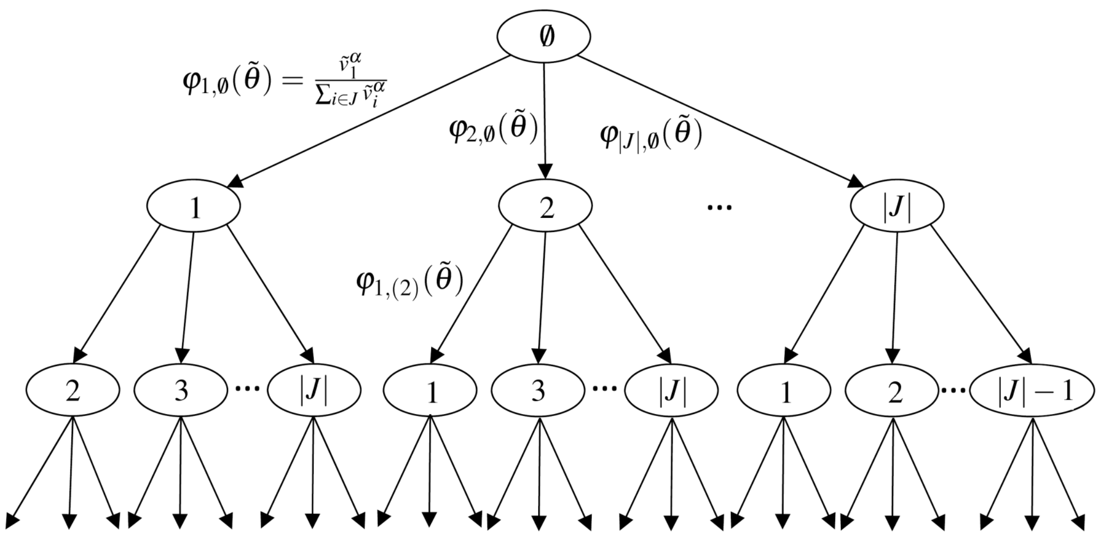
This probability equals the sum over the probabilities of all possible allocation sequences until step k times the probability that job j can be allocated after a given allocation sequence.

Overall, job j is allocated with probability $\pi_j(\tilde{\theta}) = \sum_{k=1}^{|J|} \pi_{jk}(\tilde{\theta})$.

The randomized allocation algorithm spans out an "allocation tree" that illustrates the various allocation sequences, the allocation probabilities and their dependencies, cf. Figure 4.1. Each node in this allocation tree represents a job that has been considered in the last step of the allocation heuristic. The vertex to a subsequent node indicates that the end job of that vertex is considered next. Consequently, at a given decision point (node in the allocation tree), the heuristic chooses a specific vertex with the probability specified by Equation 4.1.[6] This randomized process is now illustrated with a sample scenario.

[5]It can reasonably be assumed that $\mathbb{1}_j(\tilde{\theta}, \emptyset) = 1$, since otherwise the job would not be considered in the allocation phase.

[6]For simplification, a vertex can be omitted if this probability is zero, i.e., if there is not sufficient capacity left to accommodate the end node of the vertex.



$\emptyset$
$\varphi_{1,\emptyset}(\tilde{\theta}) = \frac{\tilde{v}_1^\alpha}{\sum_{i\in J}\tilde{v}_i^\alpha}$
$\varphi_{2,\emptyset}(\tilde{\theta})$
$\varphi_{|J|,\emptyset}(\tilde{\theta})$
1
2
$|J|$
$\varphi_{1,(2)}(\tilde{\theta})$
2
3
$|J|$
1
3
$|J|$
1
2
$|J|-1$

Figure 4.1: Allocation tree.

Example: Assume the job requests and node offers listed in Table 4.1 have been submitted to the market:

Job j	$\tilde{v}_j$	c_j	m_j	Node n	v_n	C_n	M_n
j_1	10	1	100	n_1	1	2	500
j_2	8	1	500	n_2	2	1	600
j_3	7	2	400				

Table 4.1: Sample scenario.

The node offers are ranked in non-decreasing order of their reserve prices (costs) as in the deterministic heuristic. Recall that these valuations are expressed per unit of computing power. Job j_1 is willing to pay up to \$10 for its one required CPU and 100 MB of memory. Node n_1 requests at least \$1 for each of its two CPUs and offers up to 500 MB of memory. The optimal allocation is $j_1 \to n_2$ and $j_3 \to n_1$, yielding welfare of \$20. The deterministic heuristic, however, allocates $j_1 \to n_1$ and $j_2 \to n_2$ with welfare of \$15 only.

As introduced above, the randomized heuristic does not depend on this deterministic ranking of job requests but, in each step k and given a preceding allocation sequence Δ_{k-1}, randomly chooses the job according to $\varphi_{j,\Delta_{k-1}}(\tilde{\theta})$. The chosen job is then greedily assigned to the cheapest possible offer.

Assume $\alpha = 2$. Then, in the first step, job j_1 is chosen (and thus allocated to node n_1) with probability $\pi_{j_1,1}(\tilde{\theta}) = \varphi_{j_1,\emptyset}(\tilde{\theta}) = \frac{10^2}{10^2+8^2+7^2} \approx 0.47$. The probabilities $\pi_{j_2,1}(\tilde{\theta})$ and $\pi_{j_3,1}(\tilde{\theta})$ are determined analogously.

For the second step, note that j_3 cannot be allocated to n_2 because this node does not have sufficient computing power. Thus $\pi_{j_3,2}(\tilde{\theta}) = 0$. The probability that j_1 is allocated to n_2 (in the second step) is $\pi_{j_1,2}(\tilde{\theta}) = \pi_{j_2,1}(\tilde{\theta}) \cdot 1 + \pi_{j_3,1}(\tilde{\theta}) \cdot \frac{10^2}{10^2+8^2} \approx 0.44$, and $\pi_{j_2,2}(\tilde{\theta}) = \pi_{j_1,1}(\tilde{\theta}) \cdot 1 + \pi_{j_3,1}(\tilde{\theta}) \cdot \frac{8^2}{10^2+8^2} \approx 0.56$. In summary, $\pi_1(\tilde{\theta}) = \pi_{j_1,1}(\tilde{\theta}) + \pi_{j_2,1}(\tilde{\theta}) \approx 0.91$, $\pi_2(\tilde{\theta}) \approx 0.86$ and $\pi_3(\tilde{\theta}) \approx 0.23$. The allocation tree for this sample scenario is depicted in Figure 4.2.

Job j	π_{j1}	π_{j2}	π_j
j_1	0.47	0.44	0.91
j_2	0.30	0.56	0.86
j_3	0.23	0	0.23

Table 4.2: Sample allocation probabilities.

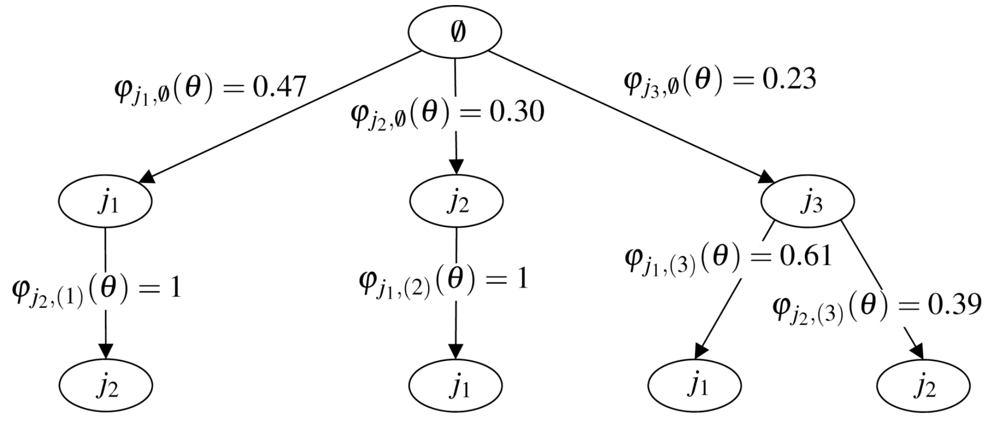

$\emptyset$
$\varphi_{j_1,\emptyset}(\theta) = 0.47$
$\varphi_{j_2,\emptyset}(\theta) = 0.30$
$\varphi_{j_3,\emptyset}(\theta) = 0.23$
j_1
j_2
j_3
$\varphi_{j_2,(1)}(\theta) = 1$
$\varphi_{j_1,(2)}(\theta) = 1$
$\varphi_{j_1,(3)}(\theta) = 0.61$
$\varphi_{j_2,(3)}(\theta) = 0.39$
j_2
j_1
j_1
j_2

Figure 4.2: Sample allocation tree.

The expected welfare generated by the randomized heuristic can now be computed as $0.47 \cdot \$15 + 0.30 \cdot \$15 + 0.23 \cdot 0.61 \cdot \$20 + 0.23 \cdot 0.39 \cdot \$18 \approx \$15.97$. Consequently, in expectation the randomized heuristic yields a higher welfare than the deterministic heuristic (\$15, see above).

As this example shows, the randomized heuristic can outperform the deterministic heuristic on expectation. However, in contrast to the discussion of the related work, it is hard to give a general bound for such complex scenarios. If the expected welfare of the randomized heuristic is higher than the welfare generated by the deterministic heuristic, and with which probability a gain is actually achieved, depend on the specific scenario. This efficiency dimension of the design problem at hand will be evaluated in more detail in Section 4.5. In the remainder of this section, the focus will be on the strategic properties and implications of this randomized pay-as-bid mechanism.

4.3.2 Monotone and Strongly Convex Allocation Probabilities

As discussed above, it is crucial for a randomized mechanism to yield monotone allocation probabilities, which are a first step towards inducing users to bid "close" to their true valuations:

Theorem 1 (Monotonicity of the randomized mechanism). *The randomized pay-as-bid mechanism is monotone.*

Proof. $\frac{\partial \varphi_{j,\Delta_{k-1}}(\tilde{\theta})}{\partial \tilde{v}_j} \geq 0$ for all Δ_{k-1} and $\tilde{\theta}_{-j}$ if $\mathbb{1}_j(\tilde{\theta}, \Delta_{k-1}) = 1$.

This directly implies that $\pi_{jk}(\tilde{\theta})$ is monotone in $\tilde{v}_j$. However, $\pi_{jk}(\tilde{\theta})$ and $\pi_{jl}(\tilde{\theta})$, $l > k$, are generally not independent. It remains to be shown that an increase in $\pi_{jk}(\tilde{\theta})$ outweighs a resulting decrease in $\sum_{l=k+1}^{|J|} \pi_{jl}(\tilde{\theta})$.

Consider the sub-tree of the allocation tree following an allocation sequence Δ_{k-1}. In each such sub-tree, the heuristic makes an allocation decision:

$$\sum_{i \in J \setminus \Delta_{k-1}} \varphi_{i,\Delta_{k-1}}(\tilde{\theta}) = 1.$$

Consequently, if $\varphi_{j,\Delta_{k-1}}(\tilde{\theta})$ increases by $\varepsilon > 0$, $\sum_{\substack{i \in J \setminus \Delta_{k-1} \\ i \neq j}} \varphi_{i,\Delta_{k-1}}(\tilde{\theta})$ decreases by ε. That is, if the probability of choosing job j increases, the probability of choosing a job other than j decreases by the same amount. Hence, there is a direct, positive effect of reporting a higher valuation in the form of a higher probability of being allocated early, and an indirect, negative effect since this inherently lowers the probability of being allocated later. However, the overall effect is non-negative, since

$$\sum_{\substack{i \in J \setminus \Delta_{k-1} \\ i \neq j}} \varphi_{i,\Delta_{k-1}}(\tilde{\theta}) \geq \sum_{\substack{i \in J \setminus \Delta_{k-1} \\ i \neq j}} \varphi_{i,\Delta_{k-1}}(\tilde{\theta}) \varphi_{j,(i_1,\ldots,i_{k-1},i)}(\tilde{\theta}),$$

and thus the negative effect is no more than ε.

Since this effect holds for all sequences Δ_{k-1} (i.e., all possible scenarios that lead to step k, and in particular also for $k = 1$), the increase in $\pi_{jk}(\tilde{\theta})$ outweighs the possible decrease in $\sum_{l=k+1}^{|J|} \pi_{jl}(\tilde{\theta})$.

Consequently, $\pi_j(\tilde{\theta}) = \sum_{k=1}^{|J|} \pi_{jk}(\tilde{\theta})$ increases monotonically in $\tilde{v}_j$. □

Besides monotonicity, in order to further induce users to bid close to their true valuation, it is desirable to have *strongly convex allocation probabilities*, that is the increase in the allocation probability is disproportionately high compared to the increase in the reported valuation. This is inherently connected to the choice of α and the competition in the market:

Corollary 1 (Strong convexity). *With the randomized pay-as-bid mechanism, the allocation probability $\varphi_{j,\Delta_{k-1}}(\tilde{\theta})$ is strongly convex in $\tilde{v}_j$ if $\alpha > 1$, $\mathbb{1}_j(\tilde{\theta},\Delta_{k-1}) = 1$, $J \setminus \{\Delta_{k-1} \cup j\} \neq \emptyset$, and*

$$\tilde{v}_j < \sqrt[\alpha]{\frac{\alpha-1}{\alpha+1} \sum_{i \in J \setminus \{\Delta_{k-1} \cup j\}} \tilde{v}_i^{\alpha}}. \tag{4.2}$$

Proof. With $J \setminus \{\Delta_{k-1} \cup j\} \neq \emptyset$, $\alpha > 1$, and $\mathbb{1}_j(\tilde{\theta},\Delta_{k-1}) = 1$, it is straightforward to see that $\frac{\partial \varphi_{j,\Delta_{k-1}}(\tilde{\theta})}{\partial \tilde{v}_j} > 0$.

Moreover,

$$\frac{\partial^2 \varphi_{j,\Delta_{k-1}}(\tilde{\theta})}{\partial \tilde{v}_j^2} = \alpha \tilde{v}_j^{\alpha-2} \left(\sum_{i \in J \setminus \Delta_{k-1}} \tilde{v}_i^{\alpha} \right) \left(\sum_{\substack{i \in J \setminus \Delta_{k-1} \\ i \neq j}} \tilde{v}_i^{\alpha} \right) \frac{(\alpha-1)\left(\sum_{i \in J \setminus \Delta_{k-1}} \tilde{v}_i^{\alpha} \right) - 2\alpha \tilde{v}_j^{\alpha}}{\left(\sum_{i \in J \setminus \Delta_{k-1}} \tilde{v}_i^{\alpha} \right)^4} > 0$$

$$\Leftrightarrow \quad (\alpha-1)\left(\sum_{i \in J \setminus \Delta_{k-1}} \tilde{v}_i^{\alpha} \right) - 2\alpha \tilde{v}_j^{\alpha} > 0$$

$$\Leftrightarrow \quad (\alpha-1)\left(\sum_{\substack{i \in J \setminus \Delta_{k-1} \\ i \neq j}} \tilde{v}_i^{\alpha} \right) - (\alpha+1)\tilde{v}_j^{\alpha} > 0$$

$$\Leftrightarrow \quad \tilde{v}_j < \sqrt[\alpha]{\frac{\alpha-1}{\alpha+1} \sum_{\substack{i \in J \setminus \Delta_{k-1} \\ i \neq j}} \tilde{v}_i^{\alpha}}.$$

□

Consequently, at each decision point Δ_{k-1} of the heuristic (i.e., decision node in the allocation tree), the allocation probability is strongly convex in $\tilde{v}_j$ given that there is at least one other job left that is competing with j and that $\tilde{v}_j$ is within the interval that is upper-bounded by Equation 4.2.

There are two key implications of this result: (i) The user's incentive to bid close to the true valuation increases with increasing competition and (ii) there is no general "optimal" choice of α that maximizes the upper bound of Equation 4.2.

The first implication is obvious; $\sum_{\substack{i \in J \setminus \Delta_{k-1} \\ i \neq j}} \tilde{v}_i^{\alpha}$ increases as the set of jobs whose reported valuations are summed up increases. The second implication is exacerbated by the fact that the optimal α is different for each decision point in the randomized heuristic's allocation tree. It can be stated, however, that the upper bound of the interval specified by Equation 4.2 converges towards the maximum reported valuation of all competing jobs as α approaches ∞:

Corollary 2 (Limit of Equation 4.2).

$$\lim_{\alpha\to\infty} \sqrt[\alpha]{\frac{\alpha-1}{\alpha+1} \sum_{i\in J\setminus\{\Delta_{k-1}\cup j\}} \tilde{v}_i^{\alpha}} = \max_{i\in J\setminus\{\Delta_{k-1}\cup j\}} \tilde{v}_i.$$

Proof. The overall limit can be partitioned as

$$\lim_{\alpha\to\infty} \sqrt[\alpha]{\frac{\alpha-1}{\alpha+1} \sum_{\substack{i\in J\setminus\Delta_{k-1}\\ i\neq j}} \tilde{v}_i^{\alpha}} = \lim_{\alpha\to\infty} \sqrt[\alpha]{\frac{\alpha-1}{\alpha+1}} \cdot \lim_{\alpha\to\infty} \sqrt[\alpha]{\sum_{\substack{i\in J\setminus\Delta_{k-1}\\ i\neq j}} \tilde{v}_i^{\alpha}}.$$

For the first limit, one gets $\lim\limits_{\alpha\to\infty} \sqrt[\alpha]{\frac{\alpha-1}{\alpha+1}} = \lim\limits_{\alpha\to\infty} e^{\frac{1}{\alpha}\ln\frac{\alpha-1}{\alpha+1}} = e^0 = 1$.

The latter limit is both lower- and upper-bounded by $\tilde{v}_{\max} = \max\limits_{\substack{i\in J\setminus\Delta_{k-1}\\ i\neq j}} \tilde{v}_i$:

- $\sqrt[\alpha]{\sum_{\substack{i\in J\setminus\Delta_{k-1}\\ i\neq j}} \tilde{v}_i^{\alpha}} \leq \sqrt[\alpha]{n\cdot\tilde{v}_{\max}^{\alpha}} = \sqrt[\alpha]{n}\cdot\tilde{v}_{\max} \to \tilde{v}_{\max}$ for $\alpha\to\infty$ and n being the number of jobs over which to sum up.
- $\sqrt[\alpha]{\sum_{\substack{i\in J\setminus\Delta_{k-1}\\ i\neq j}} \tilde{v}_i^{\alpha}} \geq \sqrt[\alpha]{\tilde{v}_{\max}^{\alpha}} = \tilde{v}_{\max}$.

Consequently, $\lim\limits_{\alpha\to\infty} \sqrt[\alpha]{\sum_{\substack{i\in J\setminus\Delta_{k-1}\\ i\neq j}} \tilde{v}_i^{\alpha}} = \tilde{v}_{\max}$ and $\lim\limits_{\alpha\to\infty} \sqrt[\alpha]{\frac{\alpha-1}{\alpha+1}\sum_{\substack{i\in J\setminus\Delta_{k-1}\\ i\neq j}} \tilde{v}_i^{\alpha}} = 1\cdot\tilde{v}_{\max}$. □

This further supports the first implication, as this maximum valuation will generally increase as the number of jobs increases over which to maximize.

The complexity of the setting with multiple jobs, multiple nodes, and varying capacity constraints precludes a further analytic evaluation, since the allocation probabilities π_j are hard to derive. In the following subsection, a simplified setting will be analyzed in order to exemplify the effect of the parameter α on the strategies of the users, the expected utilities, and the expected overall welfare.

4.3.3 Evaluation in a Simplified Setting

Consider one node only, w.l.o.g. having a reserve price of zero. There are two users who are competing for this node, meaning only the job of one of them can be allocated. Obviously, in this scenario $\pi_j(\tilde{\theta}) = \pi_{j1}(\tilde{\theta}) = \varphi_{j,\emptyset}(\tilde{\theta})$, i.e. the chance of being allocated equals the chance of being considered first.

The solution concept of Nash equilibria is chosen to analyze the incentive of users to deviate from their true valuations (Nash, 1951; Mas-Colell *et al.*, 1995). However, for a randomized allocation algorithm, this concept has to be adapted similar to the adaptation of the solution concept of truthfulness to truthfulness in expectation that was briefly discussed in Section 4.2:

Definition 2 (Nash equilibrium in expectation). *The vector* $\tilde{\theta}^{NE}$ *constitutes a* Nash equilibrium in expectation *if*

$$E(u_j(\tilde{\theta}_j^{NE}, \tilde{\theta}_{-j}^{NE} | \theta_j)) \geq E(u_j(\tilde{\theta}_j, \tilde{\theta}_{-j}^{NE} | \theta_j)), \; j \in J, \; \tilde{\theta}_j,$$

where $E(u_j(.))$ *denotes j's expected utility.*

Given a randomized allocation algorithm and users with complete information about the other reported job and node types $\tilde{\theta}_{-j}^{NE}$ and the general allocation process, no user can benefit *in expectation* by unilaterally deviating from $\tilde{\theta}_j^{NE}$. Note that Nash equilibria in expectation are distinctively different from Bayesian-Nash equilibria (Mas-Colell *et al.*, 1995; Parkes, 2001). In the latter, the participants face incomplete information about the types of the other participants. In contrast, with Nash equilibria in expectation, participants face incomplete information about the non-deterministic outcome of the mechanism as such.

Nash equilibria can be interpreted as the final outcome of a fictive, iterative process. After each stage, the users can adjust their reported types based on feedback about the other users' reported types. This process is not a repeated game in the game-theoretic sense, but users play a myopic best response strategy for this single allocation phase only until they arrive at an equilibrium point (Mas-Colell *et al.*, 1995; Sanghavi and Hajek, 2004).

For the simplified setting at hand, the following lemmata about these best responses can be stated:

Lemma 1 (Best response for $\alpha = 1$). *In this scenario and for* $\alpha = 1$*, the best response of user j to the bid of user i is given by*

$$b_j(\tilde{v}_i | v_j) = -\tilde{v}_i + \sqrt{\tilde{v}_i^2 + \tilde{v}_i v_j}.$$

Lemma 2 (Best response for $\alpha = 2$). *In this scenario and for* $\alpha = 2$*, the best response of user j to the bid of user i is given by*

$$b_j(\tilde{v}_i | v_j) = \sqrt[3]{\sqrt{\tilde{v}_i^4 v_j^2 + \tilde{v}_i^6} + \tilde{v}_i^2 v_j} - \sqrt[3]{\sqrt{\tilde{v}_i^4 v_j^2 + \tilde{v}_i^6} - \tilde{v}_i^2 v_j}.$$

The proofs are presented in Appendix B.2. With the best response functions given by Lemma 1 and Lemma 2, the Nash equilibria can now be computed as the intersections of the best response functions of the two users.

Example: Assume two unit jobs ($c_1 = c_2 = 1$). The first user has a true valuation of $v_1 = 5$ and reports the valuation $\tilde{v}_1$, $0 < \tilde{v}_1 \leq 5$ (recall that $p_1(5, \tilde{v}_2) = 5$). The second user values her job at $v_2 = 7$ and reports the valuation $\tilde{v}_2$, $0 < \tilde{v}_2 \leq 7$. The best response functions of both users for $\alpha = 1$ and $\alpha = 2$ are plotted in Figure 4.3.

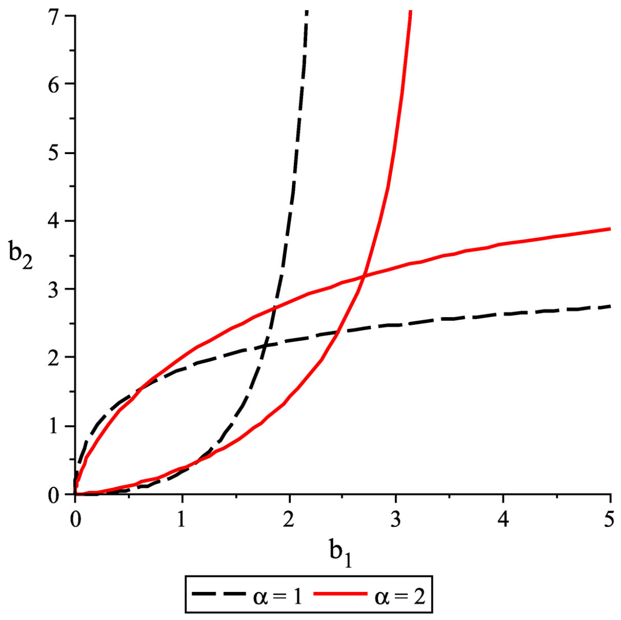


Figure 4.3: Sample best response functions for two competing and mutually exclusive users.

At the end of the imaginary iterative process, the two users have coordinated their bids to the intersections of their respective best response functions. At these intersections, no user can improve her expected utility by unilaterally deviating from her best response. Consequently, two Nash equilibria in expectation are obtained at these intersections (one per α): $\tilde{v}^{NE}_{\alpha=1} \approx (1.7748, 2.1716)$ and $\tilde{v}^{NE}_{\alpha=2} \approx (2.6917, 3.1829)$.

	$\alpha = 1$	$\alpha = 2$
$\tilde{v}^{NE}$	(1.7748, 2.1716)	(2.6917, 3.1829)
$\pi_j(\tilde{v}^{NE})$	(0.4497, 0.5503)	(0.417, 0.583)
$E(u_j(\tilde{v}^{NE}\|v))$	(1.4504, 2.6571)	(0.9626, 2.2254)
$\tilde{v}^{NE}_j / \pi_j(\tilde{v}^{NE})$	(3.94, 3.94)	(6.45, 5.45)

Table 4.3: Numerical example.

Table 4.3 lists the basic results for this sample scenario. Increasing α pushes both users towards reporting a higher valuation, thus inherently increasing their payments. This push also increases overall welfare in expectation. For $\alpha = 1$, the expected welfare is $\pi_1(\tilde{v}^{NE})v_1 + \pi_2(\tilde{v}^{NE})v_2 \approx 6.1$, whereas expected revenue is $\pi_1(\tilde{v}^{NE})\tilde{v}_1^{NE} + \pi_2(\tilde{v}^{NE})\tilde{v}_2^{NE} \approx 1.99$. For $\alpha = 2$, expected welfare and revenue increase to 6.17 and 2.98, respectively.

As pointed out earlier, one basic rationale in designing the mechanism is to give high bidders a discount in the form of disproportionately higher allocation probabilities. One metric for this is $\tilde{v}_j^{NE}/\pi_j(\tilde{v}^{NE})$ in the last row of Table 4.3, which can be interpreted as price per "unit" of allocation probability. For $\alpha = 1$, both users have to pay the same unit price. But for $\alpha = 2$, the high bidder has to pay less than the low bidder (5.45 respectively 6.45).

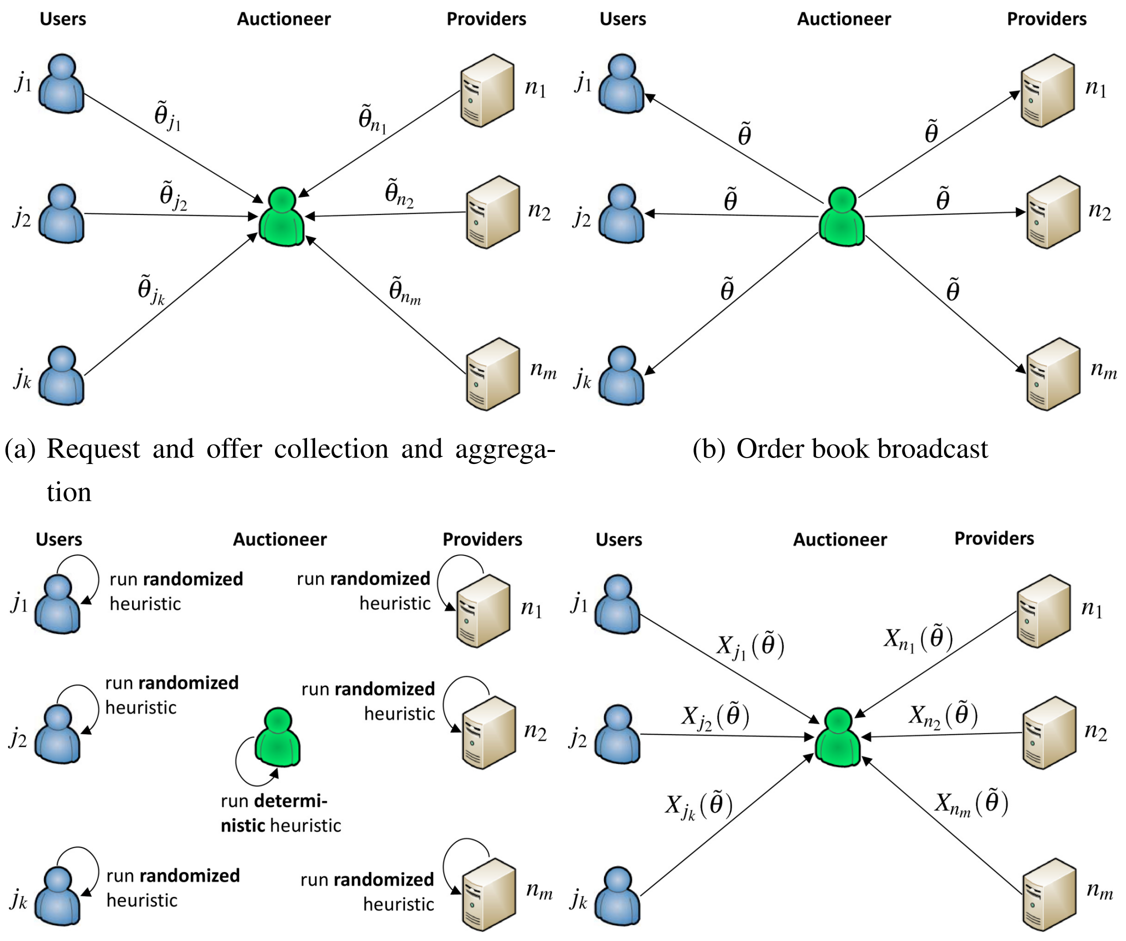
4.4 Distributed Problem Solving

As the previous results show, the parameter α plays an important role in trading off the influence of the randomization in intercepting worst cases on the one hand, and in distorting the incentives to bid close to truthfully on the other hand. It is, however, a non-trivial task to choose an "appropriate" α. As discussed above, one implication of Corollary 1 is that it is difficult to choose the α that maximizes the interval within which the job's probability of being allocated increases disproportionately to the user's reported valuation. Furthermore, from an allocation point of view, for $\alpha \to \infty$ the randomized allocation algorithm "converges" towards the deterministic heuristic again, since the jobs essentially become ordered by their reported valuations. In contrast, the smaller α, the larger the deviation from the deterministic algorithm and its vulnerabilities.

While it seems difficult to determine the best choice of the parameter α, this can be partly mitigated by running the randomized mechanism multiple times instead of just once, thereby possibly changing α between the runs. On the one hand, the scalability of the (randomized) heuristic easily allows for multiple runs of the centralized market. The auctioneer can then combine the results of the multiple runs and determine the allocation that spends the maximum welfare. On the other hand, the auctioneer might exploit the distributed nature that is inherent to any market and require the market participants to help in solving the allocation problem as kind of a "participation fee". This approach is depicted in Figure 4.4. First, the auctioneer collects all job requests and node offers. Then the consolidated order book is distributed (possibly in some anonymized way[7]) to all users and providers. Each market participant is required to run the randomized heuristic with some specific α and returns the resulting allocation to the auctioneer. Finally, the auctioneer again combines all allocations, determines the allocation that spends the

[7]For instance, the reported valuations and capacity constraints can be scaled, each parameter with a different scalar, to hinder the identification of one's own job or node. However, manipulation in the sense of trying to allocate one's own job or node might not actually be a problem after all, since the outcome is only enforced if it outperforms the other local solutions, which is a desirable overall outcome.

maximum welfare as well as the corresponding prices and payments, which are then enforced. This further allows the auctioneer to perform some sort of parameter sweep for α.


Users
Auctioneer
Providers
j_1
j_2
j_k
n_1
n_2
n_m
$\tilde{\theta}_{j_1}$
$\tilde{\theta}_{j_2}$
$\tilde{\theta}_{j_k}$
$\tilde{\theta}_{n_1}$
$\tilde{\theta}_{n_2}$
$\tilde{\theta}_{n_m}$
$\tilde{\theta}$
run **randomized** heuristic
run **determi-nistic** heuristic
$X_{j_1}(\tilde{\theta})$
$X_{j_2}(\tilde{\theta})$
$X_{j_k}(\tilde{\theta})$
$X_{n_1}(\tilde{\theta})$
$X_{n_2}(\tilde{\theta})$
$X_{n_m}(\tilde{\theta})$

(a) Request and offer collection and aggregation

(b) Order book broadcast

(c) Local solving

(d) Collection of the local solutions

Figure 4.4: Distributed randomized mechanism. The central auctioneer collects all reported job and node types and broadcasts the order book. Then the market participants run the randomized heuristic, the auctioneer runs the deterministic heuristic. Finally, the auctioneer collects the local solutions and enforces the outcome among all randomized solutions and the deterministic solution that generates the maximum welfare.

But this approach is not limited to the randomized heuristic. In many cases the deterministic heuristic clearly shows its strengths. An intuitive approach is thus to combine the randomized and the deterministic heuristic in the spirit of the *Approx-MUA* algorithm in Mu'alem and Nisan (2008). The basic idea is to run both the deterministic and the randomized heuristic on the same problem instances and then to actually enforce the allocation and prices of the mechanism that returned the maximum welfare. Thus, one obtains what may be called a "best-of-breed" *distributed randomized mechanism* by combining the mechanism design worlds of determinism and randomization. Clearly, the impact on the strategic properties is more involved. On the one hand, the desirable truthfulness of the deterministic mechanism is lost. On the other hand, coupling the deterministic with the randomized heuristic may strengthen the properties of the latter, since deviating from true valuations may prove unfavorable when the former is selected and enforced. In order to gain further insights into these issues, the following section will report about the results of a numerical simulation.

4.5 Numerical Simulations

The aim of this section is to evaluate the distributed randomized mechanism in more complex scenarios with respect to efficiency and its robustness against strategic behavior from users that misreport their valuations. The analysis in the previous chapter showed that the level of competition (i.e., the number of jobs and nodes as well as the ratio of jobs to nodes) has an important influence on these performance dimensions. Hence, in the following analysis both the choice of α as well as the level of competition will be varied.

4.5.1 Strategic Behavior

With the pay-as-bid mechanism, users can only obtain a positive utility from understating their true valuations. To go against this, the previous analysis showed that a larger α generally results in a stronger incentive to bid at least close to the true valuation.[8] Another implication is that this incentive clearly increases as the level of competition increases.

The following simulation is based on the same order books as the simulation in Subsection 3.5.2 in the previous chapter to allow for a comparison of the results. Hence, the randomized mechanism is employed in a more complex scenario as in the analysis above, now also including time constraints. The simulation consists of two settings, a setting with 20 jobs and 20 nodes and a setting with 60 jobs and 20 nodes. As in the previous chapter, the simulation is based on 200 order books per setting. Recall that, in the distributed randomized mechanism, the randomized mechanism is run once for each job and node. Additionally, the deterministic heuristic is run once. Finally, the best outcome with respect to efficiency is enforced. α is varied from 1, to 10, 30, 50, 100, and 200. The aim of this subsection is to analyze to what extent a single job request (user) can benefit from reporting an untrue valuation: In the simulation, the user of j hence reported 50%, 55%, up to 100% of her true valuation v_j.

[8]As pointed out above, this relation does not always hold since the right term of Equation 4.2 is not (strictly) monotone for any combination of $\tilde{v}_i$'s.

Bid	$\alpha = 1$			$\alpha = 10$			$\alpha = 30$			$\alpha = 50$			$\alpha = 100$			$\alpha = 200$		
	util	*succ*	*rand*	*util*	*succ*	*rand*	*util*	*succ*	*rand*	*util*	*succ*	*rand*	*util*	*succ*	*rand*	*util*	*succ*	*rand*
50%	22.09	41	194	12.69	25	200	10.51	21	197	12.12	25	193	8.35	19	180	9.14	21	175
55%	27.96	58	192	24.41	52	198	21.07	45	197	21.17	45	191	21.29	46	176	20.91	45	175
60%	36.81	88	196	34.21	81	200	34.81	84	197	32.51	79	194	32.37	78	178	32.06	76	171
65%	38.29	108	194	34.75	97	200	35.16	97	197	34.60	97	195	34.50	97	178	34.67	96	175
70%	36.22	124	195	34.42	116	199	33.23	111	198	33.45	113	195	34.08	112	178	34.56	112	175
75%	33.92	140	194	31.58	130	197	31.73	131	198	31.39	126	192	32.04	127	178	32.32	127	177
80%	28.62	152	193	26.92	143	197	26.30	138	199	27.18	138	192	28.93	141	176	29.09	141	175
85%	23.45	165	196	22.41	159	199	21.67	154	198	21.90	152	194	24.95	154	178	25.78	154	173
90%	16.50	175	195	15.59	168	199	16.03	167	198	16.19	165	192	18.97	163	177	19.73	163	173
95%	9.76	181	194	8.21	177	199	8.70	173	196	9.41	174	193	12.33	173	177	12.96	172	175
100%	1.09	191	197	0.18	187	199	0.54	186	198	2.18	186	194	5.85	187	180	7.18	187	179

Table 4.4: Simulation results for a misreporting user in the setting with 20 jobs and 20 nodes. *util* represents the mean utility of the misreporting user, *succ* the number of runs (out of 200) in which the job of the misreporting user was allocated, and *rand* the number of runs in which the distributed randomized mechanism enforced the outcome of the randomized heuristic (and not the deterministic heuristic). Note that the user can obtain a positive utility from truthfully reporting her valuation in case the outcome of the deterministic mechanism is enforced.

Bid	$\alpha = 1$			$\alpha = 10$			$\alpha = 30$			$\alpha = 50$			$\alpha = 100$			$\alpha = 200$		
	util	*succ*	*rand*	*util*	*succ*	*rand*	*util*	*succ*	*rand*	*util*	*succ*	*rand*	*util*	*succ*	*rand*	*util*	*succ*	*rand*
50%	0.92	3	11	0.80	3	189	0.61	2	198	0.80	3	198	0.61	2	198	0.20	1	197
55%	0.92	3	10	0.97	4	191	0.72	3	197	0.97	4	200	1.20	5	198	0.72	3	199
60%	1.60	6	9	1.51	7	191	1.99	8	196	1.42	6	197	1.41	6	196	1.52	7	199
65%	6.47	20	8	2.66	14	189	3.58	18	198	5.78	21	198	3.57	16	195	3.57	17	198
70%	10.85	33	5	7.94	31	189	6.75	26	197	7.68	29	198	9.29	33	199	9.28	34	198
75%	14.60	48	8	14.58	58	189	12.23	53	197	12.57	56	198	12.89	56	198	11.95	54	198
80%	18.60	68	7	13.32	68	191	13.29	71	198	12.38	67	199	13.19	72	198	12.75	68	197
85%	20.61	80	8	12.80	80	188	12.54	85	196	12.40	84	197	12.29	84	196	11.67	81	197
90%	22.67	95	6	11.09	100	186	10.21	101	197	10.00	96	197	10.15	97	197	9.89	97	196
95%	23.35	110	7	7.34	109	188	6.63	109	197	6.29	113	197	6.47	115	195	6.19	116	195
100%	23.01	121	9	1.01	122	193	0.45	119	198	0.00	126	200	0.45	127	198	0.45	125	199

Table 4.5: Simulation results for a misreporting user in the setting with 60 jobs and 20 nodes. *util* represents the mean utility of the misreporting user, *succ* the number of runs (out of 200) in which the job of the misreporting user was allocated, and *rand* the number of runs in which the distributed randomized mechanism enforced the outcome of the randomized heuristic (and not the deterministic heuristic). Note that the user can obtain a positive utility from truthfully reporting her valuation in case the outcome of the deterministic mechanism is enforced.

Table 4.4 presents the results with the distributed randomized mechanism for the setting with 20 jobs and 20 nodes, Table 4.5 for the setting with 60 jobs and 20 nodes. The tables contain the mean utility of the misreporting user depending on her percentage bid, the number of runs (out of 200) in which the user's job got allocated, and the number of runs in which the distributed randomized mechanism enforced the outcome of the underlying randomized allocation mechanism instead of enforcing the outcome of the deterministic mechanism.

In the setting with 20 jobs and 20 nodes, the user can substantially benefit from understating her true valuation. The user maximizes her expected utility by only reporting about 65% of her true valuation. While this "optimal" percentage bid is essentially the same across all α, increasing α slightly reduces the gain. That is, expected utility decreases for small percentage bids and increases for higher percentage bids, thus further increasing the incentive to bid higher. Another important aspect is that understating the true valuation implies a trade-off between higher expected utility but a smaller chance of being allocated, cf. the *succ* columns in Table 4.4. For example for $\alpha = 10$, the job of the misreporting user is allocated 177 times when she reports 95% of her true valuation, but only 97 times when reporting 65%. This suggests that the randomized mechanism will provide much stronger incentives for users that are averse to the risk of their job not being allocated, and that bidding 65% of the true valuation will be a "lower bound" for such users. Moreover, increasing α leads to allocation decisions similar to the decisions made by the deterministic heuristic. The *rand* columns in Table 4.4 show that the distributed randomized mechanism increasingly enforces the allocation decisions of the deterministic heuristic as α increases and thus the chance for "luckier" allocation decisions by the randomized heuristic decreases.

Increasing competition from 20 jobs to 60 jobs while holding the number of nodes constant reduces the potential gain from manipulation. This further confirms the analytic insights. For large α's, the misreporting user's expected utility is maximized when she reports about 80% of her true valuation. Further, increasing competition drastically increases the risk of not being allocated. With 20 jobs and truthful reports of the job valuation, the misreporting job is allocated in about 187 runs out of 200, whereas with 60 jobs the misreporting job is only allocated about 125 times. Moreover, with higher competition, this risk increases *faster* than with less competition. With the user only reporting 65% of her true valuation and $\alpha = 10$, her job gets allocated 97 times (down from 187, i.e. a decrease of about 48% in the number of allocations) in the setting with low competition, but only 14 times (down from 122, i.e. a decrease of about 88.5%) in the setting with high competition. This increased risk pushes the misreporting user towards reporting closer to her true valuation.

Interestingly, this is very different for $\alpha = 1$, where the user's expected utility is maximized when reporting almost her true valuation. This result is due to the distributed mechanism enforcing the outcome of the deterministic mechanism rather than the randomized mechanism, which makes truth-telling of the valuation a (weakly) dominant strategy in most runs. As pointed out above, for small α's the randomized heuristic generally deviates most from the allocation decisions of the deterministic heuristic. On its downside, in this setting with high competition, this leads to the randomized heuristic increasingly taking unfortunate allocation decisions and thus the distributed mechanism enforcing the outcome of the deterministic mechanism. This illustrates the potential strategic benefit of combining the randomized mechanism with the partially truthful deterministic mechanism.

4.5.2 Efficiency

The following simulation aims at comparing the efficiency generated by the distributed randomized mechanism to the efficiency generated by the stand-alone deterministic heuristic. It is based on the same order books as the simulation in Subsection 3.5.1 in the previous chapter when investigating the impact of the level of competition on the deterministic heuristic's efficiency. The simulation consists of two settings, a setting with as many jobs as nodes and a setting with twice as many jobs as nodes. In each of these settings, the number of nodes is successively increased from 20, to 40, 60, up to 200. As in the previous chapter, the simulation is based on 30 order books per setting and order book size. α is varied from 1, to 5, 10, 20, 30, 40, and 50.

Table 4.6 shows the results for the setting with the same number of jobs and nodes, Table 4.7 shows the results for the more competitive setting with twice as many jobs as nodes. In the tables, *ratio* represents the ratio of the mean efficiency generated by the distributed randomized heuristic to the mean efficiency of the deterministic solution, and *rand* the number of runs (out of 30) in which the distributed randomized mechanism enforced the solution of the randomized heuristic and not the deterministic heuristic.

For the low level of competition (among users) and small numbers of jobs and nodes, the distributed randomized heuristic outperforms the deterministic heuristic by up to 2%. This is a good result considering that the deterministic heuristic already approximates the near-optimal solution of Anytime-CPLEX by about 97%, as shown in Table 3.9 in the previous chapter, thus not leaving much leeway for improvement. Moreover, the distributed randomized mechanism enforces the allocation determined by the randomized mechanism in almost all runs. Increasing α slightly reduces the improvement. As discussed above, the randomized heuristic makes allocation decisions close to the deterministic heuristic if α is large. Consequently, the randomized heuristic becomes less likely to improve on the deterministic heuristic.

Increasing the order book size also reduces the improvement. As Table 3.9 in the previous

Jobs	Nodes	$\alpha = 1$		$\alpha = 5$		$\alpha = 10$		$\alpha = 20$		$\alpha = 30$		$\alpha = 40$		$\alpha = 50$	
		ratio	*rand*	*ratio*	*rand*	*ratio*	*rand*	*ratio*	*rand*	*ratio*	*rand*	*ratio*	*rand*	*ratio*	*rand*
20	20	1.022	29	1.020	29	1.020	30	1.017	30	1.014	30	1.012	30	1.012	30
40	40	1.014	29	1.012	30	1.011	30	1.009	30	1.009	30	1.008	30	1.008	30
60	60	1.009	29	1.008	29	1.008	29	1.007	28	1.007	28	1.006	29	1.006	29
80	80	1.010	30	1.009	30	1.009	30	1.007	30	1.007	30	1.007	30	1.007	30
100	100	1.009	30	1.008	30	1.007	30	1.007	30	1.006	30	1.006	30	1.006	30
120	120	1.007	30	1.007	30	1.006	30	1.006	30	1.006	30	1.005	30	1.005	30
140	140	1.007	30	1.006	30	1.006	30	1.005	30	1.005	30	1.005	30	1.005	30
160	160	1.006	30	1.006	30	1.006	30	1.005	30	1.005	30	1.005	30	1.005	30
180	180	1.006	30	1.006	30	1.005	30	1.005	30	1.005	30	1.005	30	1.005	30
200	200	1.005	30	1.005	30	1.005	30	1.004	30	1.004	30	1.004	29	1.004	29

Table 4.6: Simulation results with respect to efficiency in the setting with as many jobs as nodes. *ratio* represents the ratio of the distributed randomized heuristic's mean efficiency compared to the mean efficiency generated by the deterministic heuristic, *rand* the number of runs (out of 30) in which the distributed randomized heuristic selects the outcome of one of the randomized runs and not of the deterministic heuristic.

Jobs	Nodes	$\alpha = 1$		$\alpha = 5$		$\alpha = 10$		$\alpha = 20$		$\alpha = 30$		$\alpha = 40$		$\alpha = 50$	
		ratio	*rand*	*ratio*	*rand*	*ratio*	*rand*	*ratio*	*rand*	*ratio*	*rand*	*ratio*	*rand*	*ratio*	*rand*
40	20	1.014	22	1.026	28	1.028	30	1.025	30	1.025	30	1.023	30	1.022	30
80	40	1.005	14	1.014	30	1.016	30	1.016	30	1.015	29	1.015	30	1.014	30
120	60	1.002	14	1.009	28	1.011	29	1.011	29	1.011	30	1.010	30	1.011	30
160	80	1.001	9	1.008	30	1.010	30	1.010	30	1.010	30	1.009	30	1.009	30
200	100	1.001	10	1.007	29	1.009	30	1.009	30	1.008	30	1.008	30	1.008	30
240	120	1.001	6	1.005	28	1.007	30	1.007	30	1.007	30	1.007	30	1.007	30
280	140	1.000	7	1.005	29	1.007	30	1.007	30	1.007	30	1.007	30	1.007	30
320	160	1.000	0	1.004	28	1.006	29	1.006	29	1.006	30	1.006	30	1.006	30
360	180	1.000	2	1.004	27	1.006	30	1.007	30	1.007	30	1.007	30	1.006	30
400	200	1.000	2	1.004	29	1.006	29	1.006	30	1.006	30	1.006	30	1.006	30

Table 4.7: Simulation results with respect to efficiency in the setting with twice as many jobs as nodes. *ratio* represents the ratio of the distributed randomized heuristic's mean efficiency compared to the mean efficiency generated by the deterministic heuristic, *rand* the number of runs (out of 30) in which the distributed randomized heuristic selects the outcome of one of the randomized runs and not of the deterministic heuristic.

chapter shows, the approximation ratio of the deterministic heuristic improves with increasing order book sizes, as the heuristic obtains more flexibility in allocating jobs to nodes. It hence becomes harder for the distributed mechanism to improve on the deterministic heuristic.

Increasing competition has two implications. With small α's, the randomized heuristic takes unfortunate allocation decisions, i.e. these decisions become "overly independent" of the jobs' valuations. Hence, the distributed randomized heuristic enforces the result of the deterministic heuristic in most runs (cf. Table 4.7 for $\alpha = 1$). However, for larger α's the improvement actually increases compared to the setting with low competition. Firstly, the approximation ratio of the deterministic heuristic reduces for increased competition, thus leaving more room for improvement. For instance, for small numbers of jobs and users, increasing the ratio of jobs to nodes from one to two reduces the deterministic heuristic's approximation ratio from about 97% to 95% (cf. Table 3.9 in the previous chapter). Secondly, the distributed randomized heuristic is robust to increased competition, since each additional user implies an additional run of the randomized heuristic.

4.5.3 Implications

The distributed randomized mechanism is a promising approach to mitigating the potential worst case behavior of the deterministic heuristic. The numerical results show that the distributed randomized heuristic only slightly improves on the deterministic heuristic with respect to efficiency in large settings with many jobs and nodes. This is due to the stand-alone deterministic heuristic already achieving a high approximation ratio, not leaving much room for possible improvements. The randomized mechanism becomes more relevant in small-scale settings, where the deterministic mechanism becomes more vulnerable to special cases.

The choice of α depends on the competition in the system. If competition is low, the randomized heuristic should be parameterized with a small α so as to make it deviate from the possibly unfortunate allocation decisions of the deterministic heuristic. If competition is high, α should be increased as the randomized heuristic takes overly random allocation decisions with small α's.

Besides avoiding worst cases, the distributed randomized mechanism might be a possible means for boosting the providers' revenue. Note that with the randomized mechanism the single user maximizes her expected utility by reporting between 65% and 80% of her true valuation, depending on the level of competition in the market. These reported valuations inherently translate into the providers' revenue, since a pay-as-bid pricing scheme is employed. In contrast, while the deterministic mechanism with critical-value pricing is truthful for users, this truthfulness essentially comes at the expense of revenue. Especially in settings with low competition, the critical-values of jobs will just be slightly above the node's reservation prices.

It is important to note that the performance of the randomized mechanism with respect to limiting strategic user behavior will improve as users become more risk-averse. Then the increased risk of not being allocated when deviating from the true valuation will push the reported valuations still closer towards the true valuations, thus further increasing the providers' revenue.

4.6 Discussion

In the previous chapter, a deterministic heuristic with a partially truthful pricing scheme was presented. However, in special cases, the heuristic may generate efficiency far from optimal. This chapter thus elaborated on a randomized heuristic that may help in intercepting such worst cases. Critical-value pricing is impossible with a non-deterministic allocation rule, and thus the use of the simple yet promising pay-as-bid pricing rule was proposed that, together with the monotone and convex allocation algorithm induces users to bid "close" to their true valuations. Further, a distributed mechanism was proposed that leverages the distributed nature that is inherent to any market by requiring each market participant to help in solving the allocation problem as kind of a participation fee. Moreover, this approach allows to combine the mechanism design worlds of determinism and randomization and mitigates the complexity in appropriately parameterizing the randomized mechanism, i.e. choosing a good α with respect to limiting strategic behavior and deviating from the deterministic heuristic's worst case behavior. The numerical results, however, indicate that especially in large settings the average case performance of the deterministic heuristic is hard to improve on with respect to efficiency. But the randomized pay-as-bid mechanism might be an interesting means to increase the providers' revenue if the critical-values of jobs are low, especially in the case of risk-averse users.

This chapter focused purely on modeling strategic behavior on the demand side. A natural extension would thus be to also model strategic behavior of resource providers. It would be interesting to see whether randomizing the supply-side of the heuristics (i.e. the sorting of node offers) might be beneficial or if this is already too much randomization in the sense that the resulting mechanism makes overly "random" and unfortunate allocation decisions. Moreover, users were assumed to have quasi-linear utility functions and hence to be risk-neutral. This assumption in fact penalizes the randomized mechanism. As the simulation results showed, one important strategic aspect of randomization is the allocation risk. It would hence be promising to investigate randomized mechanisms for risk-averse users, as increased allocation risk will generally push risk-averse users towards truthful behavior.

Chapter 5

The Power of Preemption

5.1 Introduction

The aim of this chapter is to showcase the benefit of allowing *preemptions* regarding the performance of online market mechanisms (or simply "online mechanisms" in the remainder). As discussed in Chapters 1 and 2, online mechanisms continuously assign jobs to nodes as new jobs enter the system and / or nodes become idle. Whereas online mechanisms are more responsive than periodic offline mechanisms, online mechanisms have no information about the future input, and past allocation decisions may thus prove unfortunate. Preemption gives the mechanism more flexibility in intercepting and alleviating such unfortunate decisions. Running jobs can be suspended in favor of some more desirable job and possibly continued later on the same node. Until recently, only few grid and cluster systems provided preemptive migration. The emerging technology of virtualization becomes an important building block in clusters and grids (e.g. Figueiredo *et al.* (2003)) and provides off-the-shelf support for virtual machine migration, thus making the use of preemption and migration more accessible.

The results of this chapter show that the performance of online market mechanisms can be significantly improved by performing preemptions, which has largely been neglected in the existing literature on market-based scheduling. E.g. the *Decentralized Local Greedy Mechanism* of Heydenreich *et al.* (2006) was shown to be 3.281-competitive with respect to total weighted completion time if the users act rationally. In this chapter, it is analytically shown that the *preemptive version* of this mechanism is 2-competitive. As a by-product, preemption allows to relax the restrictions on jobs upon which this competitiveness relies. At the core of this chapter, an in-depth empirical analysis is presented of the *average case performance* of the original mechanism and its preemptive extension based on real workload traces. The empirical findings indicate that introducing preemption improves both the utility and the slowdown of the jobs.

Furthermore, this improvement does not necessarily come at the expense of low-priority jobs.

5.2 The Problem Setting

The specific problem at hand is having to schedule a set of jobs with arbitrary release dates onto m parallel *homogeneous* nodes with the aim of minimizing total weighted completion time $\sum_{j \in J} v_j e_j$, where J is the set of jobs to be scheduled.[1] $e_j \in \mathbb{R}_+$ denotes job j's completion time (the end), i.e. the point in time when j leaves the system. Job $j \in J$ is of type $\theta_j = (r_j, d_j, v_j) \in \mathbb{R}_+ \times \mathbb{R}_+ \times \mathbb{R}_+$, where r_j denotes j's release date, d_j its runtime, and v_j is its weight, which can be interpreted as j's valuation or waiting cost, that is the cost of remaining in the system for one additional unit of time.

A setting is considered in which each user submits a single job. The terms "user" and "job" are thus used interchangeably in the remainder of this chapter. While the nodes are obedient, the jobs are rational and selfish. Each job $j \in J$ aims at maximizing its individual ex post utility

$$u_j(e_j, p_j | \theta_j) = -v_j e_j - p_j,$$

where p_j is j's ex post payment. Job j may decide to strategically misreport about its type, i.e. it may report $\tilde{\theta}_j = (\tilde{r}_j, \tilde{d}_j, \tilde{v}_j) \neq (r_j, d_j, v_j)$ in order to improve its utility compared to truthful reporting. Obviously, $\tilde{r}_j \geq r_j$. Furthermore, $\tilde{d}_j \geq d_j$ since any excess runtime can easily be detected and punished by the system.[2] It is henceforth assumed that jobs are numbered according to their time of arrival, i.e. $k < j \Rightarrow \tilde{r}_k \leq \tilde{r}_j$.

It is important to note that this homogeneous setting (with respect to the nodes) will generally understate the benefit of performing preemptions. For instance, with heterogeneous nodes it can be beneficial to preempt a running job and continue it on another, faster node that has idle capacity (migration).

5.3 Baseline Model – A Decentralized Local Greedy Mechanism

Heydenreich *et al.* (2006) examine the setting at hand *without* preemption, that is $P|r_j|\sum v_j e_j$ based on the notation of Graham *et al.* (1979). They propose a *Decentralized Local Greedy Mechanism* (*DLGM*) that will be presented now for the ease of exposition:

[1] Note that the original notation uses w_j instead of v_j and C_j instead of e_j. This was modified to allow for consistency with the notation in the other chapters of this work.

[2] It is a common policy in shared computing systems to kill jobs that exceed their reported runtime.

- **Step 1 – Job report:** At its chosen release date $\tilde{r}_j$, job j communicates $\tilde{v}_j$ and $\tilde{d}_j$ to every node $n \in N$.

- **Step 2 – Tentative node feedback:** Based on the received information, the nodes communicate a tentative node-specific completion time $\hat{e}_j(n)$ and a tentative payment $\hat{p}_j(n)$ to the job. The tentativeness is due to the fact that later arriving jobs might overtake job j. This leads to a final ex post completion time $e_j(n) \geq \hat{e}_j(n)$ and a final ex post payment $p_j(n) \leq \hat{p}_j(n)$ as compensation payments by overtaking jobs might occur (see Step 3 below).

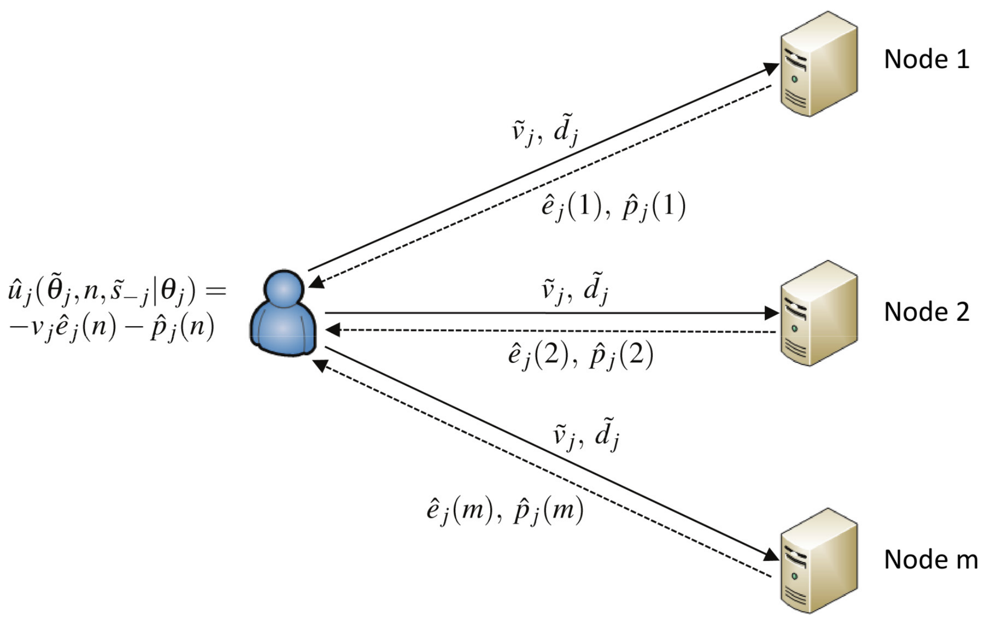
  The local scheduling on each node follows Smith's ratio rule (Smith, 1956), which has been shown to be optimal for $1||\sum v_j e_j$ with one single node and without release dates, i.e. all jobs arriving at the same time. Jobs are assigned a priority according to their ratio of weight and processing time: Job j has a higher priority than job k if (1) $\tilde{v}_j/\tilde{d}_j > \tilde{v}_k/\tilde{d}_k$ or (2) $\tilde{v}_j/\tilde{d}_j = \tilde{v}_k/\tilde{d}_k$ and $j < k$, and is inserted in front of k into the waiting queue at this node. For obtaining the tentative completion time, the remaining processing time of the currently running job and the runtimes of the higher-prioritized jobs in the local queue as well as j's own runtime have to be added to $\tilde{r}_j$. The tentative payment equals a compensation of utility loss for all jobs which would be displaced if j is queued at this node.

- **Step 3 – Queueing:** Upon receiving information about its tentative completion time and required payment from the nodes, job j makes a binding decision for a node. j is queued at its chosen node n according to its priority and pays $\tilde{v}_k\tilde{d}_j$ to each lower ranked job k at this node.

The information exchange between the job and the nodes is depicted in Figure 5.1. At a first glance, this decentralized protocol does not seem to fit to a cluster or utility computing setting where the nodes are under centralized control. However, the purpose of this decentralization is rather to be able to determine the waiting costs a new job causes and in turn appropriate prices. The protocol as such – and thus the properties of the mechanisms that will be derived below – can also be implemented by a centralized mechanism.

For evaluating and comparing market mechanisms, one needs to define the user behavior, i.e. the users' strategies s, and a metric. It is started with the former.

Under *DLGM*, j's strategy consists of reporting its type *and* choosing a node. Let $\tilde{s}_j = (\tilde{\theta}_j, n)$ be job j's strategy. Let $\tilde{s}$ be the vector containing the arbitrary strategies of all users, and let $\tilde{s}_{-j}$ be the vector containing the arbitrary strategies of all users except j. Given the tentative node



Node 1
Node 2
Node m
$\tilde{v}_j,\ \tilde{d}_j$
$\hat{e}_j(1),\ \hat{p}_j(1)$
$\hat{e}_j(2),\ \hat{p}_j(2)$
$\hat{e}_j(m),\ \hat{p}_j(m)$
$\hat{u}_j(\tilde{\theta}_j, n, \tilde{s}_{-j}|\theta_j) = -v_j\hat{e}_j(n) - \hat{p}_j(n)$

Figure 5.1: Information exchange in *DLGM* at time $\tilde{r}_j$.

feedback to job j's report of $\tilde{\theta}_j$, let $\hat{u}_j(\tilde{\theta}_j, n, \tilde{s}_{-j}|\theta_j)$ be j's *tentative utility at node n* at time $\tilde{r}_j$.[3] Heydenreich *et al.* (2006) use the concept of myopic best response equilibria in order to model the behavior of rational and selfish users:

Definition 1 (Myopic best response equilibrium). *A strategy profile $s = (s_1, \ldots, s_n)$ is called a* myopic best response equilibrium *if, for all $j \in J$, θ_j, $\tilde{s}_{-j}$, and all strategies $\tilde{s}_j$ which j could play instead of s_j,*

$$\hat{u}_j(s_j, \tilde{s}_{-j}|\theta_j) \geq \hat{u}_j(\tilde{s}_j, \tilde{s}_{-j}|\theta_j).$$

Theorem 1 (Myopic best response equilibrium under *DLGM*, Theorem 9 in Heydenreich *et al.* (2006)). *Given the types of all jobs, the strategy profile s where each job j reports $\tilde{\theta}_j = \theta_j$ and chooses the node n that maximizes its tentative utility $\hat{u}_j(\tilde{\theta}_j, n, \tilde{s}_{-j}|\theta_j) = -v_j\hat{e}_j(n) - \hat{p}_j(n)$ is a myopic best response equilibrium under* DLGM.

For the sake of completeness, the full proof of Heydenreich *et al.* (2006) is presented in Appendix C. According to this theorem, without knowledge about the future and other jobs' types, each job maximizes its *tentative* utility at time r_j by truthfully reporting its characteristics and choosing the best available node. Furthermore, if the user *truthfully* reports her type, then her ex post utility equals her tentative utility since whenever the job's tentative completion time changes, the job is immediately compensated for the exact loss of ex post utility.

[3]The notation may henceforth differ slightly for the ease of exposition if the notation is obvious from the context.

Since it is now known how jobs act in this model, the performance of *DLGM* can be evaluated as regards efficiency. A common metric for a scheduling mechanism's performance is its *competitive ratio*, in this case in the myopic best response equilibrium. This approach is adopted and, in the setting at hand, an online market mechanism's competitive ratio is defined as the largest possible ratio of the total weighted completion time generated by the specific mechanism if all jobs play their equilibrium strategy divided by the theoretical minimum of an *omniscient offline mechanism* that knows *all* the jobs' true types when making its allocation decisions, in particular also the jobs' release dates (Heydenreich *et al.*, 2006). One of the main results of Heydenreich et al. will be stated, as this will be the baseline for the following analysis:

Theorem 2 (Competitive ratio of *DLGM*, Theorem 10 in Heydenreich *et al.* (2006)). *Suppose every job is rational in the sense that it truthfully reports θ_j and selects a node that maximizes its tentative utility at arrival. Then* DLGM *is 3.281-competitive for the scheduling problem $P|r_j|\sum v_j e_j$.*

This theorem essentially captures *DLGM*'s performance without using preemption. Note that this performance ratio relies on the artificial assumption that *critical jobs*, that is jobs with long runtimes, are only released to the system later in the scheduling process. To achieve this, Heydenreich *et al.* (2006) impose the restriction $r_j \geq \alpha d_j$, and optimize the performance ratio ρ over α to obtain $\rho = 3.281$. In a real system, this restriction can be achieved by not allowing the release of such jobs before this inequality is satisfied. However, this restriction still remains somewhat artificial.

5.4 Adding Preemption

It will now be examined how introducing preemption to *DLGM* impacts the mechanism's competitive ratio. This extended *DLGM* will henceforth be referred to as *P-DLGM*.

The following notation is introduced. Let d_j continue to denote j's total runtime, but let $d_j(t)$ be its *remaining* runtime at time t. In contrast to *DLGM*, *P-DLGM* uses a *dynamic extension* of Smith's ratio rule, i.e. at time t, the jobs are ordered according to the ratio of their reported weight and the remaining runtime ($\tilde{v}_j/\tilde{d}_j(t)$). Hence, let $H_j(t) = \{k \in J | \tilde{v}_k/\tilde{d}_k(t) > \tilde{v}_j/\tilde{d}_j(t)\} \cup \{k \leq j | \tilde{v}_k/\tilde{d}_k(t) = \tilde{v}_j/\tilde{d}_j(t)\}$, i.e. $H_j(t)$ contains all jobs with higher priority than job j at time t, including j itself. Further, $L_j(t) = J \setminus H_j(t)$ is introduced, i.e. the set containing all jobs with a lower priority than j at time t. Let $j \to n$ denote that job j is assigned to node n. At time $\tilde{r}_j$, all jobs k with $k < j$ and $e_k > \tilde{r}_j$ are present in the system.

P-DLGM comprises the following three steps:

- **Step 1 – Job report:** At its chosen release date $\tilde{r}_j$, job j communicates $\tilde{v}_j$ and $\tilde{d}_j$ to every node $n \in N$.

- **Step 2 – Tentative node feedback:** Based on the received information, the nodes communicate a tentative node-specific completion time and a tentative payment to the job.

  The tentative completion time of job j at node n is determined as

  $$\hat{e}_j(n) = \tilde{r}_j + \tilde{d}_j + \sum_{\substack{k \in H_j(\tilde{r}_j) \\ k \to n \\ k < j \\ e_k > \tilde{r}_j}} \tilde{d}_k(\tilde{r}_j),$$

  i.e. the projected time that job j spends on node n equals the sum of j's own runtime and the remaining runtimes of all jobs which are queued in front of j at n at time $\tilde{r}_j$.

  The tentative compensation payment of job j at node n is determined as

  $$\hat{p}_j(n) = \tilde{d}_j \sum_{\substack{k \in L_j(\tilde{r}_j) \\ k \to n \\ k < j \\ e_k > \tilde{r}_j}} \tilde{v}_k,$$

  i.e. j's runtime multiplied by the aggregate weights of all jobs which are displaced at node n due to the addition of j at time $\tilde{r}_j$. This comprises the currently waiting jobs and, due to allowing preemption, possibly also the currently running job. Consequently, given that the job reported its true weight, it is compensated exactly for its additional waiting costs if it is delayed by some subsequent job with a higher priority.

- **Step 3 – Queueing:** Upon receiving information about its tentative completion time and required payment from the nodes, job j makes a binding decision for a node. Job j is queued at its chosen node n according to its priority or preempts the currently running job – which is then put back into this node's local queue – and pays $\tilde{v}_k \tilde{d}_j$ to each lower ranked job k at this node.

Note that in the extension to the basic *DLGM*, preemption is assumed to imply zero cost, that is jobs can be suspended in negligible time. This is a reasonable assumption since – in contrast to migrations where jobs are transferred between *different* nodes over the network (a setting investigated in Amar *et al.* (2008b)) – in *P-DLGM* jobs are suspended on one single node.

Now the main results can be stated:

Theorem 3 (Myopic best response equilibrium under *P-DLGM*). *Given the types of all jobs, the strategy profile where each job j reports $\tilde{\theta}_j = \theta_j$ and chooses a node which maximizes its tentative utility $\hat{u}_j(\theta_j, n, \tilde{s}_{-j} | \theta_j) = -v_j \hat{e}_j(n) - \hat{p}_j(n)$ is a myopic best response equilibrium under* P-DLGM *and its ex post utility equals its tentative utility.*

Proof. Due to the dynamic extension to Smith's ratio rule, the proof to Theorem 3 reduces to the proofs to Theorem 9 in Heydenreich *et al.* (2006) as the dynamic priorities can be plugged into the latter. Consequently, the proof to this theorem and its supporting lemmata and theorems do not change if preemption is introduced (cf. the proofs in Appendix C). □

Before deriving *P-DLGM*'s competitive ratio, the following lemma is stated:

Lemma 1 (Non-increasing set of higher-priority jobs).

$$\{(j,k) \in J \times J | j \in H_k(r_j), k < j\} \subseteq \{(j,k) \in J \times J | j \in H_k(r_k), k < j\}.$$

Proof. There are two possible cases (recall that $k < j \Rightarrow r_k \leq r_j$):

- $r_k = r_j \Rightarrow H_k(r_j) = H_k(r_k)$ and thus $\{(j,k)|j \in H_k(r_j), k < j\} = \{(j,k)|j \in H_k(r_k), k < j\}$.
- $r_k < r_j$ and $j \in H_k(r_j)$. Obviously, $\frac{v_j}{d_j(r_j)} = \frac{v_j}{d_j(r_k)}$ since j cannot be started prior to r_j. Moreover, $\frac{v_k}{d_k(r_j)} \geq \frac{v_k}{d_k(r_k)}$ since the dynamic weighted shortest processing time (WSPT) ratio of job k cannot decrease over time but remains unchanged if k is not executed and increases if k is executed. It follows that
$$\frac{v_j}{d_j(r_j)} = \frac{v_j}{d_j(r_k)} > \frac{v_k}{d_k(r_j)} \geq \frac{v_k}{d_k(r_k)}$$
and thus $j \in H_k(r_k)$.

□

Theorem 4 (Competitive ratio of *P-DLGM*). *Suppose that every job j plays its myopic best response strategy according to Theorem 3. Then* P-DLGM *is 2-competitive for the scheduling problem $P|r_j, pmtn|\sum v_j e_j$.*

Proof. If job j plays a myopic best response strategy (r_j, d_j, v_j), at time r_j it selects the node n that minimizes

$$\begin{aligned} -\hat{u}_j(\theta_j, n, s_{-j}|\theta_j) &= v_j \hat{e}_j(n) + \hat{p}_j(n) \\ &= v_j(r_j + d_j + \sum_{\substack{k \in H_j(r_j) \\ k \to n \\ k < j \\ e_k > r_j}} d_k(r_j)) + d_j \sum_{\substack{k \in L_j(r_j) \\ k \to n \\ k < j \\ e_k > r_j}} v_k. \end{aligned}$$

Let n_j be this node. As a result of the payment scheme, $-\hat{u}_j(., n_j, .)$ exactly corresponds to the increase of the objective value $\sum_{k \in J} v_k e_k$ which is due to the addition of j. Furthermore, any change in $u_j(.)$ that results from the assignment of some job k to node n_j after r_j, is absorbed by the payment scheme and $u_k(.)$. Thus the objective value can be expressed as $W(s|\theta) = \sum\limits_{j\in J} -\hat{u}_j(\theta_j, n_j, s_{-j}|\theta_j)$.

Since jobs are assumed to act rationally when choosing the node n at which to queue, one obtains

$$-\hat{u}_j(.,n_j,.) = \min_{n=1,\dots,m} -\hat{u}_j(.,n,.) \le \frac{1}{m}\sum_{n=1}^{m} -\hat{u}_j(.,n,.)$$

by an averaging argument, and therefore $W(.) \le \sum\limits_{j\in J} \frac{1}{m} \sum\limits_{n=1}^{m} -\hat{u}_j(.,n,.)$. Hence

$$\frac{1}{m}\sum_{n=1}^{m} -\hat{u}_j(.,n,.) = v_j(r_j+d_j) + v_j \sum_{n=1}^{m} \sum_{\substack{k\in H_j(r_j)\\ k\to n\\ k<j\\ e_k>r_j}} \frac{d_k(r_j)}{m} + d_j \sum_{n=1}^{m} \sum_{\substack{k\in L_j(r_j)\\ k\to n\\ k<j\\ e_k>r_j}} \frac{v_k}{m},$$

which can be shortened to

$$\frac{1}{m}\sum_{n=1}^{m} -\hat{u}_j^n(.) = v_j(r_j+d_j) + v_j \sum_{\substack{k\in H_j(r_j)\\ k<j\\ e_k>r_j}} \frac{d_k(r_j)}{m} + d_j \sum_{\substack{k\in L_j(r_j)\\ k<j\\ e_k>r_j}} \frac{v_k}{m}.$$

By including all jobs instead of only the jobs k for which $e_k > r_j$, and by considering the total runtime of jobs $k \in H_j(r_j), k < j$, this can be upper-bounded by

$$\begin{aligned}\frac{1}{m}\sum_{n=1}^{m} -\hat{u}_j^n(.) &\le v_j(r_j+d_j) + v_j \sum_{\substack{k\in H_j(r_j)\\ k<j}} \frac{d_k(r_j)}{m} + d_j \sum_{\substack{k\in L_j(r_j)\\ k<j}} \frac{v_k}{m}\\ &\le v_j(r_j+d_j) + v_j \sum_{\substack{k\in H_j(r_j)\\ k<j}} \frac{d_k}{m} + d_j \sum_{\substack{k\in L_j(r_j)\\ k<j}} \frac{v_k}{m}.\end{aligned}$$

By utilizing Lemma 1 and subsequently interchanging the indices j and k, the summation of the last term over all jobs in J can be rewritten as

$$\sum_{j\in J} d_j \sum_{\substack{k\in L_j(r_j)\\ k<j}} \frac{v_k}{m} = \sum_{\substack{(j,k):\\ j\in H_k(r_j)\\ k<j}} d_j \frac{v_k}{m} \le \sum_{\substack{(j,k):\\ j\in H_k(r_k)\\ k<j}} d_j \frac{v_k}{m} = \sum_{\substack{(j,k):\\ k\in H_j(r_j)\\ j<k}} d_k \frac{v_j}{m} = \sum_{j\in J} v_j \sum_{\substack{k\in H_j(r_j)\\ k>j}} \frac{d_k}{m}.$$

Therefore,

$$\begin{aligned}
W(.) &\leq \sum_{j\in J} v_j(r_j+d_j) + \sum_{j\in J} v_j \sum_{\substack{k\in H_j(r_j)\\ k<j}} \frac{d_k}{m} + \sum_{j\in J} v_j \sum_{\substack{k\in H_j(r_j)\\ k>j}} \frac{d_k}{m} \\
&= \sum_{j\in J} v_j(r_j+d_j) + \sum_{j\in J} v_j \sum_{k\in H_j(r_j)} \frac{d_k}{m} - \sum_{j\in J} v_j \frac{d_j}{m} \\
&\leq \sum_{j\in J} v_j(r_j+d_j) + \sum_{j\in J} v_j \sum_{k\in H_j(r_j)} \frac{d_k}{m}.
\end{aligned}$$

Let $W^{OPT}_{pmtn}(\theta)$ be the objective value that an omniscient offline mechanism can achieve for an instance θ of $P|r_j, pmtn|\sum v_j e_j$.

Obviously, $\sum_{j\in J} v_j(r_j+d_j) \leq W^{OPT}_{pmtn}(\theta)$.

Furthermore, consider the problem $1||\sum v_j e_j$ for a single node with speed m times the speed of any of the identical parallel nodes and with the same jobs all arriving at time zero. Without release dates, one gets that $H_j(t) = H_j := \{k \in J|\tilde{v}_k/\tilde{d}_k > \tilde{v}_j/\tilde{d}_j\} \cup \{k \leq j|\tilde{v}_k/\tilde{d}_k = \tilde{v}_j/\tilde{d}_j\}$ for all t since $\tilde{d}_j(t) \leq \tilde{d}_j$, i.e. the ordering of jobs does not change over time. Since $1||\sum v_j e_j$ is a relaxation of $P|r_j, pmtn|\sum v_j e_j$ and $\sum_{j\in J} v_j \sum_{k\in H_j} \frac{d_k}{m}$ is the objective value of the optimal solution to $1||\sum v_j e_j$, one obtains $\sum_{j\in J} v_j \sum_{k\in H_j(r_j)} \frac{d_k}{m} \leq W^{OPT}_{pmtn}(\theta)$ (Megow and Schulz, 2004), and thus $W(s|\theta) \leq 2W^{OPT}_{pmtn}(\theta)$. □

This proof is close in spirit to the proof to Theorem 10 in Heydenreich *et al.* (2006). The basic differences are that dynamic WSPT ratios are used in contrast to the static priorities used in Heydenreich *et al.* (2006) and that different reductions can be used to upper bound the competitive ratio in the last step of the proof.

One may argue that the bounds in Theorems 2 and 4 relate to different optimization problems. However, exactly this difference – introducing preemption – is the main point in this chapter, which is captured by the following theorem:

Corollary 1 (Benefit of preemption). *Suppose that every job j plays its myopic best response strategy according to Theorem 3. Then preemptions allow to improve the upper (worst case) bound on the objective value $\sum v_j e_j$ generated by a market mechanism from 3.281 to 2.*

Proof. Take Theorems 2 and 4 as well as the fact that the objective value $W^{OPT}_{pmtn}(\theta)$ of the optimal solution to $P|r_j, pmtn|\sum v_j e_j$ will always (for all θ) be less than or equal to the objective value $W^{OPT}(\theta)$ of the optimal solution to $P|r_j|\sum v_j e_j$. Consequently, if $W(s|\theta)$ is the objective value generated by *P-DLGM*, then $W(s|\theta) \leq 2W^{OPT}_{pmtn}(\theta) \leq 2W^{OPT}(\theta)$. □

With preemption, one cannot only lower this upper bound to $\rho = 2$, but additionally omit the artificial restriction on critical jobs ($r_j \geq \alpha d_j$).

An important result of Heydenreich *et al.* (2006) is that there is no payment scheme which can complement *DLGM* so as to make truthtelling a dominant strategy equilibrium where revealing the true job type and choosing the best node is not only the tentatively optimal strategy but is also optimal from an ex post perspective. This result applies also for *P-DLGM*.

Theorem 5 (Impossibility of truthful prices for *P-DLGM*). *It is not possible to turn* P-DLGM *into a mechanism with a dominant strategy equilibrium in which all jobs report truthfully by only modifying the payment scheme.*

Proof. The proof follows from Theorem 14 in Heydenreich *et al.* (2006). It relies on a simple example to show that, under *DLGM*, jobs may improve their ex post completion time by reporting $\tilde{v}_j < v_j$, which contradicts weak monotonicity, a necessary condition for truthfulness (Lavi *et al.*, 2003; Heydenreich *et al.*, 2006). In the example, all jobs arrive at the same time. Consequently, no preemption can occur and this example as well as the supporting lemmata thus also hold *with* preemption. □

An interesting open question for future research remains: Is there any truthful mechanism (in dominant strategies) at all for this setting?

5.5 Empirical Analysis

5.5.1 Experimental Setup

In the previous section, it was shown that *P-DLGM* yields a better *worst case* performance than *DLGM*. In this section, the *average case* is analyzed by means of an empirical analysis based on real workload traces.

A simulator was implemented to study online mechanisms for the scheduling in distributed computing systems. The experimental setup is similar to the analysis of fairness in economic online scheduling in Amar *et al.* (2008b). *P-DLGM* and *DLGM* are evaluated using this previous setting since this allows for comparing the results of both analyses. Moreover, the economic setting at hand is to be checked without “tailoring” a specific setting towards the advantage of *P-DLGM*.

Workload Traces

For the simulations, four workload traces were taken from the Parallel Workload Archive[4] (cf. Table 5.1). All these traces were recorded on homogeneous clusters. The DAS2-FS4 cluster

Trace	Timeframe	Jobs (original)	Jobs (serialized)	CPUs	Runtime Mean (sec.)	Runtime CV (%)
WHALE	Dec'05–Jan'07	196,417	280,433	3,072	35,658	237
REQUIN	Dec'05–Jan'07	50,442	466,177	1,536	45,674	411
LPC-EGEE	Aug'04–May'05	219,704	219,704	140	3,212	500
DAS2-FS4	Feb'03–Dec'03	32,626	118,567	64	2,236	961

Table 5.1: Workload traces.

is part of a Dutch academic grid. LPC is a French cluster that is part of the EGEE grid. The WHALE and REQUIN traces are taken from two Canadian clusters. These workloads were chosen due to the large number of jobs which will help in mitigating stochastic outliers, the availability of technical parameters such as release dates and runtimes, and because of their relative recentness, as old workloads might contain outdated applications and utilization patterns. In all of the traces the CPUs were dedicated, meaning only one job is using each CPU at the same time.[5] Parallel jobs (using more than one CPU) are treated as a *collection* of serial jobs all with the same weight, release date and runtime. The addition "serialized" in the job column of Table 5.1 indicates the number of jobs after converting such parallel jobs to serial ones.

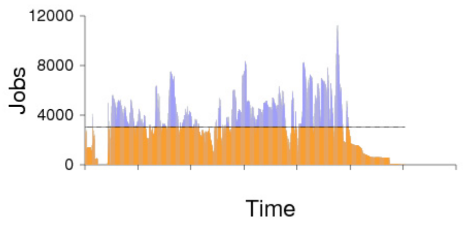
Table 5.1 contains descriptive statistics of the jobs in the traces. The homogeneity of the jobs within one trace as regards runtime is expressed by reporting the coefficients of variation (CV) of the runtimes, which normalize the standard deviation by the mean. The jobs in WHALE and REQUIN have long runtimes and are rather homogeneous, whereas the jobs in LPC-EGEE and DAS2-FS4 are short on average with DAS2-FS4 being highly heterogeneous.

To analyze the utilization patterns in these traces, they were simulated using a simple first-in-first-out scheduler. As the results in Figure 6.1 illustrate, the WHALE and the REQUIN cluster are highly utilized, a large number of jobs resides in the waiting queue most of the time. In contrast, the LPC-EGEE and the DAS2-FS4 clusters only have a small number of peaks in the waiting queue. The competition among jobs is small and CPUs are frequently idle. To measure the impact of preemption for the LPC-EGEE and the DAS2-FS4 clusters in more competitive settings, the pressure in these two workloads was increased and simulated with only 75% of the

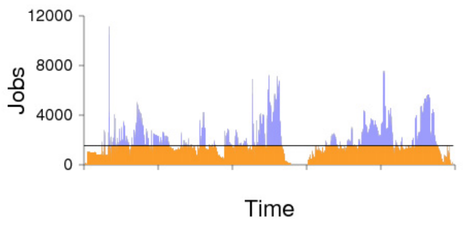
[4] `http://www.cs.huji.ac.il/labs/parallel/workload`

[5] Note that the actual job characteristics are taken from from the traces which have been measured by the system, not the user estimates.

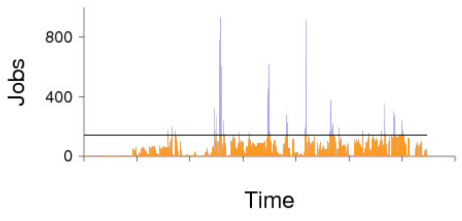
original CPUs being available.



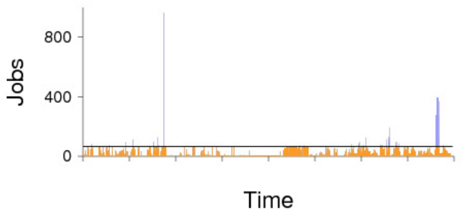
(a) WHALE



(b) REQUIN



(c) LPC-EGEE



(d) DAS2-FS4

Figure 5.2: Utilization patterns of the workload traces with a FIFO scheduler (Waiting Jobs, Running Jobs).

Waiting Cost Model

Essentially, the users' waiting costs (weights) represent the users' valuations for the jobs. The only empirical investigation of economic scheduling mechanisms which uses a time-dependent user valuation model was performed by Chun and Culler (2002). Valuations were assumed to be bimodal with the majority of jobs having valuations following a normal distribution with a low mean, and some high valuation jobs with valuations coming from a second normal distribution with a higher mean.

In order to check the validity of the results for two different valuation models, all settings were simulated for such a bimodal distribution with 80% of the job weights coming from a normal distribution with mean 30 and standard deviation 15, and 20% of the job weights coming from a normal distribution with mean 150 and standard deviation 15.[6] Consequently, on average, high-valuation jobs were assumed to be five times more important than low-valuation jobs. The simulation settings were additionally run drawing the job weights from a uniform distribution over $\{1, \ldots, 100\}$, i.e. there are 100 priority classes.

[6]Note that negative valuations were cut off.

Metrics

Since this work is about investigating economic schedulers, the evaluation cannot be based on purely technical metrics, based on a single scalar, such as makespan or the sum of completion times. Instead, metrics have to be developed that capture the viewpoint of the users and measure the dependency between the "service" a job receives from the system and its reported valuation.

Total weighted flow time describes the overall system performance and is defined as $\sum_j v_j(e_j - r_j)$. In contrast to the previous section, for the empirical analysis it was chosen to measure the total weighted flow time instead of the total weighted completion time. First, minimizing completion time is equivalent to minimizing flow time up to an additive constant of $-\sum_j v_j r_j$.[7] Second, since traces are run that cover more than one year of workloads on a per second basis, this additive constant will be very large and hence might dominate this ratio. Thus, focussing on the flow time instead of the completion time helps in determining the actual difference in system performance for *DLGM* and *P-DLGM*.

Utility per priority value describes the utility a job receives in relation to its WSPT ratio.[8] Total weighted flow time only describes the overall system performance. In contrast, this measure gives more insights into the impact of performing preemptions on the *single* jobs' utility. Which jobs suffer from preemptions, which gain, or do all jobs gain by performing preemptions regardless of their priority? To capture the utility per WSPT ratio (which is a continuous random variable), this value range is discretized as follows: All jobs are sorted with respect to their initial WSPT priorities v_j/d_j. This sorted list is then divided into 100 slices, i.e. the percentile of jobs with the lowest WSPT ratios, the second percentile and so forth. The mean utility will be reported for each percentile to compare *DLGM* and *P-DLGM*.

Bounded slowdown per valuation (BSD) also reflects the perspective of a single job. The bounded slowdown of job j is defined as

$$BSD_j = \begin{cases} \frac{e_j - r_j}{d_j} & \text{if } d_j \geq 60 \\ \frac{e_j - r_j}{60} & \text{else.} \end{cases}$$

This metric is widely used in the Computer Systems Evaluation literature (e.g. Harchol-Balter and Downey (1997); Mu'alem and Feitelson (2001)). The bounded slowdown is taken instead of the slowdown because short jobs can easily experience a large slowdown, which does not necessarily reflect a bad service. Intuitively, job j seeks to minimize BSD_j. Naturally, in economic schedulers, jobs with higher valuations (and smaller runtimes) should get smaller (bounded)

[7]The optimal schedules are identical for both metrics. However, schedules that approximate each metric can differ even if the same approximation ratio is guaranteed.

[8]Note that for jobs playing the best myopic strategy of truthful reporting the tentative utility equals the ex post utility, as shown in Theorem 7(a) in Heydenreich *et al.* (2006).

slowdowns. The rationale for looking at this metric is that this gives hints towards the mechanisms' performance if other job utility functions as the one introduced above are considered, e.g. if the importance of the jobs' waiting costs increases compared to the job payments. Additionally, the utility function in the setting at hand has many indifference points (a delay of job j for a one time unit can be compensated with a payment of v_j). It seems reasonable to assume that users have strict preferences over these "indifference" points and would like to finish earlier rather than to be compensated.

5.5.2 Experimental Results

The following results represent the means across 50 runs. Table 5.2 shows the ratio of the total weighted flow time generated by *P-DLGM* to the result produced by *DLGM* for both the uniform and the bimodal weight distribution. *P-DLGM* always outperforms *DLGM* with respect

Trace	Uniform distribution	Bimodal distribution
WHALE	1.09	1.08
REQUIN	1.06	1.05
LPC-EGEE-75%	1.07	1.07
DAS2-FS4-75%	1.22	1.20

Table 5.2: Ratio of the total weighted flow time of *DLGM* to *P-DLGM* for the uniform and the bimodal weight distributions.

to this overall performance metric. Consequently, *P-DLGM* not only improves upon *DLGM* in the worst case as shown in the previous section but also in the average case. Intuitively, the benefit of performing preemptions increases (i.e., the ratio increases) as the pressure in the system increases, that is as more jobs compete for the resources. As the results hold for both the uniform and the bimodal weight distribution, it can be hypothesized that the overall benefit of performing preemptions is robust to the assumption about a specific weight distribution.

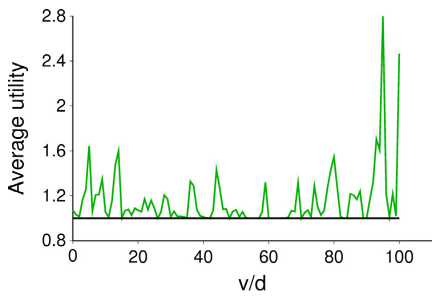
However, the results for the total weighted flow time cannot give any insight into the impact of performing preemptions on the performance of the *individual* jobs. Which jobs benefit and which jobs suffer from this feature? Figures 5.3(a), 5.3(c), 5.3(e) and 5.3(g) show the ratio of the average utility per WSPT priority percentile (see explanation in Subsection 5.5.1) generated by *DLGM* to the average utility produced by *P-DLGM* (based on the uniform weight distribution). Since both are always negative, the bigger this ratio, the better *P-DLGM* performs in comparison to *DLGM* for a given priority range. Overall, the payments across all jobs cancel out due to the presented pricing scheme that is based on compensation payments between the jobs. *P-DLGM* results in lower total waiting costs and, due to the jobs' utility functions, higher welfare. In the simulations, it can be seen that the ratio of the average utilities is almost always bigger than 1 for all four workload traces *and* priority ranges. This shows that *P-DLGM* almost always outperforms *DLGM* and that essentially all job priority percentiles benefit from performing preemptions, regardless of their WSPT priority. Note that within each percentile there might well be jobs that suffer, however, on average the jobs within the percentile benefit in most cases.

But how does the ability of preempting jobs impact the jobs' service level, as captured by the bounded slowdown? As pointed out above, this is important when considering other job utility functions where the waiting costs are more important. As Figures 5.3(b), 5.3(d), 5.3(f) and 5.3(h) show, *P-DLGM* strikingly outperforms *DLGM* regarding the bounded slowdown. On average, *P-DLGM* yields a lower bounded slowdown (better service) than *DLGM* across all priority ranges. Moreover, the bounded slowdown of *P-DLGM* is almost always close to 1 (the optimum), besides a small peak in the DAS-FS4 workload. This result can be explained by the use of the (dynamic) WSPT ratios, which divide the job weight by the runtime. This generally boosts the priority of small jobs compared to long jobs. Thus, in contrast to the static *DLGM*, the WSPT ratios in conjunction with preemptions gives the mechanism the ability to suspend long jobs in favor of short jobs. Hence, *P-DLGM* tends to result in much smaller slowdowns for short jobs but only slightly larger slowdowns for long jobs, since the slowdown is normalized by the job runtime. From an overall perspective, the mean slowdown will thus be much smaller for *P-DLGM* than for *DLGM*.

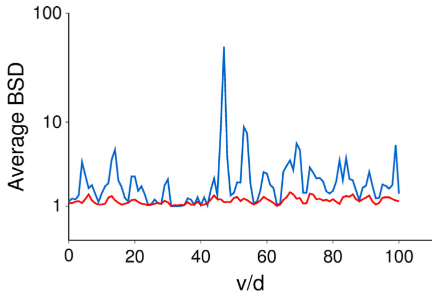
The results for the bimodal weight distribution closely resemble the results for the uniform distribution, cf. Figure 5.4. In fact, the graphs look almost identical. As pointed out above, this suggests that the overall benefit of performing preemptions is robust to the assumption about a specific weight distribution. One explanation for this robustness can be that, on average, the job weights are small compared to the job runtimes. Consequently, the jobs' WSPT priority is heavily dominated by the job runtimes and becomes less sensitive to changes in the valuation distribution.

5.6 Related Work

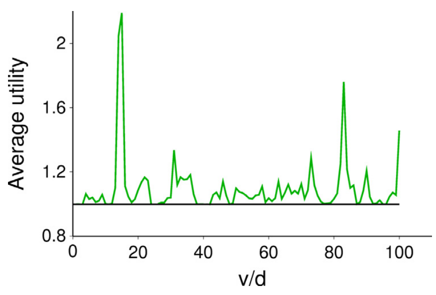
This section will first discuss related work on (technical) allocation algorithms before focusing on market mechanisms. A good overview about approximation results via *deterministic* algorithms is given in Megow (2007). For the non-preemptive setting with arbitrary job weights, release dates, and numbers of nodes ($P|r_j|\sum v_j e_j$ in the notation of this chapter), Correa and Wagner (2005) give the currently best known algorithm, which is 2.62-competitive. For the single-node setting, Goemans *et al.* (2002) obtain a 2.41-competitive algorithm. Note that Megow and Schulz (2004) and Megow (2007) also give an allocation algorithm that is 2-competitive for the



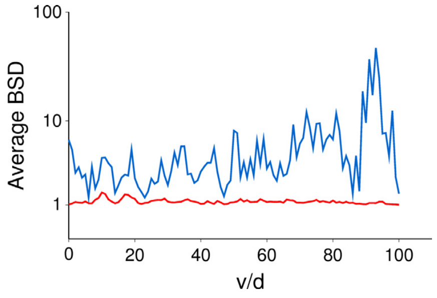
(a) WHALE Utility



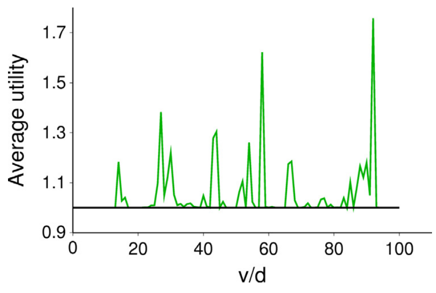
(b) WHALE BSD



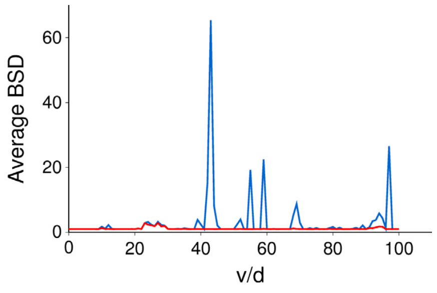
(c) REQUIN Utility



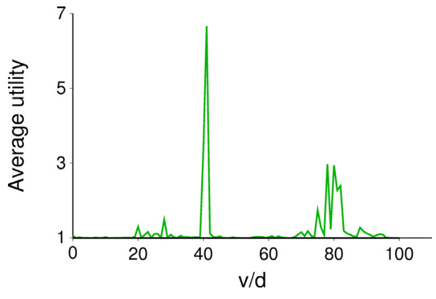
(d) REQUIN BSD



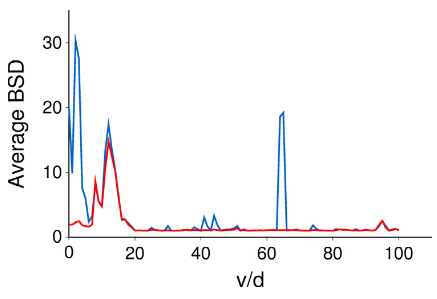
(e) LPC-EGEE-75% Utility



(f) LPC-EGEE-75% BSD



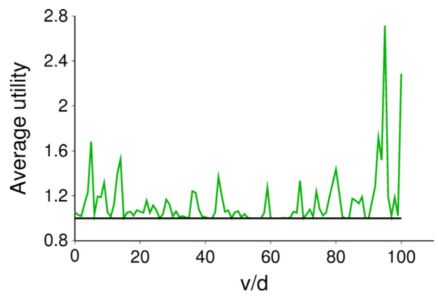
(g) DAS2-FS4-75% Utility



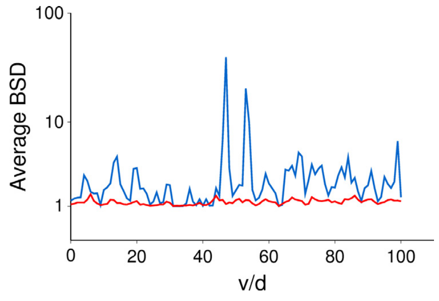
(h) DAS2-FS4-75% BSD

Figure 5.3: Left column: ratio of the average (negative) utility of *DLGM* to *P-DLGM*; right column: average bounded slowdown of *DLGM* and *P-DLGM*. v/d indicates the discretized WSPT percentiles. Both evaluations are based on the ***uniform*** weight distribution.

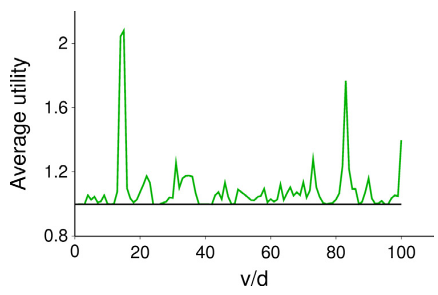
preemptive setting $P|r_j, pmtn|\sum v_j e_j$. Furthermore, they show that this bound is tight. However, they use *static priorities* when ordering jobs which are independent of the jobs' progress and



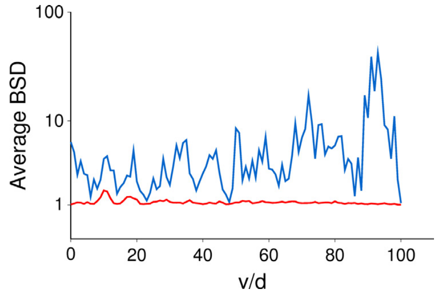
(a) WHALE Utility



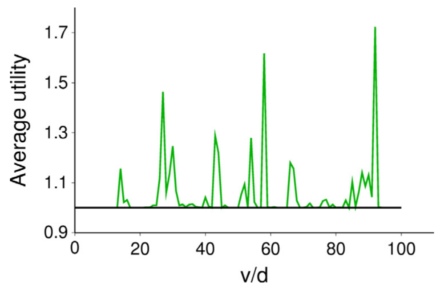
(b) WHALE BSD



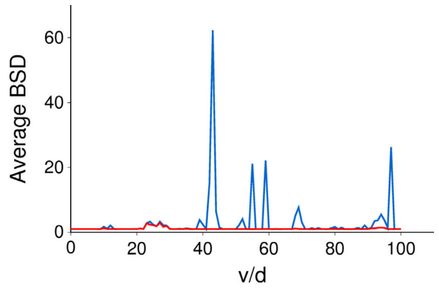
(c) REQUIN Utility



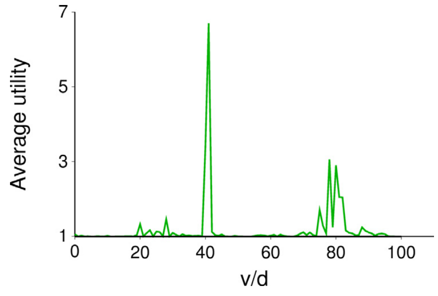
(d) REQUIN BSD



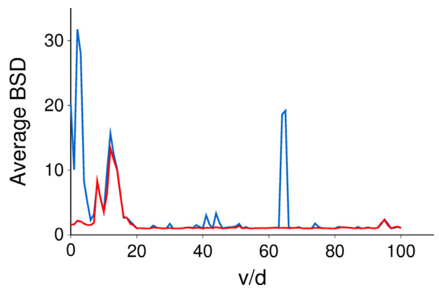
(e) LPC-EGEE-75% Utility



(f) LPC-EGEE-75% BSD



(g) DAS2-FS4-75% Utility



(h) DAS2-FS4-75% BSD

Figure 5.4: Left column: ratio of the average (negative) utility of *DLGM* to *P-DLGM*; right column: average bounded slowdown of *DLGM* and *P-DLGM*. v/d indicates the discretized WSPT percentiles. Both evaluations are based on the ***bimodal*** weight distribution.

the allocation algorithm is *centralized* as opposed to the decentralized setting used in this work. Most importantly, the latter leads to Megow and Schulz using migration (i.e. the moving of jobs across nodes) whereas *P-DLGM* only uses preemption (i.e. suspended jobs are continued on the same node). Megow (2007) proposes an algorithm that uses dynamic WSPT priorities as proposed in this work and shows that it is 2-competitive for the single-node case. Again, in this chapter a decentralized variant of this algorithm was proposed. Moreover, it was shown that the resulting market mechanism is not only 2-competitive for the single-node case but for arbitrary numbers of nodes. However, it is not clear whether this bound is tight, as the proof of the tightness of the algorithm of Megow and Schulz (2004) and Megow (2007) does not hold with dynamic WSPT priorities. It is important to note that this previous research only focuses on the allocation algorithm and does not give a pricing scheme to complement this algorithm.

In Harchol-Balter and Downey (1997), the authors showed that performing preemptive migration is more effective than static scheduling as regards the mean slowdown. Moreover, the authors argue that the mean slowdown understates the advantages of the preemptive strategy from the viewpoint of severely suffering jobs, and suggest to also use more fine-grained metrics such as the variance of slowdown and the number of severely slowed jobs. In this chapter, these metrics were extended to take into account the valuations of jobs.

As presented in the preliminaries in Chapter 2, online market mechanisms can be distinguished into mechanisms which allocate *shares* of one or more divisible goods such as bandwidth or computing power (e.g. Chun and Culler (2000); Sanghavi and Hajek (2004)) and, similar to the approach in this chapter, mechanisms that allocate *indivisible nodes*. As such, strategic behavior of users is considered by Friedman and Parkes (2003) for the allocation of bandwidth. The paper elaborates online variants of the prominent VCG mechanism to induce the users to truthfully reveal their valuations and release dates. Lavi and Nisan (2005) studied "Set-Nash" equilibria mechanisms and also showed that no ex post dominant-strategy implementation can obtain a constant fraction of the optimum. Porter (2004) is most similar to the spirit of this chapter in that it addresses the issue of preemption in economic online settings. The objective of the mechanism is to schedule strategic jobs with *deadlines* onto *one single node* so as to induce these users to truthful reports of their characteristics and to maximize the overall value returned to the users. The benefit of migration was studied in Amar *et al.* (2008b) in the context of online fair allocations in heterogenous grids: under mild assumptions, it was shown that several natural fairness and quality of service properties cannot be achieved without the ability to preempt jobs during runtime. Extensive numerical experiments are performed with real-world workloads and varying degrees of realistic migration cost. The experiments show that the performance of a fair (according to the defined criteria) scheduling algorithm is robust to even significant migration cost.

Additionally, many recent results studied the limitations of the dominant strategy approach on various scheduling settings (e.g. Christodoulou *et al.* (2007) and Mu'alem and Schapira (2007)).

This suggests that other notions of implementation should be studied. In this chapter "myopic" best responses were studied where users do not have information about the future.

In Chun and Culler (2002), the authors also find that economic scheduling algorithms improve the system performance from the users' viewpoint. They conclude that introducing a limited preemption model (in which a job is preempted at most once) does not significantly improve the overall performance. The paper does not give sufficient details about the simulation setting, e.g. the level of competition in the artificially generated workloads. More importantly, it only considers overall performance as opposed to the intimate connection of the individual performance and the valuation of single jobs as well as the predictability of the service level.

5.7 Discussion

This chapter investigated the benefit of performing preemptions in economic settings where users have time-dependent valuations. By focusing on Heydenreich et al.'s *DLGM* mechanism, it was shown that both the worst case as well as the average case economic performance of online mechanisms can be significantly improved by introducing preemptions. Virtualization technologies provide off-the-shelf support for virtual machine migration, thus making the use of preemption and migration more accessible. The results suggest that designers of distributed systems should make full use of this feature to build in more flexible and efficient allocation and pricing mechanisms. *DLGM* and *P-DLGM* have been integrated into *MOSIX*, a cluster and grid management system that supports preemptive migration and is functional at the Hebrew University in Jerusalem (Barak *et al.*, 2005).

A natural extension to preemption is migration, i.e. the moving of jobs *across* nodes during runtime instead of only the suspension and continuation on one single node. Migration will allow for still more efficient mechanisms and might also help in introducing stronger game-theoretic solution concepts (e.g. in dominant strategies) in some settings. However, this would require the introduction of migration cost since jobs are moved across the network, whereas in this chapter jobs have only been suspended in local memory or on disk (cf. Amar *et al.* (2008b)). Additionally it will be interesting to consider settings in which the nodes are paid for executing the jobs. It would also be interesting to extend the analytic and empirical results to other job strategies, e.g. based on Q-Learning (Phelps, 2007).

Chapter 6

Allocating and Pricing Shared Resources

6.1 Introduction

In Chapter 1, the vision of utility computing has briefly been introduced. One aspect of utility computing is the trend back towards centralizing computing services. In contrast to grid computing where essentially every user can also become a resource provider over time, in utility computing settings, users are serviced by few dedicated service providers such as Amazon or Sun Microsystems. Within such a centralized resource pool of one provider, the management of resources and applications becomes simpler. The resources are homogeneous and connected by a high-speed network. Moreover, virtualization technologies allow to dynamically adapt the system to changes in demand, e.g. by dynamically adjusting the size of virtual machine images regarding CPU, memory and bandwidth, or by moving virtual machines across physical machines. Ultimately, the resources might be viewed as *shared* resources that can be accessed by multiple users at the same time.

The aim of this chapter is to review and extend existing market mechanisms in the light of this scenario. The contributions of this chapter are the following:

- **Mechanism design:** From an economic point of view, a discriminatory pay-as-bid market mechanism by Sanghavi and Hajek (2004) is presented. By means of a game-theoretic analysis, this mechanism is compared to (market-based) Proportional Share, arguably the currently most prominent market mechanism for computing systems. Conditions are derived under which the Sanghavi-Hajek mechanism outperforms Proportional Share regarding both the provider's surplus and the allocative efficiency for the case of two users.

  For larger settings, the mechanisms are compared by means of an agent-based simulation based on a real-world workload trace. However, the basic mechanism does not allow for users to obtain service level guarantees for business-critical applications. Therefore, an extension to the basic mechanism is proposed that allows users to choose between (1) a fixed price and a dynamic (uncertain) service level or (2) a dynamic price and a guaranteed service level.

- **Integration into state-of-the-art service platforms:** From a technical perspective, it is shown that this mechanism is not purely a theoretical construct but that it can be integrated into state-of-the-art computing platforms to economically enrich the current allocation procedure. The basic design considerations are illustrated for the case of the N1 Grid Engine, the basis of Sun Microsystems's *network.com* platform.

6.2 Related Work

The market mechanisms that were presented in the previous chapters are based on periodically allocating either discrete shares of a compute node (e.g. assuming that the node can only be divided in whole CPU units, cf. Chapter 3 and Chapter 4) or dedicated nodes only (cf. Chapter 5). Moreover, with an allocation rule mainly based on economic reasoning, all available resources would be given to the resource requests[1] with the highest valuations. Technically, this could result in starvation[2] and in unstable systems[3], which needs to be considered. Combining the economic and the technical aspects, it can be desirable to give "better" service to high-value requests but to also allot at least "some" service to requests of low value.

Fundamentally different in their approach are mechanisms that assign *continuous resource shares* to applications based on one-dimensional input only, e.g. single scalars that represent the users' valuations. Proportional Share is currently the most prominent proxy of such mechanisms (Stoica *et al.*, 1996; Chun and Culler, 2000; Lai *et al.*, 2005). The Sanghavi-Hajek mechanism (Sanghavi and Hajek, 2004), however, can improve on Proportional Share with respect to efficiency and the provider's revenue. The setting and these two mechanisms will now be presented in more detail.

[1]In this chapter, the term "request" is used instead of the term "job" as the former does not necessarily need to be fully satisfied with respect to its requested resources if it is allocated.

[2]In scheduling theory, starvation denotes the fact that low-priority processes are prevented from doing any progress because all resources are assigned to other higher-value processes.

[3]Note that the whole allocation might need to be adjusted even if there has only been a marginal increase in a user's bid.

6.2.1 The Setting

Let vector $\tilde{v} = (\tilde{v}_1, \ldots, \tilde{v}_n)$ represent the positive reported valuations ("bids") of the users. The users receive shares $x = (x_1, \ldots, x_n)$, $x_j \in \mathbb{R}_+$, $\sum_{j=1}^{n} x_j = 1$, of the perfectly divisible good according to some specific allocation algorithm τ. Thus, $x_j = \tau_j(\tilde{v})$ is the quantity user j is allocated for a given bid vector $\tilde{v}$.

This chapter considers *pay-as-bid* mechanisms where user j pays the bid $\tilde{v}_j$ regardless of the allocation $\tau_j(\tilde{v})$. The mechanism must be *valid* in the sense that τ must be smooth, symmetric, and scale free (Sanghavi and Hajek, 2004):

- Smooth: $\tau_j(\tilde{v}_j, \tilde{v}_{-j})$ is differentiable, increasing, and concave in $\tilde{v}_j$.
- Symmetric: The allocation to a specific user does not change if the bid vector $\tilde{v}$ is transformed by a permutation.
- Scale free: The allocations to the users do not change if all bids are scaled by the same factor.

The prominent Vickrey auction is not smooth since its allocation rule is not continuous; it has a jump discontinuity at the highest bid.

As is common in mechanism design, the users are assumed to have quasi-linear utility functions:

$$u_j(\tilde{v}_j, \tau_j(\tilde{v}) | v_j) = v_j \tau_j(\tilde{v}) - \tilde{v}_j,$$

with $v_j \in \mathbb{R}_+$ being the user's true valuation for receiving the whole resource. This type of utility function has also been used by Sanghavi and Hajek (2004), Johari and Tsitsiklis (2004), and Lai *et al.* (2005) in equivalent settings. Consequently, j's *unit price* p_j, i.e. the price user j would have to pay if she got the whole resource unit ($x_j = \tau_j(\tilde{v}) = 1$), amounts to $p_j(\tilde{v}_j, \tau_j(\tilde{v})) = \frac{\tilde{v}_j}{\tau_j(\tilde{v})}$.

It is assumed that each user can make full use of the resource, i.e. there is no idle share of the resource even if there is only one user in the system. Let $u_P(\tilde{v}) = \sum_j \tilde{v}_j$ be the provider's utility function, which only depends on the users' bids and equals the sum across all user payments. It is assumed that the provider has a zero reservation price.

6.2.2 Proportional Share

Proportional Share allocates shares of unequal size, hence accounting for varying importance among users. Whereas scheduling according to pre-set, fixed shares for different users remains technical, *market-based* Proportional Share mechanisms dynamically base the resource share on the users' bids. The total amount of available resources is distributed among the users according to their bids' fraction of the overall bid: User (request) j with bid $\tilde{v}_j$ receives a fraction of $\frac{\tilde{v}_j}{\sum_{k=1}^{n} \tilde{v}_k}$ of the available resource given a group of n users is competing for resource access. It is easy to see that Proportional Share has a valid allocation algorithm according to the criteria above.

Systems using Proportional Share as allocation scheme have been proposed in Stoica *et al.* (1996), Chun and Culler (2000), and Lai *et al.* (2005). The problem with Proportional Share is that it remains allocatively inefficient, which will be illustrated in Subsection 6.3.3 by means of a numerical example. The point is that users have an incentive to shade their bids down, since the value of an increased share (resulting from a higher bid) is overcompensated by the increase in the payment.

6.2.3 The Discriminatory Pay-as-Bid Mechanism by Sanghavi and Hajek

With Proportional Share, all users pay the same unit price. In contrast, Sanghavi and Hajek (2004) propose a mechanism that gives "volume discounts" in the sense that high-bidding users pay lower unit prices.

For a given bid vector $\tilde{v} = (\tilde{v}_1, \ldots, \tilde{v}_n)$, the following allocation rule is proposed:

$$\tau_j^{sh}(\tilde{v}) = \frac{\tilde{v}_j}{\tilde{v}_{max}} \int_0^1 \prod_{k \neq j} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds,$$

with at least two users and $\tilde{v}_{max}$ being the maximum bid.

For a scenario with two users, a user l (low bidder) and a user h (high bidder), $v_l \leq v_h$, τ^{sh} simplifies to

$$\tau_l^{sh}(\tilde{v}_l, \tilde{v}_h) = \frac{\tilde{v}_l}{2\tilde{v}_h} \text{ and } \tau_h^{sh}(\tilde{v}_l, \tilde{v}_h) = 1 - \frac{\tilde{v}_l}{2\tilde{v}_h}.$$

If the pay-as-bid pricing rule is combined with the Sanghavi-Hajek allocation scheme τ^{sh}, the resulting mechanism as a whole produces *discriminatory* unit prices; the user with a lower bid pays a higher unit price than the high-bidding user.[4] This volume discount is supposed to encourage users to bid higher, and thus closer to their true valuation, compared to a scenario with uniform prices where the users can potentially benefit from shading their bids downwards.

While the Sanghavi-Hajek mechanism has been proposed for the allocation of network bandwidth, the setting essentially generalizes to the allocation of any perfectly divisible resource. In the following section, this mechanism is compared to the Proportional Share mechanism with respect to the provider's revenue and overall utility for the restricted two-user case.

[4] $p_l(\tilde{v}_l, \tau_l^{sh}(\tilde{v})) = \tilde{v}_l \frac{2\tilde{v}_h}{\tilde{v}_l} = 2\tilde{v}_h$ and $p_h(\tilde{v}_h, \tau_h^{sh}(\tilde{v})) = \frac{\tilde{v}_h}{1 - \frac{\tilde{v}_l}{2\tilde{v}_h}} = \frac{2\tilde{v}_h^2}{2\tilde{v}_h - \tilde{v}_l}$. Consequently, $p_l(.) \geq p_h(.) \Leftrightarrow 2\tilde{v}_h - \tilde{v}_l \geq \tilde{v}_h \Leftrightarrow \tilde{v}_h - \tilde{v}_l \geq 0$. This inequality holds per definition.

6.3 Comparison for Two Users

6.3.1 User Strategies and Mechanism Performance

For evaluating and comparing market mechanisms, the users' strategies, i.e. the users' reporting of $\tilde{v}$, and a metric need to be defined. For the former, the solution concept of Nash equilibria under complete information (Nash, 1951; Mas-Colell *et al.*, 1995) is chosen in line with Sanghavi and Hajek (2004) and Johari and Tsitsiklis (2004).

Definition 1 (Nash equilibrium). *A bid vector* $\tilde{v}^{NE} = (\tilde{v}_1^{NE}, \ldots, \tilde{v}_n^{NE})$ *constitutes a* Nash equilibrium *if, for all j, v_j and $\tilde{v}_j$,*

$$u_j(\tilde{v}_j^{NE}, \tau_j(\tilde{v}^{NE})|v_j) \geq u_j(\tilde{v}_j, \tau_j(\tilde{v}_1^{NE}, \ldots, \tilde{v}_{j-1}^{NE}, \tilde{v}_j, \tilde{v}_{j+1}^{NE}, \ldots, \tilde{v}_n^{NE})|v_j).$$

In a Nash equilibrium $\tilde{v}^{NE}$, no user j can benefit by unilaterally deviating from $\tilde{v}_j^{NE}$, even when given complete information about the bids of the other users.

A common metric for a mechanism's performance is its *competitive ratio* in its Nash equilibrium (also called *fractional efficiency* (Sanghavi and Hajek, 2004)):

Definition 2 (Competitive ratio). *The* competitive ratio *of the allocation mechanism τ is defined as*

$$\begin{aligned} \inf_{\{u_j\}} \inf_{\{\tilde{v}^{NE}\}} \frac{W^{\tau}(\tilde{v}^{NE}|v)}{W^{OPT}} &= \frac{\sum_j u_j(\tilde{v}_j^{NE}, \tau_j(\tilde{v}^{NE})|v_j) + u_P(\tilde{v}^{NE})}{W^{OPT}} \\ &= \frac{\sum_j \tau_j(\tilde{v}^{NE}) v_j}{W^{OPT}}, \end{aligned}$$

where W^{OPT} is the optimal efficiency if the whole resource unit is allocated to the user with the highest valuation, and $W^{\tau}(.)$ is the efficiency generated by allocation mechanism τ in its Nash equilibrium.

In this setting, the competitive ratio of a mechanism is the worst case ratio of the overall utility generated by the specific mechanism if all users play their Nash strategy $\tilde{v}_j^{NE}$, divided by the theoretical optimum W^{OPT}.[5] As shown in Johari and Tsitsiklis (2004) and Sanghavi and Hajek (2004), to analyze the competitive ratio it is sufficient to consider linear valuation functions of the form $v_j \tau_j(\tilde{v})$.

From a mechanism design perspective, the aim is to design a mechanism with maximum competitive ratio, i.e. to find the mechanism that maximizes the market's (and thus the computing

[5]Note that the uniqueness of Nash equilibria is generally not guaranteed. Consequently, the infimum over all Nash equilibria is taken for a given set of utility functions to obtain the competitive ratio.

system's) value across all users, given the specific setting at hand. For two users, the competitive ratio of τ^{sh} adds up to 87.5% (Sanghavi and Hajek, 2004). In comparison, the competitive ratio of the Proportional Share mechanism is 82.84% (cf. the proof to Lemma 4 in Johari and Tsitsiklis (2004)). Moreover, Sanghavi and Hajek (2004) show that τ^{sh} not only provides the best (worst case) competitive ratio, but that it is optimal for *any* pair of valid valuation functions[6].

For arbitrary numbers of users, it is hard to analytically determine an exact value for the worst case efficiency of the Sanghavi-Hajek mechanism. Instead, Sanghavi and Hajek (2004) numerically calculate the interval [87.03%, 87.5%] as bounds for the general competitive ratio. The proposed mechanism is still close to the theoretical maximum worst case efficiency of 87.5% (Sanghavi and Hajek, 2004). But a guarantee that the mechanism τ^{sh} is the optimal one can no longer be given. In contrast, for general numbers of users, Proportional Share was shown to have a competitive ratio of 75% (Johari and Tsitsiklis, 2004).

These previous results focus solely on efficiency. Especially in the case where the market outcome is determined by a centralized provider and not by an independent intermediary, another important performance dimension of the mechanism is the provider's revenue. In order to derive more general results, Sanghavi and Hajek (2004) and Johari and Tsitsiklis (2004) base their analyses on the general structure of the mechanisms' Nash equilibria without explicitly determining the users' bids in these equilibria.[7] However, measuring and comparing the mechanisms' revenue requires an explicit model of the users' strategies dependent on their requests' characteristics and the mechanism. The focus of the following analysis will thus be on deriving the equilibrium strategies for the two mechanisms at hand and comparing the resulting provider's revenue. Additionally, the results with respect to efficiency confirm the results of Sanghavi and Hajek (2004) about the optimality of their mechanism in the two-user case.

6.3.2 Analytical Comparison: User Strategies, Provider's Revenue, and Efficiency

This subsection will first develop an explicit model of the users' strategies and the resulting market outcome before evaluating the mechanism performance.

The following lemma confirms that there is no "distorted" Nash equilibrium for the Sanghavi-Hajek mechanism in the sense that the user with the lower valuation bids more than the user with the higher valuation.

[6]By *valid valuation function*, Sanghavi and Hajek (2004) refer to valuation functions that are differentiable, concave, and strictly increasing.

[7]For instance, Sanghavi and Hajek (2004) (cf. the proof to Lemma 3) show that for all Nash equilibria $\tilde{v}^{NE}$ the following condition must hold: $\frac{\partial u_l(.)}{\partial \tau_l(\tilde{v}^{NE})} / \frac{\partial u_h(.)}{\partial \tau_h(\tilde{v}^{NE})} = \tilde{v}_l^{NE} / \tilde{v}_h^{NE}$.

Lemma 1 (No "distorted" Nash equilibria). *In the Nash equilibrium $\tilde{v}^{sh}$ of the pay-as-bid mechanism τ^{sh}, the user l does not bid more than user h ($\tilde{v}_l^{sh} \leq \tilde{v}_h^{sh}$).*

Proof. Assume $\tilde{v}_l^{sh} > \tilde{v}_h^{sh}$. Hence user l gets allocated $1 - \frac{\tilde{v}_h^{sh}}{2\tilde{v}_l^{sh}}$ and user h gets allocated $\frac{\tilde{v}_h^{sh}}{2\tilde{v}_l^{sh}}$. In the Nash equilibrium,

$$\frac{\partial u_j(\tilde{v}_j^{sh}, \tau_j^{sh}(\tilde{v}^{sh})|v_j)}{\partial \tilde{v}_j^{sh}} = 0, \ j = l, h.$$

Consequently,

$$\frac{\partial u_h(\tilde{v}_h^{sh}, \tau_h^{sh}(\tilde{v}^{sh})|v_h)}{\partial \tilde{v}_h^{sh}} = \frac{v_h}{2\tilde{v}_l^{sh}} - 1 = 0 \Leftrightarrow \tilde{v}_l^{sh} = \frac{v_h}{2}$$

and

$$\frac{\partial u_l(\tilde{v}_l^{sh}, \tau_l^{sh}(\tilde{v}^{sh})|v_l)}{\partial \tilde{v}_l^{sh}} = \frac{v_l \tilde{v}_h^{sh}}{2(\tilde{v}_l^{sh})^2} - 1 = 0 \Leftrightarrow \tilde{v}_h^{sh} = \frac{2}{v_l}\frac{v_h}{2}\tilde{v}_l^{sh} = \frac{v_h}{v_l}\tilde{v}_l^{sh} \geq \tilde{v}_l^{sh},$$

a contradiction. □

In the following, it can thus be assumed that the user with the high valuation indeed reports the higher valuation. The strategies and outcomes can then be determined as follows:

Lemma 2 (Nash equilibrium with the Sanghavi-Hajek mechanism). *In the Nash equilibrium $\tilde{v}^{sh}$ of the Sanghavi-Hajek mechanism τ^{sh}, user l bids $\tilde{v}_l^{sh} = \frac{v_l^2}{2v_h}$ and receives a share of $\tau_l^{sh}(\tilde{v}^{sh}) = \frac{v_l}{2v_h}$, whereas user h bids $\tilde{v}_h^{sh} = \frac{v_l}{2}$, thus receiving $\tau_h^{sh}(\tilde{v}^{sh}) = 1 - \frac{v_l}{2v_h}$.*

Proof.

$$\frac{\partial u_l(\tilde{v}_l^{sh}, \tau_l^{sh}(\tilde{v}^{sh})|v_l)}{\partial \tilde{v}_l^{sh}} = \frac{v_l}{2\tilde{v}_h^{sh}} - 1 = 0 \Leftrightarrow \tilde{v}_h^{sh} = \frac{v_l}{2}$$

and

$$\frac{\partial u_h(\tilde{v}_h^{sh}, \tau_h^{sh}(\tilde{v}^{sh})|v_h)}{\partial \tilde{v}_h^{sh}} = \frac{v_h \tilde{v}_l^{sh}}{2(\tilde{v}_h^{sh})^2} - 1 = 0 \Leftrightarrow \tilde{v}_l^{sh} = \frac{2}{v_h}\left(\frac{v_l}{2}\right)^2 \Leftrightarrow \tilde{v}_l^{sh} = \frac{v_l^2}{2v_h}.$$

Moreover,

$$\frac{\partial^2 u_h(\tilde{v}_h^{sh}, \tau_h^{sh}(\tilde{v}^{sh})|v_h)}{\partial (\tilde{v}_h^{sh})^2} = -\frac{v_h \tilde{v}_l^{sh}}{(\tilde{v}_h^{sh})^3} < 0.$$

To see that $\tilde{v}_l = \tilde{v}_l^{sh}$ indeed maximizes $u_l(\tilde{v}_l, \tau_l^{sh}(\tilde{v}_l, \tilde{v}_h^{sh})|v_l)$, consider two cases:

- $\tilde{v}_l \leq \tilde{v}_h^{sh} \Rightarrow u_l(\tilde{v}_l, \tau_l^{sh}(\tilde{v}_l, \tilde{v}_h^{sh})|v_l) = \frac{v_l \tilde{v}_l}{2\tilde{v}_h^{sh}} - \tilde{v}_l = 0 = u_l(\tilde{v}_l^{sh}, \tau_l^{sh}(\tilde{v}^{sh})|v_l)$.
- $\tilde{v}_l > \tilde{v}_h^{sh} = \frac{v_l}{2} \geq \tilde{v}_l^{sh} \Rightarrow u_l(\tilde{v}_l, \tau_l^{sh}(\tilde{v}_l, \tilde{v}_h^{sh})|v_l) = v_l - \frac{v_l^2}{4\tilde{v}_l} - \tilde{v}_l$. Hence,

$$\frac{\partial u_l(.)}{\partial \tilde{v}_l} = \frac{v_l^2}{4\tilde{v}_l^2} - 1 < \frac{v_l^2}{4(\frac{v_l}{2})^2} - 1 = 0.$$

Consequently, the user l cannot increase her utility by deviating and $\tilde{v}_l = \tilde{v}_l^{sh}$ maximizes $u_l(.)$ for $\tilde{v}_h = \tilde{v}_h^{sh}$.

Inserting $\tilde{v}^{sh}$ into τ^{sh} directly yields $\tau_l^{sh}(\tilde{v}^{sh}) = \frac{v_l}{2v_h}$ and $\tau_h^{sh}(\tilde{v}^{sh}) = 1 - \frac{v_l}{2v_h}$. □

The mechanism τ^{sh} allocates the resource shares in such a way that, in the Nash equilibrium $\tilde{v}^{sh}$, the low-bidding user l is pushed to zero utility:

$$u_l(\tilde{v}_l^{sh}, \tau_l(\tilde{v}^{sh})|v_l) = v_l \tau_l(\tilde{v}^{sh}) - \tilde{v}_l^{sh} = v_l \frac{v_l}{2v_h} - \frac{v_l^2}{2v_h} = 0.$$

The high-bidding user h obtains utility of

$$u_h(\tilde{v}_h^{sh}, \tau_h(\tilde{v}^{sh})|v_h) = v_h \tau_h(\tilde{v}^{sh}) - \tilde{v}_h^{sh} = v_h - v_l,$$

while the provider's revenue amounts to

$$\tilde{v}_l^{sh} + \tilde{v}_h^{sh} = \frac{v_l^2 + v_l v_h}{2v_h}.$$

The central result of this analysis is that, from a resource provider's point of view, the Sanghavi-Hajek mechanism dominates Proportional Share under certain conditions with respect to the provider's revenue. To be able to compare the results of both mechanisms, first the provider's revenue is computed that is generated by Proportional Share in the Nash equilibrium. In doing so, it is assumed that, as in the mechanism by Sanghavi and Hajek, the Proportional Share allocation rule is complemented by the pay-as-bid pricing rule, as proposed in Chun and Culler (2000) and Johari and Tsitsiklis (2004).

Lemma 3 (Nash equilibrium with Proportional Share). *In the Nash equilibrium $\tilde{v}^{ps}$ of the Proportional Share mechanism τ^{ps}, user l bids $\tilde{v}_l^{ps} = \frac{v_l^2 v_h}{(v_l+v_h)^2}$ and receives a share of $\tau_l^{ps}(\tilde{v}^{ps}) = \frac{v_l}{v_l+v_h}$, whereas user h bids $\tilde{v}_h^{ps} = \frac{v_l v_h^2}{(v_l+v_h)^2}$, thus receiving $\tau_h^{ps}(\tilde{v}^{ps}) = \frac{v_h}{v_l+v_h}$.*

Proof. In the Nash equilibrium $\tilde{v}^{ps}$,

$$\frac{\partial u_l(\tilde{v}_l^{ps}, \tau_l^{ps}(\tilde{v}^{ps})|v_l)}{\partial \tilde{v}_l^{ps}} = v_l \frac{\tilde{v}_h^{ps}}{(\tilde{v}_l^{ps} + \tilde{v}_h^{ps})^2} - 1 = 0$$

$$\Leftrightarrow \sqrt{v_l \tilde{v}_h^{ps}} = \tilde{v}_l^{ps} + \tilde{v}_h^{ps} \Leftrightarrow \tilde{v}_l^{ps} = \sqrt{v_l \tilde{v}_h^{ps}} - \tilde{v}_h^{ps}.$$

(Note that $v_i, \tilde{v}_i^{ps} > 0$). Analogously,

$$\frac{\partial u_h(\tilde{v}_h^{ps}, \tau_h^{ps}(\tilde{v}^{ps})|v_h)}{\partial \tilde{v}_h^{ps}} = v_h \frac{\tilde{v}_l^{ps}}{(\tilde{v}_l^{ps} + \tilde{v}_h^{ps})^2} - 1 = 0 \Leftrightarrow v_h \tilde{v}_l^{ps} = (\tilde{v}_l^{ps} + \tilde{v}_h^{ps})^2$$

$$\Leftrightarrow v_h\left(\sqrt{v_l\tilde{v}_h^{ps}}-\tilde{v}_h^{ps}\right)=v_l\tilde{v}_h^{ps}\Leftrightarrow \tilde{v}_h^{ps}=v_l\left(\frac{v_h}{v_l+v_h}\right)^2=\frac{v_lv_h^2}{(v_l+v_h)^2}.$$

Inserting $\tilde{v}_h^{ps}$ above directly yields

$$\tilde{v}_l^{ps}=\frac{v_lv_h}{v_l+v_h}-\frac{v_lv_h^2}{(v_l+v_h)^2}=\frac{v_l^2v_h}{(v_l+v_h)^2}.$$

Moreover,

$$\frac{\partial^2 u_j(\tilde{v}_j,\tau_j^{ps}(\tilde{v})|v_j)}{\partial\tilde{v}_j^2}=-\frac{2\tilde{v}_iv_j}{(\tilde{v}_l+\tilde{v}_h)^3}<0,\ j=l,h.$$

Hence, $\tilde{v}_j=\tilde{v}_j^{ps}$ indeed maximizes $u_j(\tilde{v}_j,\tau_j^{ps}(\tilde{v})|v_j)$ for $j=l,h$.

Inserting $\tilde{v}_l^{ps}$ and $\tilde{v}_h^{ps}$ into τ^{ps} returns

$$\tau_l^{ps}(\tilde{v}^{ps})=\frac{\sqrt{v_l\tilde{v}_h^{ps}}-\tilde{v}_h^{ps}}{\sqrt{v_l\tilde{v}_h^{ps}}}=1-\sqrt{\frac{\tilde{v}_h^{ps}}{v_l}}=1-\frac{v_h}{v_l+v_h}=\frac{v_l}{v_l+v_h}$$

and

$$\tau_h^{ps}(\tilde{v}^{ps})=1-\tau_l^{ps}(\tilde{v}^{ps})=\frac{v_h}{v_l+v_h}.$$

□

Consequently, Proportional Share generates provider's revenue of $\tilde{v}_l^{ps}+\tilde{v}_h^{ps}=\sqrt{v_l\tilde{v}_h^{ps}}=\frac{v_lv_h}{v_l+v_h}$. Based on Lemma 2 and Lemma 3, the first central result can be stated:

Theorem 1 (Revenue comparison). *For two users with linear valuation functions with slopes v_l and v_h, $v_l,v_h\in\mathbb{R}_+$ and $v_l\le v_h$, in the unique Nash equilibria $\tilde{v}^{sh}$ and $\tilde{v}^{ps}$ the Sanghavi-Hajek mechanism generates larger provider's revenue than Proportional Share iff*

$$v_l>(\sqrt{2}-1)v_h.$$

Proof.

$$\begin{aligned}
&\frac{v_l^2+v_lv_h}{2v_h}-\frac{v_lv_h}{v_l+v_h}>0\Leftrightarrow\frac{(v_l+v_h)(v_l^2+v_lv_h)-2v_lv_h^2}{2v_h(v_l+v_h)}>0\\
\Leftrightarrow\ & v_l^3+2v_l^2v_h-v_lv_h^2>0\Leftrightarrow v_l(v_l^2+2v_lv_h-v_h^2)>0\\
\Leftrightarrow\ & v_l^2+2v_lv_h-v_h^2>0\Leftrightarrow(v_l+v_h)^2>2v_h^2\\
\Leftrightarrow\ & v_l>(\sqrt{2}-1)v_h.
\end{aligned}$$

□

Finally, these results can be used to also assess the allocative efficiency that is generated in the Nash equilibrium for two users, and in line with the results of Sanghavi and Hajek (2004) the following theorem can be stated:

Theorem 2 (Efficiency comparison). *For two users with linear valuation functions with slopes v_l and v_h, $v_l, v_h \in \mathbb{R}_+$ and $v_l \leq v_h$, in the unique Nash equilibria $\tilde{v}^{sh}$ and $\tilde{v}^{ps}$ the Sanghavi-Hajek mechanism generates an equal or larger efficiency than Proportional Share ($W^{sh}(\tilde{v}^{sh}|v) \geq W^{ps}(\tilde{v}^{ps}|v)$) for all combinations of (v_l, v_h).*

Proof. From the previous results, it is known that

$$W^{sh}(\tilde{v}^{sh}|v) = u_l(.) + u_h(.) + u_P(.) = 0 + v_h - v_l + \frac{v_l^2 + v_l v_h}{2v_h} = \frac{v_l^2 + 2v_h^2 - v_l v_h}{2v_h}$$

and

$$W^{ps}(\tilde{v}^{ps}|v) = v_l \tau_l^{ps}(\tilde{v}^{ps}) + v_h \tau_h^{ps}(\tilde{v}^{ps}) = \frac{v_l^2 + v_h^2}{v_l + v_h}.$$

Thus,

$$W^{sh}(\tilde{v}^{sh}|v) \geq W^{ps}(\tilde{v}^{ps}|v) \Leftrightarrow \frac{v_l^2 + 2v_h^2 - v_l v_h}{2v_h} - \frac{v_l^2 + v_h^2}{v_l + v_h} \geq 0$$
$$\Leftrightarrow v_l^2 + v_h^2 \geq 2 v_l v_h \Leftrightarrow (v_l - v_h)^2 \geq 0.$$

□

Consequently, the discriminatory pay-as-bid mechanism of Sanghavi and Hajek (2004) not only provides a better competitive ratio (i.e. worst case bound) than Proportional Share, but it outperforms the latter *independently* of v_l and v_h.

6.3.3 A Numerical Example

In this subsection, Proportional Share and the Sanghavi-Hajek mechanism are compared with respect to their allocation and the resulting efficiencies by means of a simple numerical example. There are two divisions within a company – divisions L and H – that submit requests to a shared pool of computing resources. Division H temporarily demands more resources than division L, which is shown by a higher valuation for the computing resources: $u_L(\tilde{v}_L, \tau_L(\tilde{v})) = 2.1 \cdot \tau_L(\tilde{v}) - \tilde{v}_L$ and $u_H(\tilde{v}_H, \tau_H(\tilde{v})) = 5 \cdot \tau_H(\tilde{v}) - \tilde{v}_H$. The provider's utility function is given by $u_P(\tilde{v}) = \tilde{v}_L + \tilde{v}_H$. Each division sends one resource request to the central market-based scheduler. The allocation of resource shares to requests and the resulting prices and utilities with both market-based mechanisms are listed in Table 6.1.

Whereas in this example the provider's revenue created by the Sanghavi-Hajek mechanism (\$1.491) is only slightly higher than with the Proportional Share mechanism (\$1.478), overall

	Proportional Share	Sanghavi-Hajek
Bids in Nash equilibrium	$\tilde{v}_L^{ps} = 0.437$, $\tilde{v}_H^{ps} = 1.041$	$\tilde{v}_L^{sh} = 0.441$, $\tilde{v}_H^{sh} = 1.05$
Allocations (shares)	$x_L^{ps} = 0.296$, $x_H^{ps} = 0.704$	$x_L^{sh} = 0.21$, $x_H^{sh} = 0.79$
Unit prices	$p_L^{ps}(.) = p_H^{ps}(.) = 1.478$	$p_L^{sh}(.) = 2.1$, $p_H^{sh}(.) = 1.329$
Utilities of the users	$u_L(.) = 0.185$, $u_H(.) = 2.479$	$u_L(.) = 0$, $u_H(.) = 2.9$
Provider's revenue	1.478	1.491
Efficiency	$W^{ps}(\tilde{v}^{ps}\|v) = 4.142$ (82.84%)	$W^{sh}(\tilde{v}^{sh}\|v) = 4.391$ (87.82%)
Theoretical optimum	$W^{OPT} = v_H = 5$	

Table 6.1: Numerical example.

utility increases by about 6% from $W^{ps}(\tilde{v}^{ps}|v) = \$4.142$ to $W^{sh}(\tilde{v}^{sh}|v) = \$4.391$. While both the high bidder and the provider benefit from switching to the Sanghavi-Hajek mechanism, this gain mainly comes at the expense of the low bidder, whose utility becomes zero, leaving him indifferent between participating or not (as analytically shown above).

In certain cases, the provider might be willing to actually sacrifice revenue in return for higher overall utility. Attributing the resources to the division with the higher valuation might be more important than creating revenue from internal sources. Especially in cases where the provider's revenue increases, an option might be to "subsidize" the lower bidder in order to convince this bidder to participate in the Sanghavi-Hajek mechanism. It would be an interesting question for future research to explore how such an incentive can be implemented in the Sanghavi-Hajek mechanism and how it changes the users' strategic considerations.

6.3.4 Intermediate Summary

The preceding analysis has shown that the Sanghavi-Hajek mechanism can improve on Proportional Share with respect to both efficiency and the provider's revenue in a simplified scenario with two users with one unit-size request each. The complexity of more general settings defies a further analytical evaluation. Indeed, as pointed out above, the competitive ratio of the Sanghavi-Hajek mechanism can no longer be explicitly bounded. The subsequent section hence reports the results of an agent-based evaluation on the basis of a real workload trace, using two user strategies to gain further insights into the two mechanisms' more general performance.

6.4 Comparison Based on Real-World Workloads

6.4.1 Experimental Setup

This subsection specifies and describes the setting in which the mechanisms were compared as well as the workload trace and the user strategies that were used in this comparison.

The Setting

Consider a setting where a resource is to be allocated to computing requests, which arrive over time. The resource has a "size" of $C \in \mathbb{N}$, e.g. measured in the number of available CPUs. This models the Amazon and Sun Microsystems scenario, where a total number of C CPUs is available in the server pool. It is again assumed that the resource is perfectly divisible.

Let $\theta_j = (r_j, c_j, d_j, v_j)$ be the "type" of request j. $r_j \in \mathbb{N}$ is the request's release date, i.e. the time (in seconds) when the request is submitted to the system. $c_j \in \mathbb{N}$ denotes the number of resource units that the user requests *in each timeslot*, $c_j \leq C$. $d_j \in \mathbb{N}$ is the request's duration, again in seconds. The request length is assumed to be independent of the resource share. This might be due to an application where the quality is continuous in the resource share, such as a rendering application where the achieved frames per second almost continuously increase with the resource share, or because unavailable resources are sourced from some outside option. Finally, the user request is completed by the valuation $v_j \in \mathbb{R}_+$, which is specified *per timeslot and resource unit*.

The users' utility functions used in Section 6.3 need to be generalized to match this more complex setting, as the user's valuation for a request is affected not only by the request of one single other user, but possibly by multiple competing users over the whole duration of the request. As above, quasi-linear utility functions are considered. In a given timeslot, each user receives her full valuation for her request only if she is allocated the full resource. If she does not receive the full share in this timeslot, her valuation decreases linearly. Hence, the user's ex post utility for request j is modeled as

$$u_j(\tilde{v}_j, x_j | \theta_j) = v_j \sum_{t=r_j}^{r_j+d_j-1} x_{jt} - \sum_{t=r_j}^{r_j+d_j-1} \tilde{v}_{jt} c_j,$$

where $\tilde{v}_j = (\tilde{v}_{jr_j}, \ldots, \tilde{v}_{jt}, \ldots, \tilde{v}_{j,r_j+d_j-1})$ is the bid vector – and hence $\tilde{v}_j c_j$ is the payment vector as a pay-as-bid pricing rule is used – containing j's bids per CPU across the request's duration and $x_j = (x_{ir_j}, \ldots, x_{jt}, \ldots, x_{j,r_j+d_j-1})$ is the vector containing the resulting shares (measured in allocated CPUs). This utility function reduces to the utility function used in Section 6.3 if two requests (users) with identical release dates, durations and resource requirements are considered. Compared to the case of Section 6.3 where a unit-size resource was studied, in this more general setting, the user j is allocated $x_{jt} = \min\{c_j, \tau_j(\tilde{v}_t) \cdot C\}$ units of the (perfectly divisible) resource, with $\tilde{v}_t$ being the bid vector in timeslot t.

Modeling the Demand

The experiment is based on a workload from the Parallel Workload Archive[8]. "DAS2 fs0" (henceforth DAS) is a workload from a Dutch computing cluster, which consists of 72 homogeneous compute nodes with dual core CPUs. This workload was used as it contains all request attributes that are needed for the simulation, such as the request's release date, its duration, and the number of CPUs used, except of the valuation.

The workload trace was modified as follows:

- *Reducing the size:* It was not possible to run the whole workload on a per second basis for 225,711 requests across a timeframe of one year. This was exacerbated by the need to tailor the user strategies towards the specific workload (as will be discussed in detail below), which is a computationally intensive task in itself. Consequently, only the first 1,000 requests of the DAS workload were considered.
- *Installing competition:* To analyze the market mechanisms, competition needed to be installed in the system. However, the DAS workload does not show any competition. It only contains information about the actual allocations but not the status of the waiting queue. Competition was induced by reducing the number C of available resources from 144 CPUs to 10 CPUs. As will be shown below, this creates a setting where demand is frequently below supply, but that also contains peaks where demand exceeds supply by a factor of up to six.
- *Filtering incomplete entries:* Incomplete request entries in the workload file that do not contain all necessary request attributes were filtered out.
- *Filtering excessive requests:* All requests that exceed 10 CPUs were omitted since these requests cannot be completely allocated in the modified system. Note that this was only the case for 16 requests.
- *Inducing user valuations:* Two settings were considered. In the first setting, the users have rather similar valuations. Each request was assigned a valuation per CPU and timeslot from a "narrow" uniform distribution between $\{10, \ldots, 60\}$. In the second setting, the users have more heterogeneous valuations; for each request, the valuation was drawn

[8]`http://www.cs.huji.ac.il/labs/parallel/workload/`

from a “wide” uniform distribution between $\{10,\ldots,160\}$, i.e. three times the support of the narrow valuation distribution.

Table 6.2 contains further descriptive statistics about the requests in the modified workload. Histograms of the CPU size and the runtimes of the requests are presented in Appendix D.1. The dynamicity and competition in the workload is illustrated in Figure 6.1. This graph shows that, when restricting the number of available CPUs to 10, the resulting workload exhibits various larger peaks where resource demand exceeds supply, but it also contains periods of time where there is no scarcity.

Request attribute	Min	Max	Mean	Median
CPU	2	8	2.08	2
Runtime (in sec.)	2	38,768	1,044.06	38

Table 6.2: Descriptive statistics of the workload trace.

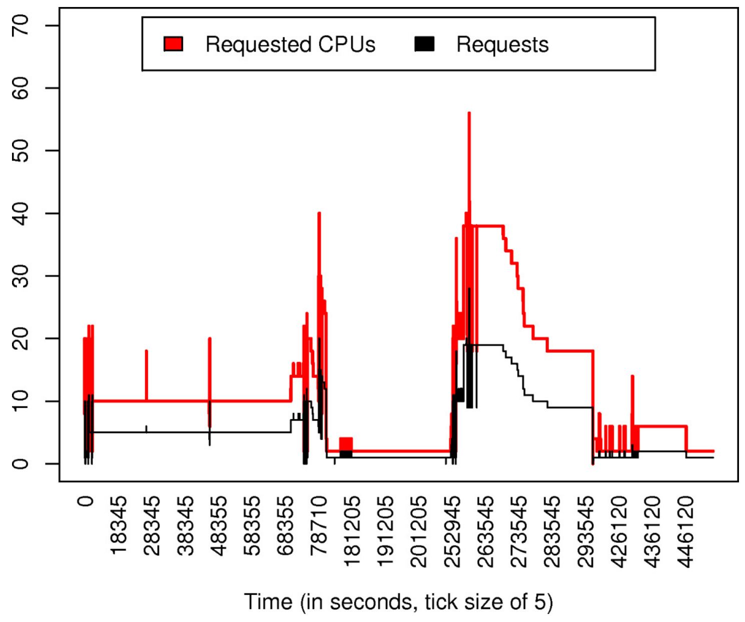


Figure 6.1: The number of requests and requested CPUs in the workload for each timeslot. Note that the graph shows the means across intervals of five timeslots. It does not include timeslots in which there was no request present.

The mechanisms’ performance might vary greatly depending on the users’ strategies. Hence, two different strategies were implemented: *Zero Intelligence Constrained* and *Q-Learning*.

Modeling the User Strategies – Zero Intelligence Constrained

Looking at the users' utility functions, users obviously only report valuations up to the true valuation v_j. Any higher bid would result in a utility loss since a pay-as-bid pricing rule is employed. With the Zero Intelligence Constrained (ZIC) strategy, users randomly report a valuation within this action space: $\tilde{v}_{jt}(v_j) = \mu_{jt}$, where $\mu_{jt} \in \{1, 2, \ldots, v_j\}$ is a discrete, uniformly distributed random variable. As Phelps (2007) reports, ZIC was able to achieve high degrees of efficiency in continuous double auctions. In its formulation, ZIC can easily be applied to the problem setting at hand.

It is important to note that strategies such as ZIC, in which the users follow a fixed strategy that is independent of the market mechanism[9], are not suited for comparing different market mechanisms. ZIC rather serves as a validity check for the implementation of the Proportional Share mechanism, the Sanghavi-Hajek mechanism, and the subsequent strategy based on reinforcement learning: Since with ZIC the users' bids are independent of the market mechanism and a pay-as-bid pricing rule is employed, both mechanisms should produce the same provider's revenue. Further, the subsequent strategy based on reinforcement learning should outperform ZIC, since otherwise the users are better off when *not* following the proposed learning algorithm.

Modeling the User Strategies – Q-Learning

Q-Learning is a prominent instantiation of reinforcement learning (see Kaelbling *et al.* (1996) for an excellent survey). It has originally been proposed by Watkins (1989) and has been shown to perform well in a wide array of applications (cf. Watkins and Dayan (1992)). The underlying reinforcement model can be briefly described as follows, cf. Figure 6.2.

The user iteratively interacts with some system. In each iteration, the user obtains an input signal s about the state of the system. Based on this input, the user determines an action a. This action changes the state of the system, and the value associated with the state change is fed back to the user by means of a reinforcement (reward) signal r. Ultimately, by means of the reinforcement signals, the user tries to learn the optimal (e.g. utility maximizing) action a^* for a given state s (the so-called "policy").

Reinforcement learning is used for approximating the optimal policy in a stationary, markovian environment. Here the probabilities of state transitions are fixed and the subsequent system state only depends on the previous state and the actions performed in the system (Kaelbling *et al.*, 1996). Q-Learning has been shown to converge to the optimal policy in a stationary,

[9]In another, similar strategy the users might bid only a fixed percentage of their true valuation, for instance. Note that with ZIC, the users bid about 50% of their true valuation on average.

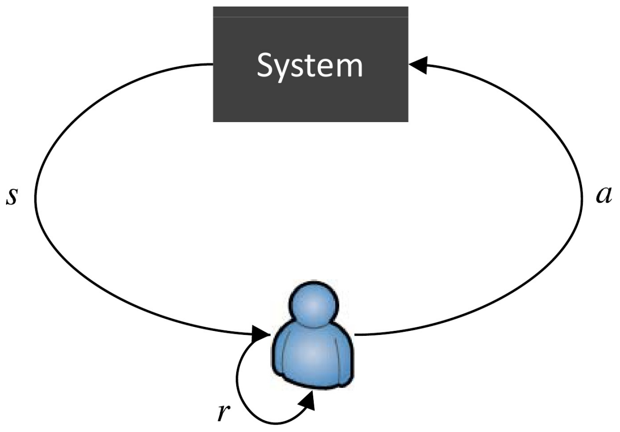


Figure 6.2: The reinforcement learning model (based on Kaelbling *et al.* (1996)).

markovian environment and under further theoretical assumptions, e.g. that each state-action pair is visited infinitely often (Watkins and Dayan, 1992). More importantly, it does not need an explicit model of the system and the learning can take place online, i.e. as the user interacts with the system (Tesauro and Kephart, 2002).

One issue with the setting considered here is that it is in fact *non*-stationary. The system changes over time as requests are finished, new requests are released, and users change their actions. Moreover, the system is only partially observable by the single users, who can assess their own state (allocation and prices) but not the system state as a whole. While the theoretical convergence guarantees of Q-Learning do not hold in this dynamic environment, this does not generally preclude the use of reinforcement learning approaches (Kaelbling *et al.* (1996); see e.g. Crites and Barto (1996), who use Q-Learning in a non-stationary, dynamic setting where the objective is to optimize elevator dispatching policies; Sandholm and Crites (1996), who use Q-Learning for the Iterated Prisoner's Dilemma; Tesauro and Kephart (2002) for the price-setting in competitive marketplaces; and Weinberg and Rosenschein (2004) for an algorithm that copes with larger numbers of players). The logic underlying the Q-Learning algorithm can still be hypothesized to model human reasoning in dynamic environments. Moreover, at a first glimpse the system might appear to be more dynamic under the used workload than it actually is. There are several long periods of time (10,000-20,000 timeslots) where the same set of requests is present in the system, cf. Figure 6.1. Further, the characteristics of requests are typically assumed to follow certain stochastic processes and distributions. This is the subject of a large body of research on workload modeling (cf. Feitelson (2002, 2008)).

The remainder of this subsection describes the design decisions that were made in implementing the Q-Learning algorithm for the setting at hand.

States and Actions

The system consists of the demand that is generated based on the modified workload trace, the fixed resource supply, the market mechanism, and the users' bidding strategies. The action space of each user consists of the bidding interval $\{1,2,\ldots,v_j\}$. The state signal corresponds to the resource share the user has received in the last bidding round (second). Since the resource share is continuous but the learning algorithm can only reasonably work with a finite number of states, for request j the resource share is discretized to ratios $\frac{x_{jt}}{c_j}$ in $\{0\%, 1-10\%, 11-20\%, \ldots, 91-100\%\}$[10], i.e. the user of the request keeps action-reward memories for 11 states. Furthermore, in order to enable the users to quickly build up a memory of the past rewards for each state and action, the action space is reduced as well by using the ratios $\frac{\tilde{v}_{jt}-\tilde{v}_{min}}{v_j-\tilde{v}_{min}}$ in $\{0\%, 1-10\%, 11-20\%, \ldots, 91-100\%\}$ with $\tilde{v}_{min}$ being the minimal admissible bid, i.e. the user keeps 11 action-reward memories for each state. Consequently, each user's "memory" consists of $11 \cdot 11$ state-action pairs.

Reinforcements

The learning process works as follows: Assume in the last bidding round t ($r_j \leq t < r_j + d_j - 1$) the user of request j bid $\tilde{v}_{jt}$ and subsequently received a (discretized) resource share of x_{jt} units. Then the user determines the reward from taking the (discretized) action $\tilde{v}_{jt}$ in state x_{jt} as

$$reward(x_{jt}, \tilde{v}_{jt} | \theta_j) = v_j x_{jt} - \tilde{v}_{jt} c_j.$$

After round t, the memory Q_j is then updated with this new information according to the update rule

$$Q_j(x,\tilde{v}) \quad \leftarrow \quad \begin{cases} (1-\alpha) \cdot Q_j(x,\tilde{v}) + \alpha \cdot [reward(x,\tilde{v}|\theta_j) + \gamma \cdot \max\limits_{k \in \{1,\ldots,v_j\}} Q_j(x',k)] \\ \qquad \text{if } x = x_{jt} \text{ and } \tilde{v} = \tilde{v}_{jt} \\ Q_j(x,\tilde{v}) \text{ else,} \end{cases}$$

where α is the learning rate according to which the new information suppresses the past experience ($0 < \alpha < 1$), γ ($0 < \gamma < 1$) is the factor for discounting future rewards, and x' is the state that followed from taking action $\tilde{v}_{jt}$ in state x_{jt} (Watkins, 1989; Watkins and Dayan, 1992).

The learning rate α determines how fast the user's actions adapt to changes in the system. If α is small, it takes the user a long time to incorporate changes in the reinforcement signal (her reward) into the Q-table. However, if α is large and the system is highly dynamic, the user incorporates short-term noise too quickly and the actions fluctuate heavily.

[10]Since the ratios are rounded to integer values, it is possible to have a ratio of zero even though each user is allocated a positive share.

The discount factor γ essentially allows a trade-off between immediate and long-term rewards. This discount factor becomes very important in non-stationary environments. In the dynamic setting considered here, the user will generally not look far ahead into the future (i.e., γ will be small) since the future state of the system is highly uncertain (Schwartz, 1993). Consequently, γ serves so as to adjust for the uncertainty in the future state of the system.

The updating is typically implemented based on lookup tables (also called the "Q-table"). The Q-table is initialized with $Q_j(x,v) = 0$ for all states x and actions v (Kaelbling, 1993).

Exploration vs. Exploitation

When designing the Q-Learning strategy, a policy must be chosen according to which the user tries to *explore* her action space in order to accumulate information about the expected outcomes of her actions, and how the user *exploits* the acquired information to benefit in the future. A so-called *epsilon-greedy* strategy is combined with a fixed number of initial explorations (Sutton and Barto, 1998).

In the first 100 bidding rounds, the user exclusively explores the state-action pairs to build up an initial perception of the system. The choice of 100 exploration rounds is due to the user having about 100 state-action pairs. After the initial 100 rounds, the user follows the epsilon-greedy strategy with $\varepsilon = 0.05$: In each round, the parameter $\beta \in [0,1]$ is randomly drawn (from a uniform distribution). If $\beta \leq \varepsilon$, the user randomly chooses her next bid from a uniform distribution over all possible actions, i.e. the user explores the action space. Otherwise, the user exploits the past experience by choosing action $\text{argmax}_k Q_j(x_{jt},k) \cdot (v_j - \tilde{v}_{min}) + \tilde{v}_{min}$.

In stationary environments, the learning rate and the exploration parameter are typically gradually lowered as time advances in order to ensure convergence of the learning process to the optimal policy. However, in this dynamic environment, continuous learning and exploration allow the user to adapt to changes in the system (and thus competition). The resulting updating and epsilon-greedy action selection rule are illustrated in Algorithm 1.

Algorithm 1 Epsilon-Greedy Q-Learning.

```
Require: Q-Table $Q_j$, allocation $x_{jt}$, bid $\tilde{v}_{jt}$, true type $\theta_j$, minimal admissible bid $\tilde{v}_{min}$, exploration counter $count$
1: $reward(x_{jt},\tilde{v}_{jt}|\theta_j) = v_j x_{jt} - \tilde{v}_{jt} c_j$ {Compute the reward from the last timeslot}
2: $Q_j(x_{jt},\tilde{v}_{jt}) = (1-\alpha)\cdot Q_j(x_{jt},\tilde{v}_{jt}) + \alpha\cdot[reward(x_{jt},\tilde{v}_{jt}|\theta_j) + \gamma\cdot \max_k Q_j(x_{j,t+1},k)]$ {Update the Q-table}
3: if $count \leq 100 \vee random(0,1) \leq \varepsilon$ then
4:     $\tilde{v}_{j,t+1} = random(0,1)\cdot(v_j - \tilde{v}_{min}) + \tilde{v}_{min}$ {Explore}
5: else
6:     $\tilde{v}_{j,t+1} = \text{argmax}_k Q_j(x_{jt},k)\cdot(v_j - \tilde{v}_{min}) + \tilde{v}_{min}$ {Exploit}
```

Dealing with Short Request Runtimes

Ideally, the user would learn separately for each request, so that the learning process takes into account the distinct attributes of this request. However, many requests in the workload only have very short runtimes (2 seconds, cf. Table 6.2). In this scenario, assigning a separate Q-table to each request is inadequate since the request runtime is too short to allow the user to build up a memory of states, actions and associated rewards specific to the request that could be exploited later on. Consequently, the requests are assigned to 10 different users so as to model a scenario where the users have different valuations while at the same time allowing the users to build up a memory (Q-table) across multiple requests. Looking at the pressure in the workload (cf. Figure 6.1), 10 users seem to be a reasonable trade-off between aggregating information across multiple requests over time while at the same time not having very many requests per user in a given timeslot (on average).

The requests are assigned to the users as follows: The uniform valuation distribution is divided into 10 intervals, e.g. $\{10,\ldots,14\}$, $\{15,\ldots,19\}$, ..., $\{50,\ldots,54\}$, $\{55,\ldots,60\}$ for the narrow valuation distribution. Each such interval is associated with exactly one user and vice versa. When the valuation for a request has been randomly drawn from the valuation distribution, this request is then assigned to the user that is associated with that valuation interval. Consequently, the users have different valuations, but the requests of one user can also have slightly different valuations. Each user maintains one single Q-table that aggregates the information across all of her requests.

Parameterizing the Q-Learning Strategy

One key issue remains: How is the Q-Learning strategy to be parameterized with respect to the learning rate α, the discount factor γ, and the exploration parameter ε so as to yield "appropriate" results? And what determines the appropriateness of a result? Based on Weidlich (2008), the following three criteria were used to compare the results of instantiations of the learning algorithm:

- **Correlation:** Do the users' bids reflect the competition in the market? The main aim in designing the learning algorithm is to induce user behavior that is somewhat rational. If aggregate resource demand is above the fixed resource supply, the average bid should go up as users compete for the scarce resources. In contrast, if aggregate demand is below supply, i.e. the resources are not scarce, the users should detect this and the bids should go down. Besides the direction in the change of the users' bids, the question is how *fast* these bids adjust to changes in the market situation and how stable the users' bids are over time if the market situation does not change.

  The correlation criterion was operationalized as follows. For a given combination of parameters, the learning algorithm was run 10 times. A larger number of runs was infeasible due to the size of the workload and the dimension of the parameter space. For each run, every 20 seconds the mean "percentage bid" was calculated for the last period under observation, as well as the mean number of requested CPUs. The percentage bid measures for each request the ratio of the reported valuation to the true valuation to allow for comparison across requests with different valuations. Then the mean Spearman correlation coefficient between resource demand and the mean percentage bids was calculated across all 10 runs in order to measure the correlation between resource scarcity and the users' bidding behavior (Fahrmeir *et al.*, 2004).

- **Rationality:** Do the users benefit from following the proposed learning process? The learning algorithm should ensure that the users benefit from its usage, i.e. maximize their individual utility.

  In order to evaluate the rationality criterion, for each run and parameter combination, the mean utility per request was calculated. Finally, the mean over all runs was computed. The aim was to search for a parameter combination that maximizes the users' mean utility.

- **Robustness:** Is the learning algorithm robust against variations in the random number seeds? Q-Learning with an epsilon-greedy based exploration and exploitation strategy is based on random choices with respect to whether the action space should be explored and if so, with which action it should be explored. An appropriate learning algorithm should be robust to changes in the random number seeds upon which these random choices are based. Simulation results can only be generalized if the algorithm is robust.

  If the learning algorithm is robust, for a given parameter combination, the market outcome across all runs should be similar. This similarity was measured by computing the coefficient of variation of the provider's revenue across all 10 runs, since this will be the key performance metric in the subsequent analysis.

Similar to Weidlich (2008), a trial and error approach was applied. The infinite parameter space was partially searched by running the learning algorithm for a specific combination of parameters, market mechanism, and input (demand). The result was then evaluated according to the three criteria above. This ultimately allowed to compare different combinations of parameters for a given market mechanism and input. The parameters have been independently varied as specified in Table 6.3. Consequently, 36 possible parameter combinations were tested and compared for both Proportional Share and the Sanghavi-Hajek mechanism and the two valuation distributions.

The results for the valuation distribution with support $\{10,\dots,160\}$ and the Proportional Share mechanism are presented in Appendices D.2, D.3 and D.4.

Parameter	Tested values
Learning rate α	$\{0.1, 0.3, 0.5, 0.7\}$
Discount γ	$\{0.3, 0.5, 0.7\}$
Exploration parameter ε	$\{0.05, 0.15, 0.25\}$

Table 6.3: Tested parameter combinations for the Q-Learning strategy.

With respect to correlation, the parameter combination $\alpha = 0.3$, $\gamma = 0.3$, and $\varepsilon = 0.05$ shows the largest correlation with the CPU demand in the system. Since the users interact with the system very frequently (on a per second basis), the fairly small exploration parameter $\varepsilon = 0.05$ is sufficient for the users to detect changes in the system. It is thus not necessary to increase the exploration parameter but this in fact reduces the convergence and thus correlation. The frequent interactions with the system also explain the relatively small learning rate α. Even at this small learning rate, the users quickly incorporate changes in the system into their Q-tables. Increasing the discount factor γ implies that users try to look ahead further into the future. This, however, reduces correlation in this setting. The reason for this might be the non-stationary system as discussed above: If the system is changing (e.g. due to finished requests or new requests arriving in the system), looking into the future becomes overly uncertain.

Figure 6.3 illustrates the mean percentage bid across all users and runs and the competition in the workload (on a logarithmic scale). The graph shows that the peaks in the users' mean percentage bids are associated with the CPU scarcity in the system. Note that an increase in demand first causes a short *drop* in the users' mean percentage bid before the bids go up. This can be explained by the fact that the Q-tables are initiated with zero values. If demand goes up and, in consequence, the users' state changes, some users might encounter state-action pairs in their Q-tables that have not yet been explored, thus initially only submitting the minimum bid. As the resource scarcity persists and the users explore these regions of the state-action space, the bids increase and the mean percentage bid reflects the peak in the resource demand.

The parameter combination $\alpha = 0.3$, $\gamma = 0.3$, and $\varepsilon = 0.05$ also maximizes the users' mean utility, which reflects the rationality criterion above. The reasoning behind the relatively small exploration parameter and discount factor are analogous to the discussion for the correlation criterion. Increasing the exploration parameter does not lead to higher correlation but only makes the users deviate from the "rational" bid; a large discount factor leads the users into making foresights that are highly uncertain.

All parameter combinations are very robust to changes in the random number seeds. The coefficient of variation is never larger than 6.5%. The parameter combination $\alpha = 0.3$, $\gamma = 0.3$, and $\varepsilon = 0.05$ only exhibits a coefficient of variation of 1.3%. Interestingly, increasing the exploration parameter (and thus the impact of the random choices) leads to less variation in the

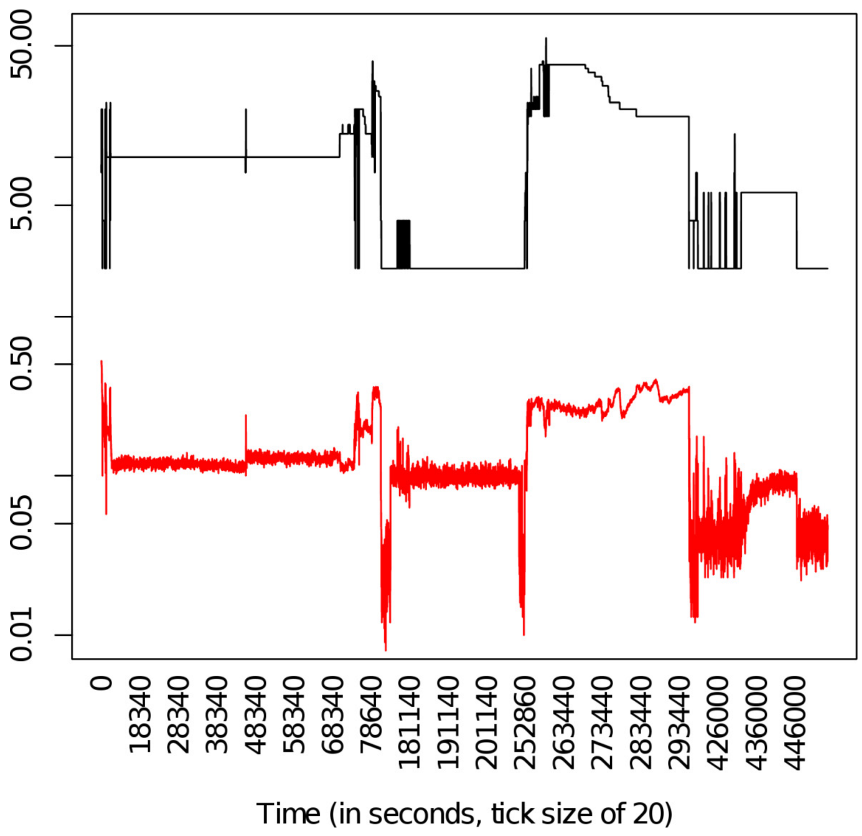


Figure 6.3: Correlation of the **competition in the workload** (measured as the total number of requested CPUs) and the **mean percentage bid across all users**. Note that the graph shows the means across intervals of 20 timeslots *on a logarithmic scale*. It does not include timeslots in which there was no request present.

overall payments. This, however, can be explained by the large number of requests and bidding rounds (more than 400,000), which in fact make the payment component, which can be attributed to the users exploring the state-action space, *more* predictable since the impact of the random choices converges to its mean.

Overall, the parameter combination $\alpha = 0.3$, $\gamma = 0.3$, and $\varepsilon = 0.05$ shows a good trade-off between the criteria presented above. It accurately reflects the situation in the market (correlation), leads to the best outcome for the users among the tested parameter combinations (rationality), and is robust to variations in the underlying random number seeds.

The results essentially hold if the Sanghavi-Hajek and / or the narrow valuation distribution are used, however the differences between the various parameter combinations become smaller. Hence, the parameter combination $\alpha = 0.3$, $\gamma = 0.3$, and $\varepsilon = 0.05$ was applied to all settings.

Further test runs showed that the results are robust against using initial rewards in the Q-tables different from zero.

6.4.2 Empirical Analysis

The aim of this experiment is to compare the proposed user strategies as well as the resulting performance of Proportional Share and the Sanghavi-Hajek mechanism. Essentially, the objective is to confront the analytic results with a more complex (and more realistic) scenario. The experiment was performed by generating 30 workloads for each valuation distribution based on the 1,000 requests taken from the original DAS workload. Each combination of a user strategy and allocation mechanism was then fed with the same workloads.

User Strategies

The performance of the user strategies was evaluated using the individual metric of utility, which is measured per request. Table 6.4 shows the mean utility per request and run for each combination of a user strategy, valuation distribution (henceforth called "setting"), and allocation mechanism.

Setting \ Mechanism	*PS*	*SH*	*PS*	*SH*
Valuation distribution	$\{10,\ldots,60\}$		$\{10,\ldots,160\}$	
ZIC	7,615	6,404	22,226	19,310
Q-Learning	30,406	28,245	75,503	70,635

Table 6.4: Mean utility per request. *PS* stands for Proportional Share, *SH* for the Sanghavi-Hajek mechanism.

For both settings, Q-Learning outperforms ZIC. Looking at the revenues listed in Table 6.5, users drastically overbid with ZIC (recall that the revenue corresponds to the aggregated bids of all users since pay-as-bid pricing is used). This essentially validates the implementation of Q-Learning as users benefit from "following" the proposed logic instead of simply issuing random bids.

Switching from Proportional Share to the Sanghavi-Hajek mechanism reduces the mean utility in all settings. This is due to the users bidding higher with the Sanghavi-Hajek mechanism than with Proportional Share. Moreover, the Sanghavi-Hajek mechanism results in slightly lower efficiency, cf. Table 6.6, as will be discussed below.

Mechanism Performance

For the comparison of the mechanisms, the aggregate metrics of provider's revenue and efficiency are used. Table 6.5 shows the mean revenue per run for each setting and mechanism combination, whereas Table 6.6 contains the mean efficiency ratios, i.e. the resulting efficiency divided by the theoretical optimum. Two-tailed sign tests were performed to test the differences in revenue (cf. Theorem 1) and efficiency (cf. Theorem 2) between the Sanghavi-Hajek mechanism and Proportional Share for statistical significance. The alternative hypothesis was that the median of the differences in revenue (efficiency) is not equal to zero (Fahrmeir *et al.*, 2004). The results (p-values) are indicated in the tables.

Setting \ Mechanism	*PS*	*SH*	*SH/PS*	*PS*	*SH*	*SH/PS*
Valuations		$\{10,\ldots,60\}$			$\{10,\ldots,160\}$	
ZIC	38,996,633	38,999,034	1.000	90,435,690	90,435,073	1.000
Q-Learning	16,077,240	17,147,600	1.067***	36,146,847	38,270,621	1.059***

Table 6.5: Mean revenue. *PS* stands for Proportional Share, *SH* for the Sanghavi-Hajek mechanism. *** denotes significance at the level of p = 0.01 for a two-tailed sign test.

Setting \ Mechanism	*PS*	*SH*	*PS*	*SH*
Valuations	$\{10,\ldots,60\}$		$\{10,\ldots,160\}$	
Q-Learning	0.829***	0.809	0.819***	0.799

Table 6.6: Mean efficiency ratio for the Q-Learning strategy. *PS* stands for Proportional Share, *SH* for the Sanghavi-Hajek mechanism. *** denotes significance at the level of p = 0.01 for a two-tailed sign test.

With ZIC, both mechanisms generate about the same revenue. This is due to the fact that ZIC is independent of the specific market mechanism and further validates the implementation of ZIC and the mechanisms. For the narrow valuation distribution, the revenue should be approximately $\tilde{v}_{mean} \cdot c_{mean} \cdot d_{mean} \cdot 1000 = \$17.5 \cdot 2.08 \cdot 1{,}044.06 \cdot 1000 = \$38{,}000{,}000$. For the wide valuation distribution, it should be about \$92,300,000. This is close to the results obtained in the simulations.

The results for the Q-Learning strategy reveal that the Sanghavi-Hajek mechanism consistently generates higher revenue than Proportional Share in both settings where the users follow the Q-Learning strategy. Consequently, in this regard the analytic result for the case of two users holds for this more complex scenarios. The increase adds up to about to 6.7% with the narrow valuation distribution and 5.9% with the wide valuation distribution. Both improvements are statistically significant at a level below p = 0.01. It is interesting that the increase in revenue is smaller if the user valuations are more heterogeneous. This resembles the analytic result where the Sanghavi-Hajek mechanism only generates higher revenue than Proportional Share in case the valuation of the low-value user is not too far from the valuation of the high-value user (cf. Theorem 1).

Table 6.6 shows the mean ratio of the efficiency generated by the specific mechanism and the theoretical optimum, in which the resource is allocated exclusively to the requests with the highest (true) valuation. The table only contains the results for the more realistic and successful (with respect to individual utility) Q-Learning strategy. For both valuation distributions, Proportional Share produces a slightly higher efficiency than the Sanghavi-Hajek mechanism.

There are three reasons for the efficiency in these share-based mechanisms being below the theoretical optimum. The first reason is obvious. By construction, even if all users truthfully report their valuations, the mechanisms allocate shares to low-value requests. This inherently leads to suboptimal allocations. Secondly, this effect is strengthened by the users shading down their bids for requests with a high bid and small resource requirements. These requests can be allocated more resources than the users actually requested. However, excessive resources are of no value to the users, and these users will thus generally submit lower bids for high-value requests ("bid shading") *relative* than for low-value requests with high resource consumption. Finally, allocations in excess of the requests' CPU demand essentially constitute a waste since the current allocation rule does not re-allocate this excess allocation to other requests.

The latter two effects also explain why the Sanghavi-Hajek mechanism leads to slightly lower efficiency than Proportional Share. Due to the volume discount for high bids, the Sanghavi-Hajek mechanism further strengthens the bid shading effect for high-value requests that have small resource requirements and weakens the effect for requests with low-valuations and large resource requirements. The Sanghavi-Hajek mechanism thus encourages users with low valuations to bid more aggressively (compared to Proportional Share) in order to obtain a volume discount. The resulting bigger share for low-valuation requests directly reduces efficiency. Moreover, the volume discount in the Sanghavi-Hajek mechanism also increases the possible "waste" resulting from allocation in excess of the users' demand.

The effect of more heterogeneous user valuations is the following. When user valuations are more heterogeneous, this slightly lowers the efficiency of both mechanisms. The mechanisms are designed to allocate a share to *every* user. Consequently, the more heterogeneous the users are, the larger is the deviation from the theoretical optimum.

Implications of the Simulation Results

In summary, the experimental analysis with larger numbers of user requests and realistic request attributes yields promising results. It shows that both mechanisms are approximately equal with respect to efficiency; Proportional Share only yields a slightly higher efficiency. However, the Sanghavi-Hajek mechanism can lead to significantly larger provider's revenue. These results imply that Proportional Share might be more appropriate for scenarios where revenue is only of minor importance and fairness is important (regarding identical unit prices). For instance, within enterprises, the efficient but fair division of scarce computing resources of a shared computing center will be more important than generating revenue. In contrast, if commercial considerations become dominant (e.g. in the case of Amazon and Sun Microsystems), the Sanghavi-Hajek mechanism can be a viable tool for increasing revenue.

6.5 Service Levels Guarantees for Business-Critical Applications

The main advantages of both Proportional Share and the Sanghavi-Hajek mechanism are the low informational requirements (users do not need to specify the length of their request in advance), avoiding "starvation" and promoting "fairness" in a wider sense, meaning that all requests are served, or at least to a certain extent. The major drawback, however, is the uncertain resource allocation. Requests are not guaranteed a specific share of the resource(s), which can be critical for applications that require a minimum service level, e.g. with respect to bandwidth or storage. In this section, the basic Sanghavi-Hajek mechanism is extended by adopting the main idea behind the Augmented Proportional Share mechanism of Stoica *et al.* (1997), which supports resource reservations. Users can thus obtain a guaranteed share of the resource. The resulting mechanism essentially allows a trade-off between predictable costs but uncertain service levels and uncertain costs but guaranteed service levels.

6.5.1 The Allocation

Assume again one perfectly divisible resource but now there are two classes of service requests, the class *SH* of pay-as-bid (Sanghavi-Hajek) requests as before and the class *RES* of *reservation requests*. In contrast to the pay-as-bid requests, reservation requests do not specify their valuation but the share $x_j \in (0,1]$ of the resource that they want reserved for them.

The reservation requests hold a share of $x_{RES} = \sum_{j \in RES} x_j$ in total. Consequently, a share of $x_{SH} = 1 - x_{RES}$ can be allocated to the pay-as-bid requests. This logic follows the approach in Stoica *et al.* (1997). The difference is in calculating the shares of the pay-as-bid requests and in determining the payments of the reservation requests. The share x_{SH} is distributed among the pay-as-bid requests according to the Sanghavi-Hajek scheme as presented above. The pay-as-bid requests hence pay $\tilde{v}_{SH} = \sum_{j \in SH} \tilde{v}_j$ in total.

6.5.2 The Pricing of Reservation Requests

The mechanism of Stoica *et al.* (1997) uses Proportional Share for the pay-as-bid class. Determining the prices for the reservation requests is straightforward: They simply pay the same (uniform) unit price as the requests in the pay-as-bid class. However, when using the Sanghavi-Hajek mechanism for the pay-as-bid class, this is no longer possible since the unit price for the pay-as-bid requests is discriminatory as discussed above.

Reserving resources typically involves paying a premium since the provider looses degrees of freedom in serving and pricing other requests and because service guarantees are more difficult to provide and manage than a best effort service. This approach is adopted and reservation requests are charged a premium that depends on the situation in the pay-as-bid class: Each reservation request is charged a unit price corresponding to the *maximum* unit price in the pay-as-bid class:

$$\forall j \in RES : p_j = \frac{\tilde{v}_j}{x_j} = \max_{k \in SH} \frac{\tilde{v}_k}{x_k}.$$

Due to the discriminatory allocation rule of the Sanghavi-Hajek mechanism, this maximum unit price is always paid by the pay-as-bid request with the lowest bid:

Corollary 1 (Maximum unit price). *The maximum unit price is paid by the pay-as-bid request with the lowest bid:*

$$\operatorname*{argmax}_{j \in SH} \frac{\tilde{v}_j}{x_j} = \operatorname*{argmin}_{j \in SH} \tilde{v}_j.^{11}$$

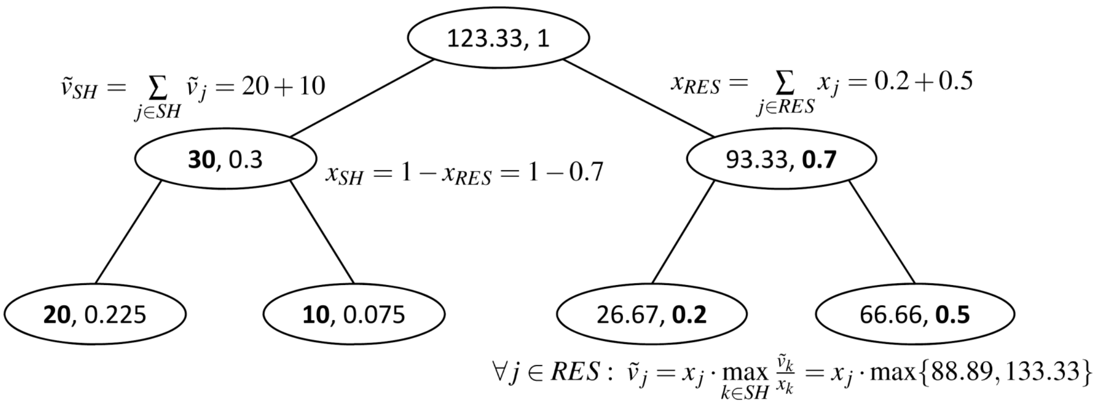
The proof is given in Appendix D.5. This implements a desirable economic reasoning: If resource demand is low (and the maximum unit price in the pay-as-bid class hence is generally also low), reserving a share of the resource is cheaper than if the resource is scarce. Analogously, reserving resources becomes more expensive as demand increases.

6.5.3 A Numerical Example

The following numerical example illustrates the mechanism's functionality. Assume that there are currently two pay-as-bid requests and two reservation requests present in the system, cf. Figure 6.4.

The reservation requests reserve a share of 70% of the resource. The remaining 30% are allocated among the pay-as-bid requests according to the basic Sanghavi-Hajek mechanism. The high-bidding user who pays \$20 thus receives a share of 22.5% and the low-bidding user (\$10)

[11] Note that the argmin and argmax operators might return a *set* of requests in case there are multiple requests with the same bid.

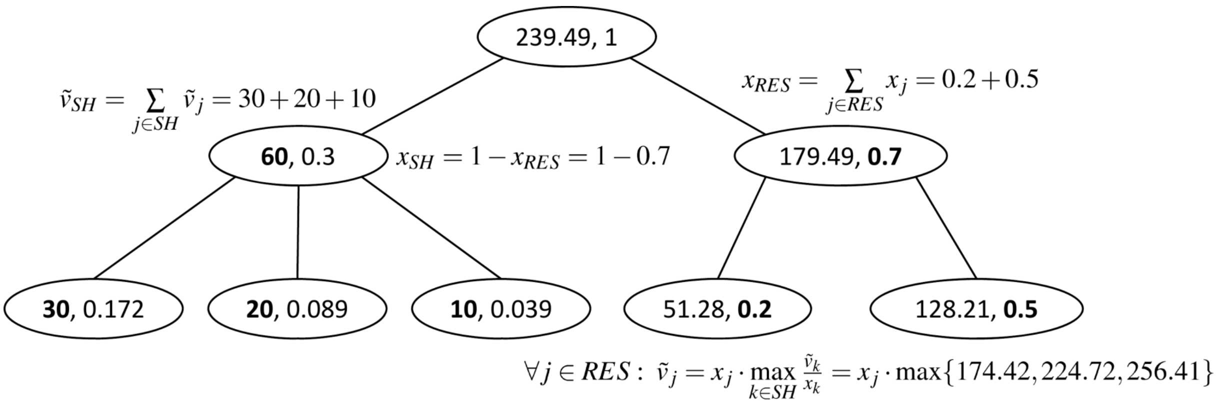


123.33, 1
$\tilde{v}_{SH} = \sum_{j \in SH} \tilde{v}_j = 20 + 10$
$x_{RES} = \sum_{j \in RES} x_j = 0.2 + 0.5$
30, 0.3
$x_{SH} = 1 - x_{RES} = 1 - 0.7$
93.33, **0.7**
20, 0.225
10, 0.075
26.67, **0.2**
66.66, **0.5**
$\forall j \in RES: \ \tilde{v}_j = x_j \cdot \max_{k \in SH} \frac{\tilde{v}_k}{x_k} = x_j \cdot \max\{88.89, 133.33\}$

Figure 6.4: Sample initial scenario.

receives 7.5%. Consequently, the highest unit price of the pay-as-bid requests is paid by the low-bidding user with \$133.33. According to the presented pricing logic, this is also the unit price of the reservation requests. Overall, the mechanism accrues revenue of \$123.33.

Now assume that another pay-as-bid request with weight \$30 is submitted to the system. Figure 6.5 illustrates how the increased competition in the pay-as-bid class pushes the unit prices in both the pay-as-bid class and the reservation class.



239.49, 1
$\tilde{v}_{SH} = \sum_{j \in SH} \tilde{v}_j = 30 + 20 + 10$
$x_{RES} = \sum_{j \in RES} x_j = 0.2 + 0.5$
60, 0.3
$x_{SH} = 1 - x_{RES} = 1 - 0.7$
179.49, **0.7**
30, 0.172
20, 0.089
10, 0.039
51.28, **0.2**
128.21, **0.5**
$\forall j \in RES: \ \tilde{v}_j = x_j \cdot \max_{k \in SH} \frac{\tilde{v}_k}{x_k} = x_j \cdot \max\{174.42, 224.72, 256.41\}$

Figure 6.5: Sample scenario – Addition of a pay-a-bid request.

The pay-as-bid class still gets 30% of the resource. However, the increased competition pushes the maximum unit price in the pay-as-bid class from \$133.33 to \$256.41. Overall, revenue almost doubles to \$239.49.

This example illustrates how prices (and thus revenue) dynamically adapt to changes in the market situation, without the provider having to react manually. The pay-as-bid requests receive an uncertain service level at a certain cost, while the reservation requests receive a certain service level at uncertain costs.

6.5.4 Special Situations

There are two special scenarios to the allocation and pricing logic above. The first one arises when a new reservation request j enters the system but cannot be served since there are not sufficient resources available, i.e. $x_j > 1 - \sum_{k \in RES \setminus j} x_k$. In this case, the request can either be rejected or be held in a waiting queue until a sufficiently big resource share becomes available.

The second special case occurs when there are no pay-as-bid requests. The logic above can no longer be applied to determine the prices of reservation requests. There are two possible cases. If all reservation requests can be served, the resource is not scarce and might be shared among the reservation requests at some minimum unit price (reservation price) set by the provider. If there are reservation requests $\overline{RES} \subset RES$ which cannot be served, one possibility to determine the unit price for the served reservation requests is to multiple the provider's reservation price by a scarcity factor $\frac{\sum_{j \in RES} x_j}{\sum_{j \in RES \setminus \overline{RES}} x_j}$.

6.6 Integration into State-of-the-Art Schedulers

The following section is dedicated to the integration of market mechanisms that are based on allocating "shares" (such as Proportional Share and the Sanghavi-Hajek mechanism) into technical scheduling systems. First, the scheduling logic of the *Sun N1 Grid Engine* (*N1GE*) – a state-of-the-art cluster and grid management system by Sun Microsystems – will be briefly presented. Then two possible approaches towards integrating market-based schedulers into this process will be briefly discussed: (1) a modular extension of *N1GE*, with an additional market-based policy and (2) the displacement of the current technical scheduler with the market mechanism.

6.6.1 Sun N1 Grid Engine

N1GE is a resource management and scheduling system developed by Sun Microsystems (Sun Microsystems, 2008). It administers and dynamically allocates the shared pool of resources such as computing power, memory, and licensed software within an organization.

The *N1GE* scheduler consists of a waiting queue with pending requests and a technical scheduler that subsequently assigns waiting requests to idle resources. The user submits a request together with a specification of the technical requirements. After receiving the requests, the scheduler places the requests in the waiting list. The position of a request in the waiting list is determined by the request's priority. This priority value is calculated by the scheduler using a pre-defined and static mix of different policies. A sample policy mix might include manually (by the administrator) set shares for individual users, user groups, a department or a project (also called *entitlement policy*), an increase in priority for requests that will reach their deadline soon or that have been waiting for a long time (*urgency policy*). Additionally, users might be able to sort their own requests (*custom policy* or *POSIX*) (Chaubal, 2005). An example policy mix might be as follows:

$$P_{mix,j} = P_{e,i}W_e + P_{u,j}W_u + P_{c,j}W_c,$$

where $P_{mix,j}$ is the overall dispatch priority for request j, $P_{e,i}$ is the entitlement priority for the user i who submitted request j and W_e is the entitlement weighting factor. $P_{u,j}$, W_u, $P_{c,j}$ and W_c are defined accordingly for the urgency and custom priorities. If the user submits multiple requests, the entitlement priority of the user might be split equally across all her requests, for instance, as depicted in the example in Figure 6.6.

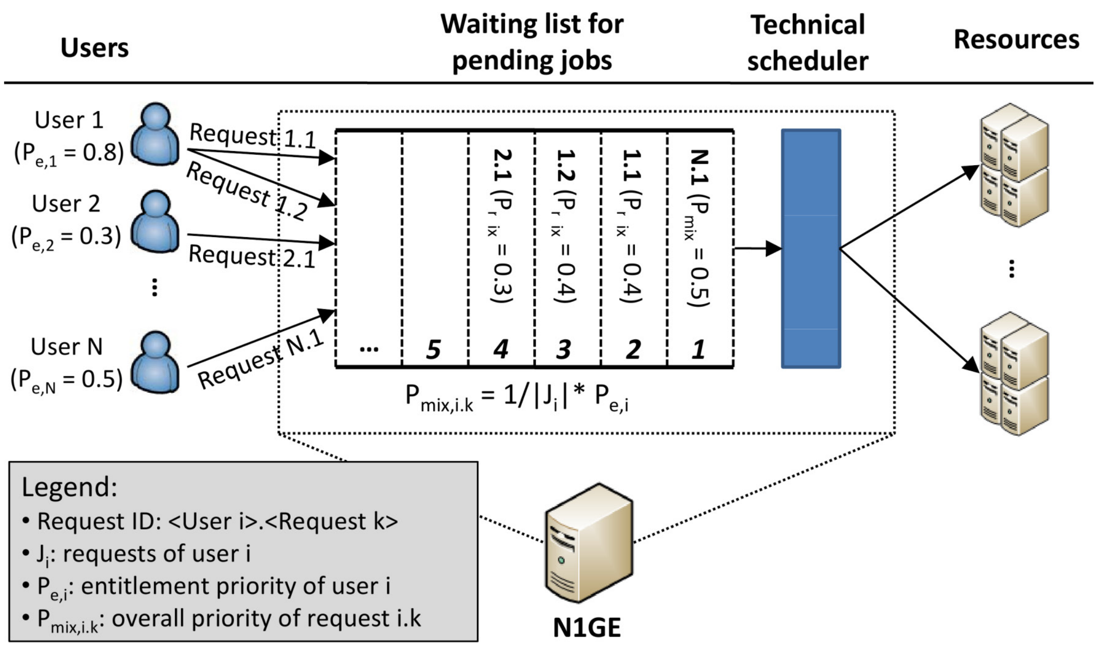


Figure 6.6: The technical *N1GE* scheduler.

There are two possible approaches towards integrating market mechanisms such as Proportional Share and the Sanghavi-Hajek mechanism into *N1GE*'s scheduling process that will now be discussed.

6.6.2 The Pay-as-Bid Mechanism as Additional Policy

The objective of this approach is to use the market-based mechanism as an instrument for partitioning the priority value that determines the order of requests in *N1GE*'s waiting queue. Analogue to the current *N1GE* system, the users submit their requests along with a specification document to state their resource requirements. In addition, the users send a one-dimensional bid (a single real number) to signal their valuation of the submitted request. These bids are then used by the market mechanism to allocate shares of the priority value, which then again are used to determine the order in the waiting list of pending requests. Whenever a resource (e.g. a compute node) becomes idle, the technical scheduler traverses the sorted waiting list and picks the first feasible request for execution. This procedure is depicted in Figure 6.7.

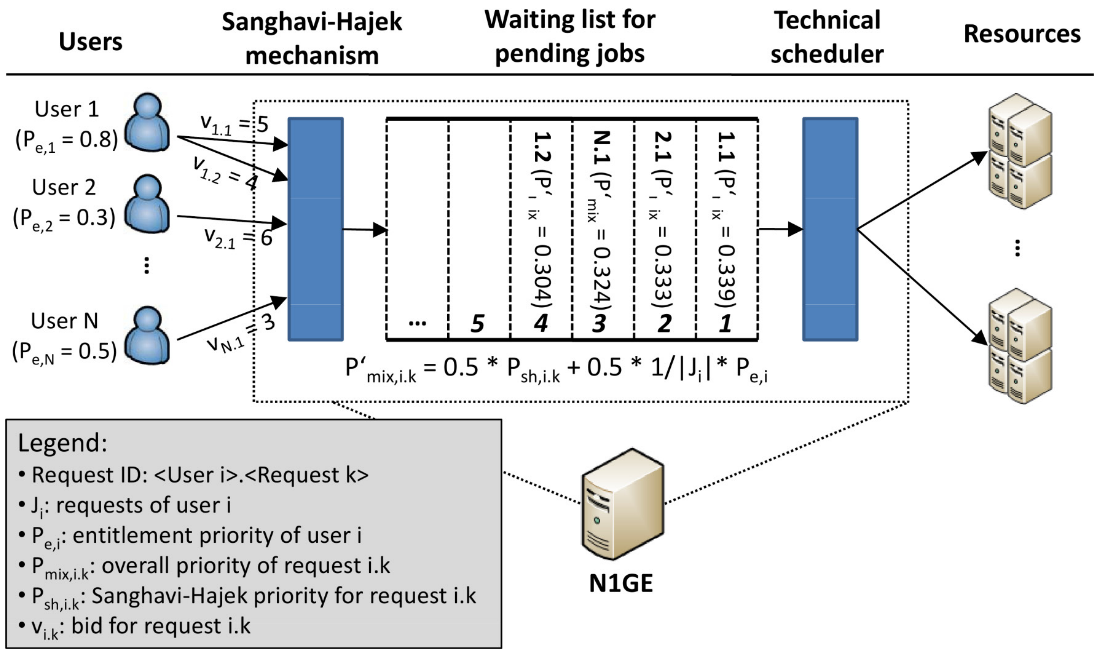

Users
Sanghavi-Hajek mechanism
Waiting list for pending jobs
Technical scheduler
Resources
User 1 (P_e,1 = 0.8)
User 2 (P_e,2 = 0.3)
User N (P_e,N = 0.5)
v_1.1 = 5
v_1.2 = 4
v_2.1 = 6
v_N.1 = 3
1.2 (P'_ix = 0.304) 4
N.1 (P'_mix = 0.324) 3
2.1 (P'_ix = 0.333) 2
1.1 (P'_ix = 0.339) 1
... 5
P'_mix,i.k = 0.5 * P_sh,i.k + 0.5 * 1/|J_i| * P_e,i
Legend:
• Request ID: <User i>.<Request k>
• J_i: requests of user i
• P_e,i: entitlement priority of user i
• P_mix,i.k: overall priority of request i.k
• P_sh,i.k: Sanghavi-Hajek priority for request i.k
• v_i.k: bid for request i.k
N1GE

Figure 6.7: The pay-as-bid mechanism as additional policy.

In addition to this new policy, the currently available policies in *N1GE* are still at the administrator's disposal. This can rather easily be achieved by adding the new *Sanghavi-Hajek policy* to the existing policy mix, for instance. The value that is determined by the market-based mechanism is weighted and added to the total priority value of a request:

$$P'_{mix,j} = P_{mix,j} + P_{sh,j}W_{sh}$$

Thereby, $P'_{mix,j}$ is the new dispatch priority for request j, $P_{mix,j}$ the priority value generated with the current *N1GE* policy mix, and $P_{sh,j}$ and W_{sh} are the Sanghavi-Hajek priority and the corresponding weighting factor for this new policy, respectively.

The major drawback of this approach is that – by only allocating and pricing the priority value – the market mechanism cannot dynamically revise past allocation decisions once a request has been assigned to a physical resource.

6.6.3 The Pay-as-Bid Mechanism as Scheduler

In this scenario the market mechanism is applied as a scheduler to directly allocate the resources to the users. In contrast to the scenario assumed so far, the mechanism does not calculate the respective priority value for each user, but it determines the actual share of resources a user gets. This procedure is depicted in Figure 6.8.

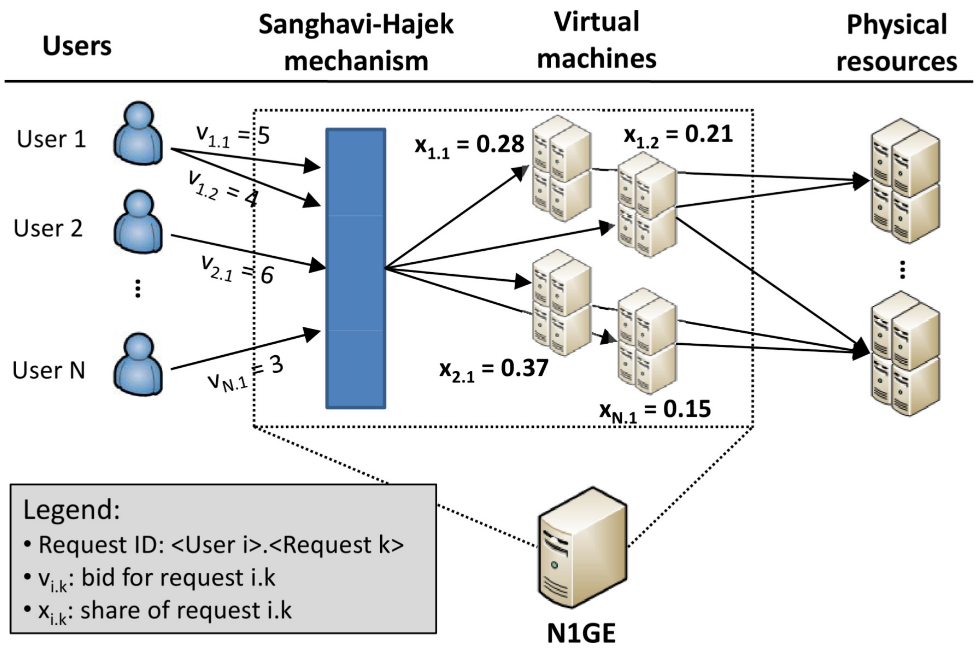


Figure 6.8: The pay-as-bid mechanism as scheduler.

The major challenge that has to be mastered in this scenario is the architectural limitation of *N1GE*, which only allows a single task to be executed on each physical resource at a time. Therefore resources can no longer be assumed as being perfectly divisible. This problem can be tackled from the technical side. Employing virtualization tools could restore the perfect divisibility of the resources. With recent advances in virtualization technology it is possible to almost perfectly split one physical resource into multiple virtual instances (Barham *et al.*, 2003). Moreover, there has been research on dividing a *pool* of dedicated compute nodes among user requests over time so that each request is approximately allocated its share (Lai *et al.*, 2005; Amar *et al.*, 2007).

Compared to the policy-based approach, incorporating the market mechanism directly into the scheduler – especially by using virtualization technologies – has the advantage that the shares of requests that have been allocated can be dynamically resized.

6.6.4 Implications of the Approaches

Introducing the market mechanism as an additional policy constitutes a modular extension of *N1GE*. Most of the current architecture and structure remains unchanged. The policy mix still covers the existing policies. In contrast, the direct scheduling scenario requires the modification and replacement of core components. Consequently, the implementation effort is much higher with this approach. In addition, the allocated resources are limited to a single type whereas the modular extension supports heterogeneous resources. On the other hand, giving the market mechanism direct access to the resources entails many advantages. For instance, a waiting list is dispensable since all requests get a share of the resource. Furthermore, the additional in-between scheduler can be omitted, which removes complexity from the scheduling process. Both scenarios offer promising enhancements to *N1GE*. The modular extension offers a quick solution for incorporating the discriminatory pay-as-bid mechanism, whereas the higher flexibility of the second approach comes at the expense of higher implementation effort.

To guarantee a sound concurrence of the *N1GE* system and the market mechanism, a number of extensions and modifications have to be considered, in particular the *re-evaluation of the priority values* of requests as new requests enter the system or requests are finished, *bid updating* to allow for strategic user behavior, especially to prevent the starvation of requests, and *feedback of market information* to the users.

6.7 Discussion

The Sanghavi-Hajek mechanism and the proposed extensions to the mechanism are a promising addition to the *N1GE* scheduler. The Sanghavi-Hajek mechanism allows flexible reactions to changes in the demand and supply situation. Moreover, it offers an elaborated pricing scheme where prices reflect the current market situation and induce users to report their valuations to the system. The administrator no longer needs to adjust the user priorities manually but the users can directly express the urgency of their requests. In comparison to other market mechanisms, for instance the mechanisms presented in the previous chapters, the Sanghavi-Hajek mechanism scores with its ease-of-use; users only need to report few technical and economic parameters, hence making it a reasonable choice especially for online settings.

While the mechanism makes strong assumptions about the divisibility of resources, this chapter discussed two basic design options for how the mechanism can be integrated into state-of-the-art computing platforms. In the first approach, the scarce good to be allocated is essentially the priority value that determines the position of requests in the waiting queue. The second approach relies on virtualization technologies and load-balancing techniques so as to make the physical resources themselves almost perfectly divisible.

Compared to (market-based) Proportional Share, arguably the currently most prominent market mechanism for the setting in which the Sanghavi-Hajek mechanism operates, the Sanghavi-Hajek offers a potential increase in the provider's revenue. On its downside, from the user perspective the Sanghavi-Hajek mechanism might be more complicated to grasp and not fair in the sense that it discriminates users with respect to unit prices.

The game-theoretic analysis of the simplified two-user scenario built on the solution concept of ex post Nash equilibria, where the users are assumed to have complete information about the other user's types, i.e. the user types are public information. As pointed out by Parkes (2001), this is a somewhat peculiar assumption. If the user types are public, the mechanism could easily determine the optimal allocation without having to rely on the user reports. Consequently, from a game-theoretic point of view, it would be interesting to consider more realistic solution concepts in the future, such as Bayesian-Nash equilibria where the users only have expectations about the other users' types (Mas-Colell *et al.*, 1995).

Further work also needs to be done with respect to learning agents (users) in a dynamic and only partially-observable environment. While traditional reinforcement learning concepts and methods are not perfectly suited for such settings, the simulations reported in this chapter yielded promising results. Developing software agents that trade in dynamic and inherently non-stationary and non-markovian environments is a very challenging but also important area for future research.

In this chapter, a framework was developed that allows to integrate requests that require service level guarantees. Further work needs to be done on analyzing the impact of these extensions on the mechanism's economic properties.

Finally, a decentralized version of the mechanism would be desirable to support decentralized waiting queues as well. This might be necessary to keep the *N1GE* scheduler applicable for very large computing systems with thousands of cores, which will be demanded in the near future given the current trend towards centralizing computing resources in few providers' computing centers.

Chapter 7

Conclusions and Future Work

This chapter concludes the work at hand. First, the key points of this thesis are summarized by reviewing the research questions from Chapter 1. Subsequently, possible extensions of this work and open questions for future research are highlighted.

7.1 Summary of Contributions

The main objective of this work was to investigate market-based scheduling mechanisms for specific application scenarios of distributed computing systems. The aim was to design markets that are both feasible in practice and theoretically sound. These two perspectives are oftentimes complex and somewhat conflicting. If the scenario is close to reality, its complex structure might defy further theoretical analyses. On the other hand, abstract models with nice theoretical properties (in particular possibility results) might not be truly applicable to reality. The work at hand therefore makes the following contributions:

- The general problem setting – designing market-based schedulers for computational resources – was structured according to the characteristics of the applications and the resources. By doing so, the problem was divided into a set of more specific and manageable scenarios.
- For each of these scenarios, this work investigated market mechanisms by applying a range of methods, from theoretical analyses, over empirical studies, to numerical simulations. Either novel market mechanisms were developed that improve over existing research, or existing mechanisms were extended so as to provide a better fit to the given setting.

- In consequence, this research partially bridged the gap between theory and practice. On the one hand, theoretical research about market mechanisms was applied to real-world scenarios. On the other hand, the characteristics of the specific computing environment, such as the importance of preemption and migration, might also feed back into the research on market mechanisms in general.

Based on the results developed and presented in this work, the four research questions, which were introduced in Chapter 1, can be recapitulated as follows.

Research Question 1:

How can a scalable, two-sided market mechanism for clearing heterogeneous grid settings be designed so that strategic behavior from users and providers is limited?

Chapter 3 studied market mechanisms in offline grid settings and developed a deterministic heuristic and the proportional critical-value pricing scheme. The analytical and numerical evaluation showed that this heuristic is highly scalable and yields near-optimal resource allocations in realistic scenarios. Moreover, the mechanism generates prices that are truthful on the demand side and approximately truthful on the supply side with respect to valuations. Limiting strategic behavior is important because with truthful prices, market participants will not game the mechanism by reporting an untrue valuation, and the mechanism can thus optimize over the correct input. The developed mechanism improves over previous work especially because it allows the trading between multiple users and providers without artificially restricting the providers' strategy space (cf. Bapna *et al.* (2008)). Moreover, it improves on the previous mechanisms' vulnerability against manipulations from users and providers, while only slightly sacrificing allocative efficiency.

Research Question 2:

How can a randomization of the heuristic allocation algorithm improve efficiency? How can the pricing scheme be modified to limit strategic behavior from users?

The greedy heuristic proposed in Chapter 3 is based on a deterministic ranking of job requests. In Chapter 4, this deterministic heuristic was converted so as to *randomly* choose the job requests in the allocation phase. This randomization can help in avoiding worst cases of the deterministic heuristic. While previous research showed that it is theoretically possible to turn a randomized heuristic into a truthful (in expectation) mechanism, it is practically infeasible to determine these prices. Moreover, the previous research only considered simplified scenarios and the results can thus not be transferred to the more complex scheduling scenario at hand. Consequently, in Chapter 4 a randomized mechanism was developed that induces users to at least bid “close” to truthfully. Moreover, it was shown how this randomized mechanism can be combined with the deterministic mechanism to a distributed randomized mechanism. The numerical results show that the distributed randomized mechanism is a promising addition to the stand-alone deterministic mechanism especially in smaller settings. In large settings, the deterministic heuristic obtains a close approximation ratio (regarding the optimal solution), thus not leaving much room for improvement. This further underlines the strength of the deterministic heuristic.

Research Question 3:

What is the benefit of performing preemptions in economic online settings with dedicated resources?

Chapter 5 showed that preemptions allow the design of more efficient economic online mechanisms both with respect to worst case efficiency as well as “average” efficiency in realistic scenarios. This result was derived in a setting that in fact puts preemptive mechanisms at a disadvantage, as only homogeneous resources are considered. Moreover, the mechanism knows the job runtimes, which is a strong assumption, since oftentimes the user will not be able to actually specify her job’s runtime. The benefit of preemption and especially migration will thus generally be still higher when heterogeneous resources are taken into account and when the job runtimes are unknown to the mechanism.

Research Question 4:

How should shared resources be allocated and priced in online settings?

A mechanism by Sanghavi and Hajek (2004), which was originally proposed for the allocation of bandwidth, was applied and extended to the cluster / utility computing setting. The mechanism was compared to Proportional Share, the currently most prominent mechanism for such scenarios. It was shown both analytically and with an agent-based simulation that the Sanghavi-Hajek mechanism can outperform Proportional Share with respect to allocative efficiency and, in particular, the provider’s revenue. Moreover, a framework was developed that allows a trade-off between predictable costs but uncertain service levels and uncertain costs but guaranteed service levels, to extend the range of possible application scenarios for the Sanghavi-Hajek mechanism.

7.2 Possible Extensions and Open Questions

7.2.1 Specific Extensions to This Work

There are several directions in which the specific research in this work should be continued and extended.

Real Workloads

Since there is currently no workload of a real grid or utility computing market available, especially with respect to economic attributes (user valuations, reserve prices), the market mechanisms had to be evaluated by means of randomly generated and modified workloads. While the general properties of the mechanisms will hold, it would be interesting to see the runtime and approximation behavior for real workloads. In this regard, it should be noted that currently a real-world marketplace for computer resources is being launched: *Zimory*[1], a spin-off of the Deutsche Telekom Laboratories, aims at establishing a “marketplace where anyone is able to sell and buy server capacities world-wide”.

Strategic Behavior

Additionally to having to use artificial workloads, the numerical analysis of the mechanisms' strategic properties was based on letting only one user or provider misreport. This of course does not consider the interdependencies between the strategies of the market participants. In more complex and realistic settings, multiple users and providers will misreport in parallel and / or form coalitions. Moreover, a provider can be a user at the same time and vice versa. Such behavior could possibly be analyzed by means of agent-based simulations and laboratory experiments. Furthermore, in Chapter 5, the providers have been assumed to be obedient. In an extended scenario, these providers can have reservation prices and report and act strategically.

The Benefit of Migration in Heterogeneous Settings

As pointed out above, a natural extension to the investigation of preemption in online markets would be to consider migration, i.e. the moving of jobs *across* compute nodes during runtime, especially if these nodes are heterogeneous (Amar *et al.*, 2008b). Migration allows for still more flexibility in the allocation and pricing schemes and might thus also help in implementing stronger game-theoretic solution concepts.

[1] `http://www.zimory.de/`

Non-Clairvoyant Market Mechanisms

The analysis of preemption in Chapter 5 was based on the strong assumption that the job runtimes are known a priori. However, oftentimes the user will have no information or not even an estimate of the job runtime. Scheduling mechanisms that work without information about the job runtimes are called "non-clairvoyant schedulers" (Motwani *et al.*, 1993). Non-clairvoyant market mechanisms would thus be a further important addition to the set of market mechanisms for scheduling problems. Moreover, such mechanisms would also be easier to use from the user's perspective as less information is required. First results on such mechanisms for grid settings show that non-clairvoyant mechanisms with the ability of migrating jobs can even outperform existing clairvoyant, static mechanisms that know the exact job runtimes a priori (Amar *et al.*, 2008a,b). However, this existing work focuses on the allocation algorithm only. Future research is needed that develops a suitable pricing scheme that can complement the allocation algorithm.

Decentralized Market Mechanisms

Except for the *P-DLGM* mechanism in Chapter 5, all mechanisms considered in this work rely on a centralized entity that collects the private information of users and providers and then determines the allocations and prices. However, a central entity can constitute a single-point of failure and bottleneck especially in large-scale grid settings. It would thus be a promising area for future research to design decentralized market protocols for such settings, possibly based on existing work on peer-to-peer networks (e.g. Milojicic *et al.* (2002)). Sample research questions comprise algorithms that distribute the information about users and providers and the system's state in a decentralized manner, as well as market protocols that converge towards a market equilibrium in a timely manner, if such an equilibrium exists.

7.2.2 Complementary Research

Besides these specific extensions, there are more general research areas that are complementary to the mechanisms and the scheduling process that have been considered in this work.

Preference Elicitation

While market mechanisms exhibit compelling features, it is typically assumed that the market participants know their *true* valuations. However, it is a complex task for users and providers to assess their valuation for a certain resource or service or even a complex combination of services. This comprises questions such as "What am I willing to pay for using a server with application X, a dual-core processor and 2 GB of memory for one hour?". It might be even more difficult to assess the valuation for a certain offering *relative* to another offering: "If I can only get a server with a single-core processor, what is my valuation for this offering as opposed to the dual-core server?" There is currently not much research available in this area, which is surprising as it is a prerequisite for any market-based approach (Shneidman *et al.*, 2005). Popular techniques for estimating a user's *valuation* for a certain resource or service are Conjoint Analyses (Luce and Tukey, 1964; Green and Rao, 1971) and the Analytical Hierarchy Process (Saaty, 1990), for instance. The main problem with these approaches is that they quickly become infeasible with increasing numbers of resource and service attributes and attribute values. This is exacerbated by the large number of transactions especially in resource-near markets.

Automated Trading

When the valuation and resource characteristics are known, another complex task is to efficiently communicate these preferences to the market (Shneidman *et al.*, 2005). By equipping users with automated tools such as bidding agents, the communication with the market can be drastically simplified since human users and providers do not constantly need to monitor the market outcome and update their requests and offers (MacKie-Mason and Wellman, 2006). The tools do not need to be (and cannot be) fully automated black boxes but will be hybrid models; the users must understand the underlying logic to be able to parameterize the tools according to their technical and economic preferences. Moreover, the users will not give away the total control about the trading process (and might not be allowed to do so for legal reasons) but will regularly interact with the bidding agents. From a technical perspective, it would be interesting to analyze how such tools can be implemented into real systems. From an economic point of view, these tools need to be tailored to the specific market mechanisms at hand, and vice versa (cf. Neumann *et al.* (2008b)). Consequently, the tasks of designing a market mechanism and of designing the agents that trade on this market are interdependent, iterative, and evolutionary.

SLA Enforcement

As presented in Chapter 2, after the market mechanism has determined the outcome, this is encoded in an SLA. Research in the design of market mechanisms assumes the ex post compliance of market participants: After the mechanism has determined the allocation and the resulting prices, the market participants are typically assumed to adhere to the market's decisions. In reality, however, this is not self-evident due to the possibility of technical failures and moral hazard problems. For instance, in case a provider has committed to a certain SLA, it might later well be in the provider's interest to intentionally violate the SLA in favor of another, more desirable user (cf. Byde *et al.* (2003)). A subsequent task is thus to monitor the execution of the job to assure that it executes according to the SLA. An interesting area for future research is the question of how to penalize the user or the provider in case the job or the node violates the SLA (van Dinther *et al.*, 2009; Rana *et al.*, 2008). How should violations of certain job or resource / service attributes be aggregated and translated into monetary compensations? Is there any way to extend the economic properties of the market mechanism (such as incentive compatibility) to the subsequent stages of the overall process? Moreover, analogously to the mechanism design problem, another layer of complexity is added by the characteristics of the application domain at hand. In computational grids, for instance, when a job fails or a wrong result is returned to the user, it is hard to detect whether this was due to intentional misbehavior of the resource provider or due to technical reasons, which are neither controlled by the user nor the provider, programming errors of the user, etc.

Reputation Mechanisms

While the SLA enforcement works with contractual and monetary penalties to incentivize users and providers to comply with the market outcome, reputation mechanisms aim at building trust via feedback mechanisms (Anandasivam and Neumann, 2008; Resnick *et al.*, 2000; Bolton *et al.*, 2002; Dellarocas, 2003). Trust is a vital requirement for a market to be sustainable, in particular if a user only rarely transacts with the same provider, and vice versa. Reputation mechanisms aggregate the perceived users' and providers' transaction history and thus might give indications towards these users' and providers' future performance and compliance. This expected performance can be taken into account when making allocation and pricing decisions and thus in turn impacts the future allocation and pricing decisions. For example, jobs of users and nodes of providers with a low reputation might receive a lower priority in the scheduling process. Ultimately, users and providers have to trade off the present benefit of not complying with an SLA with the future loss. It is an interesting challenge for future research to design and implement such feedback mechanisms into the market's allocation and pricing decisions.

Markets Mechanisms for Complex Services

This work has focused on market mechanisms for resource-near markets. However, there is a strong trend towards not only dynamically sourcing low-level resources from external providers, but also software services and complementary services. This trend is oftentimes comprised under the terms "cloud computing" and "Software as a Service" (Saas). Such more complex services are located on Tier 2 of the market structure elaborated in Chapter 2 (cf. Figure 2.4).

Cloud computing is a new concept for distributed computing. While there is no established definition yet, there is a general consensus that cloud computing is more than distributed, high-performance computing. Boss *et al.* (2007) stress that clouds are not limited to high-performance environments, but also support "interactive, user-facing applications" such as Web applications and three-tier architectures. Lawton (2008) briefly describes the type of applications that is run on clouds: Web-based applications that are accessed via browsers but with the look-and-feel of desktop programs. For Klems *et al.* (2008), cloud computing builds upon virtualization and Web technologies to deliver "compute utilities as on-demand services with variable pricing schemes, enabling a new consumer mass market." Buyya *et al.* (2008) point out that there are different types of clouds, compute clouds for data processing and storage clouds for data storage. This distinction, however, is not sufficient as clouds go beyond the provisioning of physical resources. Consequently, a further distinction should be made into infrastructure clouds and application clouds (Weinhardt *et al.*, 2008).

Cloud computing is similar to the well-known trends of application service providing (ASP) and SaaS in that it focuses on the provisioning of services that are hosted in a more or less centralized way and that can be accessed and payed for dynamically by many users (Knolmayer, 2000; Gillan *et al.*, 1999; Turner *et al.*, 2003; Papazoglou, 2003; Sääksjärvi *et al.*, 2005). However, cloud computing is more than ASP and SaaS since it not only supports the access to software services, but also to infrastructure services, such as computing power and storage. Moreover, cloud computing systems can be the underlying platform for the hosting of software services, especially when these services have to scale quickly. By means of virtualization technologies, cloud computing allows the customization of services by the end user. This view is supported by Weiss (2007), for whom cloud computing is not a fundamentally new paradigm. It rather draws on existing technologies and approaches, such as virtualization, utility computing, SaaS, distributed computing, and centralized data centers. What is new is that cloud computing combines and integrates these approaches to a comprehensive concept and architecture.

In the future, cloud providers will try to differentiate themselves and build a market by offering services on top of physical resources. Here services might comprise tools for making the resources easily accessible, high qualities of service, and security, but also programming environments and application services. It will be interesting for future research to investigate business and pricing models for such service markets (cf. Blau *et al.* (2008)). Sample questions comprise: How should resources and services be bundled to more complex offerings? How can SLAs be determined for such complex services? Is price discrimination appropriate? Moreover, this leads to the following research area.

Market Concatenation

As introduced in Chapter 2, there can be multiple market mechanisms both across the tiers of the market structure as well as within one tier. Up to now, each of these mechanisms has been investigated in isolation. But in practice, these mechanisms may be closely intertwined. For example, a mechanism for the pricing of a complex service may depend on multiple mechanisms for raw services and physical resources. An interesting question for future research might thus be to model and investigate these dependencies: How does strategic behavior change if a user acts on multiple markets at the same time? And how does the computational complexity of the markets on one service layer affect the tractability of the markets on another service layer?

Revenue Considerations

In this work, the dominant design objective was allocative efficiency, i.e. maximizing the overall utility of all users and providers. However, especially in utility computing and cloud computing settings with potentially only a few rather centralized providers, the objective in the design of market mechanisms or pricing schemes in general will rather shift to revenue maximization (for the provider, cf. Anandasivam and Neumann (2009)). Moreover, it is not possible to design a pricing scheme for each resource provider in isolation, since the pricing decisions of one provider might cause users to move from or to another provider. Consequently, there is the need for an explicit model of the competition between the resource providers and the price elasticity of demand.

7.3 Final Remarks

Markets for computer resources are an interesting and promising setting for investigating market mechanisms, since markets can be purposefully designed and implemented from scratch instead of passively studying existing markets. In settings where computer resources are scarce, markets can be a viable tool towards increasing the efficiency of the resource scheme. The system engineer faces a key problem: How should the scarce resources be allocated between the competing users and providers so as to maximize the overall system value? In order to maximize this value, the system engineer inherently depends on the users and providers to reveal their private information about resource demand and supply. The users and providers, however, will selfishly try to maximize their individual benefit instead of working towards one common goal. For instance, without "proper" pricing mechanisms (as studied in this work) they will generally benefit from overstating their priorities. Here markets have the power to align the incentives of the users and providers with the system / market engineer's goal. Markets can induce users and providers to make proper use of the resources, reveal their true demand and supply situation, and offer incentives to providers to contribute scarce resources to the system. Markets can then aggregate this dispersed information and determine allocations and prices that are both desirable from an overall perspective, but also from the viewpoint of the individual users and providers. This ultimately follows the concept of the “invisible hand” by Adam Smith (Smith, 1904).

The focus of this work was on Market Engineering, especially the micro and IT structure of the market, i.e. the rules that govern the information exchange between the market participants and the market itself and that decide about the market outcome based on this input. The aim was to develop, implement, and test allocation algorithms that take into account the technical constraints and capabilities of the application scenarios in order to determine allocations that are technically feasible and economically desirable. Moreover, in line with Smith's concept of the “invisible hand”, pricing mechanisms were developed that aim at aligning the market participants' incentives with the overall objective of welfare and / or revenue maximization.

However, as this research outlook shows, there are many complementary research questions that need to be addressed in a comprehensive and integrated manner. Only if these challenges are mastered, it is that markets for computer resources and services will find their way from theory to practice.

Appendix A

Appendix to Chapter 3

The complete allocation problem considered in this chapter is the following:

$$\begin{aligned}
&\max_X W = \sum_{j\in J} c_j \sum_{n\in N}\sum_{t\in T} x_{jnt}(v_j - v_n)\\
&\text{subject to } \sum_{n\in N} x_{jnt} \leq 1,\ j\in J, t\in T, r_j \leq t \leq e_j,\\
&\qquad\qquad \sum_{j\in J} x_{jnt} c_j \leq C_n,\ n\in N, t\in T, R_n \leq t \leq E_n,\\
&\qquad\qquad \sum_{j\in J} x_{jnt} m_j \leq M_n,\ n\in N, t\in T, R_n \leq t \leq E_n,\\
&\qquad\qquad \sum_{u=r_j}^{e_j}\sum_{n\in N} x_{jnu} = d_j \sum_{n\in N} x_{jnt},\ j\in J, t\in T, r_j \leq t \leq e_j,\\
&\qquad\qquad x_{jnt} \in \{0,1\},\ j\in J, n\in N, t\in T, r_j \leq t \leq e_j, R_n \leq t \leq E_n, v_n \leq v_j.
\end{aligned}$$

Appendix B

Appendix to Chapter 4

B.1 Pseudo-Code of the Randomized Heuristic

Algorithm 2 Randomized Allocation Heuristic.

```
Require: Set $J$ containing all job requests; set $N$ containing all node offers, sorted in non-decreasing order of $v_n$; $\alpha \geq 1$.
Ensure: Feasible allocation schedule $X = (x_{jn})$.
 1: $X = O, J' = \emptyset$
 2: while $|J'| < |J|$ do
 3:     $v = 0$
 4:     for all $j \in J \setminus J'$ do
 5:         if isFeasible($j$,$N$) then
 6:             $v+ = v_j^\alpha$
 7:     if $v == 0$ then
 8:         break
 9:     $r$ = random([0,$v$]), $tmp = 0$
10:     for all $j \in J \setminus J'$ do
11:         if !isFeasible($j$,$N$) then
12:             continue
13:         $tmp+ = v_j^\alpha$
14:         if $r \leq tmp$ then
15:             allocate($j$,$N$)
16:             $J' = J' \cup j$
17:             break
18: return $X$
```

B.2 Proofs of Lemma 1 and Lemma 2

Proof to Lemma 1. The expected utility of j is given as

$$E(u_j(\tilde{v}|\theta_j)) = c_j \frac{\tilde{v}_j}{\tilde{v}_j + \tilde{v}_i}(v_j - \tilde{v}_j).$$

Hence,

$$\begin{aligned}
\frac{\partial E(u_j(\tilde{v}|\theta_j))}{\partial \tilde{v}_j} &= -\frac{c_j(-v_j\tilde{v}_i + \tilde{v}_j^2 + 2\tilde{v}_j\tilde{v}_i)}{(\tilde{v}_j+\tilde{v}_i)^2} = 0 \\
\Leftrightarrow \qquad & \tilde{v}_j^2 + 2\tilde{v}_i\tilde{v}_j - v_j\tilde{v}_i = 0 \\
\Leftrightarrow \qquad & \tilde{v}_j = -\tilde{v}_i + \sqrt{\tilde{v}_i^2 + \tilde{v}_i v_j},
\end{aligned}$$

since all parameters are positive by definition.

Moreover,

$$\frac{\partial^2 E(u_j(\tilde{v}|\theta_j))}{\partial \tilde{v}_j^2} = -\frac{2c_j\tilde{v}_i(\tilde{v}_i + v_j)}{(\tilde{v}_j + \tilde{v}_i)^3} < 0.$$

Consequently, $b_j(\tilde{v}_i|v_j) = -\tilde{v}_i + \sqrt{\tilde{v}_i^2 + \tilde{v}_i v_j}$ maximizes $E(u_j(\tilde{v}|\theta_j))$ for $\alpha = 1$. □

Proof to Lemma 2. The expected utility of j is given as

$$E(u_j(\tilde{v}|\theta_j)) = c_j \frac{\tilde{v}_j^2}{\tilde{v}_j^2 + \tilde{v}_i^2}(v_j - \tilde{v}_j).$$

Hence,

$$\begin{aligned}
\frac{\partial E(u_j(\tilde{v}|\theta_j))}{\partial \tilde{v}_j} &= c_j\frac{\tilde{v}_j\left(2\tilde{v}_i^2 v_j - \tilde{v}_j^3 - 3\tilde{v}_j\tilde{v}_i^2\right)}{\left(\tilde{v}_j^2+\tilde{v}_i^2\right)^2} = 0 \\
\Leftrightarrow \qquad & \tilde{v}_j^3 + 3\tilde{v}_i^2\tilde{v}_j - 2\tilde{v}_i^2 v_j = 0.
\end{aligned}$$

Cardano's formula (cf. Nickalls (1993)) gives the only real root of this cubic polynomial as

$$\begin{aligned}
\tilde{v}_j &= \sqrt[3]{\tilde{v}_i^2 v_j + \sqrt{\tilde{v}_i^4 v_j^2 + \tilde{v}_i^6}} + \sqrt[3]{\tilde{v}_i^2 v_j - \sqrt{\tilde{v}_i^4 v_j^2 + \tilde{v}_i^6}} \\
&= \sqrt[3]{\sqrt{\tilde{v}_i^4 v_j^2 + \tilde{v}_i^6} + \tilde{v}_i^2 v_j} - \sqrt[3]{\sqrt{\tilde{v}_i^4 v_j^2 + \tilde{v}_i^6} - \tilde{v}_i^2 v_j} \\
&= b_j(\tilde{v}_i|v_j).
\end{aligned}$$

Moreover,

$$\begin{aligned}
\frac{\partial^2 E(u_j(\tilde{v}|\theta_j))}{\partial \tilde{v}_j^2} &= \frac{2c_j\tilde{v}_i^2(-3v_j\tilde{v}_j^2 + v_j\tilde{v}_i^2 + \tilde{v}_j^3 - 3\tilde{v}_j\tilde{v}_i^2)}{(\tilde{v}_j^2 + \tilde{v}_i^2)^3} < 0 \\
&\Leftrightarrow \; -3v_j\tilde{v}_j^2 + v_j\tilde{v}_i^2 + \tilde{v}_j^3 - 3\tilde{v}_j\tilde{v}_i^2 < 0.
\end{aligned}$$

Inserting $b_j(\tilde{v}_i|v_j)$ into the left hand side of the latter equation always yields negative values. This has been verified graphically and numerically for $0 < v_j \leq 100$ and $0 < \tilde{v}_i \leq 100$ (with increments of 0.01 each).

Consequently, the second derivative $\frac{\partial^2 E(u_j(\tilde{v}|\theta_j))}{\partial \tilde{v}_j^2}$ is always negative within reasonable intervals and $b_j(\tilde{v}_i|v_j)$ maximizes $E(u_j(\tilde{v}|v_j))$ for $\alpha = 2$. □

Appendix C

Appendix to Chapter 5

The following results from Heydenreich *et al.* (2006) are repeated for the sake of completeness.

Lemma 4 (Truthful weight report and node selection, Theorem 6 in Heydenreich *et al.* (2006)). *Regard any type vector θ, any strategy profile s and any job j such that j reports $(\tilde{r}_j, \tilde{d}_j, \tilde{v}_j)$ and chooses node $\tilde{m}$. Then changing the report to $(\tilde{r}_j, \tilde{d}_j, v_j)$ and choosing a node that maximizes its tentative utility at time $\tilde{r}_j$ does not decrease j's tentative utility under* DLGM.

Proof. First consider the single node case, i.e. $m = 1$. Suppose, at the arrival time $\tilde{r}_j$ of job j jobs $k_1, k_2, \ldots, k_r$ with corresponding reported processing times $\tilde{d}_1, \tilde{d}_2, \ldots, \tilde{d}_r$ and reported weights $\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_r$ are queueing to be processed on the node, but none of them has started being processed yet. Without loss of generality let $\tilde{v}_1/\tilde{d}_1 \geq \tilde{v}_2/\tilde{d}_2 \geq \ldots \geq \tilde{v}_r/\tilde{d}_r$. Given the reported processing time $\tilde{d}_j$, job j could receive any position in front of, between or behind the already present jobs in the priority queue by choosing its weight appropriately. Therefore, it has to decide for every job k_s, $s \in \{1, \ldots, r\}$, whether it wants to be placed in front of k_s or not. Displacing k_s would increase $\hat{p}_j(1)$ by $\tilde{v}_s \tilde{d}_j$, whereas $\hat{e}_j(1)$ is decreased by $\tilde{d}_s$. Thus, j's tentative utility changes by $v_j \tilde{d}_s - \tilde{v}_s \tilde{d}_j$ if j displaces k_s compared to not displacing k_s. Therefore, it is rational for j to displace k_s if and only if $v_j \tilde{d}_s - \tilde{v}_s \tilde{d}_j > 0$, which is equivalent to $v_j/\tilde{d}_j > \tilde{v}_s \tilde{d}_s$. As the node schedules according to WSPT, j is placed at the position that maximizes its tentative utility when reporting v_j.

For $m > 1$, recall that j can select a node itself. As reporting the truth maximizes its tentative utility on every single node, and as j can then choose the node that maximizes its tentative utility among all nodes, truth-telling and choosing a node maximizing $\hat{u}_j(.)$ will maximize j's tentative utility. $\square$

Lemma 5 (Preservation of the tentative utility, Lemma 7(a) in Heydenreich *et al.* (2006)). *Consider any job $j \in J$. Then, under* DLGM, *for all reports of all other users as well as all choices* *of nodes of the other users, the following is true:*

If j reports $\tilde{v}_j = v_j$, then the tentative utility when queued at any of the nodes will be preserved over time, i.e. it equals j's ex post utility.

Proof. Note that whenever j's tentative completion time changes, j is immediately compensated for that by a payment. If $\tilde{v}_j = v_j$ then the payment exactly equals the loss in utility. □

Lemma 6 (Truthful reports of release dates and runtimes, Theorem 8 in Heydenreich *et al.* (2006)). *Consider the restricted strategy space where all $j \in J$ report $\tilde{v}_j = v_j$. Then the strategy profile where all jobs j truthfully report $\tilde{r}_j = r_j$, $\tilde{d}_j = d_j$ and choose a node that maximizes $\hat{u}_j(.)$ is a dominant strategy equilibrium under* DLGM.

Proof. Start with $m = 1$. Suppose $\tilde{v}_j = v_j$, fix any pretended release date $\tilde{r}_j$ and regard any $\tilde{d}_j > d_j$. Let $u_j(.)$ denote j's (ex post) utility when reporting d_j truthfully and let $\tilde{u}_j(.)$ be its (ex post) utility for reporting $\tilde{d}_j$. As $\tilde{v}_j = v_j$, the ex post utility equals in both cases the tentative utility at decision point $\tilde{r}_j$ according to Lemma 5. Thus regard the latter utilities. Clearly, according to the WSPT-priorities, j's position in the queue at the node for report d_j will not be behind its position for report $\tilde{d}_j$. Divide the jobs already queuing at the node at j's arrival into three sets: Let $J_1 = \{k \in J | k < j, b_k > \tilde{r}_j, \tilde{v}_k/\tilde{d}_k \geq v_j/d_j\}$ (where b_k is the time when k is started to be processed), $J_2 = \{k \in J | k < j, b_k > \tilde{r}_j, v_j/d_j > \tilde{v}_k/\tilde{d}_k \geq v_j/\tilde{d}_j\}$ and $J_3 = \{k \in J | k < j, b_k > \tilde{r}_j, v_j/\tilde{d}_j > \tilde{v}_k/\tilde{d}_k\}$. That is, J_1 comprises the jobs that are in front of j in the queue for both reports, J_2 consists of the jobs that are only in front of j when reporting $\tilde{d}_j$ and J_3 includes only jobs that queue behind j for both reports. Therefore,

$$\begin{aligned}
\tilde{u}_j(.) - u_j(.) &= -\sum_{k \in J_1 \cup J_2} v_j \tilde{d}_k - \sum_{k \in J_3} \tilde{d}_j \tilde{v}_k - v_j \tilde{d}_j - \left(-\sum_{k \in J_1} v_j \tilde{d}_k - \sum_{k \in J_2 \cup J_3} d_j \tilde{v}_k - v_j d_j \right) \\
&= \sum_{k \in J_2} (d_j \tilde{v}_k - v_j \tilde{d}_k) - \sum_{k \in J_3} (\tilde{d}_j - d_j) \tilde{v}_k - (\tilde{d}_j - d_j) v_j.
\end{aligned}$$

According to the definition of J_2, the first term is smaller than or equal to zero. As $\tilde{d}_j > d_j$, the whole right hand side becomes non-positive. Therefore $\tilde{u}_j(.) \leq u_j(.)$, i.e. truthfully reporting d_j maximizes j's ex post utility on a single node.

Now fix $\tilde{v}_j = v_j$ and any $\tilde{d}_j \geq d_j$ and regard any false release date $\tilde{r}_j > r_j$. There are two effects that can occur when arriving later than r_j at the node. Firstly, jobs queued at the node already at time r_j may have been processed or may have started receiving service by time $\tilde{r}_j$. But, either j would have had to wait for those jobs anyway or it would have increased its immediate utility at decision point r_j by displacing a job and paying the compensation. So, j cannot gain from this effect by lying. The second effect is that new jobs have arrived at the node between r_j and $\tilde{r}_j$. Those jobs either delay j's completion time and j looses the payment it could have received from those jobs by arriving earlier. Or the jobs do not delay j's completion time, but j has to pay the jobs for displacing them when arriving at $\tilde{r}_j$. If j arrived at time r_j, it would not have to pay for displacing such a job. Hence, j cannot gain from this effect either and the immediate utility at decision point r_j will be at least as large as its immediate utility at decision point $\tilde{r}_j$. Therefore, for a single node, j maximizes its immediate utility at decision point $\tilde{r}_j$ by choosing $\tilde{r}_j = r_j$. As $\tilde{v}_j = v_j$, it follows from Lemma 5 that choosing $\tilde{r}_j = r_j$ also maximizes the job's ex post utility on a single node.

For $m > 1$ note that on every node, the immediate utility of job j at decision point $\tilde{r}_j$ is equal to its ex post utility and that j can select a node itself that maximizes its immediate utility and therefore its ex post utility. Therefore, given that $\tilde{v}_j = v_j$, a job's ex post utility is maximized by choosing $\tilde{r}_j = r_j$, $\tilde{d}_j = d_j$ and choosing a node that minimizes the immediate increase in the objective function. □

Theorem 1 (Myopic best response equilibrium under *DLGM*, Theorem 9 in Heydenreich *et al.* (2006)). *Given the types of all jobs, the strategy profile s where each job j reports $\tilde{\theta}_j = \theta_j$ and chooses the node n that maximizes its tentative utility $\hat{u}_j(\tilde{\theta}_j, n, \tilde{s}_{-j}|\theta_j) = -v_j\hat{e}_j(n) - \hat{p}_j(n)$ is a myopic best response equilibrium under* DLGM.

Proof. Regard job j. According to the proof of Lemma 4, $\hat{u}_j(.)$ on any node is maximized by reporting $\tilde{v}_j = v_j$ for any $\tilde{r}_j$ and $\tilde{d}_j$. According to Lemma 6, $\tilde{r}_j = r_j$, $\tilde{d}_j = d_j$ and choosing a node that maximizes j's tentative utility at time $\tilde{r}_j$ maximize j's ex post utility if j truthfully reports $\tilde{v}_j = v_j$. According to Lemma 5 this ex post utility is equal to $\hat{u}_j(.)$ if j reports $\tilde{v}_j = v_j$. Therefore, any job j maximizes $\hat{u}_j(.)$ by truthful reports and choosing the node as claimed. □

Appendix D

Appendix to Chapter 6

D.1 Workload Statistics

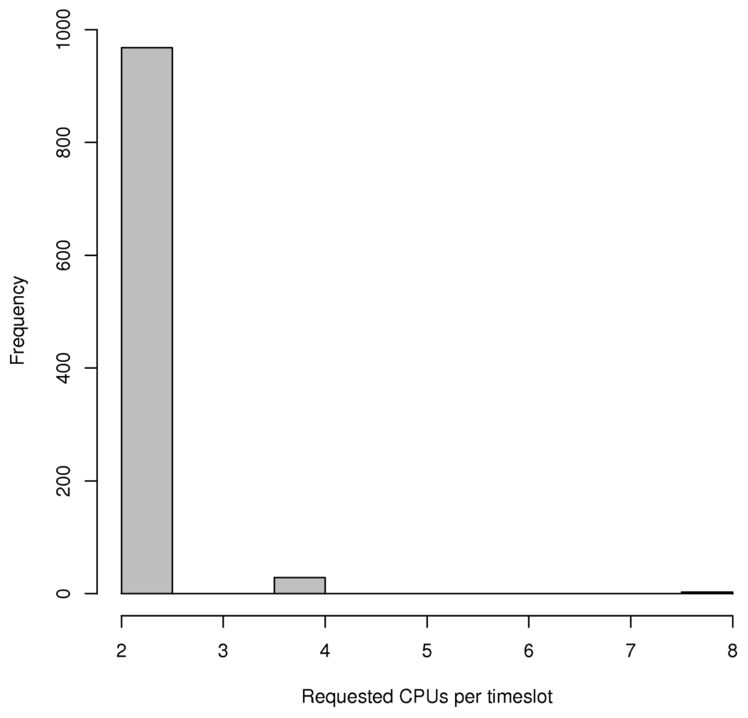


Figure D.1: Histogram of the CPU size of the requests in the workload.

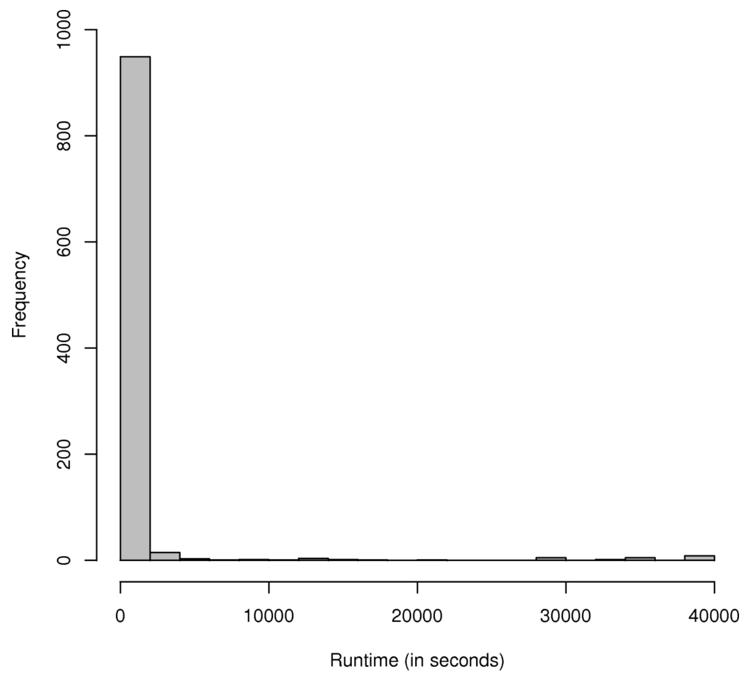


Figure D.2: Histogram of the runtime of the requests in the workload.

D.2 Correlation – Spearman Correlation Coefficient

The following table presents the mean Spearman correlation coefficients across 10 runs. The workloads were based on the valuation distribution with support $\{10,\ldots,160\}$ and were fed into the Proportional Share mechanism.

$\gamma \setminus \varepsilon$	0.05	0.15	0.25
Learning rate $\alpha = 0.1$			
0.3	0.799	0.766	0.733
0.5	0.781	0.771	0.74
0.7	0.751	0.753	0.743
Learning rate $\alpha = 0.3$			
0.3	0.808	0.752	0.725
0.5	0.807	0.752	0.727
0.7	0.797	0.76	0.734
Learning rate $\alpha = 0.5$			
0.3	0.796	0.747	0.723
0.5	0.795	0.747	0.726
0.7	0.785	0.752	0.731
Learning rate $\alpha = 0.7$			
0.3	0.774	0.735	0.715
0.5	0.771	0.74	0.721
0.7	0.773	0.745	0.732

Table D.1: Mean Spearman correlation coefficients for varying α, γ, and ε.

D.3 Rationality – Mean Request Utility

The following table presents the mean request utility across 10 runs. The workloads were based on the valuation distribution with support $\{10,\dots,160\}$ and were fed into the Proportional Share mechanism.

$\gamma \setminus \varepsilon$	0.05	0.15	0.25
Learning rate $\alpha = 0.1$			
0.3	67,395	61,408	55,394
0.5	61,447	57,152	52,950
0.7	52,982	50,214	47,098
Learning rate $\alpha = 0.3$			
0.3	69,055	62,598	55,237
0.5	65,544	59,803	53,877
0.7	57,546	55,257	50,807
Learning rate $\alpha = 0.5$			
0.3	67,525	60,471	53,716
0.5	63,810	58,650	52,427
0.7	59,037	55,251	50,019
Learning rate $\alpha = 0.7$			
0.3	65,265	58,451	51,804
0.5	61,622	56,598	50,639
0.7	58,203	54,055	48,484

Table D.2: Mean request utility for varying α, γ, and ε.

D.4 Robustness Against Varying Random Number Seeds

The following results present the coefficient of variation for the total payments across 10 runs. The workloads were based on the valuation distribution with support $\{10,\dots,160\}$ and were fed into the Proportional Share mechanism.

$\gamma \setminus \varepsilon$	0.05	0.15	0.25
Learning rate $\alpha = 0.1$			
0.3	0.029	0.012	0.005
0.5	0.065	0.02	0.005
0.7	0.062	0.015	0.011
Learning rate $\alpha = 0.3$			
0.3	0.013	0.005	0.003
0.5	0.035	0.007	0.005
0.7	0.03	0.01	0.005
Learning rate $\alpha = 0.5$			
0.3	0.01	0.004	0.004
0.5	0.025	0.006	0.004
0.7	0.032	0.009	0.005
Learning rate $\alpha = 0.7$			
0.3	0.016	0.006	0.003
0.5	0.04	0.009	0.004
0.7	0.022	0.008	0.004

Table D.3: Coefficient of variation for the total payments across all runs for varying α, γ, and ε.

D.5 Proof of Corollary 1

Proof. The left term can be rewritten as follows:

$$\begin{aligned}
\max_{j\in SH} \frac{\tilde{v}_j}{x_j} &= \max_{j\in SH} \frac{\tilde{v}_j}{\tau_j^{sh}(\tilde{v})} \\
&= \max_{j\in SH} \frac{\tilde{v}_j}{\frac{\tilde{v}_j}{\tilde{v}_{max}} \int\limits_0^1 \prod\limits_{\substack{k\in SH\\k\neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds} \\
&= \max_{j\in SH} \frac{\tilde{v}_{max}}{\int\limits_0^1 \prod\limits_{\substack{k\in SH\\k\neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds}.
\end{aligned}$$

Thus, one gets that

$$\operatorname*{argmax}_{j\in SH} \frac{\tilde{v}_j}{x_j} = \operatorname*{argmin}_{j\in SH} \int_0^1 \prod_{\substack{k\in SH\\k\neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds.$$

It is now shown that

$$\operatorname*{argmin}_{j\in SH} \int_0^1 \prod_{\substack{k\in SH\\k\neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds = \operatorname*{argmin}_{j\in SH} \tilde{v}_j. \tag{D.1}$$

Assume the term

$$\int_0^1 \prod_{k\in SH} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds$$

must be minimized by removing one sub-term $\left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right)$. This is achieved by removing the term $\left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right)$ with the minimal $\tilde{v}_k$ across all $k \in SH$. Consequently,

$$\operatorname*{argmin}_{j\in SH} \tilde{v}_j \subseteq \operatorname*{argmin}_{j\in SH} \int_0^1 \prod_{\substack{k\in SH\\k\neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds.$$

The fact that the left hand side of Equation D.1 is a subset of the right hand side can be shown by contradiction.

Assume there is a request

$$i \in \operatorname*{argmin}_{j\in SH} \int_0^1 \prod_{\substack{k\in SH\\k\neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds$$

with $i \notin \operatorname{argmin}_{j\in SH} \tilde{v}_j$, i.e. $\tilde{v}_i > \min_{j\in SH} \tilde{v}_j$.

However, then the value of the term

$$\int_0^1 \prod_{\substack{i \in SH \\ i \neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds$$

can be made smaller by interchanging i with an element in $\operatorname{argmin}_{j \in SH} \tilde{v}_j$. Thus it can be concluded that

$$i \notin \operatorname*{argmin}_{j \in SH} \int_0^1 \prod_{\substack{k \in SH \\ k \neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds,$$

and hence

$$\operatorname*{argmin}_{j \in SH} \int_0^1 \prod_{\substack{k \in SH \\ k \neq j}} \left(1 - s\frac{\tilde{v}_k}{\tilde{v}_{max}}\right) ds \subseteq \operatorname*{argmin}_{j \in SH} \tilde{v}_j.$$

$\square$

Bibliography

Adar, E. and Huberman, B. (2000). Free Riding on Gnutella. *First Monday*, **5**(10), 2.

Amar, L., Barak, A., Levy, E., and Okun, M. (2007). An On-line Algorithm for Fair-Share Node Allocations in a Cluster. In *Proceedings of the International Symposium on Cluster Computing and the Grid*, pages 83–91, 14–17, Rio de Janeiro, Brazil.

Amar, L., Stößer, J., Levy, E., Shiloh, A., Barak, A., and Neumann, D. (2008a). Harnessing Migrations in a Market-based Grid OS. In *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing*, page 85, 29 September – 1 October 2008, Tsukuba, Japan.

Amar, L., Mu'alem, A., and Stößer, J. (2008b). On the Importance of Migration for Fairness in Online Grid Markets. In *Proceedings of the 9th IEEE/ACM International Conference on Grid Computing*, page 65, 29 September – 1 October 2008, Tsukuba, Japan.

Amar, L., Mu'alem, A., and Stößer, J. (2008c). The Power of Preemption in Economic Online Markets. In *Proceedings of the 5th International Workshop on Grid Economics and Business Models*, pages 41–57, 25–26 August 2008, Las Palmas, Spain.

Anandasivam, A. and Neumann, D. (2008). Reputation-Based Pricing for Grid Computing in eScience. In *Proceedings of the 16th European Conference on Information Systems*, 9–11 June, Galway, Ireland.

Anandasivam, A. and Neumann, D. (2009). Managing Revenue in Grids. In *Proceedings of the 42nd Hawaii International Conference on System Sciences*, 5–8 January, Waikoloa, Hawaii, USA. Forthcoming.

Archer, A. and Tardos, E. (2001). Truthful Mechanisms for One-Parameter Agents. In *Proceedings of the 42nd IEEE Symposium on Foundations of Computer Science*, pages 482–491, 14–17 October, Las Vegas, Nevada, USA.

Archer, A., Papadimitriou, C., Talwar, K., and Tardos, E. (2003). An Approximate Truthful Mechanism for Combinatorial Auctions with Single Parameter Agents. In *Proceedings the*

14th Annual ACM-SIAM Symposium on Discrete Algorithms, pages 205–214, 12–14 January, Baltimore, ML, USA.

Arndt, O., Freisleben, B., Kielmann, T., and Thilo, F. (2000). A Comparative Study of Online Scheduling Algorithms for Networks of Workstations. *Cluster Computing*, **3**(2), 95–112.

AuYoung, A., Chun, B., Snoeren, A., and Vahdat, A. (2004). Resource Allocation in Federated Distributed Computing Infrastructures. In *Proceedings of the 1st Workshop on Operating System and Architectural Support for the On-Demand IT Infrastructure*, 9 October, Boston, MA, USA.

Awerbuch, B., Azar, Y., and Meyerson, A. (2003). Reducing Truth-Telling Online Mechanisms to Online Optimization. In *Proceedings of the 35th Annual ACM Symposium on Theory of Computing*, pages 503–510, 9–11 June, San Diego, CA, USA.

Baker, M., Fox, G., and Yau, H. (1995). Cluster Computing Review. Technical Report CRPC-TR95623, Northeast Parallel Architectures Center, Syracuse University.

Bapna, R., Das, S., Garfinkel, R., and Stallaert, J. (2008). A Market Design for Grid Computing. *INFORMS Journal on Computing*, **20**(1), 100–111.

Barak, A., Shiloh, A., and Amar, L. (2005). An Organizational Grid of Federated MOSIX Clusters. In *Proceedings of the 5th of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, pages 350–357, 9–12 May, Cardiff, UK.

Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R., Pratt, I., and Warfield, A. (2003). Xen and the Art of Virtualization. In *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pages 164–177, 19–22 October, Bolton Landing, NY, USA. ACM Press New York, NY, USA.

Beladym, C. (2007). In the Data Center, Power and Cooling Costs More than the IT Equipment it Supports. *Electronics Cooling Magazine*, **13**(1). `http://electronics-cooling.com/articles/2007/feb/a3/`.

Bell, G. and Gray, J. (2002). What's Next in High-Performance Computing? *Communications of the ACM*, **45**(2), 91–95.

Blau, B. and Schnizler, B. (2008). In *Proceedings of the Multikonferenz Wirtschaftsinformatik*, 26–28 February, Munich, Germany.

Blau, B., Lamparter, S., Neumann, D., and Weinhardt, C. (2008). Planning and Pricing of Service Mashups. In *Proceedings of the IEEE Joint Conference on E-Commerce Technology and*

Enterprise Computing, E-Commerce and E-Services, pages 19–26, 21–24 July, Washington, DC, USA.

Bodenbenner, P., Stößer, J., and Neumann, D. (2007). A Pay-as-Bid Mechanism for Pricing Utility Computing. In *Proceedings of the 20th Bled eConference, Merging and Emerging Technologies, Processes, and Institutions*, 4–6 June, Bled, Slovenia.

Bolton, G., Katok, E., and Ockenfels, A. (2002). How Effective are Online Reputation Mechanisms? Discussion Papers on Strategic Interaction 2002-25, Max Planck Institute of Economics, Strategic Interaction Group.

Bonabeau, E. (2002). Agent-Based Modeling: Methods and Techniques for Simulating Human Systems. *Proceedings of the National Academy of Sciences*, **99**(90003), 7280.

Bonorden, O., Gehweiler, J., and Heide, F. (2006). A Web Computing Environment for Parallel Algorithms in Java. *Lecture Notes in Computer Science*, **3911**, 801.

Borodin, A. and El-Yaniv, R. (1998). *Online Computation and Competitive Analysis*. Cambridge University Press.

Boss, G., Malladi, P., Quan, D., Legregni, L., and Hall, H. (2007). Cloud Computing. Technical report, IBM High Performance On Demand Solutions.

Böttcher, J., Drexl, A., Kolisch, R., and Salewski, F. (1999). Project Scheduling Under Partially Renewable Resource Constraints. *Management Science*, **45**(4), 543–559.

Bridge, P. and Sawilowsky, S. (1999). Increasing Physicians' Awareness of the Impact of Statistics on Research Outcomes Comparative Power of the t-test and Wilcoxon Rank-Sum Test in Small Samples Applied Research. *Journal of Clinical Epidemiology*, **52**(3), 229–235.

Brunner, R., Freitag, F., and Navarro, L. (2008). Towards the Development of a Decentralized Market Information System: Requirements and Architecture. In *Proceedings of the 22nd IEEE International Symposium on Parallel and Distributed Processing*, 14–18 April, Miami, FL, USA.

Burke, S., Campana, S., Lorenzo, P., Nater, C., Santinelli, R., and Sciaba, A. (2008). gLite 3.1 User Guide. CERN-LCG-GDEIS-722398.

Buyya, R., Abramson, D., Giddy, J., and Stockinger, H. (2002). Economic Models for Resource Management and Scheduling in Grid Computing. *Concurrency and Computation: Practice and Experience*, **14**(13-15), 1507–1542.

Buyya, R., Yeo, C., Venugopal, S., Broberg, J., and Brandic, I. (2008). Cloud Computing and Emerging IT Platforms: Vision, Hype, and Reality for Delivering Computing as the 5th Utility. `http://www.gridbus.org/reports/CloudITPlatforms2008.pdf`.

Byde, A., Salle, M., and Bartolini, C. (2003). Market-Based Resource Allocation for Utility Data Centers. Technical Report HPL-2003-188, HP Lab, Bristol.

Carr, N. (2003). IT Doesn't Matter. *Harvard Business Review*, **811**(5), 41–53.

Carr, N. (2005). The End of Corporate Computing. *MIT Sloan Management Review*, **46**(3), 67.

Chaubal, C. (2005). Scheduler Policies for Job Prioritization in the Sun N1 Grid Engine 6 System. Technical report, Sun BluePrints Online, Sun Microsystems, Inc., Santa Clara, CA, USA.

Chekuri, C. and Khanna, S. (2006). A PTAS for the Multiple Knapsack Problem. *SIAM Journal on Computing*, **35**(3), 713–728.

Christodoulou, G., Koutsoupias, E., and Vidali, A. (2007). A Lower Bound for Scheduling Mechanisms. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1163–1170, 7–9 January, New Orleans, LA, USA.

Chun, B. and Culler, D. (2000). Market-based Proportional Resource Sharing for Clusters. Technical Report CSD-00-1092, Computer Science Division, University of California at Berkeley.

Chun, B. N. and Culler, D. E. (2002). User-Centric Performance Analysis of Market-Based Cluster Batch Schedulers. In *Proceedings of the 2nd of the IEEE/ACM International Symposium on Cluster Computing and the Grid*, page 30, 21–24 May, Berlin, Germany.

Cirne, W., Brasileiro, F., Andrade, N., Costa, L., Andrade, A., Novaes, R., and Mowbray, M. (2006). Labs of the World, Unite!!! *Journal of Grid Computing*, **4**(3), 225–246.

Correa, J. and Wagner, M. (2005). LP-Based Online Scheduling: From Single to Parallel Machines. *Integer Programming and Combinatorial Optimization*, **3509**, 196–209.

Crites, R. and Barto, A. (1996). Improving Elevator Performance Using Reinforcement Learning. *Advances in Neural Information Processing Systems*, pages 1017–1023.

Czajkowski, K., Foster, I., and Kesselman, C. (1999). Resource Co-Allocation in Computational Grids. In *Proceedings of the 8th IEEE International Symposium on High Performance Distributed Computing*, pages 219–228, 3–6 August, Redondo Beach, CA, USA.

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

Degermark, M., Köhler, T., Pink, S., and Schelén, O. (1997). Advance Reservations for Predictive Service in the Internet. *Multimedia Systems*, **5**(3), 177–186.

Dellarocas, C. (2003). The Digitization of Word-Of-Mouth: Promise and Challenges of Online Feedback Mechanisms. *Management Science*, **49**(10), 1407–1424.

Dobzinski, S. (2007). Two Randomized Mechanisms for Combinatorial Auctions. *Lecture Notes in Computer Science*, **4627**, 89–103.

Dobzinski, S., Nisan, N., and Schapira, M. (2006). Truthful Randomized Mechanisms for Combinatorial Auctions. In *Proceedings of the 38th Annual ACM Symposium on Theory of Computing*, pages 644–652, 21–23 May, Washington, DC, USA.

Dongarra, J., Sterling, T., Simon, H., and Strohmaier, E. (2005). High-Performance Computing: Clusters, Constellations, MPPs, and Future Directions. *Computing in Science & Engineering*, pages 51–59.

Emmerich, W., Butchart, B., Chen, L., Wassermann, B., and Price, S. (2005). Grid Service Orchestration Using the Business Process Execution Language (BPEL). *Journal of Grid Computing*, **3**(3), 283–304.

Eymann, T., Neumann, D., Reinicke, M., Schnizler, B., Streitberger, W., and Veit, D. (2006). On the Design of a Two-Tiered Grid Market Structure. In *Business Applications of P2P and Grid Computing, Multikonferenz Wirtschaftsinformatik*, 20-22 February, Passau, Germany.

Fahrmeir, L., Künstler, R., Pigeot, I., and Tutz, G. (2004). *Statistik*. Springer-Verlag Berlin Heidelberg. In German.

Feitelson, D. (2002). Workload Modeling for Performance Evaluation. *Lecture Notes in Computer Science*, **2459**, 114–141.

Feitelson, D. (2008). Workload Modeling for Computer Systems Performance Evaluation. Draft. Available at `http://www.cs.huji.ac.il/~feit/wlmod/`.

Figueiredo, R. J., Dinda, P. A., and Fortes, J. A. B. (2003). A Case For Grid Computing On Virtual Machines. In *Proceedings of the 23rd International Conference on Distributed Computing Systems*, pages 550–559, 19-22 May, Providence, RI, USA.

Foster, I. (2002). What is the Grid? A Three Point Checklist. *Grid Today*, **1**(6), 22–25. `http://www.gridtoday.com/02/0722/100136.html`.

Foster, I. and Kesselman, C. (1997). Globus: a Metacomputing Infrastructure Toolkit. *International Journal of High Performance Computing Applications*, **11**(2), 115.

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

Foster, I., Kishimoto, H., Savva, A., Berry, D., Grimshaw, A., Horn, B., Maciel, F., Siebenlist, F., Subramaniam, R., Treadwell, J., *et al.* (2006). The Open Grid Services Architecture. Technical report, Open Grid Forum. Version 1.5.

Freitag, S., Yahyapour, R., Jankowski, G., and Januszewski, R. (2008). Virtualization Management for Grids and SOA. White paper WHP-0005, CoreGRID.

Friedman, E. and Parkes, D. (2003). Pricing WiFi at Starbucks: Issues in Online Mechanism Design. In *Proceedings of the 4th ACM Conference on Electronic Commerce*, pages 240–241, 9–12 June, San Diego, CA, USA. ACM Press New York, NY, USA.

Gillan, C., Graham, S., Levitt, M., McArthur, J., Murray, S., Turner, V., Villars, R., and Whalen, M. (1999). The ASP's Impact on the IT Industry: An IDC-Wide Opinion. IDC Bulletin No. 20323, `http://www.amsys.net/pdf/idpwhitepaper.pdf`.

Globus, A. (2001). Towards 100,000 CPU Cycle-Scavenging by Genetic Algorithms. Technical report, NAS technical report NAS-0-011, October 2001.

Goemans, M., Queyranne, M., Schulz, A., Skutella, M., and Wang, Y. (2002). Single Machine Scheduling with Release Dates. *SIAM Journal on Discrete Mathematics*, **15**(2), 165–192.

Goldberg, A., Hartline, J., Karlin, A., Saks, M., and Wright, A. (2006). Competitive Auctions. *Games and Economic Behavior*, **55**(2), 242–269.

Graham, R. L., Lawler, E. L., Lenstra, J. K., and Kan, A. H. G. R. (1979). Optimization and approximation in deterministic sequencing and scheduling theory: a survey. *Annals of Discrete Mathematics*, **5**, 287–326.

Green, J. and Laffont, J. (1979). *Incentives in Public Decision-making*. North-Holland.

Green, P. and Rao, V. (1971). Conjoint Measurement for Quantifying Judgmental Data. *Journal of Marketing Research*, **8**(3), 355–363.

Guo, Z., Koehler, G., and Whinston, A. (2007). Market-Based Optimization Algorithms for Distributed Systems. *Management Science*, **53**(8), 1345.

Harchol-Balter, M. and Downey, A. (1997). Exploiting Process Lifetime Distributions for Dynamic Load Balancing. *ACM Transactions on Computer Systems*, **15**(3), 253–285.

Heydenreich, B., Müller, R., and Uetz, M. (2006). Decentralization and Mechanism Design for Online Machine Scheduling. *Lecture Notes in Computer Science*, **4059**, 136–147.

Huang, K. and Thulasiram, R. (2005). Parallel Algorithm for Pricing American Asian Options with Multi-Dimensional Assets. In *Proceedings of the 19th International Symposium on High Performance Computing Systems and Applications*, pages 177–185, 15–18 May, Guelph, ON, Canada.

Hurwicz, L. (1972). On Informationally Decentralized Systems. *Decision and Organization*, pages 297–336.

Hurwicz, L. (1975). On the Existence of Allocation Systems Whose Manipulative Nash Equilibria are Pareto Optimal. Presented at the 3rd World Congress of the Econometric Society (unpublished).

Hurwicz, L. and Walker, M. (1990). On the Generic Nonoptimality of Dominant-Strategy Allocation Mechanisms: A General Theorem That Includes Pure Exchange Economies. *Econometrica*, **58**(3), 683–704.

IEEE (1990). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. IEEE, , New York, NY, USA.

Insight Research (2006). Grid Computing – A Vertical Market Perspective 2006–2011. Market study, The Insight Research Corportation, New Jersey, USA. Executive summary.

Irwin, D., Grit, L., and Chase, J. (2004). Balancing Risk and Reward in a Market-Based Task Service. In *Proceedings of the 13th IEEE International Symposium on High Performance Distributed Computing*, 4–6 June, Honolulu, Hawaii, USA.

Johari, R. and Tsitsiklis, J. (2004). Efficiency Loss in a Network Resource Allocation Game. *Mathematics of Operations Research*, **29**(3), 407–435.

Joseph, J., Ernest, M., and Fellenstein, C. (2004). Evolution of Grid Computing Architecture and Grid Adoption Models. *IBM Systems Journal*, **43**(4), 624.

Kaelbling, L. (1993). Learning to Achieve Goals. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 1094–1098, 28 August – 3 September, Chambéry, France.

Kaelbling, L., Littman, M., and Moore, A. (1996). Reinforcement Learning: A Survey. *Journal of Artificial Intelligence Research*, **4**, 237–285.

Klems, M., Nimis, J., and Tai, S. (2008). Do Clouds Compute? A Framework for Estimating the Value of Cloud Computing. In *Proceedings of the 7th Workshop on e-Business*, 13 December, Paris, France. Forthcoming.

Knolmayer, G. (2000). Application Service Providing (ASP). *Wirtschaftsinformatik*, **42**(3), 443–446. In German.

Lai, K. (2005). Markets Are Dead, Long Live Markets. *ACM SIGecom Exchanges*, **5**(4), 1–10.

Lai, K., Rasmusson, L., Adar, E., Zhang, L., and Huberman, B. (2005). Tycoon: An Implementation of a Distributed, Market-Based Resource Allocation System. *Multiagent and Grid Systems*, **1**(3), 169–182.

Lavi, R. and Nisan, N. (2005). Online Ascending Auctions for Gradually Expiring Items. In *Proceedings of the 16th Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 1146–1155, 23–25 January, Vancouver, BC, Canada.

Lavi, R., Mu'alem, A., and Nisan, N. (2003). Towards a Characterization of Truthful Combinatorial Auctions. In *Proceedings of the 44th Annual IEEE Symposium on Foundations of Computer Science*, page 574, 11–14 October, Cambridge, MA, USA. IEEE Computer Society Washington, DC, USA.

Lawton, G. (2008). Moving the OS to the Web. *Computer*, **41**(3), 16–19.

Lehmann, D., O'Callaghan, L., and Shoham, Y. (2002). Truth Revelation in Approximately Efficient Combinatorial Auctions. *Journal of the ACM*, **49**(5), 577–602.

Luce, R. and Tukey, J. (1964). Simultaneous Conjoint Measurement: A New Type of Fundamental Measurement. *Journal of Mathematical Psychology*, **1**(1), 1–27.

MacKie-Mason, J. K. and Wellman, M. (2006). Automated Markets and Trading Agents. In L. Tesfatsion and K. Judd, editors, *Handbook of Computational Economics, vol. 2: Agent-Based Computational Economics*. North-Holland.

Marena, M., Marazzina, D., and Fusai, G. (2008). Option Pricing, Maturity Randomization and Grid Computing. In *Proceedings of the 22nd IEEE International Symposium on Parallel and Distributed Processing*, pages 1–8, 14–18 April, Miami, FL, USA.

Martello, S. and Toth, P. (1990). *Knapsack Problems: Algorithms and Computer Implementations*. John Wiley & Sons, Inc. New York, NY, USA.

Mas-Colell, A., Whinston, M., and Green, J. (1995). *Microeconomic Theory*. Oxford University Press New York.

Maskin, E. and Riley, J. (2000). Asymmetric Auctions. *The Review of Economic Studies*, **67**(3), 413–438.

Megow, N. (2007). *Coping with Incomplete Information in Scheduling – Stochastic and Online Models*. Ph.D. thesis, Technische Universität Berlin.

Megow, N. and Schulz, A. (2004). On-line Scheduling to Minimize Average Completion Time Revisited. *Operations Research Letters*, **32**(5), 485–490.

Milojicic, D., Kalogeraki, V., Lukose, R., Nagaraja, K., Pruyne, J., Richard, B., Rollins, S., and Xu, Z. (2002). Peer-to-Peer Computing. *HP Laboratories Palo Alto, March*.

Minoli, D. (2005). *A Networking Approach to Grid Computing*. Wiley Hoboken, NJ.

Motwani, R., Phillips, S., and Torng, E. (1993). Non-Clairvoyant Scheduling. In *Proceedings of the 4th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 422–431, 25–27 January, Austin, TX, USA. Society for Industrial and Applied Mathematics Philadelphia, PA, USA.

Moulin, H. (2008). Proportional Scheduling, Split-proofness, and Merge-proofness. *Games and Economic Behavior*, **63**(2), 567–587.

Mu'alem, A. and Feitelson, D. (2001). Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. *IEEE Transactions on Parallel and Distributed Systems*, **12**(6), 529–543.

Mu'alem, A. and Nisan, N. (2008). Truthful Approximation Mechanisms for Restricted Combinatorial Auctions. *Games and Economic Behavior*, **64**(2), 612–631.

Mu'alem, A. and Schapira, M. (2007). Setting Lower Bounds on Truthfulness. In *Proceedings of the 18th Annual ACM-SIAM Symposium on Discrete algorithms*, pages 1143–1152, 7–9 January, New Orleans, LA, USA. Extended abstract.

Myerson, R. and Satterthwaite, M. (1983). Efficient Mechanisms for Bilateral Trading. *Journal of Economic Theory*, **29**(2), 265–281.

Nash, J. (1951). Non-Cooperative Games. *Annals of Mathematics*, **54**(2), 286–295.

Neumann, D. (2004). *Market Engineering – A Structured Design Process for Electronic Markets*. Ph.D. thesis, Universität Karlsruhe (TH).

Neumann, D., Lamparter, S., and Schnizler, B. (2006a). Automated Bidding for Trading Grid Services. In *Proceedings of the 14th European Conference on Information Systems*, 12–14 June, Gothenburg, Sweden.

Neumann, D., Veit, D., and Weinhardt, C. (2006b). Grid Economics: Market Mechanisms for Grid Markets. In T. Barth and A. Schüll, editors, *Grid Computing: Konzepte, Technologien, Anwendungen*, pages 64–83. Vieweg Verlag. In German.

Neumann, D., Stößer, J., and Weinhardt, C. (2007a). Bridging the Grid Adoption Gap – Developing a Roadmap for Trading Grids. In *Proceedings of the 20th Bled eConference, Merging and Emerging Technologies, Processes, and Institutions*, 4–6 June, Bled, Slovenia.

Neumann, D., Stößer, J., Anandasivam, A., and Borissov, N. (2007b). SORMA – Building an Open Grid Market for Grid Resource Allocation. *Lecture Notes in Computer Science*, **4685**, 194.

Neumann, D., Stößer, J., Weinhardt, C., and Nimis, J. (2008a). A Framework for Commercial Grids – Economic and Technical Challenges. *Journal of Grid Computing*, **6**(3), 325–347.

Neumann, D., Borissov, N., Stößer, J., and See, S. (2008b). Best Myopic vs. Rational Response: An Evaluation of an Online Scheduling Mechanism. In *Proceedings of the 70. Wissenschaftliche Jahrestagung des Verbands der Hochschullehrer fr Betriebswirtschaft e.V.*, page 53, 15–17 May, Berlin, Germany.

Neumann, D., Stößer, J., and Weinhardt, C. (2008c). Bridging the Adoption Gap–Developing a Roadmap for Trading in Grids. *Electronic Markets*, **18**(1), 65–74.

Nickalls, R. (1993). A New Approach to Solving the Cubic: Cardan's Solution Revealed. *The Mathematical Gazette*, **77**, 354–359.

Nisan, N. (2006). Bidding Languages for Combinatorial Auctions. In P. Cramton, Y. Shoham, and R. Steinberg, editors, *Combinatorial Auctions*, chapter 9, pages 215–233. MIT Press, Cambridge, MA.

Nisan, N. and Ronen, A. (2001). Algorithmic Mechanism Design. *Games and Economic Behavior*, **35**(1-2), 166–196.

Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA.

Ong, T., Lim, T., Lee, B., and Yeo, C. (2002). Unicorn: Voluntary Computing Over Internet. *ACM SIGOPS Operating Systems Review*, **36**(2), 36–51.

Papazoglou, M. (2003). Service-Oriented Computing: Concepts, Characteristics and Directions. In *Proceedings of the 4th International Conference on Web Information Systems Engineering*, 10–12 December, Rome, Italy.

Parkes, D. (2001). *Iterative Combinatorial Auctions: Achieving Economic and Computational Efficiency*. Ph.D. thesis, University of Pennsylvania.

Parkes, D., Kalagnanam, J., and Eso, M. (2002). Achieving Budget-Balance with Vickrey-Based Payment Schemes in Combinatorial Exchanges. Technical Report RC 22218 W0110-065, IBM T.J. Watson Research Center.

Phelps, S. (2007). *Evolutionary Mechanism Design*. Ph.D. thesis, University of Liverpool.

Porter, R. (2004). Mechanism Design for Online Real-Time Scheduling. In *Proceedings of the 5th ACM Conference on Electronic Commerce*, pages 61–70, 17–20 May, New York, NY, USA. ACM New York, NY, USA.

Ramsey, P. (1980). Exact Type 1 Error Rates for Robustness of Student's t Test with Unequal Variances. *Journal of Educational and Behavioral Statistics*, **5**(4), 337.

Rana, O., Warnier, M., Quillinan, T., Brazier, F., and Cojocarasu, D. (2008). Managing Violations in Service Level Agreements. In D. Talia, R. Yahyapour, and W. Ziegler, editors, *Grid Middleware and Services*, pages 349–358. Springer US.

Rappa, M. (2004). The Utility Business Model and the Future of Computing Services. *IBM Systems Journal*, **43**(1), 32–42.

Regev, O. and Nisan, N. (2000). The POPCORN market – Online Markets for Computational Resources. *Decision Support Systems*, **28**(1-2), 177–189.

Resnick, P., Zeckhauser, B., Friedman, E., and Kuwabara, K. (2000). Reputation Systems. *Communications of the ACM*, **12**(43), 45–48.

Roberts, D. and Postlewaite, A. (1976). The Incentives for Price-Taking Behavior in Large Exchange Economies. *Econometrica*, **44**(1), 115–127.

Roth, A. (2002). The Economist as Engineer: Game Theory, Experimentation, and Computation as Tools for Design Economics. *Econometrica*, **70**(4), 1341–1378.

Roth, A. and Ockenfels, A. (2002). Last-Minute Bidding and the Rules for Ending Second-Price Auctions: Evidence from eBay and Amazon Auctions on the Internet. *American Economic Review*, **92**(4), 1093–1103.

Sääksjärvi, M., Lassila, A., and Nordström, H. (2005). Evaluating the Software as a Service Business Model: From CPU Time-Sharing to Online Innovation Sharing. In *Proceedings of the IADIS International Conference e-Society*, pages 177–186, 27–30 June, Qawra, Malta.

Saaty, T. (1990). *Multicriteria Decision Making – The Analytic Hierarchy Process: Planning, Priority Setting, Resource Allocation*. RWS Publications.

Sandholm, T. and Crites, R. (1996). On Multiagent Q-Learning in a Semi-Competitive Domain. *Lecture Notes in Computer Science*, pages 191–205.

Sandholm, T., Suri, S., Gilpin, A., and Levine, D. (2005). CABOB: A Fast Optimal Algorithm for Winner Determination in Combinatorial Auctions. *Management Science*, **51**(3), 374–390.

Sanghavi, S. and Hajek, B. (2004). Optimal Allocation of a Divisible Good to Strategic Users. In *Proceedings of the 43rd IEEE Conference on Decision and Control*, 14–17 December, Atlantis, Bahamas.

Sarmenta, L. (2001). *Volunteer Computing*. Ph.D. thesis, Massachusetts Institute of Technology.

Sawilowsky, S. and Blair, R. (1992). A More Realistic Look at the Robustness and Type II Error Properties of the t-test to Departures from Population Normality. *Psychological bulletin*, **111**(2), 352–360.

Schnizler, B. (2007). *Resource Allocation in the Grid – A Market Engineering Approach*. Ph.D. thesis, Universität Karlsruhe (TH).

Schnizler, B., Neumann, D., Veit, D., and Weinhardt, C. (2008). Trading Grid Services – A Multi-Attribute Combinatorial Approach. *European Journal of Operational Research*, **187**(3), 943–961.

Schwartz, A. (1993). A Reinforcement Learning Method for Maximizing Undiscounted Rewards. In *Proceedings of the 10th International Conference on Machine Learning*, 27–29 June, Amherst, MA, USA.

Shao, G., Berman, F., and Wolski, R. (2000). Master/Slave Computing on the Grid. In *Proceedings of the 9th Heterogeneous Computing Workshop*, page 3.

Shneidman, J., Ng, C., Parkes, D., AuYoung, A., Snoeren, A., Vahdat, A., and Chun, B. (2005). Why Markets Could (But Don't Currently) Solve Resource Allocation Problems in Systems. In *Proceedings of the 10th Conference on Hot Topics in Operating Systems*, page 7, 12–15 June, Santa Fe, NM, USA.

Smith, A. (1904). *An Inquiry Into the Nature and Causes of the Wealth of Nations*. Methuen & Co. Based on E. Cannan's 1904 compilation of Smith's 5*th* edition of the book (1789).

Smith, V. (1982). Microeconomic Systems as an Experimental Science. *American Economic Review*, **72**(5), 923–955.

Smith, W., Foster, I., and Taylor, V. (1998). Predicting Application Run Times Using Historical Information. *Lecture Notes on Computer Science*, **1459**(122-142), 203.

Smith, W. E. (1956). Various Optimizers for Single-Stage Production. *Naval Resource Logistics Quarterly*, **3**, 59–66.

Snelling, D., van den Berghe, S., von Laszewski, G., Wieder, P., MacLaren, J., Brooke, J., Nicole, D., and Hoppe, H. (2002). A Unicore Globus Interoperability Layer. *Computing and Informatics*, **21**(4), 399–411.

SORMA Consortium (2007). Preliminary Specification and Design Documentation of the SORMA Components. Deliverable D2.1 of the EU FP6 project 034286 "SORMA – Self-Organizing ICT Resource Management".

Stoica, I., Abdel-Wahab, H., Jeffay, K., Baruah, S., Gehrke, J., and Plaxton, C. (1996). A Proportional Share Resource Allocation Algorithm for Real-Time, Time-Shared Systems. In *Proceedings of the 17th IEEE Real-Time Systems Symposium*, pages 288–299, 4–6 December, Washington, DC, USA.

Stoica, I., Abdel-Wahab, H., and Jeffay, K. (1997). On the Duality between Resource Reservation and Proportional Share Resource Allocation. *Multimedia Computing and Networking Proceedings, SPIE Proceedings Series*, **3020**, 207–214.

Stößer, J. (2008). A Randomized Pay-as-Bid Mechanism for Grid Resource Allocation. In *Proceedings of the IEEE Joint Conference on E-Commerce Technology and Enterprise Computing, E-Commerce and E-Services*, pages 11–18, 21–24 July, Washington, DC, USA.

Stößer, J. and Neumann, D. (2007). GREEDEX – A Scalable Clearing Mechanism for Utility Computing. In *Proceedings of the Networking and Electronic Commerce Research Conference*, pages 221–231, 18–21 October, Riva Del Garda, Italy.

Stößer, J. and Neumann, D. (2008). GREEDEX – A Scalable Clearing Mechanism for Utility Computing. *Electronic Commerce Research*, **8**(4), 235–253.

Stößer, J., Neumann, D., and Anandasivam, A. (2007a). A Truthful Heuristic for Efficient Scheduling in Network-Centric Grid OS. In *Proceedings of the 15th European Conference on Information Systems*, pages 1052–1063, 7–9 June, St. Gallen, Switzerland.

Stößer, J., Neumann, D., and Weinhardt, C. (2007b). Market-based Pricing in Grids: On Strategic Manipulation and Computational Cost. Presented at the Journal of AIS Sponsored Theory Development Workshop, 12 December, Montréal, Canada.

Stößer, J., Bodenbenner, P., See, S., and Neumann, D. (2008). A Discriminatory Pay-as-Bid Mechanism for Efficient Scheduling in the Sun N1 Grid Engine. In *Proceedings of the 41st Hawaii International Conference on System Sciences*, page 382, 7–10 January, Waikoloa, Hawaii, USA.

Sun Microsystems (2008). Sun N1 Grid Engine 6 User's Guide. Sun Microsystems, Inc., Santa Clara, CA, USA, `http://docs.sun.com/app/docs/doc/817-6117/`.

Sutherland, I. (1968). A Futures Market in Computer Time. *Communications of the ACM*, **11**(6), 449–451.

Sutton, R. and Barto, A. (1998). *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, USA.

Tesauro, G. and Kephart, J. (2002). Pricing in Agent Economies Using Multi-Agent Q-Learning. *Autonomous Agents and Multi-Agent Systems*, **5**(3), 289–304.

The451Group (2007). Grid Computing – The State of the Market. Market study, The451Group, Boston, USA. Executive summary.

Turner, M., Budgen, D., and Brereton, P. (2003). Turning Software into a Service. *Computer*, **36**(10), 38–44.

van Dinther, C., Blau, B., and Conte, T. (2009). Strategic Behavior in Service Networks under Price and Service Level Competition. In *Proceedings of the 9. Internationale Tagung Wirtschaftsinformatik*, 25–27 February, Vienna, Austria. Forthcoming.

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

Walker, M. (1980). On the Nonexistence of a Dominant Strategy Mechanism for Making Optimal Public Decisions. *Econometrica*, **48**(6), 1521–1540.

Wang, D. (2007). Meeting Green Computing Challenges. In *Proceedings of the International Symposium on High Density Packaging and Microsystem Integration*, pages 1–4, 26–28 June, Shanghai, China.

Watkins, C. (1989). *Learning From Delayed Rewards*. Ph.D. thesis, Cambridge University.

Watkins, C. and Dayan, P. (1992). Q-learning. *Machine Learning*, **8**(3), 279–292.

Weidlich, A. (2008). *Engineering Interrelated Electricity Markets – An Agent-Based Computational Approach*. Ph.D. thesis, Universität Karlsruhe (TH).

Weinberg, M. and Rosenschein, J. (2004). Best-Response Multiagent Learning in Non-Stationary Environments. In *Proceedings of the 3rd International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 506–513, 19-23 August, New York, NY, USA.

Weinhardt, C., Holtmann, C., and Neumann, D. (2003). Market Engineering. *Wirtschaftsinformatik*, **45**(6), 635–640.

Weinhardt, C., Anandasivam, A., Blau, B., and Stößer, J. (2008). The Advent of Cloud Computing in the Service World. Submitted to IT Professional.

Weiss, A. (2007). Computing in the Clouds. *netWorker*, **11**(4), 16–25.

Wellman, M., Walsh, W., Wurman, P., and MacKie-Mason, J. (2001). Auction Protocols for Decentralized Scheduling. *Games and Economic Behavior*, **35**(1-2), 271–303.

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