

Market-Based Pricing in Grids: On Strategic Manipulation and Computational Cost

Jochen Stößer* and Christof Weinhardt

Karlsruhe Institute of Technology, Universität Karlsruhe (TH), Germany

Dirk Neumann

Albert-Ludwigs-Universität Freiburg, Germany

Abstract

Grid technologies and the related concepts of utility computing and cloud computing enable the dynamic sourcing of computer resources and services, thus allowing enterprises to cut down on hardware and software expenses and to focus on key competencies and processes. Resources are shared across administrative boundaries, e.g. between enterprises and/or business units. In this dynamic and inter-organizational setting, scheduling and pricing become key challenges. Market mechanisms show promise for enhancing resource allocation and pricing in grids. Current mechanisms, however, are not adequately able to handle large-scale settings with strategic users and providers who try to benefit from manipulating the mechanism. In this paper, a market-based heuristic for clearing large-scale grid settings is developed. The proposed heuristic and pricing schemes find an interesting match between scalability and strategic behavior.

Keywords: Auctions and bidding; Pricing; Scheduling; Heuristics; Grid

1 Introduction

Information technology (IT) departments are facing a challenging dilemma. On the one hand, they need to support increasingly complex applications requiring massive amounts of computer resources, such as systems for enterprise resource planning, engineering applications, or demand forecasts. On the other hand, IT departments are under constant pressure to cut down on hardware and software expenses as well as energy for electricity and cooling. But to be able to accommodate peak loads, they 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 companies' data storage capacity is being wasted (Carr, 2005).

Grid computing – and the related concepts of utility computing and cloud computing – offer a promising way out of this dilemma by allowing organizations to respond quickly to peaks in demand for computer resources. According to Foster (2002), a grid “coordinates resources that are not subject to centralized control [. . .], uses standard, open, general-purpose protocols and interfaces [. . .], and delivers non-trivial qualities of service [. . .].” It permits computer

*Corresponding author. Address: Englerstr. 14, 76131 Karlsruhe, Germany; E-Mail: jochen.stoesser@kit.edu; Phone: +49-721-608-8372; Fax: +49-721-608-8399.

resources to be shared across administrative boundaries, e.g. between enterprises or among business units and departments within one enterprise. Enterprises need only accommodate the basic load; peak loads are sourced externally on demand.

In this dynamic and inter-organizational setting, the scheduling and pricing of resource requests and offers, i.e. deciding which user can access what resource at what time and at what cost, become key challenges. To this end, market mechanisms could revolutionize resource allocation in grids in several respects (Shneidman *et al.*, 2005; Lai, 2005): (1) They dynamically and efficiently match resource requests to resource offers based on the users' valuations, (2) they establish previously unknown prices that reflect the scarcity of the respective resources, and (3) they provide incentives to users to shift their demand in off-peak periods where market prices are lower, thus achieving some kind of user-driven load balancing. The advantages are straightforward: lower hardware and software costs, increased user involvement, more efficient use of scarce resources, incentives to contribute idle resources to the grid in return for the market price, and (dynamic) pricing based on actual resource consumption. 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.

Contribution of this paper. This paper proposes a market-based heuristic that offers an important 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 heuristic is highly scalable but still yields near-optimal allocations. Compared to previous heuristics (e.g. tsfGRID by Bapna *et al.* (2008)), the mechanism allows for strategic behavior on both sides of the market. Proportional Critical-Value Pricing constitutes the main increment compared to previous pricing schemes (e.g. k -Pricing by Schnizler *et al.* (2008)). If complemented by this novel pricing scheme, the mechanism achieves truthfulness on the demand side of the market while limiting strategic behavior on the supply side.

Structure of this paper. Section 2 describes the setting and the resulting domain-specific and economic requirements and design desiderata. In Section 3, related work in the field of market mechanisms in grids is presented. In Section 4, a scalable heuristic for clearing grid markets is designed. In Section 5, the allocation problem and the mechanism are evaluated in more detail by means of a numerical experiment. Section 6 discusses the implications of these findings. Finally, Section 7 summarizes the paper and highlights future research directions.

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.

Double-sided market: The mechanism must support the trading between multiple strategic users and multiple strategic resource providers that issue job requests and offers of compute nodes. 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 compute 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). 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: 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 (and have to) 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 (see Appendix A in the electronic companion for formal definitions):

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

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.

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. For example, users do not have to pay a participation fee or price without receiving any resources in return.

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 (and even untrue) $\tilde{\theta}_{-i}$. 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. Moreover, it ensures that the participants' incentives are aligned towards the overall design goal. 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 .¹ Analogously, the utility of the provider of node n is $u_n(o, \theta | \theta_n) = p_n(\theta) - v_n(X(\theta))$.

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 the design desiderata (cf. e.g. Parkes (2001)). Among the most prominent is the Myerson-Satterthwaite Impossibility Theorem, which 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 (Myerson and Satterthwaite, 1983). 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 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 (Lehmann *et al.*, 2002; 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 work 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, (partially) truthful and at least approximately allocatively efficient. This trade-off will be tested both analytically and numerically.

¹Note 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.

3 Related Work

The focus of this work is on systems for computation-intensive applications, such as in high-performance computing. Research in the market-based allocation of compute resources can best be separated according to the stakeholders in a grid market: the users that aim to obtain resources via the market, the resource providers that contribute compute clusters and that possibly pursue local scheduling strategies and distributed markets, and intermediaries that act as resource brokers by running centralized grid markets.

3.1 Bidding Strategies of Users

From a mechanism design perspective, it is desirable to develop market mechanisms that align the users' goals (utility maximization) with the overall design goal (welfare maximization).

For some mechanisms, theoretic analyses show that users cannot benefit from deviating from some equilibrium strategy (cf. Mas-Colell *et al.* (1995); Nisan *et al.* (2007)). The strongest concept in this regard is truthfulness as introduced above. Other weaker solution concepts are Nash, Bayes-Nash and myopic best response equilibria. If the complexity of the setting does not allow the theoretic analysis of a mechanism, or in case it is impossible to construct a mechanism that implements strong game-theoretic solution concepts, the fallback approach is to apply bidding heuristics or approaches from the design of software agents, e.g. the Zero-Intelligence Plus strategy or more sophisticated algorithms that rely on reinforcement learning techniques such as Q-Learning (cf. Phelps (2007) and the references therein).

As will be shown below, the mechanism that we propose in this paper incorporates strong incentives to truthfully report resource constraints and valuations. As pointed out above, this tremendously reduces the strategic complexity from the users' perspective.

3.2 Distributed Markets

Besides the work on the users' trading strategies, there is also ample research on distributed (or decentralized) markets for managing the allocation process in computing systems. We will subsequently briefly discuss some of the arguably most prominent *distributed* mechanisms.

In the Spawn system by Waldspurger *et al.* (1992), each resource provider runs a sealed-bid second-price auction for slices of CPU time. The users run configurable bidding processes that bid for these time slices in the various distributed auctions. Similarly, the Popcorn market described in Regev and Nisan (2000), besides implementing two double auctions, also comprises a sealed-bid second-price auction where the users bid for CPU time of the resource provider(s). In the Tycoon system, the resource providers run a local resource scheduler called "Auction Share" that allocates CPU time based on a variant of the prominent Proportional Share Scheduler (Lai *et al.*, 2005; Chun and Culler, 2000). Another promising distributed mechanism is the Decentralized Local Greedy Mechanism (DLGM) by Heydenreich *et al.* (2006). In this mechanism, the users compete for the positions in the providers' local scheduling queues. When complemented by a pricing scheme based on compensation payments (for delays) among the jobs,

truthful behavior is a myopic best strategy for users. Moreover, DLGM is 3.281-competitive with respect to total weighted completion time if compared to the omniscient (optimal) scheduler, and it is 2-competitive when preemptions are supported (Heydenreich *et al.*, 2006; Amar *et al.*, 2008b).

3.3 Centralized Grid Brokers

The advantage of distributed markets is that they do not rely on a central entity that might fail or become a bottleneck as the system load increases. The drawback is that the information about resource demand and supply is also distributed, which can lead to local and thus inefficient outcomes. This is probably one of the main reasons why most existing schedulers are centralized, such as the Maui cluster scheduler² and the GridWay meta-scheduler³. Moreover, the existing distributed mechanisms do only support the trading of one type of resource (CPU time). However, 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.

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 paper. 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 9 in Appendix B in the electronic companion, the mechanisms proposed for grids are not fully able to account for dependencies between multiple grid resources in large-scale settings with strategic users and providers. 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.

²<http://www.clusterresources.com/products/maui/>

³<http://www.gridway.org/>

⁴Note, 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).

⁵The 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.

4 The Mechanism

First, the bidding language will be introduced, which determines how job requests and node offers are expressed. In order to enable the true usability of market-based approaches, human users will be released from the burden of having to issue requests and offers manually. Instead, software agents will hide the system’s complexity from human users by automatically trading resources based on the current resource consumption of applications and configurable bidding rules, which automatically derive corresponding valuations (MacKie-Mason and Wellman, 2006). 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

4.1 Expressing Job Requests and Node Offers

In our model, computing power is the central scarce resource to be traded. Let N be the set of compute node offers that 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. 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. We assume that nodes can 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)). It is important to note that this paper considers independent batch jobs that do not need to be coordinated and synchronized, e.g. because of serial or parallel dependencies. The coordination of such dependencies would tremendously exacerbate the mechanism’s complexity and most likely violate the scalability requirement. Consequently, this complexity is shifted from the mechanism to the users. If users have jobs with parallel dependencies, one possibility

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

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 1: Sample job requests and node offers.

is to bundle these jobs to one large job. If there are precedence constraints, the users have to make sure to submit the jobs in the right order, e.g. by means of intelligent submission agents.

Example: The sample job requests and node offers listed in Table 1 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. She 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.

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:

$$\max_{X=(x_{jnt})} W = \sum_{j \in J} c_j \sum_{n \in N} \sum_{t \in T} x_{jnt} (v_j - v_n) \quad (1)$$

$$\text{subject to} \quad \sum_{n \in N} x_{jnt} \leq 1, \quad j \in J, t \in T, r_j \leq t \leq e_j \quad (2)$$

$$\sum_{j \in J} x_{jnt} c_j \leq C_n, \quad n \in N, t \in T, R_n \leq t \leq E_n \quad (3)$$

$$\sum_{j \in J} x_{jnt} m_j \leq M_n, \quad n \in N, t \in T, R_n \leq t \leq E_n \quad (4)$$

$$\sum_{u=r_j}^{e_j} \sum_{n \in N} x_{jnu} = d_j \sum_{n \in N} x_{jnt}, \quad j \in J, t \in T, r_j \leq t \leq e_j \quad (5)$$

$$x_{jnt} \in \{0, 1\}, \quad j \in J, n \in N, t \in T, r_j \leq t \leq e_j, R_n \leq t \leq E_n, v_n \leq v_j \quad (6)$$

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. The constraints have the following interpretation. Constraint (2)

ensures that a job can only be allocated to one node at a time. Constraints (3) and (4) specify that the jobs allocated to one node are not allowed to consume more resources than are available on this node. Constraint (5) enforces atomicity, i.e. a job is either fully executed in all requested timeslots or it is not executed at all. Finally, Constraint (6) 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.

This problem is a special case in between the Multiple Knapsack Problem (MKP) and the Generalized Assignment Problem. 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 (5) essentially interconnects a separate allocation problem for each timeslot, where each of these separate allocation problems is already more complex than MKP. 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. However, 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. Due to the structural connection between combinatorial auctions and knapsack problems, the heuristic is similar to earlier work on first-fit algorithms for knapsack problems, e.g. by Johnson *et al.* (1974), Sahni (1975), and Martello and Toth (1990). In this work, the basic heuristic is extended to the setting of double-sided multi-attribute auctions with timeslots. The allocation scheme is specified in Appendix C in the electronic companion. 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 1) of the exact formalization of the allocation problem for each job assignment. In the light of the economic design desiderata introduced above, in particular with respect to limiting strategic behavior, it is paramount to implement a *monotonic allocation scheme* in the sense that allocated job requests (node offers) would also be allocated when reporting a higher (lower) valuation (Lavi *et al.*, 2003). This reduces the set of possible heuristics tremendously, as for e.g. most randomized algorithms are not (deterministically) monotonic (Mu’alem and Nisan, 2008).

Example: Applying the greedy allocation scheme to the example above generates the allocation schedule depicted in Table 2 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 2: 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 1 (Competitive ratio). *The greedy heuristic has a competitive ratio of $\frac{W^{greedy}}{W^{OPT}} = 0$.*

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, $W^{greedy}/W^{OPT} \rightarrow 0$ for $k \rightarrow 0$. \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 (5)), 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 5 will thus report the results of an extensive numerical evaluation.

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.

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

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 correctly⁸; 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 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.

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. 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.

4.3.1 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 $p_j(\theta) = \phi_j(\theta)c_jd_j$ if j is allocated and $p_j(\theta) = 0$ else.

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:

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

Theorem 2 (Truthfulness of Critical-Value Pricing). *The greedy heuristic with Critical-Value Pricing is truthful with respect to job valuations:*

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

See Appendix D.2 in the electronic companion for the proof. 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. 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 3 (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.*

See Appendix D.3 in the electronic companion for the proof. This is clearly a strong negative result as it extends across any payment scheme for the greedy heuristic. As a consequence of Theorem 3, 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

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: Sample prices and payments.

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 prices and payments generated by Proportional Critical-Value Pricing for the sample job requests and node offers are listed in Table 3.

4.3.2 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 (also cf. Satterthwaite and Williams (1993)). 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 $p_j(\theta) = \sum_{n \in N} \sum_{t \in T} x_{jnt} c_j (v_j - k \cdot (v_j - v_n))$. The payment to node n is $p_n(\theta) = \sum_{j \in J} \sum_{t \in T} x_{jnt} c_j (v_n + (1 - k) \cdot (v_j - v_n))$.

k -Pricing has two main advantages. The distribution of welfare among users and providers can be flexibly pre-defined by setting the factor k , 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. The prices and payments generated by Proportional Critical-Value Pricing and k -Pricing with $k = 0.5$ are listed in Table 3. 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).

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

Unfortunately, this evaluation cannot be based on real grid workload traces. To the best of our knowledge, the only publicly available traces are cluster workloads available in the Parallel Workload Archive.⁹ But this data exhibits various problems. The traces are often incomplete and do not contain all of the required parameters. Another 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 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.

5.1 What Makes Instances Hard?

The computational tractability of the allocation problem mainly depends on the number of job requests and node offers. In addition to these parameters, the heuristic’s approximation of the optimal solution also depends on the level of “competition” in the market, especially 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.

5.1.1 Data Generation

Two parameters were varied: the number of job requests and node offers (20 nodes, 40 nodes, up to 200 nodes) and the ratio of jobs to nodes (one job per node, two jobs per node, and three jobs per node).

Based on the model and the corresponding bidding language, there is a set of further parameters upon which problem instances (“order books”) need to be generated. Table 4 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.

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)

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)
End time	1 + Binomial(5, 0.5)	1 + Binomial(8, 0.5)
Valuation	Uniform{10, ..., 20}	Uniform{7, ..., 12}

Table 4: Simulation setting.

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 $1 + \text{Binomial}(5, 0.5)$ 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.

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.

5.1.2 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 5 shows the impact of varying the competition in the market.

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

Nodes	$ J / N = 1$			$ J / N = 2$			$ J / N = 3$		
	<i>time</i>	<i>CV</i>	<i>exec</i>	<i>time</i>	<i>CV</i>	<i>exec</i>	<i>time</i>	<i>CV</i>	<i>exec</i>
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 5: 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*)[†]” indicates that this happened in *n* out of 30 cases. “(*n*)[‡]” indicates that CPLEX only found a *suboptimal* solution within 30 minutes in *n* out of 30 cases.

of 30 runs for 280 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 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 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 efficiency will become larger. Thus, to measure the inherent trade-off between computational complexity and allocative efficiency for the greedy heuristic, Table 6 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.

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

Nodes	$ J / N = 1$			$ J / N = 2$			$ J / N = 3$		
	<i>ratio</i>	<i>CV</i>	<i>exec</i>	<i>ratio</i>	<i>CV</i>	<i>exec</i>	<i>ratio</i>	<i>CV</i>	<i>exec</i>
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) [†]	–	–
140	0.987	0.06	0.996	0.978 (9) [‡]	0.04	0.872	– (30) [†]	–	–
160	0.987	0.04	0.995	0.982 (18) [‡]	0.03	0.862	– (30) [†]	–	–
180	0.988	0.05	0.996	0.985 (24) [‡]	0.05	0.860	– (30) [†]	–	–
200	0.991	0.05	0.994	– (16) [†]	–	–	– (30) [†]	–	–

Table 6: 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 of the heuristic’s efficiency across all 30 runs, and *exec* the ratio of jobs that the heuristic allocates compared to the number of submitted job requests.

time, the heuristic closely approximates the optimal allocative efficiency in the average case, as opposed to the negative result for the worst case (cf. Theorem 1).

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 4. 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.¹⁰ The p-values

¹⁰Due to the large sample size, the t-test is very robust to violations of the normality assumption underlying the

of the tests are indicated in the tables.

5.2.1 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 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 7 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 11 in Appendix E in the electronic companion 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. 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.

As stated in Theorem 2, 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

test (Ramsey, 1980; Sawilowsky and Blair, 1992).

Percentage bid	<i>k</i> -Pricing						Critical-Value Pricing	
	<i>k</i> = 0.3		<i>k</i> = 0.5		<i>k</i> = 0.7		<i>abs</i>	<i>rel</i>
	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>		
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 7: 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 $p = 0.01$.

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 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.

5.2.2 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 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

Percentage bid	<i>k</i> -Pricing						Proportional	
	<i>k</i> = 0.3		<i>k</i> = 0.5		<i>k</i> = 0.7		Critical-Value Pricing	
	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>
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 8: 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 $p = 0.01$.

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.

Table 8 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 12 in Appendix E in the electronic companion 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. 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 generally 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 12 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 2, 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.

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 paper, the design of grid markets was tackled from both ends: *scalability* and *strategic behavior*.

Our analysis has shown that exact mechanisms, which always optimally solve the allocation problem, are infeasible in practice due to their combinatorial NP-hardness. In contrast, the greedy heuristic is highly scalable in these sense that it can determine the allocations and prices for several hundred jobs and nodes in a few seconds. If employed in interval scheduling mode, Sun Microsystems’s *NIGE* scheduler is typically executed every 15 seconds. In our analysis, if complemented by 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 998 nodes at 485 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.

For simplification, in this paper we assume that the timeslots have uniform length and are synchronized. In reality the independent resource providers might apply varying local clock-

¹¹as of 06.04.2009, <http://www.planet-lab.org/>

ings according to their individual settings and needs. This actually is another argument in favor of a scalable mechanism (in the sense introduced above). Our assumption will generally become less restrictive as the length of the timeslots decreases, since this reduces the conflict (overlap) between the local clockings. As discussed above, the timeslots are one major cause for the complexity of the allocation problem, since Constraint (5) essentially interconnects a separate allocation problem for each timeslot. Consequently, compared to exact mechanisms, the scalability of the proposed heuristic allows to have more fine-grained timeslots and thus to reduce the conflicts caused by varying local clockings of the resource providers.

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 participants, welfare can be distributed among users and providers according to the parameter k , which can act as an adjusting screw. The question is, of course, how k should be set. With Proportional Critical-Value 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, the 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. If k – and thus the users' surplus – is small, both users and providers have a significant incentive to understate their true valuations. 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.

In large 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 settings, the importance of computational considerations naturally diminishes, and in line with the results for increasing market sizes, strategic considerations gain in importance. 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.

7 Conclusion and Future Work

In Section 2, the need to cope with several domain-specific and economic requirements and design desiderata was discussed that need to be considered when designing a market mechanism for computational grids. As summarized in Table 9, 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 Proportional Critical-Value Pricing present a considerable 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 possible limitation is the assumption that a job can be migrated between nodes at zero cost. However, zero cost migration is a common assumption in computer system analyses (cf. Karger *et al.* (1997); Amar *et al.* (2008a)). It can thus be hypothesized that the results also hold in realistic settings. In productive computing clusters, for instance, the use of virtualization technologies that provide off-the-shelf support for migration as well as the increase in memory and network capacity make migration a standard tool in systems management. Besides these rather technical issues, 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 possibly agent-based simulations, the effect of *multiple* misreporting users and providers 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.

References

- Amar, L., Stöber, 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*, pages 85–94.
- Amar, L., Mu’alem, A., and Stöber, J. (2008b). The Power of Preemption in Economic Online Markets. In *Proceedings of the 5th International Workshop on Grid Economics and Business Models*, pages 41–57.
- 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.
- 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*.
- Bapna, R., Das, S., Garfinkel, R., and Stallaert, J. (2008). A Market Design for Grid Computing. *INFORMS Journal on Computing*, **20**(1), 100–111.
- Carr, N. (2005). The End of Corporate Computing. *MIT Sloan Management Review*, **46**(3), 67.
- Chekuri, C. and Khanna, S. (2006). A PTAS for the Multiple Knapsack Problem. *SIAM Journal on Computing*, **35**(3), 713–728.

- 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.
- de Vries, S. and Vohra, R. (2003). Combinatorial Auctions: A Survey. *INFORMS Journal on Computing*, **15**(3), 284.
- Feitelson, D. (2002). Workload Modeling for Performance Evaluation. *LNCS*, **2459**, 114–141.
- 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>.
- 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.
- Hurwicz, L. (1972). On Informationally Decentralized Systems. *Decision and Organization*, pages 297–336.
- Johnson, D., Demers, A., Ullman, J., Garey, M., and Graham, R. (1974). Worst-Case Performance Bounds for Simple One-Dimensional Packing Algorithms. *SIAM Journal on Computing*, **3**, 299.
- Karger, D., Stein, C., and Wein, J. (1997). Scheduling algorithms. In M. J. Atallah, editor, *Handbook of Algorithms and Theory of Computation*. CRC Press.
- 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., 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.
- Lehmann, D., O’Callaghan, L., and Shoham, Y. (2002). Truth Revelation in Approximately Efficient Combinatorial Auctions. *Journal of the ACM*, **49**(5), 577–602.
- 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.
- 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.
- Mu’alem, A. and Nisan, N. (2008). Truthful Approximation Mechanisms for Restricted Combinatorial Auctions. *Games and Economic Behavior*, **64**, 612–631.

- Myerson, R. and Satterthwaite, M. (1983). Efficient Mechanisms for Bilateral Trading. *Journal of Economic Theory*, **29**(2), 265–281.
- 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.
- Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V. (2007). *Algorithmic Game Theory*. Cambridge University Press, New York, NY, USA.
- 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.
- 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.
- Regev, O. and Nisan, N. (2000). The POPCORN market – Online Markets for Computational Resources. *Decision Support Systems*, **28**(1-2), 177–189.
- Roberts, D. and Postlewaite, A. (1976). The Incentives for Price-Taking Behavior in Large Exchange Economies. *Econometrica*, **44**(1), 115–127.
- Sahni, S. (1975). Approximate Algorithms for the 0/1 Knapsack Problem. *Journal of the ACM*, **22**(1), 115–124.
- 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.
- Satterthwaite, M. and Williams, S. (1993). The Bayesian Theory of the k -Double Auction. In D. Friedman and J. Rust, editors, *The Double Auction Market: Institutions, Theories, and Evidence*, pages 99–124. Addison-Wesley.
- 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., 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.
- 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.
- 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.

A Formal Definitions of the Economic Design Desiderata

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:*

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

where \bar{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 $\tilde{\theta}_{-i}$:*

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

B Overview of Related Work

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 (CDAs)	✓	✓	×	×	×	✓	✓	×
MACE	×	✓	✓	✓	×	✓	✓	×
Bellagio	×	×	✓	✓	×	✓	✓	×
Bapna et al. (exact)	×	•	•	✓	×	✓	✓	✓
Bapna et al. (heuristic)	✓	•	•	✓	×	✓	✓	×
This work	✓	✓	•	✓	×	✓	✓	•

Table 9: Mapping of the existing mechanisms to the requirements. ✓ satisfied, • partly satisfied, × not satisfied.

C Pseudo-Code of the Heuristic's Allocation Algorithm

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  $(R_n \leq t \leq E_n) \ \&\& \ (c_j \leq C_n(t)) \ \&\& \ (m_j \leq M_n(t)) \ \&\& \ (v_n \leq v_j)$  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 accommo-
   dated 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 ca-
   pacity */
14: return  $X$ 

```

D Proofs

D.1 Proof of Theorem 1

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, $W^{greedy}/W^{OPT} \rightarrow 0$ for $k \rightarrow 0$. \square

D.2 Proof of Theorem 2

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 10).

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

Table 10: 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 $u_j(\tilde{\theta}|\theta_j) = (v_j - \phi_j(\tilde{\theta}))c_j d_j < 0 = u_j(\theta_j, \tilde{\theta}_{-j}|\theta_j)$. \square

D.3 Proof of Theorem 3

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

E Misreporting the Valuation under High Demand

Percentage bid	<i>k</i> -Pricing						Critical-Value Pricing	
	<i>k</i> = 0.3		<i>k</i> = 0.5		<i>k</i> = 0.7			
	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>
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 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 $p = 0.01$.

Percentage bid	<i>k</i> -Pricing						Proportional Critical-Value Pricing	
	<i>k</i> = 0.3		<i>k</i> = 0.5		<i>k</i> = 0.7			
	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>	<i>abs</i>	<i>rel</i>
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 12: 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 $p = 0.05$.