A heuristic approach for the allocation of resources in large-scale computing infrastructures

Kevin Lee[1,*,†], Georg Buss[2,‡] and Daniel Veit[3,§]

[1]*School of Science and Technology, Nottingham Trent University, Nottingham, UK*
[2]*Business School, University of Mannheim, Mannheim, Germany*
[3]*Faculty of Business and Economics, University of Augsburg, Germany*

SUMMARY

An increasing number of enterprise applications are intensive in their consumption of IT but are infrequently used. Consequently, either organizations host an oversized IT infrastructure or they are incapable of realizing the benefits of new applications. A solution to the challenge is provided by the large-scale computing infrastructures of clouds and grids, which allow resources to be shared. A major challenge is the development of mechanisms that allow efficient sharing of IT resources. Market mechanisms are promising, but there is a lack of research in scalable market mechanisms. We extend the multi-attribute combinatorial exchange mechanism with greedy heuristics to address the scalability challenge. The evaluation shows a trade-off between efficiency and scalability. There is no statistical evidence for an influence on the incentive properties of the market mechanism. This is an encouraging result as theory predicts heuristics to ruin the mechanism's incentive properties.

Received 13 May 2014; Revised 3 July 2015; Accepted 3 September 2015

KEY WORDS: market mechanisms; heuristics; large-scale computing

1. INTRODUCTION

IT, as an enabler of information systems, plays a key role in adding value to the activities of organizations. It is fundamental in supporting operations, management, and research and provides the foundation for applications that enable conducting business and research in new ways [1,2]. Sophisticated applications are often intensive in the consumption of IT resources (e.g., business analytics [3]). From the perspective of a single organization, the demand for such applications is variable, meaning that either the full IT capacity is not always used or peaks in demand cannot be served. Studies covering peaks in demand require maintaining an IT infrastructure operated at an average capacity utilization of 5 to 35 percent [4–6].

The need for cost efficiency while maintaining reliability, flexibility and scalability of the computing infrastructure is reflected in large-scale computing paradigms like grid computing [7] and cloud computing [4,8]. Both paradigms share the vision of providing an environment that enables the dynamic allocation and aggregation of distributed IT resources, to execute computing intensive applications [9]. The paradigm shift from local IT resources to remotely operated IT resources allows for the outsourcing of parts of an organization's IT infrastructure or to offer idle IT resources to others.

The set of IT services offered in large-scale computing infrastructures differs highly in the level of abstraction [4,10,11]. The range is from low-level infrastructure services looking much like physical hardware to specific high-level application-oriented services such as application domain-specific

*Correspondence to: Kevin Lee, School of Science and Technology, Nottingham Trent University, Nottingham, UK.
†E-mail: kevin.lee@ntu.ac.uk ‡Present address: Munich, Germany
§ veit@wiwi.uni-augsburg.de

platforms or specialized data services. The focus on infrastructure services that provide access to computing resources like CPU, memory, storage, other hardware, combinations of hardware, or aggregated resources. The spectrum of consumers is very board as each type of application is executed on a set of computing resources. The services can be described by a standardized vocabulary, which facilitates an automated matching of supply and demand. The resource consumers are assumed to be capable of estimating the resource requirements for specific tasks.

It is a major challenge to develop mechanisms that allow for the efficient sharing of resources. In particular, modern large-scale infrastructures such as grid computing and federated cloud computing infrastructures consist of many simultaneous resource consumers and providers. One of the defining characteristics of these modern infrastructures is distributed ownership, making traditional centralized scheduling and allocation approaches not applicable; for example, it is not possible to use a round-robin allocation scheme when you do not control all the resources. In particular, hybrid clouds introduce additional complexity to scheduling due to the ability to use multiple resource providers [12,13].

Efficient allocation of resources for shared ownership infrastructures is becoming vital to their usability. Price-based market mechanisms are considered a promising approach to this [14–16]. Allocations computed from prices are comprehensible for resource consumers, help shifting demand from high load periods to low load periods, and incentivize resource providers to contribute computing resources due to demand.

This paper focuses on the application of heuristic optimization methods for the efficient sharing of resources using market mechanisms. For many cases of NP-hard problems, it may be possible to have algorithms that produce reasonably fast close-to-optimal solutions [17,18]. The impact on the economic requirements will be analyzed in detail. The paper is organized as follows. Section 2 presents a background on market mechanisms for resource exchange. Section 3 formally defines the problem of multi-attribute combinatorial exchange. Section 4 proposes a heuristic-based approach to the resource allocation problem. Section 5 evaluates the heuristics from a theoretical and an empirical perspective. Section 6 concludes the paper with a summary of the major results.

2. BACKGROUND

Large-scale computing infrastructures include grid [7] and cloud computing [7]. Grids are federated cluster services generally consisting of resource cluster pools in different geographical locations managed by large governmental and educational organizations. Cloud computing is an extension of this into the commercial world with subscription-based resource provision. Cloud computing provides abstracted resources, elastic resource capacity, and programmable self-service interface, all using a pay-per-use pricing model.

Provision large-scale computing resources were previously restricted to a small number of providers. The emergence and rapid growth of private and hybrid clouds have dramatically increased the number of large-scale resource providers and consumers. This has been enabled through the availability of cloud computing toolkits such as Eucalyptus [19], Nimbus [20], and OpenNebula [21].

Accessing and provisioning resources in this heterogeneous environment are a challenge. Within the cloud, provisioning is achieved by specialist schedulers such as in OpenStack [22] and Haizea in OpenNebula [23]. Traditional scheduling techniques have been used for grid scheduling, focusing on time and cost trade-offs [24]; however, these are not suitable for clouds. In particular, they are not suitable where hybrid clouds enable the utilization of resources from many providers [25]. Although the cloud gives the illusion of unlimited resources, it is about more than access to infinite resources; there are also aspects of cost, programmability and flexibility, which need to be considered when deciding whether to consume resources. The rapid increase in available large-scale resources has increased the need for marketplaces to allow the trading of these resources. There is also the necessity for market mechanisms to support these marketplaces [26]. There are attempts to meet the market for on-demand and resource-flexible resources, most notably with Amazon spot-instances [27]; however, this is relatively isolated and is yet to spread into the wider cloud community.

The application of market mechanisms to large-scale computing infrastructures requires the consideration of economic and domain-specific requirements. From an economic perspective, an ideal

market mechanism should implement resource allocation efficiency, individual rationality, incentive compatibility, and budget balancing [17,28]. For practical application, computational tractability is a key concern. [29] specifies that a suitable mechanism must be double-sided to provide resource consumers and resource providers with the ability to trade actively and it should allow for the trading of multiple, mutually exclusive bids on bundles of computing resources. Bundle bids remove the exposure risk [30]. Resource-intensive applications may require the aggregation of the capacities of several resource providers (co-allocation) with constraints specifying the parameters of co-allocation and resource coupling. Usually, resource consumers have to meet deadlines but are flexible in the consumption of the resources within a certain time frame.

This problem can be framed as a combinatorial auction or a combinatorial exchange problem. [31] introduced a time-phased combinatorial auction format with double-sided fair allocations (both exact and heuristic-based). [29] fits the multi-attribute combinatorial exchange to the domain. In a combinatorial auction, a *reserve price* is used to ensure a minimal price for resources, an *overall surplus* is calculated to optimize the auction, *efficiency loss* is when the resources are undervalued due to equilibrium, and the notion of *welfare* is used to ensure a minimal level of resource provision.

The combinatorial auction mechanism is not scalable in the number of resource consumers and providers. GreedEX is designed as a scalable exchange-based mechanism [32,33], which does not allow for co-allocation and the trading resources other than a combination of CPU and storage. Mossmann proposes a combinatorial exchange mechanism that is specialized to auctioning computing resources for small-size workflows [34]. Co-allocation is not considered, and the mechanism is not scalable. Because multi-attribute combinatorial exchange fits the domain-specific and economic requirements, the best heuristics are considered to induce scalability to the mechanism. Figure 1 compares these approaches in relation to common economic requirements. Figure 2 compares these approaches with respect to combinatorial exchange domain-specific requirements.

Approach	AE	BB	IR	IC	CT
Fair Allocation (Exact)	•	✓	✓	✓	•
Fair Allocation (Heuristic)	•	✓	✓	•	✓
Multi-Attribute Combinatorial Exchange	•	✓	✓	•	•
GreedEx (Exact)	•	✓	✓	•	•
GreedEx (Heuristic)	•	✓	✓	≈	✓
Mossmann	•	✓	✓	•	•

✓: fulfilled, •: not fulfilled , ≈ : approximately fulfilled, ○ : not evaluated

Figure 1. Comparison between combinatorial mechanism approaches and economic requirements. AE, allocative efficiency; BB, budget balance; IR, individual rationality; IC, incentive compatibility; CT, computational tractability.

					Time				Co-Alloc.		
Approach	Mechanism	Trading Object	Double Sided	Quality Attributes	Amount	Frame	Bundles	Multiple Bundles	Enabled	Split	Coupling
Fair Allocation (Exact)	Combinatorial Auction	CPU, Storage	•	single	✓	•	✓	•	✓	•	•
Fair Alloc. (Heuristic)	Combinatorial Auction	CPU, Storage	•	single	✓	•	✓	•	✓	•	•
Multi-Attribute Combinatorial Exchange	Combinatorial Exchange	arbitrary	✓	multiple	✓	✓	✓	✓	✓	✓	✓
GreedEx (Exact)	Combinatorial Exchange	CPU, Storage	✓	single	✓	•	✓	•	•	•	•
GreedEx (Heuristic)	Combinatorial Exchange	CPU, Storage	✓	single	✓	•	✓	•	•	•	•
Mossman	Combinatorial Exchange	arbitrary	✓	multiple	✓	✓	✓	✓	•	•	•

✓: fulfilled, •: not fulfilled , ○: not evaluated

Figure 2. Comparison of combinatorial mechanisms with domain-specific requirements. ✓, fulfilled; •, not fulfilled; ○, not evaluated.

Solving the multi-attribute combinatorial exchange winner determination problem (MWDP) to optimality is shown to be an NP-hard problem by [29]. There have been attempts to mitigate computational intractability, by (i) the reduction of the complexity of the bidding language [35], (ii) the design of an indirect revelation mechanism [36], and (iii) the utilization of heuristics to solve the winner determination problem [37].

3. MULTI-ATTRIBUTE COMBINATORIAL EXCHANGE

The bidding language specifies the syntax and the semantics of the bids submitted to multi-attribute combinatorial exchange. The notations follow the work of [29]. The notation is specified in Figure 3 and described in the succeeding text.

The set $G = \{g_1, \ldots, g_{|G|}\}$ specifies all the computing resources $g_k \in G$ available. A bundle S_i denotes a subset of all the resources in G. The set $S = \{S_1, \ldots, S_{|S|}\}$ of bundles covers all the possible subsets $S_i \subseteq G$. A computing resource $g_k \in G$ itself is defined by a set of $|A_k|$ cardinal quality attributes $A_{g_k} = \{a_{g_k,1} \ldots, a_{g_k,|A_k|}\}$ where a_{g_kj} denotes the jth attribute of resource g_k. The set of time slots $T = \{0, \ldots, |T|\}$ determines the range of discrete time slots the respective bundles S_i are to be traded in. $t \in T$ denotes a single time slot out of the set T.

A resource consumer n out of the set $N = \{1, \ldots, |N|\}$ of resource consumers is allowed to submit an order of multiple bundle bids $B_n = \{B_n(S_i) \oplus \ldots \oplus B_n(S_j)\}$. The respective bundle bids are Exclusive OR (XOR) concatenated.

$$B_n(S_i) = \Big\{ \big\langle v_n\left(S_i\right), s_n\left(S_i\right), e_n\left(S_i\right), l_n\left(S_i\right), q_n\left(S_i, g_1, a_{g_1,1}\right), \ldots, q_n\left(S_i, g_{|G|}, a_{g_G,|A_k|}\right), \\ \gamma_n\left(S_i, g_1\right), \ldots, \gamma_n\left(S_i, g_{|G|}\right), \varphi_n\left(S_i, g_1, g_2\right), \ldots, \varphi_n\left(S_i, g_{|G|}, g_{|G|-1}\right) \big\rangle \Big\}$$

The valuation $v_n(S_i)$ specifies the maximum amount the resource consumer n is willing to pay for each time slot she is allocated the bundle S_i. The number of slots the resources are required for is given by $s_n(S_i)$ where $s_n(S_i) <= |T| + 1$. A resource consumer bid defines a period of time slots within which the required slots have to be allocated. The period is given by $e_n(S_i) \in T$ for the earliest possible time slot and $l_n(S_i) \in T$ for the latest possible time slot. The minimum quality restrictions for each of the attributes a_{g_kj} characterizing the resources g_k contained in a bundle bid S_i are given by $q_n(S_i, g_k, a_{g_kj}) \geqslant 0$. In addition bundle bids may contain two types of fulfillment constraints. (i) A co-allocation constraint $\gamma_n(S_i, g_k) > 0$ that specifies the maximum number of resource provider bundle bids allowed to allocate a required resource g_k. (i) The coupling constraint $\varphi_n(S_i, g_k, g_j)$ requires a pair of resources g_k, g_j to be allocated from the same resource provider bid. The constraint equals one ($\varphi_n(S_i, g_k, g_j) = 1$) in case the computing resources have to be allocated from the same resource provider bid and equals zero ($\varphi_n(S_i, g_k, g_j) = 0$) otherwise.

A potential resource provider m out of the set $M = \{1, \ldots, |M|\}$ of resource providers may submit an order of multiple bundle bids $B_m = \{B_m(S_i) \vee \ldots \vee B_m(S_j)\}$. A resource provider bundle bid is considered to offer computing resources that are located on the same machine. The bundle bids are of a similar form as resource consumer bundle bids but OR concatenated. Any number of

Variable	Description
g	A computing resource
G	All computing resources available
a	Attributes of computing resources
A	The set of all attributes for a given resource g
S_i	A bundle denoting a subset of all resources G
S	The set of all bundles
t	A single time slot
T	The set of all time slots
n	A resource consumer
N	The set of all resource consumers
B	A set of bundle bids
v	Valuation for a resource bundle S

Figure 3. The bidding language notation.

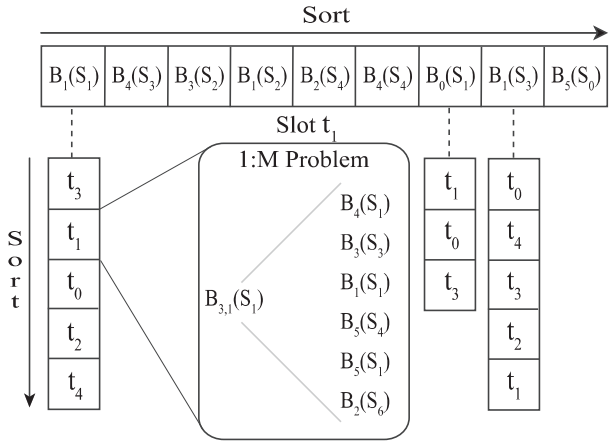


Figure 4. Problem representation of the multi-attribute combinatorial exchange winner determination problem.

resource provider bids may be part of the final allocation. A single resource provider bundle bid does not specify co-allocation and coupling constraints. Instead of a valuation, a reservation price $r_m(S_i)$ is specified. The maximum quality of a computing resource g_k is given for each attribute by $q_m(S_i, g_k, a_{g_k j}) \geqslant 0$. The MWDP, which determines the allocation of resource consumer to resource provider bids, is detailed in [29]. Given a collection of resource consumers' and resource providers' bundle orders the MWDP is to identify a set of winning bids out of the total set of bids. An optimal set of winning resource consumer and resource provider bids determines an allocation that maximizes the overall surplus. Free disposal of computing resources is if co-allocation or coupling constraints are not present.

An instance of the MWDP is represented as depicted in Figure 4. The approach is to split up the $|N| : |M|$ allocation problem subject to $|T|$ time slots to several $1 : |M|$ allocation problems to be solved for a single time slot t. The partial results are aggregated to an overall allocation. More specifically, the resource consumer orders are split into the single bundle bids $B_n(S_i)$. The single resource consumer bundle bids $\{B_1(S_1), \ldots, B_n(S_i)\}$ are stored in a sortable set RCB. For each resource consumer bid $B_n(S_i)$, a sortable list, $Slot(B_n(S_i))$, of potential time slots t is kept. The group of potential time slots is bounded by $e_n(S_i)$ and $l_n(S_i)$. For each of these time slots, the available resource provider bundle bids $B_m(S_i)$ are stored in the set $RPB_t(S_i)$. Formally, the problem representation R is constructed from the sets RCB, $Slot(B_n(S_i))$ and $RPB_t(S_i)$.

4. HEURISTIC SOLUTION TO RESOURCE ALLOCATION

A solution to the multi-attribute combinatorial exchange as defined in Section 3 using a greedy type of heuristic is proposed in Algorithm 1. The resource consumer bundle bids are processed in the order that is given by RCB. The decision whether to include a single resource consumer bundle bid in an allocation requires an additional passing of the respective set of time slots $Slot(B_n(S_i))$. The time slots the resource consumer bundle bid is valid for are checked in the given order. A check of a time slot requires solving the $1 : |M|$ allocation problem. A check of a time slot is valid only if there is a valid solution to the $1 : |M|$ allocation problem. As soon as the amount of computing resources requested is available for a sufficient number of time slots, the resource consumer bundle bid and the fractions of the respective resource provider bundle bids are included into the allocation. Therefore, problem representation does not encode any infeasible allocation. The construction process is continued with the subsequent resource consumer bundle bid. The evaluation of a resource consumer bundle bid is skipped if the respective resource consumer is already part of the allocation with a bundle bid previously checked. The ordering of the resource consumer bundle bids influences the chance of being part of an allocation. The earlier a bundle bid is evaluated, the less is the risk that potential resource provider capacities are already taken.

Algorithm 1 The proposed greedy heuristic-based combinatorial exchange algorithm

```
s ← 0                \\welfare allocation
a ← ∅                \\allocation
RCB                  \\set of resource consumer bundle bids
RPB_t(S_i)           \\set of resource provider bundle bids available in time slot t for the bundle S_i
Slot(B_n(S_i))       \\set of slots for a specific resource consumer bundle bid
w_1                  \\ranking criterion resource consumer bundle bids
w_2                  \\ranking criterion for time slots

\\iterate over all resource consumer bids
while hasNext(RCB) do
   B_n(S_i) ←next(RCB)

    \\test if the resource consumer n is part of the allocation
   if n ∈ a then
      continue with the next iteration
   end if

   s' ← 0
   a' ← ∅
   s_n(S_i) ← getNumberRequiredSlots(B_n(S_i))

   \\iterate over the slots
   while hasNext(Slot(B_n(S_i))) do
      t ← next(Slot(B_n(S_i)))
      a'' ← ∅
      a'' ←solve(B_n(S_i), RPB_t(S_i)) \\solve the 1 : |M| allocation problem
      if isFeasilbe(a'') then
         a' ∪ a''            \\add the surplus to surplus of the allocation
         s_n(S_i) ← s_n(S_i) − 1      \\save that a slot was successfully allocated
         s' ← s' + surplus(a')
      end if
      \\test if the allocation for B_n(S_i) is completed
      if s_n(S_i) == 0 then
         a ∪ +a'
         s ← s + s'
         continue with the next iteration
      end if
   end while
end while
```

In Figure 4, $B_3(S_1)$ takes the leading position in *RCB* the evaluation process is started with. The time slots, the resource consumer bundle bid is valid for, are given by $e_n(S_1) = 0$ and $l_n(S_1) = 4$. The time slots are checked in the order: 3, 1, 0, 2, 4. Once the resources are allocated for sufficient number time slots, the allocation process is stopped. $B_{3,1}$ and the fractions of the respective resource provider bids are added to the allocation. $B_4(S_3)$ is the next bid to be evaluated followed by $B_3(S_2)$. In case $B_3(S_1)$ is part of the allocation, $B_3(S_2)$ is not subject to any further evaluation (XOR constraint). The procedure is continued to the end of the sequence of resource consumer bundle bids.

Resource consumer bundle bids are sorted prior to allocating resource consumers to resource providers. The time slots are sorted at the time a bundle bid is evaluated. The $1 : |M|$ allocation problem equals an instance of the MWDP with a single resource consumer n, submitting a single bundle bid that is valid for a single time slot t. Furthermore, the resource providers offer their resources for the same single time slot to this specific resource consumer only. Formally multi-attribute combinatorial exchange is reduced to

$$\max v_n(S_i) - \sum_{m \in M} \sum_{S_i \in S} r_m(S_i) y_m \tag{1}$$

Additionally, the respective constraints are reduced to

$$\text{s.t.} \sum_{S_i \ni g_k} q_n\left(S_i, g_k, a_{g_k,j}\right) - \sum_{S_i \ni g_k} \sum_{m \in M} y_m\left(S_i\right) q_m\left(S_i, g_k, a_{g_k,j}\right) \leqslant 0 \quad \forall g_k \in G, \forall a_{g_k,j} \in A_{g_k} \tag{2}$$

$$\sum_{S_i \ni g_k} \sum_{m \in M} d_m(S_i) - \sum_{S_i \ni g_k} \gamma\left(S_i, g_k\right) \leqslant 0, \qquad \forall g_k \in G \tag{3}$$

$$\sum_{S_j \ni g_k, g_l} \varphi_n\left(S_j, g_k, g_l\right) \left(\sum_{S_i \ni g_k} d_m(S_i) - \sum_{S_i \ni g_l} d_m(S_i) \right) = 0 \qquad \forall m \in M, \forall g_k, g_l \in G \tag{4}$$

$$\sum_{S_j \ni g_k, g_l} \varphi_n\left(S_j, g_k, g_l\right) \left(\sum_{S_i \ni g_k} \sum_{m \in M} d_m(S_i) + \sum_{S_i \ni g_l} \sum_{m \in M} d_m(S_i) - 2 \right) \leqslant 0, \qquad \forall g_k, g_l \in G \tag{5}$$

An allocation is described by the variables $y_m(S_i) \in [0,1]$ and $d_m(S_i) \in \{0,1\}$. The real-valued variable $y_m(S_i)$ denotes the percentage of bundle S_i allocated from resource provider m to resource consumer n for a given time slot. In the context of the variable $y_m(S_i)$, the binary variable $d_m(S_i) = 1$ if the bundle S_i is allocated from resource provider m to resource consumer n and $d_m(S_i) = 0$ otherwise. The variable bound $c_m(S_i)$ allows for the specification of resource provider capacities, $y_m(S_i)$, equal or smaller than one. The objective function, which is defined in (1), is designed to maximize the surplus of an allocation. To model the domain-specific requirements, the objective function is subject to the constraints (2)–(5). Constraint (2) controls for the fulfillment of the capacity requirements of an allocated resource consumer bundle S_i by the respective resource provider's bundles. Constraint (3) guarantees that the limit $\gamma_n(S_i, g_k)$ for aggregating a specific computing resource g_k from different resource provider bundles is met. The constraints (4) and (5) implement the coupling functionality. Constraint (6) limits the capacity of a provider bundle bid to $c_m(S_i)$. The constraints (8) and (9) model the relation between the variables $y_m(S_i)$ and $d_m(S_i)$. For a specific resource provider m and a bundle S_i, the respective variable $d_m(S_i)$ is greater than zero only if $y_m(S_i)$ is greater than zero.

$$y_m(S_i) \leqslant c_m(S_i), \ \forall m \in M, \forall S_i \in S \tag{6}$$

$$y_m(S_i) - d_m(S_i) \leqslant 0 \tag{7}$$

$$d_m(S_i) - y_m(S_i) < 1 \tag{8}$$

$$y_m \geqslant 0 \tag{9}$$

$$d_m(S_i) \in 0, 1 \tag{10}$$

In case neither coupling nor collocation constraints exist (3)–(5), the problem reduces from a combinatorial to a linear, continuous, optimization problem, which is solved efficiently by a linear programming solver. In comparison to the original $|N| : |M|$ allocation problem, the complexity is reduced significantly. In case of constraints, the $1 : |M|$ allocation problem is an NP-hard optimization problem. The set partitioning problem is reducible to the $1 : |M|$ allocation problem [38]. Consider a problem instance with a single resource consumer bid that is valued with zero. Each of the resources of the bundle is described by a single quality attribute. Each of these resources is restricted with respect to co-allocations to 1. The resource provider bundle bids are valid for a subset of the resources requested and offer exactly the level of quality demanded. This scenario is equal to solving the set partitioning problem.

Prior to introducing the ordering procedures, several concepts have to be introduced. The factor

$$flex(B_n(S_i)) = \frac{s_n(S_i)}{l_n(S_i) - e_n(S_i) + 1}$$

measures the flexibility of a resource consumer bundle bid in a time scheduling sense. The parameter

$$acon(S_i) = \sum_{S_i \ni g_k} \max_{a_{g_k j} \in A_j} q_n\left(S_i, g_k, a_{g_k,i}\right)$$

measures the average consumption of computing resources for a given bundle bid S_i by aggregating the maximum quality requirements for each resource. The parameter

$$wac(S_i) = \frac{\sum_{S_i \ni g_k} \sum_{t=e_n(S_i)}^{l_n(S_i)} \max_{a_{g_k j} \in A_j} \frac{q_n\left(S_i, g_k, a_{g_k j}\right)}{\sum_{RPB_t(S_i) \ni S_j} q_m\left(S_j, g_k, a_{g_k j}\right)}}{l_n(S_i) - e_n(S_i) + 1}$$

measures the weighted average consumption of resources per time slot in dependency of the amount of resources offered.

The attractiveness of a resource consumer bundle bid can be assessed by the descending order according to the following criteria:

1 $w_1(B_n(S_i)) = v_n(S_i) * (flex(Si))^{0.5}$:
The weight is determined by the bundle valuation scaled by the parameter *flex*. Highly flexible bundle bids are more likely to fit into an allocation while some computing resources are taken.
2 $w_2(B_n(S_i)) = \frac{v_n(S_i) * s_n(S_i) * (flex(S_i))^{0.5}}{acon(S_i)}$:
The valuation is scaled by the flexibility and the number of slots the respective computing resources are requested for. It is adjusted by the average demand for computing resources.
3 $w_3(B_n(S_i), RPB_t(S_i)) = \frac{v_n(S_i) * s_n(S_i) * (flex(S_i))^{0.5}}{wac(S_i)}$:
The weight of resource consumer bundle bids is based on resource scarcity with the requested amount of resources weighted by the available supplies.

The options for sorting the slots of a resource consumer bundle bid are as follows:

1 $w_a(B_n(S_i), RPB_t(S_i)) = \frac{\sum_{S_j \ni RPB_t(S_i)} \frac{r_m(S_j)}{|g_k \in S_j|}}{|RPB_t(S_i)|}$:
The time slots are weighted according to the ascending average resource provider reservation price for a single computing resource.
2 $w_b(B_n(S_i), RPB_t(S_i)) = w_{a1} * \sum_{S_i \ni g_k} \max_{a_{g_k j} \in A_j} \frac{q_n(S_i, g_k, a_{g_k j})}{\sum_{RPB_t(S_i) \ni S_j} q_m(S_i, g_k, a_{g_k j})}$:
The time slots are sorted according to the ascending average resource provider reservation price for a single resource weighted by the aggregated average resource demand versus supply ratios. The demand versus supply ratio for a single resource is defined as the sum of maximum ratios between demand quality and the aggregated supplies.
3 $w_c(B_n(S_i), RPB_t(S_i))$ = optimal allocation:
This w weights the time slots according to the result of the optimal solution to the $1 : |M|$ allocation problem in descending order.

The combinations $(w_1, w_a), (w_2, w_b)$, and (w_3, w_c) are chosen for evaluation. The combinations reflect the trade-off of the complexity of evaluation versus the power of additional information included into the evaluation process ((w_1, w_a) is labeled GreedyLow, (w_2, w_b) is entitled GreedyMedium, and (w_3, w_c) is named GreedyComplex).

5. EVALUATION

5.1. Illustrative example

The representation may exclude feasible allocations from the search space, which is illustrated by two examples. The first example is composed of two equal bundle bids submitted by two independent

resource consumers (Figure 5. The offer of the resource providers matches the resource consumers' needs (Figure 6. The welfare maximizing allocation involves a sharing of the computing resources. The welfare is $2 + 2 - 1 - 1.1 = 1.9$. The heuristic optimization methods allocate the resources offered by resource provider 1 completely to the resource consumer 1. This is the welfare maximizing decision considering a single resource consumer bid. The second resource consumer is not part of the allocation as the remaining capacities are not sufficient to serve their needs. The welfare drops from 1.9 to $2 - 1 = 1$.

The second example is made up from a single resource consumer bid and two resource provider bids (Figures 7 and 8. Each of the resource provider bids covers the need of the resource consumer for a single but different time slot. The welfare maximizing allocation yields a welfare of $2 * 1 - 1.1 - 0.5 = 0.4$. In contrast, the heuristics check the two time slots according to a predefined order. For the first time slot, an allocation is determined that yields a welfare of $1-0.5 = 0.5$. For the second time slot, no allocation is provided because of a negative welfare of $1 - 1.1 = -0.1$. In consequence, the heuristic optimization method does not provide an allocation. The welfare drops by 0.5.

Instances of the MWDP exist where feasible allocations are not included in the search space of the heuristic optimization methods. The effect has to be determined empirically considering realistic problem instances.

The asymptotic running time of an algorithm is given in dependence of the size of the input to the algorithm. In theory, the size of the input to the MWDP depends on the number of resource consumers $|N|$, the number of resource providers $|M|$, the number of time slots $|T|$, the number of computing resources $|G|$, and the number of attributes $|A|$. However, the number of time slots, the number of computing resources, and the number of attributes are fixed numbers considering real world problem instances. The sorting of the resource consumer and provider bids runs in polynomial time in case the $1 : |M|$ problem is not solved optimality. Solving the $1 : |M|$ problem to its optimality is subject to polynomial time consumption if no constraints on the resource consumer bid are present, but it is subject to exponential time consumption otherwise. The exact asymptotic behavior is not available for the commercial mixed-integer linear programming (MILP) solver utilized for solving the $1 : |M|$ problem to optimality. The actual impact on scalability is evaluated by means of an empirical analysis.

5.2. Experimental evaluation

This section presents computational experiments to solve the MWDP problem to optimality using CPLEX 12.1 with a single CPU.

N	S_j	$v_n(S_j)$	$q_n(S_j, g_k, a_i^k)$	$e_n(S_j)$	$l_n(S_j)$	$s_n(S_j)$
1	S_1	2	$g_1, a_1^1 = 1; g_2, a_1^1 = 2$	1	1	1
2	S_1	2	$g_1, a_1^1 = 1; g_2, a_1^1 = 2$	1	1	1

Figure 5. Suboptimal allocation of computing resources: resource consumer bids.

M	S_j	$r_n(S_j)$	$q_n(S_j, g_k, a_i^k)$	$e_n(S_j)$	$l_n(S_j)$
1	S_1	1	$g_1, a_1^1 = 2; g_2, a_1^1 = 2$	1	1
2	S_1	1.1	$g_1, a_1^1 = 0; g_2, a_1^1 = 2$	1	1

Figure 6. Suboptimal allocation of computing resources: resource provider bids.

N	S_j	$v_n(S_j)$	$q_n^N(S_j, g_k, a_i^k)$	$e_n^N(S_j)$	$l_n^N(S_j)$	$s_n(S_j)$
1	S_1	1	$g_1, a_1^1 = 1$	1	2	2

Figure 7. Suboptimal allocation of computing resources: resource consumer bid.

M	S_j	$r_n(S_j)$	$q_n^N(S_j, g_k, a_i^k)$	$e_n^N(S_j)$	$l_n^N(S_j)$
1	S_1	0.5	$g_1, a_1^1 = 1$	1	1
2	S_1	1.1	$g_1, a_1^1 = 1$	2	2

Figure 8. Suboptimal allocation of computing resources: resource provider bids.

Inst.	Res.	Attr.	Time	Constraints	XOR
I_1	5	3	[0, 7],[1, 4]	none	none
I_2	5	3	[0, 7],[1, 4]	none	two
I_3	5	3	[0, 15],[1, 6]	none	none
I_4	5	3	[0, 7], [1, 4]	Coup.: 90%, 2 Split: 90%, 80%, [1, 3]	none

Figure 9. Parameters for the generation of different types of problem instances.

The generation of meaningful artificial problem instances for combinatorial exchange mechanisms requires a realistic, consistent, and economically motivated modeling of the bidding behavior of the market participants [39,40]. To generate economically motivated bundle bids, the arbitrary mode of the Combinatorial Auction Test Suite [39,41] is used. Having created a number of resource bundles and the respective valuations (reservation prices), the characteristics of the attributes have to be defined. An economic motivation requires the valuation for a bundle bid to be by trend positively correlated to the attribute requirements. This is modeled drawing the attribute values from a normal distribution $N(\mu, \sigma^2)$ where μ and σ^2 depend on the valuation and the number of goods requested [42]: $\mu = x \cdot v_i(S_j) \cdot (y + 0.5)$ and $\sigma^2 = z \cdot v_i(S_j) \cdot G_{S_j}$. The parameters x and z are scaling factors set to $x = 3$ and $z = 0.15$. The parameter $y \in [0, 1]$ is picked from a uniform distribution and models differences in price estimates. The split constraints are drawn from a uniform distribution. The coupling constraints are set to a fixed number of resources. The number of time slots as well as the earliest and latest possible time slot for the allocation are drawn from a uniform distribution.

For the balanced problem instances, the number of resource consumers equals the number of resource providers. For the competitive ones, which model peaks in demand, the number of resource consumers exceed the number of resource providers threefold. The balanced-type and the competitive-type problem instances are both subcategorized into four categories. I_1 represents the base line problem instances, I_2 differs in the number of XOR bids submitted by the resource consumers, I_3 is varied in the number of time slots, and I_4 is characterized by constraints on the computing resources.

The parameters are summarized in Figure 9. For the base line scenario (I_1), the earliest and the latest slots are chosen out of the interval [0, 7]. The number of required slots is chosen from the interval [1, 4]. For I_2-type problem instances, each resource consumer is assumed to submit two mutually exclusive bundle bids. For the instances of type I_3, interval for the earliest and latest is enlarged. In I_4-type problem instances, constraints are present. Ninety percent of the resource consumer bids are subject to coupling and split constraints. In case of coupling constraints, two resources are coupled. In case of split constraints, 80 percent of the resources of the bundle are constrained, which allows a maximum split in the interval of [1, 3].

For the investigation of allocative efficiency and scalability, each of the four subcategories of competitive and balanced instance types is differentiated further into six categories. Thirty problem instance are generated per category. The investigation of strategic behavior is based on competitive and balanced I_1 problem instances with a single resource consumer or provider manipulating its valuation or reservation price. The instances are made up from 20 resource consumers and 20 resource providers for the balanced problem instance types. The evaluation is based on small-scale instances because theory implicates the effect of manipulation to be strongest for these instances [43,44]. Fifty problem instances are generated per scenario. Each of the optimization methods is given a 3-min time frame to solve any problem instance.

Instance	I_1	I_2	I_3	I_4
	0/1.5/>1.5	0/1.5/>1.5	0/1.5/>1.5	0/1.5/>1.5
		balanced		
10/10	30/0/0	30/0/0	30/0/0	30/0/0
30/30	30/0/0	29/1/0	30/0/0	30/0/0
50/50	30/0/0	3/27/0	30/0/0	30/0/0
70/70	30/0/0	1/29/0	23/7/0	22/8/0
90/90	29/1/0	0/29/1	9/21/0	18/12/0
110/110	25/5/0	0/29/1	3/27/0	0/28/2
		competitive		
30/10	30/0/0	30/0/0	30/0/0	30/0/0
90/30	30/0/0	0/30/0	22/8/0	28/2/0
150/50	23/7/0	0/26/4	2/28/0	4/26/0
210/70	10/20/0	0/4/26	0/30/0	3/23/4
270/90	0/29/1	0/0/30	0/29/1	0/3/27
330/110	0/29/2	0/0/30	0/29/1	0/1/29

Figure 10. Quality of the allocations provided by the exact optimization method for the balanced and competitive problem instances.

5.2.1. Results of allocative efficiency. Test runs show that the welfare maximizing allocation is computable within a reasonable time frame for small-scale problem instances only. A limitation of the study to these small-scale problem instances would introduce bias to the evaluation. There is a tendency to underestimate the performance of the heuristics [40]. To resolve the challenge, the evaluation is conducted according to the efficiency loss, which is incurred in comparison to the welfare of the best known allocation. The best known allocation is to be determined by any of the optimization methods.

If the exact optimization method does not provide the optimal allocation within the reasonable time frame of 3 min, the solution process is interrupted and the current best allocation is noted. In addition, the upper bound on the welfare of the optimal allocation is logged. In case a nonoptimal allocation is identified, its quality can be judged by the respective upper bound. Prior tests provide evidence for a very close match of the welfare maximizing allocation if the upper bound deviates by less than 1.5 percent from the welfare of the identified allocation. Therefore, those allocations serve as a meaningful benchmark for the quality of the heuristic optimization methods.

In Figure 10, the results for the performance of the exact optimization method with respect to balanced and competitive problem instances are given, detailing the number of problem instances solved to optimality, the number of problem instances solved with a maximum deviation of 1.5 percent to the optimum quality, and the number of problem instances for which no meaningful bound on the quality of the allocation is provided are reported. Starting from a size of 150 (210) resource consumer bids, I_2 (I_4) competitive-type problem instances exist, and no allocation is determined that fits the bound of 1.5 percent. For many of these problem instances, not even a feasible allocation is computable within the 3-min time frame.

To fully judge the approximation of the welfare maximizing allocation by the greedy-type optimization methods, a lower bound in terms of a randomized search procedure, GreedyRandom, is introduced. GreedyRandom operates the same way as the greedy approaches presented but the sorting processes are randomized. A comparison shows whether the greedy-type optimization methods perform superior to simply guessing an allocation.

Figures 11 and 12 present an overview on the mean values of the efficiency loss. The results provide evidence that a consistent ordering of the optimization methods according to the mean allocative efficiency loss is possible for all types of problem instances: GreedyComplex outperforms GreedyMedium, which outperforms GreedyLow. The ordering is confirmed at a significance level of $p < 0.01$ with a Wilcoxon rank-sum test applied to each subcategory. The difference in solution quality between the optimization methods depends on the type of problem instance investigated. The range of the results for the constrained I_4-type problem instances is observed to be considerably less diverse. Constraints minimize the set of feasible allocations and direct the different greedy optimiza-

tion methods to similar allocations. The choice of the optimization method has a larger effect for unconstrained problem instances.

To assess the efficiency loss due to the heuristic optimization methods, the results are compared with the allocations provided by the exact approach. To prevent the falsification of the results, the focus is on problem instance types for which a majority of the problem instances is solved with an efficiency loss smaller than 1.5 percent. The comparison is limited to the balanced problem instance types and the competitive I_1 and I_3 problem instance types. GreedyComplex matches the reference values computed by the exact optimization method best. A comparable drop of the allocation quality between the balanced and competitive problem instances is observed for all greedy-type optimization methods. The greedy optimization methods show only for a single small-scale problem instance the worst case behavior of identifying not any feasible allocation although one exists. On an overall average, GreedyComplex, GreedyMedium, and GreedyLow approximate the allocation and provided the exact optimization method with an efficiency loss of 13, 17, and 25 percent.

An evaluation of GreedyComplex, GreedyMedium, and GreedyLow with respect to the minimum quality benchmark GreedyRandom indicates that GreedyComplex and GreedyMedium provide a measurable improvement to GreedyRandom while GreedyLow performs comparable or even worse as GreedyRandom. Further statistical evidence for the ranking of GreedyComplex, GreedyMedium, and GreedyLow is provided by comparing the optimization methods for each of the problem instances with GreedyRandom applying the Wilcoxon rank-sum test (Figures 11 and 12. GreedyComplex is found to significantly outperform GreedyLow for 42 (88 percent) of the problem instances categories at a significance level of $p < 0.01$ and for an additional five (10 percent) of the problem instances categories at a significance level of $p < 0.05$. A strong outlier at a significance level of $p < 0.1$ is present for the balanced I_4 problem instance category with the minimum number of resource consumers (2 percent). This is reasoned by the small number of constrained resource consumer bids that considerably restricts the number of feasible allocations. The probability that a

	Instance	Exact	GreedyComplex	GreedyMedium	GreedyLow	GreedyRandom
I_1	10/10	0.00	0.05***	0.06**	0.13	0.13
	30/30	0.00	0.09***	0.14***	0.22	0.23
	50/50	0.00	0.08***	0.13***	0.21**	0.18
	70/70	0.00	0.10***	0.13***	0.24*	0.20
	90/90	0.00	0.10***	0.14***	0.25**	0.21
	110/110	0.00	0.11***	0.15***	0.24	0.20
	Avg.	0.00	0.09	0.12	0.21	0.19
I_2	10/10	0.00	0.08***	0.12**	0.15	0.23
	30/30	0.00	0.17***	0.20***	0.33	0.36
	50/50	0.00	0.17***	0.20***	0.29**	0.33
	70/70	0.00	0.15***	0.18***	0.31	0.31
	90/90	0.04	0.10***	0.13***	0.25	0.26
	110/110	0.05	0.08***	0.12***	0.26	0.25
	Avg.	0.02	0.13	0.16	0.26	0.29
I_3	10/10	0.00	0.06***	0.09**	0.17	0.15
	30/30	0.00	0.09***	0.14**	0.22	0.19
	50/50	0.00	0.08***	0.13***	0.22**	0.17
	70/70	0.00	0.08***	0.14**	0.22***	0.17
	90/90	0.00	0.09***	0.15**	0.23***	0.17
	110/110	0.00	0.09***	0.15	0.22***	0.17
	Avg.	0.00	0.08	0.14	0.21	0.17
I_4	10/10	0.00	0.08*	0.08	0.14	0.13
	30/30	0.00	0.13**	0.15	0.14**	0.19
	50/50	0.00	0.14***	0.16*	0.18	0.21
	70/70	0.00	0.16***	0.18**	0.19	0.22
	90/90	0.00	0.16***	0.19*	0.19	0.22
	110/110	0.08	0.16**	0.17	0.15***	0.22
	Avg.	0.01	0.14	0.15	0.16	0.20

Figure 11. Mean efficiency loss for balanced problem instances. *Significance at level $p < 0.1$; **significance at level $p < 0.05$; ***significance at level $p < 0.01$ comparing the specific greedy-type method to GreedyLow with a Wilcoxon rank-sum test.

	Instance	Exact	GreedyComplex	GreedyMedium	GreedyLow	GreedyRandom
I_1	30/10	0.00	0.14**	0.17	0.16	0.22
	90/30	0.00	0.15***	0.20***	0.31	0.29
	150/50	0.00	0.21***	0.24***	0.37**	0.33
	210/70	0.00	0.22***	0.26***	0.40***	0.34
	270/90	0.03	0.20***	0.23***	0.38***	0.33
	330/110	0.07	0.21***	0.25***	0.41***	0.36
	Avg.	0.02	0.19	0.22	0.34	0.31
I_2	30/10	0.00	0.15***	0.18***	0.24*	0.31
	90/30	0.00	0.19***	0.23***	0.39	0.41
	150/50	0.15	0.11***	0.13***	0.31	0.31
	210/70	0.88	0.01***	0.04***	0.29***	0.24
	270/90	1.00	0.01***	0.03***	0.27	0.26
	330/110	1.00	0.01***	0.02***	0.30*	0.26
	Avg.	0.50	0.08	0.10	0.30	0.30
I_3	30/10	0.00	0.12***	0.15*	0.26	0.22
	90/30	0.00	0.17***	0.22***	0.32	0.29
	150/50	0.00	0.18***	0.23***	0.34***	0.29
	210/70	0.00	0.18***	0.24***	0.34*	0.31
	270/90	0.03	0.20***	0.24***	0.36**	0.31
	330/110	0.03	0.20***	0.24***	0.36***	0.30
	Avg.	0.01	0.17	0.22	0.33	0.29
I_4	30/10	0.00	0.12**	0.14	0.18	0.20
	90/30	0.00	0.20***	0.25**	0.20***	0.31
	150/50	0.00	0.22***	0.25*	0.25***	0.29
	210/70	0.13	0.21***	0.22**	0.22***	0.27
	270/90	0.90	0.05***	0.07***	0.07***	0.16
	330/110	0.97	0.03***	0.04***	0.06***	0.13
	Avg.	0.33	0.14	0.16	0.16	0.23

Figure 12. Mean efficiency loss for competitive problem instances. *Significance at level $p < 0.1$, **significance at level $p < 0.05$; ***significance at level $p < 0.01$ comparing the specific greedy-type method with GreedyLow with a Wilcoxon rank-sum test.

randomly drawn allocation is of high quality and does not considerably differ from an allocation provided by GreedyComplex is increased.

The improvement on the efficiency loss provided by GreedyMedium in comparison with GreedyRandom is significant at a level of $p < 0.01$ for 29 (60 percent) of the instance categories, at a level of $p < 0.05$ for nine (19 percent) of the instance categories, and at a level of $p < 0.1$ for four (8 percent) of the instance categories. It is not significant for the remaining six (13 percent) categories of problem instances. The results of GreedyMedium significantly differ from the results of GreedyRandom for balanced and competitive I_1-, I_2-, and I_3-type problem instances. It is observed that a differentiation between the efficiency loss incurred by both optimization methods is possible for balanced I_3-type and competitive I_4-type problem instances but at a lower level of significance of $p < 0.05$. The results provide evidence that for the balanced I_4-type problem instances, a differentiation is not possible at a high level of significance. The comparison between GreedyLow and GreedyRandom shows that for 22 (46 percent) of problem instances categories, no significant difference in the quality of the allocation computed.

In summary, the greedy-type optimization methods perform significantly different. On average, GreedyComplex approximates the allocation with a mean efficiency loss of 13 percent followed by GreedyMedium (17 percent) and GreedyLow (25 percent). The optimization methods are comparably robust in solving different problem instances. GreedyComplex significantly improves on all kinds of problem instances. GreedyMedium in comparison with GreedyRandom significantly improves in allocation quality on most of the problem instances types but falls short for three of the balanced I_4 instances. GreedyLow cannot be shown to outperform GreedyRandom to a significant extent. As a result, the sorting provided by GreedyLow does not capture the nature of the problem instances well and is therefore considered not to be a valuable approach.

5.2.2. Results of scalability. Figure 13 shows the results for the running time in seconds as an average value over the solution times for the problem instances solved. The results for the exact

optimization approach clearly indicate that the effort of solving a problem instance to optimality is subject to exponential growth in the number of resource consumer and resource provider bids. Computationally most expensive, whether balanced or competitive, are the I_2-type problem instances followed by the I_3-, I_4-, and I_1-type problem instances.

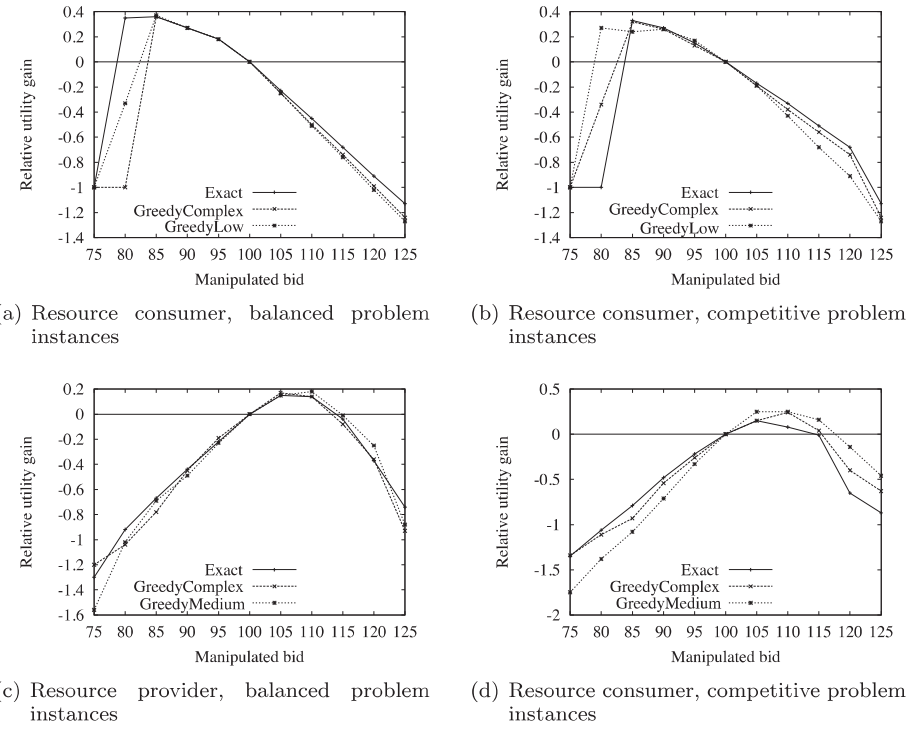
It is to be noted that the average values starting from 50 to 70 resource consumer bids do not reflect the exponential running time of the exact optimization approach as the solution process is interrupted after 3 min no matter if the optimal allocation, an intermediate allocation, or no allocation is provided (cf. Figure 10. A comparison of the running time results of the exact optimization approach with the running time results of the greedy optimization methods shows that the greedy optimization methods strikingly outperform the exact approach. The greedy optimization methods provide results for each of the problem instances tested while the exact method consumes the total amount of 3 min for 49 percent of the problem instances and does not guarantee a feasible allocation at all.

The results show that GreedyComplex on average consumes a significantly higher amount of time to determine an allocation in comparison with GreedyMedium and GreedyLow. Many times, GreedyMedium in comparison with GreedyLow consumes an additional amount of time to determine an allocation. The relation is reversed for I_2 problem instances. None of the three approaches tested consumes approximately 3 min for the identification of an allocation. While the difference for average time consumption between GreedyLow and GreedyMedium is in the range of 2 to 3 ms, GreedyComplex requires approximately double the amount of time required by GreedyLow or GreedyMedium to determine an allocation. On average, I_1 problem instances require the least effort to be solved followed by I_3-, I_2-, and I_4-type problem instances, which require the highest effort.

The growth in solution time with respect to the increase in the number of resource consumer and resource provider bids is approximated best by a quadratic function. Statistical evaluation confirms the validity of the estimated functions. For the I_1 instances, the coefficient of determination (R^2), is

Inst.	Balanced					Competitive				
Type	Size	Exact	Greedy Complex	Greedy Medium	Greedy Low	Size	Exact	Greedy Complex	Greedy Medium	Greedy Low
I_1	10/10	0.04	0.05	0.01	0.01	30/10	0.14	0.06	0.03	0.03
	30/30	0.51	0.13	0.05	0.05	90/30	8.73	0.31	0.15	0.14
	50/50	8.62	0.21	0.09	0.09	150/50	91.03	0.64	0.31	0.31
	70/70	13.12	0.33	0.15	0.14	210/70	148.50	1.05	0.53	0.53
	90/90	43.47	0.50	0.23	0.21	270/90	180.00	1.58	0.80	0.79
	110/110	75.74	0.70	0.34	0.30	330/110	180.00	2.17	1.11	1.05
	Avg.	23.58	0.32	0.15	0.13	Avg.	101.40	0.97	0.49	0.47
I_2	10/10	0.18	0.03	0.01	0.02	30/10	3.91	0.12	0.05	0.06
	30/30	58.10	0.18	0.07	0.08	90/30	180.00	0.59	0.26	0.28
	50/50	175.73	0.38	0.14	0.16	150/50	180.00	1.19	0.57	0.59
	70/70	178.60	0.63	0.24	0.25	210/70	180.00	1.99	0.93	1.00
	90/90	180.00	0.89	0.39	0.40	270/90	180.00	2.85	1.40	1.42
	110/110	180.00	1.20	0.50	0.53	330/110	180.00	3.97	1.93	1.98
	Avg.	128.77	0.55	0.22	0.24	Avg.	150.65	1.78	0.86	0.88
I_3	10/10	0.06	0.03	0.01	0.01	30/10	0.68	0.08	0.04	0.04
	30/30	4.12	0.15	0.07	0.07	90/30	74.43	0.47	0.23	0.22
	50/50	34.10	0.31	0.14	0.13	150/50	179.03	0.96	0.47	0.47
	70/70	94.57	0.49	0.23	0.21	210/70	180.00	1.55	0.81	0.76
	90/90	154.57	0.72	0.34	0.31	270/90	180.00	2.25	1.16	1.12
	110/110	174.00	1.00	0.47	0.45	330/110	180.00	3.20	1.64	1.52
	Avg.	76.90	0.45	0.21	0.20	Avg.	132.38	1.42	0.72	0.69
I_4	10/10	0.02	0.02	0.01	0.01	30/10	0.06	0.05	0.02	0.02
	30/30	0.55	0.15	0.07	0.07	90/30	33.06	0.45	0.24	0.22
	50/50	7.68	0.39	0.23	0.19	150/50	160.93	1.61	0.86	0.88
	70/70	79.96	1.04	0.54	0.49	210/70	180.00	3.38	1.56	1.70
	90/90	147.13	1.35	0.67	0.67	270/90	174.20	5.32	2.70	2.55
	110/110	180.00	2.58	1.30	1.15	330/110	180.00	8.42	4.52	4.29
	Avg.	69.84	0.92	0.47	0.43	Avg.	121.38	3.20	1.65	1.61

Figure 13. Comparison of the mean runtime in seconds for balanced and competitive problem instances.



(a) Resource consumer, balanced problem instances

(b) Resource consumer, competitive problem instances

(c) Resource provider, balanced problem instances

(d) Resource consumer, competitive problem instances

Figure 14. Median of the relative utility gain for a manipulating resource consumer or a manipulating resource provider against balanced and competitive problem instances. (a) Resource consumer, balanced problem instances; (b) resource consumer, competitive problem instances; (c) resource provider, balanced problem instances; (d) resource consumer, competitive problem instances.

at minimum 0.922 indicating a very good fit. The F-test shows at a significance level of $p < 0.01$ that the estimated parameters are of relevance in explaining the postulated relation between the number of bids included into a problem instance and the solution time. A t-test provides evidence that each of the β_2 parameters differs at a significance level of $p < 0.01$ from zero, emphasizing the fit of the quadratic function. The results for the functions fitted to the I_4 problem instances differ in R^2 (at minimum 0.598) indicating a good, but worse, fit in comparison with the functions fitted to the I_1 problem instances. The worse fit is partly caused by the higher standard deviation of the running time for I_4 problem instances.

Because the determination of an allocation to I_1 instances takes on average the minimum amount of time and the determination of an allocation to I_4 instances takes the maximum amount of time, the corresponding fitted functions for balanced problem instances are used to provide an estimate of the bounds on scalability for the heuristic optimization methods. According to these estimates, GreedyComplex is limited to 2018 resource consumer and resource provider bids, GreedyMedium is limited to 2850 resource consumer and resource provider bids, and GreedyLow is applicable up to 3191 resource consumer and resource provider bids. In case constraints on bids for computing resources are allowed (I_4-type problem instances), scalability is reduced to 850 (GreedyComplex), 1228 (GreedyMedium), and 1325 (GreedyLow) resource provider and resource consumer bids.

5.2.3. Results of incentive compatibility. The scenario is illustrated with an example considering balanced problem instances and GreedyComplex for solving the multi-attribute combinatorial exchange winner determination problem. The results for a resource consumer manipulating their valuation are given in Figure 14(a). Understating her valuation for a bundle bid from 75 to 80 percent of the real valuation results at minimum 50 percent of the cases in a total loss of the utility that would have been gained if the valuation would have been reported truthfully. A less aggressive manipulation strategy understating the true valuation by at most 15 percent is beneficial.

Percentage bid	Resource consumer				Resource provider			
	μ	Med.	$Q_{0.2}$	$Q_{0.8}$	μ	Med.	$Q_{0.2}$	$Q_{0.8}$
75	-0.37	-1.00	-1.00	0.62	-15.94	-1.30	-2.69	-0.73
80	-0.17	0.35	-1.00	0.58	-12.39	-0.92	-1.79	-0.58
85	0.01	0.36	-1.00	0.66	-9.00	-0.67	-1.14	-0.37
90	0.03	0.27	-1.00	0.50	-4.43	-0.44	-0.77	-0.22
95	0.12	0.18	0.11	0.32	-2.05	-0.22	-0.40	-0.13
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
105	-0.40	-0.23	-0.37	-0.13	0.10	0.15	-0.14	0.31
110	-0.80	-0.45	-0.75	-0.26	0.04	0.14	-0.43	0.42
115	-1.19	-0.68	-1.12	-0.36	-0.02	-0.04	-0.73	0.46
120	-1.59	-0.91	-1.49	-0.52	-0.26	-0.37	-1.00	0.26
125	-1.99	-1.13	-1.88	-0.67	-0.45	-0.74	-1.00	0.07

Figure 15. Relative resource consumer and resource provider utility gain for balanced *I*-type problem instances applying GreedyComplex.

Percentage bid	Resource consumer				Resource provider			
	μ	Med.	$Q_{0.2}$	$Q_{0.8}$	μ	Med.	$Q_{0.2}$	$Q_{0.8}$
75	-0.37	-1.00	-1.00	0.59	-4.73	-1.35	-3.80	-0.92
80	-0.30	-1.00	-1.00	0.50	-3.46	-1.06	-2.27	-0.62
85	-0.10	0.33	-1.00	0.47	-2.31	-0.79	-1.62	-0.40
90	0.12	0.27	0.16	0.45	-1.08	-0.48	-0.92	-0.25
95	0.04	0.15	0.10	0.24	-0.22	-0.22	-0.41	-0.03
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
105	-0.30	-0.17	-0.32	-0.12	0.06	0.15	-0.29	0.35
110	-0.59	-0.33	-0.64	-0.24	0.08	0.08	-0.87	0.57
115	-0.89	-0.51	-0.95	-0.35	0.08	-0.01	-1.00	0.60
120	-1.19	-0.68	-1.27	-0.47	-0.27	-0.65	-1.00	0.58
125	-1.48	-1.13	-1.62	-0.59	-0.43	-0.87	-1.00	0.23

Figure 16. Relative resource consumer and resource provider utility gain for competitive *I*-type problem instances applying the exact optimization method.

The utility gained is raised in more than 50 percent of the cases compared with reporting the valuation truthfully. Overstating the valuation by 5 to 25 percent results in more than 80 percent of the cases tested in a loss of utility (17). In case the valuation is overstated by 25 percent, a resource consumer incurs a minimum loss of 124 percent for 50 percent of the problem instances analyzed. A detailed overview on the results for the exact optimization method as well as GreedyComplex and GreedyMedium is presented in Figures 15–20.

The results for the exact and the heuristic optimization methods provide evidence that depending on the degree of manipulation, an average positive utility gain is achievable for resource consumers who are understating their valuation and resource providers who are overstating their reservation price. This holds for both competitive and balanced problem instance types.

Considering the exact optimization method and balanced (competitive) problem instances, a resource consumer realizes an average utility gain in case the valuation is not manipulated by more than 15 (10) percent. A resource provider does not incur a utility loss for balanced (competitive) problem instances in case the reservation price is manipulated by less than 10 (15) percent. A manipulating resource provider reaps an average utility gain up to a manipulation factor of 15 percent. The mean results for GreedyMedium deviate to a higher degree. A resource consumer incurs an average utility gain manipulating the respective valuation up to 20 percent considering balanced and competitive problem instance types. For resource providers, manipulation is on average beneficial up to 20 percent for balanced problem instance types and up to 25 percent for competitive problem instances.

A comparison of the mean values indicates for some degrees of manipulation a significant difference in the mean utility gain with respect to the different optimization methods (e.g., Figures 18 and 15). An analysis of the mean values and the respective median values indicates that this is caused by a small number of problem instances where the respective bidder achieves a disproportional high utility gain or loss. The conjecture is emphasized considering the bounds of the upper and lower quintiles for each sample. For all tested sets, these bounds are relatively close to the respective median. Further statistical support is provided by comparing the distribution of the utility gains between the

Percentage bid	Resource consumer				Resource provider			
	μ	Med.	$Q_{0.2}$	$Q_{0.8}$	μ	Med.	$Q_{0.2}$	$Q_{0.8}$
75	-0.41	-1.00	-1.00	0.61	-6.11	-1.20	-2.42	-0.68
80	-0.27	-1.00	-1.00	0.55	-4.62	-1.04	-1.92	-0.48
85	0.00	0.36	-1.00	0.68	-2.95	-0.78	-1.26	-0.38
90	0.01	0.27	-1.00	0.50	-0.17	-0.45	-0.84	-0.28
95	0.07	0.18	0.10	0.31	-0.04	-0.19	-0.42	-0.09
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
105	-0.46	-0.25	-0.53	-0.13	0.12	0.17	-0.14	0.39
110	-0.92	-0.50	-1.05	-0.26	0.08	0.14	-0.30	0.40
115	-1.39	-0.74	-1.58	-0.40	0.03	-0.08	-0.61	0.54
120	-1.82	-0.99	-2.10	-0.52	-0.32	-0.36	-1.00	0.23
125	-2.28	-1.24	-2.76	-0.66	-0.53	-0.93	-1.00	-0.13

Figure 17. Relative resource consumer and resource provider utility gain for competitive *I*-type problem instances applying GreedyComplex.

Percentage bid	Resource consumer				Resource provider			
	μ	Med.	$Q_{0.2}$	$Q_{0.8}$	μ	Med.	$Q_{0.2}$	$Q_{0.8}$
75	-0.36	-1.00	-1.00	0.60	-6.27	-1.34	-4.26	-0.54
80	-0.25	-0.34	-1.00	0.52	-3.93	-1.11	-3.23	-0.41
85	-0.01	0.32	-1.00	0.51	-2.37	-0.93	-2.55	-0.28
90	0.06	0.26	-1.00	0.49	-0.84	-0.54	-1.52	-0.24
95	0.07	0.13	0.09	0.28	-0.35	-0.26	-0.49	-0.12
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
105	-0.28	-0.19	-0.37	-0.12	0.16	0.15	-0.19	0.43
110	-0.59	-0.38	-0.67	-0.23	0.11	0.24	-0.63	0.66
115	-0.90	-0.56	-1.05	-0.33	0.07	0.04	-0.98	0.80
120	-1.20	-0.74	-1.33	-0.46	-0.20	-0.40	-1.00	0.51
125	-1.48	-1.24	-1.60	-0.56	-0.39	-0.63	-1.00	0.10

Figure 18. Relative resource consumer and resource provider utility gain for balanced *I*-type problem instances applying the exact optimization method.

Percentage bid	Resource consumer				Resource provider			
	μ	Med.	$Q_{0.2}$	$Q_{0.8}$	μ	Med.	$Q_{0.2}$	$Q_{0.8}$
75	-0.06	-1.00	-1.00	0.69	-6.91	-1.56	-2.78	-0.87
80	0.01	-0.33	-1.00	0.69	-5.30	-1.02	-1.95	-0.61
85	0.13	0.37	-1.00	0.75	-3.61	-0.69	-1.43	-0.34
90	0.07	0.27	-1.00	0.55	-0.74	-0.49	-0.89	-0.28
95	0.06	0.18	0.08	0.34	-0.39	-0.23	-0.46	-0.12
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
105	-0.56	-0.25	-0.77	-0.14	0.12	0.15	-0.15	0.37
110	-1.07	-0.51	-1.55	-0.27	0.18	0.18	-0.20	0.52
115	-1.72	-0.76	-2.09	-0.41	0.08	-0.01	-0.65	0.58
120	-2.54	-1.02	-2.55	-0.54	-0.21	-0.25	-1.00	0.22
125	-3.05	-1.27	-3.49	-0.69	-0.54	-0.88	-1.00	-0.01

Figure 19. Relative resource consumer and resource provider utility gain for competitive *I*-type problem instances applying GreedyMedium.

degrees of manipulation and the types of problem instances with a Kruskal–Wallis test. None of the tests provides significant evidence for rejecting the null hypothesis of equal variances.

5.3. Discussion of the results of greedy optimization methods

Allocative efficiency. The results show that GreedyLow is a valuable approach for determining high-quality allocations because it does not outperform GreedyRandom. GreedyComplex is found to minimize the efficiency loss in comparison with GreedyMedium. It improves on all kind of problem instances in comparison with GreedyRandom. GreedyMedium is found to outperform GreedyRandom on all but balanced I_4-type problem instances. Considering real world problem instances, constraints on computing resources are likely as computational tasks are parallelizable to a certain degree only. Splitting each consumer bid to an unlimited number of resource providers is an assumption of a theoretical nature. Consequently, GreedyComplex is argued to be applicable to the multi-attribute combinatorial exchange approach while GreedyMedium is of limited applicability.

Percentage bid	Resource consumer				Resource provider			
	μ	Med.	$Q_{0.2}$	$Q_{0.8}$	μ	Med.	$Q_{0.2}$	$Q_{0.8}$
75	-0.36	-1.00	-1.00	0.55	-14.15	-1.75	-4.70	-0.68
80	1.15	0.27	-1.00	0.53	-8.42	-1.38	-4.02	-0.43
85	1.15	0.24	-1.00	0.66	-4.90	-1.08	-2.55	-0.32
90	1.07	0.26	-1.00	0.54	-2.01	-0.71	-1.55	-0.19
95	0.52	0.17	0.08	0.35	-0.46	-0.33	-0.57	-0.12
100	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
105	-0.54	-0.19	-0.63	-0.11	0.32	0.25	-0.02	0.49
110	-1.16	-0.43	-1.06	-0.23	0.34	0.25	-0.49	0.99
115	-1.81	-0.68	-1.90	-0.35	0.45	0.16	-0.78	1.05
120	-3.24	-0.91	-2.36	-0.46	0.38	-0.14	-1.00	1.11
125	-4.12	-1.27	-2.91	-0.56	0.12	-0.46	-1.00	0.85

Figure 20. Relative resource consumer and resource provider utility gain for competitive *I*-type problem instances applying GreedyMedium.

A significant loss in efficiency is observed comparing the results for the balanced and competitive problem instance types. This is caused by the enlargement of the decision space, which can lead to a situation where the heuristic optimization methods are guided to a local maximum. Furthermore, the problem representation does not cover the entire search space.

Scalability. The results for the exact optimization method clearly indicate the limits due to computational tractability. An efficiency loss of 100 percent for larger problem instances, which is similar to a breakdown of the market, is not tolerable for the practical application of the mechanism. In consequence, as constraints on bids as well as multiple bids by a single resource consumer are considered to appear frequently, determining the optimal allocation is a feasible approach for balanced problem instances up to 70 resource consumer bids and for competitive problem instances up to 90 resource consumer bids. The estimates provided for the greedy-type optimization methods indicate an applicability to a minimum of 850 (GreedyComplex) or 1228 (GreedyMedium) and a maximum of 2018 (GreedyComplex) or 2080 (GreedyMedium) resource consumer bids. The estimate is not accurate but rather conservative because a quadratic function tends to overestimate the influence of the dependent variable [45].

Incentive compatibility. The results show that the effect of manipulation by a resource consumer or a resource yields on average a comparable relative utility gain for all optimization methods over all types of problem instances tested, which is an important and interesting finding. The potential relative gain incurred by manipulation is basically caused by the pricing scheme and independence of the optimization method applied. The results show an efficiency loss in the majority of the cases with the heuristic optimization method of the multi-attribute combinatorial exchange approach to tackle the winner determination problem. An allocation that is provided by a heuristic optimization method does not ensure that the traded computing resources are assigned to the resource consumers who value them most and, at the same time, are supplied by the resource providers who value them least. However, the theoretical worst case scenario of a total efficiency loss is found to be highly unlikely as indicated by the average efficiency loss for all types of problem instances. A comparison of the different heuristic optimization methods shows the efficiency loss to be smallest for GreedyComplex followed by GreedyMedium and GreedyLow. The multi-attribute combinatorial exchange mechanism is not incentive compatible but is reported not to incentivize resource consumers, and resource providers strongly manipulate their valuations or reservation prices [29]. From a theoretical perspective, heuristic optimization methods ruin the incentive properties of incentive compatible exchange mechanisms [35]. A plausible conjecture is that heuristic optimization methods negatively affect the incentive properties of the multi-attribute combinatorial exchange mechanism. However, no statistical evidence for such an effect could be found in the results of this work. The application of heuristic optimization methods to the multi-attribute combinatorial exchange approach is a trade-off between efficiency and scalability that does not negatively affect the property of incentive compatibility. The advances in scalability at the cost of an efficiency loss are a step forward in making a theoretically valuable mechanism applicable for large-scale computing infrastructures.

Limitations. Effort was taken to realistically model important details of possible real world problem instances to provide an unbiased evaluation of the approaches presented. However, the statistical

generation of valuations and reservation prices, which can make hard problem instances computationally easy [39], necessitates a confirmation of the findings presented when data from real world problem instances become available. The benefits of strategic behavior are investigated considering a single manipulating resource consumer or a single manipulating resource provider. The possible benefits of forming strategic coalitions between participants require further investigation [37]. The prototypical implementation of the framework, which the heuristic optimization methods are integrated into, is designed for scientific testing [46]. Consequently, the running time results for the heuristics have to be interpreted as lower bounds while the results for the exact optimization method are obtained from established code.

6. CONCLUSION

Large-scale computing infrastructures enable the sharing of distributed computing resources providing the potential for more cost-efficient utilization of IT. A key concern in leveraging large-scale computing infrastructures is the scheduling of available capacities. This paper argued for a market-based approach to resource scheduling, focused on developing an efficient allocation. Current approaches work well for a limited number of resource consumers and providers, but they necessitate a rethinking to make them applicable to larger scales. The objective of the research in this paper is the conceptualization, implementation, and evaluation of a scalable market mechanism that is suitable for the needs of large-scale computing infrastructures. The multi-attribute combinatorial exchange approach is not scalable in the number of participants but fits both the domains-specific and the economic requirements. We provide a generic representation of the allocation problem based on the multi-attribute combinatorial exchange approach, which is suitable for greedy optimization methods. Three greedy approaches differing in the computational complexity are customized to the problem representation.

The theoretical evaluation of the problem representation shows a reduction of the search space; that is, the efficient allocation of resource consumers to resource providers may not be encoded. The empirical evaluation measured the actual efficiency loss, the scalability, and the influence on incentive compatibility. There is no statistical evidence for an influence on the incentive properties of the multi-attribute combinatorial exchange mechanism; that is, the application of the evaluated heuristic optimization methods presents a trade-off between efficiency and scalability. The average efficiency loss for the recommended greedy heuristic is 13 percent. The scope of the mechanism is increased from 70 resource consumer and provider bids to a minimum of 850 resource consumer and provider bids. In consequence, using heuristics is a step forward in making theoretically valuable mechanisms applicable for large-scale computing infrastructures.

REFERENCES

1. Brynjolfsson E, Saunders A. *Wired for Innovation: How Information Technology Is Reshaping the Economy*. MIT Press: Cambridge, MA, 2010.
2. Hey T. The next scientific revolution. *Harvard Business Review*. 2010; **88**(11):57–63.
3. Davenport TH, Harris JG. *Competing on Analytics – The New Science of Winning*. Harvard Business School Publishing: Boston, 2007.
4. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. *Communications of the ACM* 2010; **53**(4):50–58.
5. Siegle L. Let it rise: a special report on corporate IT. *The Economist*. 2008. (Available from: http://www.economist.com/node/12411882) [Accessed on 25 July 2010].
6. Carr NG. The end of corporate computing. *MIT Sloan Management Review* 2005; **46**(3):67–73.
7. Foster I, Kesselman C, Tuecke S. The anatomy of the grid: enabling scalable virtual organizations. *International Journal of High Performance Computing Applications* 2001; **15**(3):200–221.
8. M. Vaquero L, Rodero-Merino L, Caceres J, Lindner M. A break in the clouds: towards a cloud definition. *ACM SIGCOMM Computer Communication Review* 2009; **39**(1):50–55.
9. Foster I, Yong Z, Raicu I, Lu S. Cloud computing and grid computing 360-degree compared. *In Proceedings of the IEEE Grid Computing Environments Workshop, 2008. GCE '08*: Austin, TX, USA, 2008; 1–10.
10. Neumann D, Stösser J, Weinhardt C, Nimis J. A framework for commercial grids – economic and technical challenges. *Journal of Grid Computing* 2008; **6**(3):325–347.

11. Eymann T, Neumann D, Reinicke M, Schnizler B, Streitberger W, Veit D. On the design of a two-tiered grid market structure. *In Proceedings of the Multikonferenz Wirtschaftsinformatik 2006*: Passau, Germany, 2006.
12. Van den Bossche R, Vanmechelen K, Broeckhove J. Cost-optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. *IEEE 3rd International Conference on Cloud Computing (CLOUD)*, Miami, Florida, USA, 2010; 228–235.
13. Rahman M, Li X, Palit H. Hybrid heuristic for scheduling data analytics workflow applications in hybrid cloud environment. *IEEE International Symposium on Parallel and Distributed Processing Workshops and PhD Forum (IPDPSW)*, Anchorage, Alaska, USA, 2011; 966–974.
14. Lai K. Markets are dead, long live markets. *SIGecom Exchanges*. 2005; **5**(4):1–10.
15. Shneidman J, Ng C, Parkes DC, AuYoung A, Snoeren AC, Vahdat A, Chun B. Why markets could (but don't currently) solve resource allocation problems in systems. *Proceedings of the 10th Conference on Hot Topics in Operating Systems*, vol. 10 USENIX Association: Santa Fe, NM, USA, 2005; 37–42.
16. Smidt S. Flexible pricing of computer services. *Management Science* 1968; **14**(10):B–581–B–600.
17. Nissan N. Introduction to mechanism design (for computer scientists). *Algorithmic Game Theory*, Nissan N, Roughgarden T, Tardos E, Vazirani VV (eds). Cambridge University Press: New York, NY, 2007; 209–242.
18. Garey MR, Johnson DS. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman: San Francisco, CA, 1979.
19. Nurmi D, Wolski R, Grzegorczyk C, Obertelli G, Soman S, Youseff L, Zagorodnov D. The eucalyptus open-source cloud-computing system. In *9th IEEE/ACM International Symposium on Cluster Computing and The Grid, 2009. CCGRID'09* IEEE, Shanghai, China, 2009; 124–131.
20. Marshall P, Keahey K, Freeman T. Elastic site: using clouds to elastically extend site resources. *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, IEEE Computer Society, Melbourne, Australia, 2010; 43–52.
21. Milojičić D, Llorente IM, Montero RS. Opennebula: a cloud management tool. *IEEE Internet Computing* 2011; **15**(2):11–14.
22. Pepple Ken. *Deploying Openstack*. O'Reilly Media, 2011. http://www.amazon.co.uk/Deploying-OpenStack-Ken-Pepple/dp/1449311059
23. Sotomayor B, Montero RS, Llorente IM, Foster I. Virtual infrastructure management in private and hybrid clouds. *IEEE Internet Computing* 2009; **13**(5):14–22.
24. Garg SK, Buyya R, Siegel HJ. Time and cost trade-off management for scheduling parallel applications on utility grids. *Future Generation Computer Systems* 2010; **26**(8):1344–1355.
25. Reiss C, Tumanov A, Ganger GR, Katz RH, Kozuch MA. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. *Proceedings of the Third ACM Symposium on Cloud Computing* ACM, San Jose, California, 2012; 7.
26. Alrawahi AS, Lee K. Multi-attribute combinatorial marketplaces for cloud resource trading. *2012 Second International Conference on Cloud and Green Computing (CGC)* IEEE, Xiangtan, Hunan, China, 2012; 81–88.
27. Amazon E. *Amazon Elastic Compute Cloud (Amazon EC2)*. Amazon Elastic Compute Cloud (Amazon EC2), Dispon, 2013.
28. Dash RK, Jennings NR, Parkes DC. Computational-mechanism design: a call to arms. *IEEE Intelligent Systems* 2003; **18**(6):40–47.
29. Schnizler B, Neumann D, Veit D, Weinhardt C. Trading grid services – a multi-attribute combinatorial approach. *European Journal of Operational Research* 2008; **187**(3):943–961.
30. Pekeĉ A, Rothkopf MH. Combinatorial auction design. *Management Science* 2003; **49**(11):1485–1503.
31. Bapna R, Das S, Garfinkel R, Stallaert J. A market design for grid computing. *INFORMS Journal on Computing* 2008; **20**(1):100–111.
32. Stösser J, Neumann D, Weinhardt C. Market-based pricing in grids: on strategic manipulation and computational cost. *European Journal of Operational Research*. 2010; **203**(2):464–475.
33. Stösser J, Neumann D. Greedex – a scalable clearing mechanism for utility computing. *Electronic Commerce Research*. 2008; **8**(4):235–253.
34. Mossmann M, Stösser J, Ouorou A, Gourdin E, Krishnaswamy R, Neumann D. A combinatorial exchange for complex grid services. *Economic Models and Algorithms for Distributed Systems*, Neumann D, Rana OF, Altmann J, Baker M (eds). Birkhäuser: Basel, 2010; 221–237.
35. Lehmann D, Müller R, Sandholm T. The winner determination problem. *Combinatorial Auctions*, Crampton P, Shoham Y, Steinberg R (eds). MIT Press: Cambridge, MA, 2006; 555–596.
36. de Vries S, Vohra RV. Combinatorial auctions: a survey. *INFORMS Journal on Computing* 2003; **15**(3):284–309.
37. Parkes DC. Iterative combinatorial auctions: achieving economic and computational efficiency. *Ph.D. Thesis*, University of Pennsylvania, 2001.
38. Lewis M, Kochenberger GA, Alidaee B. A new modeling and solution approach for the set-partitioning problem. *Computers & Operations Research* 2008; **35**(3):807–813.
39. Leyton-Brown K, Shoham Y. A test suite for combinatorial auctions. *Combinatorial Auctions*, Crampton P, Shoham Y, Steinberg R (eds). MIT Press: Cambridge, MA, 2006; 853–903.
40. Rardin RL, Uzsoy R. Experimental evaluation of heuristic optimization algorithms: a tutorial. *Journal of Heuristics* 2001; **7**(3):261–304.
41. Leyton-Brown K, Pearson M, Shoham Y. Towards a universal test suite for combinatorial auction algorithms. In *Proceedings of the 2nd ACM Conference on Electronic Commerce*, ACM: Minneapolis, MN, USA, 2000; 66–76.

42. Schnizler B. Resource allocation in the grid: a market engineering approach. *Dissertation*, University of Karlsruhe (TH), 2007.
43. Cripps MW, Swinkels JM. Efficiency of large double auctions. *Econometrica*. 2006; **74**(1):47–92.
44. Roberts DJ, Postlewaite A. The incentives for price-taking behavior in large exchange economies. *Econometrica* 1976; **44**(1):115–127.
45. Backhaus K, Erichson B, Plinke W, Weiber R. *Multivariate Analysemethoden*, vol. 12. Springer: Heidelberg, 2008.
46. Hooker JN. Testing heuristics: we have it all wrong. *Journal of Heuristics* 1995; **1**(1):33–42.