A Truthful Dynamic Workflow Scheduling Mechanism for Commercial Multicloud Environments

Hamid Mohammadi Fard, Radu Prodan, and Thomas Fahringer

Abstract—The ultimate goal of cloud providers by providing resources is increasing their revenues. This goal leads to a selfish behavior that negatively affects the users of a commercial multicloud environment. In this paper, we introduce a pricing model and a truthful mechanism for scheduling single tasks considering two objectives: monetary cost and completion time. With respect to the social cost of the mechanism, i.e., minimizing the completion time and monetary cost, we extend the mechanism for dynamic scheduling of scientific workflows. We theoretically analyze the truthfulness and the efficiency of the mechanism and present extensive experimental results showing significant impact of the selfish behavior of the cloud providers on the efficiency of the whole system. The experiments conducted using real-world and synthetic workflow applications demonstrate that our solutions dominate in most cases the Pareto-optimal solutions estimated by two classical multiobjective evolutionary algorithms.

Index Terms—Workflow scheduling, multicloud environment, game theory, reverse auction, truthful mechanism

1 Introduction

Task scheduling is a major challenge in parallel and distributed systems. Task scheduling techniques in distributed systems are usually based on trusting the accuracy of the information about the status of resources (e.g., their load). This information is usually submitted by resource providers to a central database which is accessible to schedulers. Although in some environments, such as Grids established between academic partners, this information might be considered as complete and accurate, there are other cases when we cannot rely on this assumption.

Nowadays, the number of commercial cloud providers is rapidly increasing. In a commercial multicloud environment, individual providers are focused toward increasing their own revenue and do not care about the utility of users and other providers. In such an environment, we cannot trust the information presented by the providers. As the influence of the Internet in solving large-scale problem is growing, the scenario of having multiple self-interested agents is becoming more and more popular. In a commercial multicloud environment consisting of a set of selfish cloud providers with private information about their resources, application schedulers should act carefully and not trust the information submitted by providers about the status of their resources, as there is always the possibility of misrepresenting private information. For example, the providers might exaggerate the speed of their resources to increase their chance of earning more revenues.

Game theory is a mathematical approach to analyze the decisions of agents in a problem modeled as a game [1]. Reverse game theory [2] or algorithmic mechanism design is a subfield of game theory dealing with multiple selfish agents [3]. In an abstract view, algorithmic mechanism design is the intersection between game theory and computer science. The general assumption in mechanism design is that all agents clearly know the mechanism rules. Each agent has a privately known function called type which is unknown for the mechanism and to other agents. All agents take part in the game using their private information. Moreover, each agent has a benefit or loss depending on the outcome of the mechanism called valuation. While the mechanism designer targets the minimization of the social cost of the game, the major challenge appears when the goals of agents are in conflict with it. While in mechanism design we are dealing with self-interested agents, one main goal is avoiding that the agents lie. In game theory literature, such mechanisms that force the self-interested agents to tell the truth are called truthful mechanisms, dominant strategies [3] or strategy-proofs [4]. An efficient mechanism, by use of payment mechanisms, motivates the agents to tell the truth about their private information.

In this context, workflow applications emerged in the last decade as an attractive paradigm for programming distributed computing infrastructures such as computational Grids and Clouds. However, scheduling of workflows in a commercial multicloud environment still remains an open issue to be addressed. In this work, we address the problem of dynamic scheduling of workflow applications in commercial multicloud environments using algorithmic mechanism design. We are looking for an efficient equilibrium of the scheduling game in a reasonable time. Each

• *The authors are with the Distributed and Parallel Systems Group, Institute for Computer Science, University of Innsbruck, Technikerstr. 21a, A-6020 Innsbruck, Austria. E-mail: {hamid, radu, tf}@dps.uibk.ac.at.*

Manuscript received 28 Feb. 2012; revised 19 June 2012; accepted 23 Aug. 2012; published online 31 Aug. 2012.
Recommended for acceptance by V.B. Misic, R. Buyya, D. Milojicic, and Y. Cui.
For information on obtaining reprints of this article, please send e-mail to: tpds@computer.org, and reference IEEECS Log Number TPDSSI-2012-02-0161.
Digital Object Identifier no. 10.1109/TPDS.2012.257.

resource is assumed as a selfish strategic agent owning some private information that might be misrepresented to increase revenue. Considering that scheduling in distributed systems is an NP-hard problem, our challenge as mechanism designers is to propose an efficient scheduling mechanism that yields low completion time (makespan) and monetary cost for execution of workflow applications compared to the state of the art.

To address the scheduling problem in a commercial multicloud environment using reverse auctions, we first propose a new truthful mechanism for scheduling single tasks on the set of resources. Afterwards, we tailor the proposed mechanism to dynamically schedule workflow applications. Our motivation to develop an auction-based scheduling mechanism for the commercial multicloud environment is fivefold:

1. it can deal with the noncooperation nature of scheduling game in multicloud environment;
2. it harnesses the selfish behavior of cloud providers;
3. the mechanism needs a small amount of public knowledge while each cloud provider has a large amount of private information;
4. the mechanism has a Nash equilibrium;
5. the approach is dynamic and easy to implement.

We analyze the truthfulness of the mechanism theoretically and the time and communication complexities both theoretically and experimentally. Finally, we evaluate the proposed mechanism by comparing it with two classical multiobjective evolutionary algorithms (MOEAs). The research presented in this paper is an extension of our previous work [5] which includes an extended model, two formal proofs about the time and communication complexity of our mechanism, new experiments, an illustrative example and more detailed description of our entire approach.

This paper is organized as follows: Section 2 is devoted to related work. We formally describe the model and the research problem in Sections 3 and 4. Section 5 proposes a truthful mechanism for dynamic scheduling of a single task in commercial multicloud environments, tailored in Section 6 for workflow scheduling. We illustrate our scheduling mechanism by means of a simple example in Section 7 and experimentally evaluate it in Section 8. In Section 9, we conclude the paper.

2 Related Work

We summarize the related work in three main areas: multicloud computing, auction-based truthful scheduling mechanisms in distributed systems, and multiobjective workflow scheduling.

2.1 Multicloud Computing

The research domain of multicloud computing is relatively young with no extensive literature available. The concept of sky computing has been introduced in [6] for building a virtual site distributed on several clouds. In [7], important challenges and architectural elements for utility-oriented federation of multiple cloud computing environments is investigated. The role of cloud brokers responsible for splitting the user requests to multiple providers with the goal of decreasing the cost for users is discussed in [8], [9].

2.2 Auction-Based Scheduling

Market-based scheduling of resources in distributed systems is a well-studied research filed. A Grid economy survey has been published in [10]. Several economical models like commodities market and auctions have been proposed for distributed systems. For instance, several auction mechanisms for selfish tasks competing for distributed resources have been investigated in [11]. The quality of the equilibrium solutions with respect to execution time is studied theoretically as well. In [12], a hierarchical noncooperative game model for Grids is proposed, where users are selfish agents.

A static load balancing mechanism based on the well-known Vickrey-Clarke-Groves (VCG) mechanism [1] for distributed systems is proposed in [13]. In [14], the authors present a dynamic solution for the problem of load balancing in Grid systems using a game theoretic approach, where the mechanism spreads the load on each computing node. They modeled load balancing as a constrained minimization problem and presented an algorithm that minimizes the average completion time of tasks. In the problems discussed in [3] and [4], the mechanisms only care about the outcome. Moreover, the payments are used only to enforce agents tell the truth.

The authors in [15] introduce a dynamic market-based model for resource allocation in Grids. The proposed bidding algorithm is described based on myopic equilibrium strategies. They analyze rational strategies of users in a repeated auction-based mechanism in which users look for required resources by updating their bids. The impact of antisocial agents to inflict losses on the other agents participating in a task scheduling mechanism on related machines (i.e., rather than maximizing their own profit) is analyzed in [16]. This work assumes that antisocial agents know the bid value of the winner which is against the assumption of presence of private information as in our work. A truthful mechanisms in single-dimension domain with $(1+\varepsilon)$-approximation for time (i.e., the solution is guaranteed to be within $1+\varepsilon$ factor of the optimum) is proposed in [17].

The authors of [18] propose a continuous double auction mechanism for scheduling tasks in Grids. The authors of [19] approach the scheduling problem of selfish organizations in Grids with respect to makespan similar to the Prisoner's Dilemma game [1]. They conclude that a strong control on the community is necessary to attain an acceptable performance. In [20], we propose a new instantiation of the negotiation protocol between the scheduler and resource manager using a market-based Continuous Double Auction model.

In [21], a market-based resource allocation model for batch jobs in cloud computing clusters was proposed. The social cost of this problem is the sum of the values gained by allocated users.

2.3 Multiobjective Workflow Scheduling

Workflow scheduling in distributed systems has been extensively studied. Since this problem is NP-hard, most of the current workflow scheduling methods use heuristics to cover the complexity. The major part of workflow scheduling approaches such as HEFT [22] address a single objective, typically the makespan.

Many existing multicriteria workflow scheduling approaches transform the problem to a single objective optimization problem by using various methods such as bounded objective optimization, lexicographical method, scalarization, and goal programming [23].

Among the multicriteria algorithms, we investigated those that target optimizing makespan and cost. LOSS and GAIN [24] schedule directed acyclic graphs (DAGs) by considering a budget constraint. They use a two-phase optimization process by first optimizing one criterion (makespan in LOSS and budget in GAIN) and then by rescheduling to find a solution that satisfies the budget constraint. The bicriteria workflow scheduling method proposed in [25] minimizes the execution cost of a workflow while meeting a deadline by finding a subdeadline for every workflow task. All these methods can be classified as bounded objective optimizations that consider one constrained objective and optimize another objective with respect to the defined constraints.

In [26], a bicriteria implementation of four general multiobjective optimization algorithms for workflow scheduling is presented. The output of every algorithm is a set of solutions (Pareto-set approximation) rather than a single one. Besides the high time complexity of the genetic algorithms, selecting the most adequate solution from the Pareto set remains a problem.

In [27], a multiobjective list scheduling approach for workflow applications is proposed. Based on a set of objectives constraints and weights defined by user, the algorithm attempts to find an appropriate Pareto solution in the region of interest for the users. The algorithm is customized and analyzed for four objectives: makespan, cost, reliability, and energy.

The authors of [28] propose two market-oriented scheduling policies for scheduling a set of independent tasks in hybrid Clouds. The goal is to satisfy an application deadline by extending the local computational capacity with cloud resources. In [29], a workflow scheduling and cloud provisioning heuristic that attempts to meet a specified budget as a soft constraint is proposed.

All aforementioned workflow scheduling algorithms are static approaches that do not consider the dynamic load of resources in a real environment. Moreover, all approaches suppose that the resource information published by the providers is correct, which is doubtful in commercial clouds, as discussed before. Furthermore, we should recall that due to the different pricing models [30], the proposed algorithms designed for utility Grids are not straightly useable in clouds. Since the problem in this paper involves multiple clouds, we need a new pricing model that can cope with the environment requirements. Our research is distinct from the related work by proposing a truthful mechanism for dynamic workflow scheduling in a commercial multicloud environment in presence of selfish providers. In most cases, the solution proposed by our mechanism is Pareto optimal and its efficiency is proved theoretically and practically.

3 Model

We model a *workflow application* as a directed acyclic graph $DAG([n],[e])$, where $[n]$ is the set of nodes representing n dependent tasks and $[e]$ indicates the control and data flow dependencies between tasks. Each task i is characterized by its workload $load(i)$ expressed, for example, in Million Instruction Per Second (MIPS). Formally, $[e]=(i, j, comm(i,j))$ where $comm(i,j)$ is the output of tasks i to j. Each workflow has a task with no predecessors called $entry$ task and a task with no successors called $exit$ task.

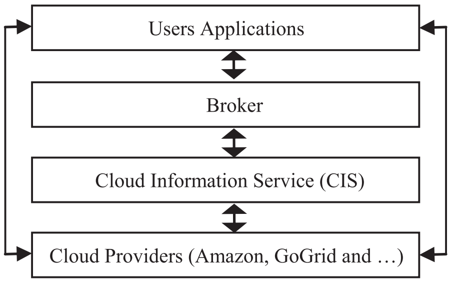

Users Applications
Broker
Cloud Information Service (CIS)
Cloud Providers (Amazon, GoGrid and …)

Fig. 1. Multicloud architecture.

The commercial multicloud environment $[m]$ comprises m selfish cloud providers, abstractly depicted in Fig. 1. In the bottom layer, we have a set of selfish cloud providers such as Amazon and GoGrid. On top of this layer, we design a *cloud information directory service* (CIS) whose task is to store information about the resources belonging to the existing cloud providers (similar to a Grid information service). This service is directly used in the *brokerage layer* implementing our proposed pricing model and scheduling mechanism by selecting the most adequate resources in terms of execution time and monetary cost for users. The top layer is the *user application* connected to the brokerage layer for scheduling purposes and to the bottom layer for submitting tasks.

We assume without loss of generality that each cloud provider has only one resource. The goal of each provider is to maximize its profit by executing as many tasks as possible. Hereafter, we use terms provider, resource, player, and agent interchangeably.

4 Problem Overview

In this section, we define the workflow scheduling problem with respect to two simultaneous minimization objectives: makespan and monetary cost of the execution. We denote each individual schedule of a task i as a pair (i,j) meaning that the task i is assigned to resource j. We denote the schedule of the entire workflow as $sched=\{(i,j)|i\in[n]\}$.

We assume the resource j is able to calculate the real completion time $t^{real}_{i,j}$ and the real computing cost $c^{real}_{i,j}$ for executing every task i. For each resource, the real completion time of a task is sum of two components: the time to transfer the input data and the effective execution time. To calculate the real completion time of a task, the resource must consider a number of internal details such as the virtual machine startup overhead, latency delay, current load, computing power, availability, ready time, communication bandwidth, task workload, and so on. We suppose that these calculations are performed by each provider internally and, therefore, ignore them in our model. The real cost of executing a task on a resource is again internally calculated by the provider based on its

TABLE 1
Objective Function Classification Based on [31]

Objective	*Structural Dependency*	*Aggregation*	*Complexity*
Makespan	Structure-dependent	Additive	$O(m^n)$
Monetary cost	Structure-independent	Additive	$O(m \cdot n)$

business model and by considering internal maintenance, operational, or energy consumption costs. Before the execution of task i, $t_{i,j}^{real}$ is private information for the resource j and not accessible to any user or other resource. After the completion of the task, the user will be aware of $t_{i,j}^{real}$, but will still not know $c_{i,j}^{real}$. Instead, the user will only know the price calculated by the broker).

Based on the real completion time of a single task, we define the workflow makespan as the time required for executing the whole workflow without violating the control and data flow dependencies between its tasks:

$$Makespan(DAG, sched) = \max_{i \in [n] \wedge (i,j) \in sched} t_{i,j}^{real}. \tag{1}$$

Similarly, we define the cost as the sum of the costs of executing all workflow tasks:

$$Cost(DAG, sched) = \sum_{i \in [n] \wedge (i,j) \in sched} t_{i,j}^{real}. \tag{2}$$

In both formulas, the time and cost for data transfers are implicitly covered in the real completion time and cost. The two objectives considered in our scheduling problem, namely monetary cost and makespan, differ with respect to several important classification criteria [31], summarized in Table 1:

- *Structural dependency* indicates if the objective function depends on the workflow structure;
- *Aggregation* indicates how the objective values of single tasks are aggregated for the entire workflow;
- *Complexity* indicates the time complexity of finding the best solution.

A background on multiobjective optimization theory is given in Appendix A.1 of the supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TPDS.2012.257.

5 BOSS: Biobjective Scheduling Strategy

The current cloud providers usually charge users based on a pay-as-you-go pricing model. With respect to our multiprovider cloud model and the two considered objectives (makespan and monetary cost), we suggest a new pricing model and truthful scheduling mechanism to find the "best" resource for executing a task called *biobjective scheduling strategy* implemented in the brokerage layer in Fig. 1. The mechanism is based on a reverse auction which is a common tool in a market with lots of sellers. Each auction is based on some rules which define the identity of the winner. The most famous result in this area is the VCG which applies to utilitarian (sum of the valuations of the agents) objectives only. We give in Appendix A.2 of the supplemental material, available in the online supplemental material, a brief background on the game-theoretic concepts behind our auctioning approach, for a better understanding.

In BOSS, each task i as an auctioneer starts an auction to select a proper resource for its execution. The task announces to the resources its workload, the dependencies with other tasks, and the required input and output. Each resource j bids by the strategy $s_{i,j} = (t_{i,j}, c_{i,j})$ which is a combination of its proposed time $t_{i,j}$ and proposed cost $c_{i,j}$. The strategy $s_{i,j}$ means that the resource j claims to complete the task i until the time $t_{i,j}$ with the cost of $c_{i,j}$. Each resource is allowed to bid by more than one strategy in every auction. For example, if a resource has the ability of dynamic voltage and frequency scaling (DVFS), it can propose several strategies with different completion times and costs. Without loss of generality, we assume only one strategy for each resource in every auction. The strategy profile denoted by $s_{i,.}$ comprises the strategies of all agents in the auction initiated by task i: $s_{i,.} = \bigcup_{j \in [m]} (t_{i,j}, c_{i,j})$. The strategy profile $s_{i,.}$ not including the strategy $s_{i,j}$ is denoted by $s_{i,-j}$. Formally, $s_{i,-j} = s_{i,.} - s_{i,j}$.

The winner w in every auction is selected based on the minimum product of cost and time across all agents. Therefore, w is the winner of the auction i (where the auctioneer is task i) if $t_{i,w} \cdot c_{i,w} = \min_{j \in [m]} \{t_{i,j} \cdot c_{i,j}\}$. The reason for using the multiplication aggregation to select the winner is twofold: 1) it takes both objectives into consideration and 2) the truthfulness of the mechanism is dependent on this function, as we will show in Theorem 1. After selecting the winner, the task is submitted to the winning resource. After its completion, the resource will be paid based on the following payment function inspired from the Vickrey's second price auction:

$$pay(i,j) = \begin{cases} \dfrac{t_{i,x} \cdot c_{i,x}}{t_{i,j}}, & \text{if } j = w \wedge t_{i,j}^{real} \le t_{i,j}; \\ f(s_{i,.}), & \text{if } j = w \wedge t_{i,j}^{real} > t_{i,j}; \\ 0, & \text{if } j \ne w, \end{cases} \tag{3}$$

where $f(s_{i,.})$ is a function of the strategy profile (see Lemma 1 and Appendix B of the supplemental material, available in the online supplemental material), and x is the resource that proposes the second smallest bid or $t_{i,x} \cdot c_{i,x} = \min_{j \in [m], j \ne w} \{t_{i,j} \cdot c_{i,j}\}$. In Theorem 1, we will prove that this payment function causes the truthfulness of the mechanism.

Obviously, the utility of each resource is calculated based on the following quasilinear function:

$$utility(i,j) = \begin{cases} pay(i,j) - c_{i,j}^{real}, & \text{if } j = w; \\ 0 \quad , & \text{if } j \ne w \end{cases} \tag{4}$$

describing the earning of the winner as the difference between the payment from the mechanism and the real cost of the resource in this execution.

Lemma 1. *The inequality $f(s_{i,.}) \le \min_j \{c_{i,j}^{real}\}$ is a necessary condition for the truthfulness of BOSS.*

Proof. Assuming a resource k such that $f(s_{i,.}) > c_{i,k}^{real}$, there is always a possibility for each resource to by a shorter time and win the game. If the resource k bids by the shortest time and wins the game, it will be paid $f(s_{i,.})$

based on the mechanism. The utility of the resource k is $utility(i,k) = f(s_{i,.}) - c_{i,k}^{real}$, where $f(s_{i,.}) > c_{i,k}^{real}$, thus $utility(i,k) > 0$. The fact that k is able to deviate from the mechanism and gain more is against the definition of truthful mechanisms. Consequently, the presence of the task k is impossible such that $f(s_{i,.}) > c_{i,k}^{real}$. Therefore, $f(s_{i,.}) \leq \min_j \{c_{i,j}^{real}\}$ is a necessary condition for BOSS to be truthful. □

As explained in Appendix B of the supplemental material, available in the online supplemental material, $f(s_{i,.})$ is an always-negative function. In other words, $f(s_{i,.})$ could be viewed as the penalty function for the liar winner. Nevertheless, instead of penalizing the liar winner, we could simply ban it from participating in the next auctions. Nevertheless, in a real world, there are many situations in which the resources might not be able to complete the execution of tasks exactly in the promised time. Thus, using a penalty function seems more practical. Finding an efficient penalty function is a practical problem related to various parameters of the system which is not in the scope of this work. However, we propose a customized penalty function with some discussion on its fairness in Appendix C of the supplemental material, available in the online supplemental material. In this context, we assume an abstract penalty function $f(s_{i,.})$ that satisfies the inequality mentioned in Lemma 1.

Theorem 1. *Strategy profile $s_{i,.} = \bigcup_{j\in[m]} (t_{i,j}^{real}, c_{i,j}^{real})$ is Nash equilibrium.*

Proof. We disprove that in each auction initiated by the auctioneer i every agent j cannot deviate from its strategy $s_{i,j} = (t_{i,j}^{real}, c_{i,j}^{real})$ unilaterally and gain more. Assume a new strategy profile $s'_{i,.} = \{s'_{i,j}, s_{i,-j}\}$, meaning that the agent j plays a new strategy $s'_{i,j} = (t'_{i,j}, c'_{i,j}) \neq (t_{i,j}^{real}, c_{i,j}^{real})$ and that the other strategies remain unchanged. We consider four possibilities for the new strategy $s'_{i,j} : t'_{i,j} > t_{i,j}^{real}$, $t'_{i,j} < t_{i,j}^{real}$, $c'_{i,j} > c_{i,j}^{real}$, and $c'_{i,j} < c_{i,j}^{real}$ which we investigate in the following in two situations: $j = w$ and $j \neq w$:

1. *Case 1:* $t'_{i,j} > t_{i,j}^{real}$ for $j = w$ not only decreases the chance of j to win, but also decreases the revenue of j. If $j \neq w$, this deviation ($t'_{i,j} > t_{i,j}^{real}$) has no impact on the revenue and the odds of winning for j. Thus, $t'_{i,j} > t_{i,j}^{real}$ is an irrational deviation for both winner and loser.
2. *Case 2:* $t'_{i,j} < t_{i,j}^{real}$ increases the odds of winning for $j \neq w$ and has no impact on odds of winning for $j = w$. Nevertheless, the winner will always be penalized in this situation and this deviation is irrational too.
3. *Case 3:* $c'_{i,j} > c_{i,j}^{real}$ for $j = w$ decreases the chance of j to win and has no effect on the revenue of j. In case of $j \neq w$, this deviation has no impact on the revenue and the odds of winning. Consequently, this deviation is also an irrational deviation for both winner and loser.
4. *Case 4:* $c'_{i,j} < c_{i,j}^{real}$ in case of $j = w$ is irrational since this strategy has no effect on the odds of winning and the revenue of j. For $j \neq w$, we have $t_{i,w}^{real} \cdot c_{i,w}^{real} < t_{i,j}^{real} \cdot c_{i,j}^{real}$. In this case, there is a possibility for j to change its strategy from $s_{i,j}$ to $s'_{i,j} = (t_{i,j}^{real}, c'_{i,j})$ such that $c'_{i,j} < c_{i,j}^{real}$ and win the game. When j wins the auction, the strategy of the old winner w will be the second smallest bid. In other words, the winner of the previous strategy will be x in the new strategy. Consequently, j will be paid the amount
$$pay(i,j) = \frac{t_{i,w}^{real} \cdot c_{i,w}^{real}}{t_{i,j}^{real}}.$$
Since
$$c_{i,j}^{real} > \frac{t_{i,w}^{real} \cdot c_{i,w}^{real}}{t_{i,j}^{real}},$$
this results in $utility(i,j) < 0$ according to (4). Consequently, this deviation is also irrational for all agents.

Since all aforementioned underbidding and overbidding cases are irrational for all resources, the strategy profile $s_{i,.} = \bigcup_{j\in[m]} (t_{i,j}^{real}, c_{i,j}^{real})$ is Nash equilibrium. In other words, the BOSS mechanism is truthful. □

6 Dynamic Workflow Scheduling

We propose in Algorithm 1 an extension to the BOSS mechanism for dynamic workflow scheduling. The proposed algorithm is a list scheduling heuristic that first orders the tasks in a list (lines 2-11) and then schedule them in the established order (lines 12-17). Finally, we pay the cost of using resources according to the proposed mechanism (line 18). The workflow tasks are ordered according to their bottom level (B-level), defined in graph theory as the longest path from each task to the $exit$ task, including the task itself. We calculate the rank of a task i according to the following recursive function:

$$rank(i) = \begin{cases} load(i) + \max_{j\in succ(i)} \{comm(i,j) + rank(j)\}, & \text{if } i \neq exit \\ load(i) & \text{if } i = exit, \end{cases} \tag{5}$$

where $load(i)$ represents the workload of task i and $comm(i,j)$ is the output of tasks i to j. Function $succ(i)$ returns the immediate successors of task i (see Section 3). The reason for using the makespan values for ordering the tasks is twofold: it is the structure-dependent objective (see Table 1) that preserves the precedence relationships in the ordered list and it usually has a direct impact on the cost since the longer tasks need more resource usage. For scheduling the ordered list of tasks, we assume a repetitive BOSS mechanism in which n tasks sequentially follow BOSS.

Preposition 1. *For scheduling n ordered tasks in the repetitive BOSS, the strategy profile $s = \bigcup_{i\in[n]\wedge j\in[m]} (t_{i,j}^{real}, c_{i,j}^{real})$ is Nash equilibrium.*

Proof. In Theorem 1, we prove that $s_{i,.} = \bigcup_{j\in[m]} (t_{i,j}^{real}, c_{i,j}^{real})$ is Nash equilibrium. Because $t_{i,j}^{real}$ is the completion time of task i on resource j, the already scheduled tasks on the resource j simply appear as the current load of resource j and are taken into consideration by resource j in the calculation of $t_{i,j}^{real}$. Since the costs of the previous executed tasks have no impact on the cost of the new coming tasks, there is no change in $c_{i,j}^{real}$ with respect to the previous executions. Consequently, s is Nash equilibrium based on Theorem 1. □

Although finding the equilibrium in a game is a valuable result, the mechanism is useless if is not able to do it in reasonable time. Since the mechanism is negotiation based, the communication complexity is critical too. We therefore analyze in the following the computation and communication complexities of the BOSS mechanism.

Algorithm 1: Workflow-tailored BOSS algorithm.

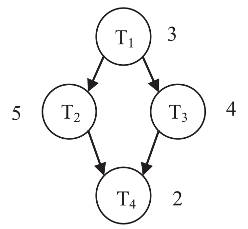
```
Input: The workflow application $DAG([n],[e])$
Output: The schedule of $DAG$ on commercial multi-Cloud $[m]$
begin
    $tasks \longleftarrow [n]$;        /* Assign the tasks to tasks list */
    $ranks \longleftarrow \emptyset$;              /* Set ranks list to empty */
    $tempTask \longleftarrow exit$;          /* Start with the exit task */
    while $tasks \neq \emptyset$ do
        $Add(ranks, CalculateRank(tempTask))$;      /* Add the
        rank of tempTask to ranks */
        $tempTask \longleftarrow Tail(tasks)$;      /* Assign the tail of
        tasks list to tempTask (the ranks are calculated
        in a bottom-up direction in the workflow) */
        $Remove(tasks, tempTask)$;    /* Remove tempTask from
        tasks list */
    end
    $tasks \longleftarrow [n]$;       /* Assign the tasks to tasks list */
    $Sort(tasks, ranks)$  /* Sort all tasks in descending ranks
    order */
    while $tasks \neq \emptyset$ do
        $auctioneer \longleftarrow Head(tasks)$;      /* Auctioneer is the
        first task in tasks list */
        $w \longleftarrow BOSS(auctioneer)$;  /* Start the BOSS auction
        by the auctioneer and select the winner as w */
        $Submit(auctioneer, w)$;        /* Submit the auctioneer
        task to the winner resource */
        $Remove(tasks, auctioneer)$;  /* Remove auctioneer from
        tasks list */
    end
    $Pay()$;                      /* pay the cost of winners */
end
```

Function CalculateRank(*task*).

```
Input: $task \in [n]$
Output: rank of $task$
begin
    if $task = exit$ then
        return $load(task)$
    else
        $ranks \longleftarrow \emptyset$;             /* Set ranks list to empty */
        $successors \longleftarrow succ(task)$;     /* Assign the successors
        list to immediate successors of task */
        while $successors \neq \emptyset$ do
            $j \longleftarrow head(successors)$;          /* Assign head of
            successors to task j */
            $Add(ranks, Comm(task, j) + CalculateRank(j))$;
            /* Assign the successors of task in successors
            list */
            $Remove(successors, j)$;      /* Remove task j from
            successors list */
        end
        return $\max(ranks) + load(task)$    /* Return sum of the
        workload of task and the maximum item of the
        ranks list */
    end
end
```

Preposition 2. *The time complexity of the repetitive BOSS mechanism for scheduling n tasks on m resources by strategy profile s is $O(n \cdot m)$.*

Proof. In each auction initiated by i, the main step is to find two resources w and x from the m possible choices. Obviously, the time complexity of this step is $O(m)$. Consequently, in n repetitive auctions, the final time complexity is $O(n \cdot m)$. □



(a) Workflow.

	R_1		R_2	
Task	*Time*	Cost	*Time*	*Cost*
T_1	4	3	2	5
T_2	6	6	4	7
T_3	6	6	4	7
T_4	2	2	1	3

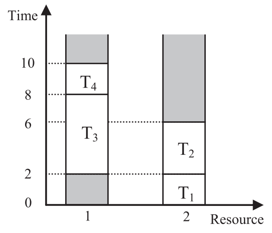
(b) Time and cost matrices on two resources.

Task	*Rank*
T_1	$2+5+3=10$
T_2	$2+5=7$
T_3	$2+4=6$
T_4	2

(c) Task ranks.

Task	*Strategy profile*	*Winner*	*Payment to winner*
T_1	$s_{1,1}=(4,3)$ $s_{1,2}=(2,5)$	R_2	$pay(1,2)=6$
T_2	$s_{2,1}=(8,6)$ $s_{2,2}=(6,7)$	R_2	$pay(2,2)=8$
T_3	$s_{3,1}=(8,6)$ $s_{3,2}=(10,7)$	R_1	$pay(3,1)=8.75$
T_4	$s_{4,1}=(10,2)$ $s_{4,2}=(9,3)$	R_1	$pay(4,1)=2.7$

(d) Workflow scheduling summary.



(e) Gantt chart.

Fig. 2. Example workflow.

Preposition 3. *The communication complexity of the repetitive BOSS mechanism to schedule n tasks on m resources by strategy profile s is $O(n \cdot m)$.*

Proof. In each auction, the auctioneer sends the auction initiation message to the agents comprising the workload and the required data transfer of the dependent tasks. Sending initiation messages to m agents has a communication complexity of $O(m)$. The bids submitted by the agents also have a communication complexity of $O(m)$. Therefore, the communication complexity for each individual auction is $O(m)$. Consequently, the communication complexity for scheduling n tasks is $O(n \cdot m)$. □

7 EXAMPLE

In the following, we present an illustrative example to better understand the proposed algorithm. Let us assume the workflow presented in Fig. 2a, where the number besides each task indicates the workload used for its rank calculation. For simplicity of the example, we assume no data transfers between tasks. The time and cost of two resources for executing the workflow tasks are shown in Fig. 2b. First, we calculate the ranks of the workflow tasks, as shown in Fig. 2c. Based on the descending order of the ranks, the tasks should be scheduled according to the ordered list: $B = \{T_1, T_2, T_3, T_4\}$.

The task T_1 starts the first auction. Its start time is assumed to be zero. The strategy profile is $s_{1,.} = \{s_{1,1} = (4, 3), s_{1,2} = (2,5)\}$. The winner w is the resource R_2, because

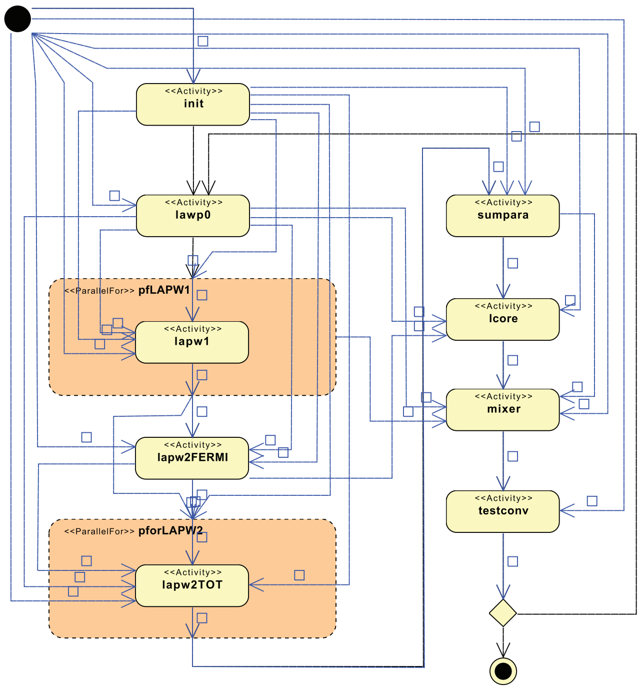


Fig. 3. WIEN2k ASKALON implementation.

$4 \cdot 3 > 2 \cdot 5$ and the resource with the second smallest bid x is the resource 1. The payment value for w is calculated by the mechanism as: $pay(1,2) = \frac{4 \cdot 3}{2} = 6$.

In the second auction, the resources must take the completion time of T_1 into consideration because T_2 is dependent on it. Since T_1 is executed on the resource 2, $t^{real}_{1,2} = 2$. Consequently, the completion time of T_2 on R_1 is calculated as: $t^{real}_{2,1} = 2 + 6 = 8$ and $t^{real}_{2,2} = 2 + 4 = 6$. Thus, the strategy profile in the second auction is $s_{2,.} = \{s_{2,1} = (8, 6), s_{2,2} = (6, 7)\}$. In this case, the winner w is resource R_2 and the resource with x is R_1 because $8 \cdot 6 > 6 \cdot 7$. The payment value for w is calculated by the mechanism as: $pay(2,2) = \frac{8 \cdot 6}{6} = 8$.

Next, since T_3 is dependent only on T_1 which has already finished, it can immediately start the third auction after the second. The resource R_1 is able to start the task at time 2, while the R_2 is already executing T_2 and can start T_3 at time 6. The real completion time for two resources are: $t^{real}_{3,1} = 2 + 6 = 8$ and $t^{real}_{3,2} = 6 + 4 = 10$. The strategy profile is: $s_{3,.} = \{s_{3,1} = (8, 6), s_{3,2} = (10, 7)\}$. Consequently, the winner w is R_1 and x is R_2, as $8 \cdot 6 < 10 \cdot 7$. The payment value for w is calculated as: $pay(3,1) = \frac{10 \cdot 7}{8} = 8.75$.

Last, T_4 is dependent on both T_2 and T_3, thus the auction will start at time 8 when both tasks are completed. The real completion time for the two resources are: $t^{real}_{4,1} = 8 + 2 = 10$ and $t^{real}_{4,2} = 8 + 1 = 9$. Thus, the strategy profile is: $s_{4,.} = \{s_{4,1} = (10, 2), s_{4,2} = (9, 3)\}$. Consequently, the winner w is R_1 and x is R_2, as $10 \cdot 2 < 9 \cdot 3$. The payment value for w calculated by the mechanism is: $pay(4,1) = \frac{9 \cdot 3}{10} = 2.7$.

The final Gantt chart depicted in Fig. 2e and the schedule is again summarized in Fig. 2d. Consequently, $sched = \{(T_1, R_2), (T_2, R_2), (T_3, R_1), (T_4, R_1)\}$, the workflow makespan is 10 and the total cost is the payment sum for all tasks: $6 + 8 + 8.75 + 2.7 = 25.45$. As expected, the algorithm enforces a truthful mechanism between the resource providers and also the mechanism establishes the payments that cover costs of the providers. We investigate in Section 8 the optimality of the proposed solutions compared to approximated Pareto set.

TABLE 2
Resource Cost Summary Based on Amazon Prices

Resource	*Cost*($)
Computation	0.08–1.00 (per hour)
Network (In/Out)	0.01 (per GB)
Storage	0.15 (per TB per hour)

8 Evaluation

The most popular metric to measure the inefficiency of an equilibrium in game theory is the price of anarchy, defined as the ratio between the worst objective function value of an equilibrium of the game and the optimal outcome [1]. Since our problem is a multiobjective optimization, there is no single optimal solution but a Pareto set of nondominated solutions. Consequently, a theoretical analysis of the price of anarchy is impossible and we must rely on experimental analysis.

8.1 Experimental Setup

We performed the experiments with two types of workflows: a real-world workflow called WIEN2k [32] and a set of synthetic randomly generated workflows implemented using the GridSim simulator [33]. In the following, we only present a part of the results, while the rest of results for the randomly generated workflows are covered in Appendix D of the supplemental material, available in the online supplemental material.

WIEN2k is a material science workflow for performing electronic structure calculations of solids using density functional theory. As shown in Fig. 3, WIEN2k has a balanced shape containing two parallel sections with sequential synchronization tasks in between. We extracted task workloads and data sizes to be transferred between the tasks from existing historical data of running this workflow using the ASKALON [34] environment in the Austrian Grid (http://www.austriangrid.at). ASKALON is part of the SHIWA European project (http://www.shiwa-workflow.eu/) targeting workflow interoperability having P-GRADE, MOTEUR, and Triana as additional pilot partners, which will guarantee that our implementation is not isolated, but portable across different workflow languages and systems.

To simulate the resource environment using a broader variety of resources than available in the Austrian Grid (which consists of a number of homogeneous parallel computers), we generated the resource speeds using a uniform distribution from 200 to 1,200 MIPS with an increment of 50. We estimated the costs of resources using the real Amazon Web Services prices (http://aws.amazon.com), as summarized in Table 2. For a fair comparison, we use in all experiments the pricing model proposed in BOSS. We assume that the resources are fully connected by a 10 GB Ethernet network. Based on the model, the bandwidth has no impact on the quality of results, since

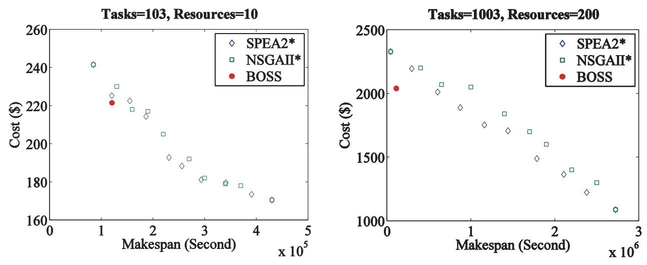


Fig. 4. WIEN2k sample objective space.

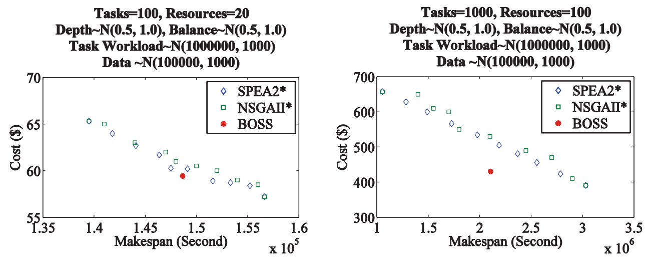


Fig. 5. Sample objective space for randomly generated workflows.

the providers are aware of the network specifications and consider it in their bids.

8.2 Experimental Results

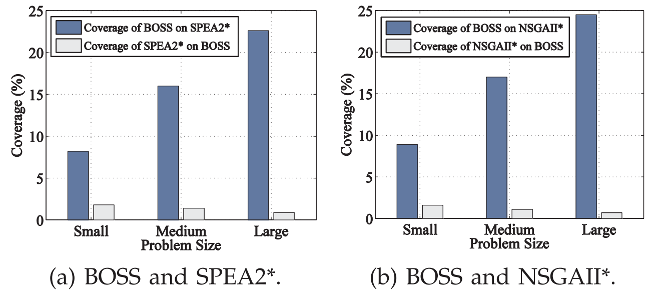
To compare the solution of BOSS with the Pareto-optimal solutions, we use the Pareto sets estimated by two multiobjective evolutionary algorithms: SPEA2* and NSGAII* [26] that are based on the general SPEA2 [35] and NSGAII [36] genetic algorithms. SPEA2 and NSGAII are categorized as a-posteriori multiobjective optimization techniques [23], because they do not assume any a-priori knowledge about the Pareto-set solutions. A posteriori multiobjective optimization techniques first generate (or approximate) the whole Pareto set and then select the adequate solution based on the user's requirements. In the implementation, we defined two fitness functions for the time and cost. Each solution is modeled by two sequences: task assignments and execution orders. Afterwards, the algorithms follow the general SPEA2 and NSGAII algorithms. Our implementation extends the jMetal library [37] and uses the default setting proposed in [26] (e.g., 10 Pareto-set solutions).

For a deeper understanding of the objective space, Figs. 4 and 5 present the results of running two WIEN2k and two randomly generated instances of different sizes using BOSS, SPEA2*, and NSGAII* schedulers. We can observe that BOSS algorithm generates nondominated solutions compared to the Pareto frontiers approximated by the SPEA2* and NSGAII* and in some cases, the solution of BOSS even dominates them. Although theoretically no solution is able to dominate the Pareto set, this is possible in our case since the Pareto sets generated by the two evolutionary algorithms are only estimations.

Nevertheless, we observed that in a few experiments the solutions of BOSS are dominated by the Pareto sets generated by SPEA2* and NSGAII*. Therefore, we need to investigate that how often this situation happens using the coverage metric inspired from multiobjective optimization [38]. The coverage of a set A on a set B indicates how many members of B are dominated by the members of A. We therefore repeated the experiments for 50 WIEN2k and

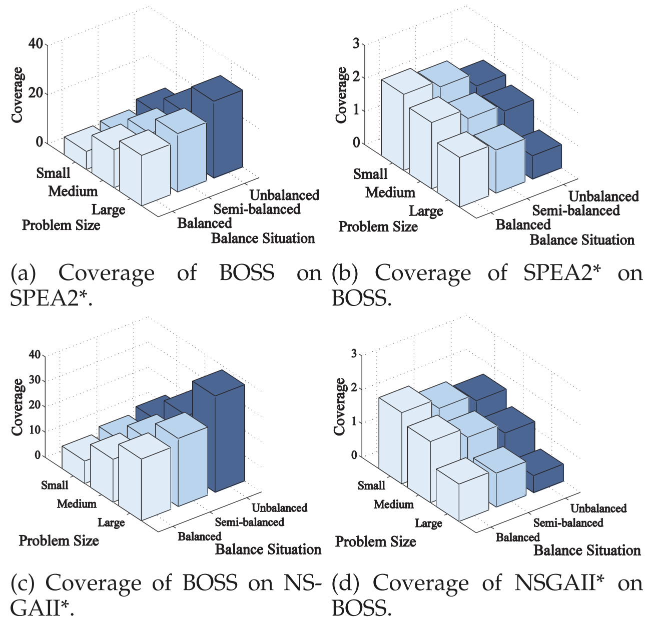
TABLE 3
Problem Size Classification

Small	*Medium*	*Large*
$10 \le n \le 200$	$500 \le n \le 1000$	$2000 \le n \le 3000$
$10 \le m \le 50$	$100 \le m \le 500$	$800 \le m \le 1200$



(a) BOSS and SPEA2*. (b) BOSS and NSGAII*.

Fig. 6. Average coverage for WIEN2k.



(a) Coverage of BOSS on SPEA2*. (b) Coverage of SPEA2* on BOSS.

(c) Coverage of BOSS on NSGAII*. (d) Coverage of NSGAII* on BOSS.

Fig. 7. Average coverage for randomly generated workflows.

1,000 randomly generated workflows categorized in three classes: small, medium, and large the problem sizes, as shown in Table 3, where n is the number of tasks and m the number of resources.

Figs. 6 and 7 prove that the average coverage of BOSS is considerably larger than SPEA2* and NSGAII*, in other words, the quality of the solutions of BOSS is better. We observe that for large problem sizes, approximately 23 percent of the solutions of SPEA2* are dominated by the solutions of BOSS, while only around 1 percent of the solutions of BOSS are dominated by SPEA2*. Figs. 6 and 7 further demonstrate that by increasing the problem size, the coverage of BOSS is increasing compared to SPEA2* and NSGAII*. Moreover, Fig. 7 shows that when the shape of the workflows is more imbalanced, the coverage of BOSS gets better. Another observation is that the coverage values in Fig. 6 are approximately equal to the coverage of balanced workflows in Fig. 7 which is because of the balanced of shape of WIEN2k workflows.

TABLE 4
The Average Execution Time for WIEN2k

	Small	Medium	Large
BOSS	0.4 s	4.8 s	7.4 s
SPEA2*	364 s	17826 s	26744 s
NSGAII*	437 s	19479 s	29662 s

The last important observation in our experiments is that BOSS is a much faster algorithm than SPEA2* and NSGAII*, as illustrated in Table 4 for WIEN2k workflows and in Fig. 8 for randomly generated workflows. We use in Fig. 8 the average between SPEA2* and NSGAII*, as both algorithms presented in our experiments approximately equal execution times.

Although both MOEAs generate 10 solutions in each execution compared to one solution in BOSS, the results show considerable differences in execution times (results in Fig. 8b are scaled by a factor of 10^4). The reason for this high difference is the second degree polynomial time complexity of BOSS, while the MOEAs are complex evolutionary algorithms.

9 Conclusion

In this paper, we first proposed a pricing model and truthful mechanism for dynamic scheduling of a single task in commercial multicloud environment. We used the mechanism for scheduling of scientific workflows with respect to optimization of two objectives, makespan and monetary cost, and proved the truthfulness of the mechanism theoretically. Since studying the price of anarchy in a multiobjective problem is not practical, we experimentally investigated it and discovered that the generated solutions of our proposed mechanism are approximately Pareto optimal. Moreover, the results present a better coverage compared to the approximated Pareto set generated by two classical MOEAs: SPEA2 and NSGAII. Finally, we analyzed the complexity of our approach and experimentally observed much smaller execution times compared to the investigated MOEAs. As a next step, we intend to study the impact of the Bayesian-Nash equilibrium [1] on having the old distribution of private information of agents as a public knowledge.

Acknowledgments

This work was funded by the Austrian Science Fund (FWF), project TRP 72-N23, and by the Standortagentur Tirol, project RainCloud.

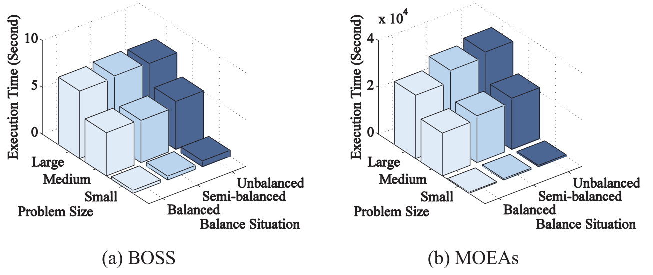


Fig. 8. Average execution time of randomly generated workflows.

References

[1] N. Nisan, T. Roughgarden, E. Tardos, and V. Vazirani, *Algorithmic Game Theory.* Cambridge Univ. Press, 2007.

[2] D. Fudenberg and J. Tirole, *A Standard Text for Graduate Game Theory.* MIT Press, 1991.

[3] N. Nisan and A. Ronen, "Algorithmic Mechanism Design," *Proc. Symp. Theory of Computing (STOC),* pp. 129-140, 1999.

[4] A. Archer and E. Tardos, "Truthful Mechanisms for One-Parameter Agents," *Proc. 42nd IEEE Symp. Foundations of Computer Science,* pp. 482-491, 2001.

[5] H.M. Fard, R. Prodan, G. Moser, and T. Fahringer, "A Bi-Criteria Truthful Mechanism for Scheduling of Workflows in Clouds," *Proc. Third IEEE Int'l Conf. Cloud Computing Technology and Science (Cloudcom '11),* pp. 599-605, 2011.

[6] K. Keahey, M. Tsugawa, A. Matsunaga, and J. Fortes, "Sky Computing," *IEEE Internet Computing,* vol. 13, no. 5, pp. 43-51, Sept./Oct. 2009.

[7] R. Buyya, R. Ranjan, and R.N. Calheiros, "InterCloud: Utility-Oriented Federation of Cloud Computing Environments for Scaling of Application Services!" *Proc. 10th Int'l Conf. Algorithms and Architectures for Parallel Processing (ICA3PP),* 2010.

[8] S.K. Nair, S. Porwal, T. Dimitrakos, A.J. Ferrer, J. Tordsson, T. Sharif, C. Sheridan, M. Rajarajan, and A.U. Khan, "Towards Secure Cloud Bursting, Brokerage and Aggregation," *Proc. Eighth IEEE European Conf. Web Services (ECOWS '10),* pp. 189-196, 2010.

[9] I. Houidi, M. Mechtri, W. Louati, and D. Zeghlache, "Cloud Service Delivery across Multiple Cloud Platforms," *Proc. IEEE Int'l Conf. Services Computing (SCC),* pp. 741-742, July 2011.

[10] R. Buyya, D. Abramson, and S. Venugopal, "The Grid Economy," *Proc. IEEE,* vol. 93, no. 3, pp. 698-714, Mar. 2005.

[11] M.P. Wellman, W.E. Walsh, P.R. Wurman, and J.K. MacKie-Mason, "Auction Protocols for Decentralized Scheduling," *Games & Economic Behavior,* vol. 35, nos. 1/2, pp. 271-303, 2001.

[12] Y.-K. Kwok, K. Hwang, and S. Song, "Selfish grids: Gametheoretic Modeling and Nas/Psa Benchmark Evaluation," *IEEE Trans. Parallel and Distributed Systems,* vol. 18, no. 5, pp. 621-636, May 2007.

[13] D. Grosu and A.T. Chronopoulos, "Algorithmic Mechanism Design for Load Balancing in Distributed Systems," *IEEE Trans. Systems, Man and Cybernetics, Part B,* vol. 34, no. 1, pp. 77-84, Feb. 2004.

[14] A. Danak and S. Mannor, "Bidding Efficiently in Repeated Auctions with Entry and Observation Costs," *Proc. IEEE Int'l Conf. Game Theory for Networks,* pp. 299-307, May 2009.

[15] A. Danak and S. Mannor, "Efficient Bidding in Dynamic Grid Markets," *IEEE Trans. Parallel and Distributed Systems,* vol. 22, no. 9, pp. 1483-1496, Sept. 2011.

[16] N. Garg, D. Grosu, and V. Chaudhary, "Antisocial Behavior of Agents in Scheduling Mechanisms," *IEEE Trans. System, Man, Cybernetics, Part A: Systems and Humans,* vol. 37, no. 6, pp. 946-954, Nov. 2007.

[17] N. Andelman, Y. Azar, and M. Sorani, "Truthful Approximation Mechanisms for Scheduling Selfish Related Machines," *Proc. 22nd Symp. Theoretical Aspects of Computer Science,* 2005.

[18] H. Izakian, A. Abraham, and B.T. Ladani, "An Auction Method for Resource Allocation in Computational Grids," *Future Generation Computer Systems,* vol. 26, no. 2, pp. 228-235, Feb. 2010.

[19] K. Rzadca, D. Trystram, and A. Wierzbicki, "Fair Game-Theoretic Resource Management in Dedicated Grids," *Proc. Int'l Symp. Cluster Computing and the Grid,* pp. 343-350, 2007.

[20] R. Prodan, M. Wieczorek, and H.M. Fard, "Double Auction-Based Scheduling of Scientific Applications in Distributed Grid and Cloud Environments," *J. Grid Computing,* vol. 9, no. 4, pp. 531-548, 2011.

[21] N. Jain, I. Menache, J. Naor, and J. Yaniv, "A Truthful Mechanism for Value-Based Scheduling in Cloud Computing," *Proc. Int'l Conf. Algorithmic Game Theory (SAGT),* pp. 178-189, 2011.

[22] H. Topcuouglu, S. Hariri, and M.-y. Wu, "Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing," *IEEE Trans. Parallel and Distributed Systems,* vol. 13, no. 3, pp. 260-274, Mar. 2002.

[23] R.T. Marler and J.S. Arora, "Survey of Multi-Objective Optimization Methods in Engineering," *Structural and Multidisciplinary Optimization,* vol. 26, no. 6, pp. 369-395, Apr. 2004.

[24] R. Sakellariou, H. Zhao, E. Tsiakkouri, and M. Dikaiakos, "Scheduling Workflows with Budget Constraints," *Proc. CoreGRID Workshop Integrated Research in GRID Computing,* S. Gorlatch and M. Danelutto, eds., pp. 189-202, 2007.

[25] J. Yu, R. Buyya, and C.K. Tham, "Cost-Based Scheduling of Scientific Workflow Applications on Utility Grids," *Proc. First Int'l Conf. e-Science and Grid Computing,* pp. 140-147, 2005.

[26] J. Yu, M. Kirley, and R. Buyya, "Multi-Objective Planning for Workflow Execution on Grids," *Proc. Eighth Int'l Conf. Grid Computing,* pp. 10-17, 2007.

[27] H.M. Fard, R. Prodan, J.J.D. Barrionuevo, and T. Fahringer, "A Multi-Objective Approach for Workflow Scheduling in Heterogeneous Computing Environments," *Proc. 12th IEEE/ACM Int'l Symp. Cluster, Cloud and Grid Computing (CCGrid '12),* pp. 300-309, 2012.

[28] M.A. Salehi and R. Buyya, "Adapting Market-Oriented Scheduling Policies for Cloud Computing," *Proc. 10th Int'l Conf. Algorithms and Architectures for Parallel Processing (ICA3PP),* pp. 351-362, May 2010.

[29] M. Mao and M. Humphrey, "Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows," *Proc. Int'l Conf. High Performance Computing, Networking, Storage and Analysis (SC '11),* 2011.

[30] M. Armbrust, A. Fox, R. Griffith, A. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the Clouds: A Berkeley View of Cloud Computing," Technical Report No. UCB/EECS-2009-28, Univ. of California at Berkley, USA, Feb. 2009.

[31] M. Wieczorek, A. Hoheisel, and R. Prodan, "Towards a General Model of the Multi-Criteria Workflow Scheduling on the Grid," *Future Generation Computer Systems,* vol. 25, no. 3, pp. 237-256, Mar. 2009.

[32] P. Blaha, K. Schwarz, G. Madsen, D. Kvasnicka, and J. Luitz, "WIEN2k: An Augmented Plane Wave plus Local Orbitals Program for Calculating Crystal Properties," Inst. of Physical and Theoretical Chemistry, TU Vienna, 2001.

[33] A. Sulistio, U. Cibej, S. Venugopal, B. Robic, and R. Buyya, "A Toolkit for Modelling and Simulating Data Grids: An Extension to GridSim," *J. Concurrency and Computation: Practice and Experience,* vol. 20, no. 13, pp. 1591-1609, Sept. 2008.

[34] T. Fahringer, R. Prodan, R. Duan, F. Nerieri, S. Podlipnig, J. Qin, M. Siddiqui, H.L. Truong, A. Villazon, and M. Wieczorek, "Askalon: A Grid Application Development and Computing Environment," *Proc. sixth IEEE/ACM Int'l Conf. Grid Computing,* pp. 122-131, 2005.

[35] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization" *Proc. Int'l Conf. Evolutionary Methods for Design, Optimisation and Control with Application to Industrial Problems (EUROGEN '01),* pp. 95-100, 2002.

[36] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II," *IEEE Trans. Evolutionary Computation,* vol. 6, no. 2, pp. 182-197, Apr. 2002.

[37] J.J. Durillo, A.J. Nebro, and E. Alba, "The jMetal Framework for Multi-Objective Optimization: Design and Architecture," *Proc. IEEE Congress on Evolutionary Computation (CEC '10),* pp. 4138-4325, July 2010.

[38] E. Zitzler and L. Thiele, "Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach," *IEEE Trans. Evolutionary Computation,* vol. 3, no. 4, pp. 257-271, Nov. 1999.

Hamid Mohammadi Fard received the bachelor's and master's degrees in computer science from Iran. Between 1998 and 2009, he worked in Iran's telecommunication industry. Since 2009, he has been currently working toward the PhD degree at the Institute of Computer Science, University of Innsbruck, Austria, where he is also a researcher.

Radu Prodan received the PhD degree from the Vienna University of Technology, Austria, in 2004. He is currently an associate professor at the Institute of Computer Science, University of Innsbruck, Austria. His research in the area of parallel and distributed systems comprises programming methods, compiler technology, performance analysis and scheduling. He is the author of more than 70 journal and conference papers and one book. He was the recipient of the IEEE best paper award.

Thomas Fahringer is a full professor of computer science at the Institute of Computer Science, University of Innsbruck. His research interests are programming languages, performance tools, debuggers, compiler analysis and program optimization for parallel and distributed systems. He participated in numerous EU-funded, international and national projects. He is the author of more than 160 papers including 30 journal articles and three books. He was the recipient of three IEEE/ACM best paper awards.