Int. J. Appl. Math. Comput. Sci., 2011, Vol. 21, No. 2, 243–257
DOI: 10.2478/v10006-011-0018-x

amcs

MODERN APPROACHES TO MODELING USER REQUIREMENTS ON RESOURCE AND TASK ALLOCATION IN HIERARCHICAL COMPUTATIONAL GRIDS

JOANNA KOŁODZIEJ *, FATOS XHAFA **

* Department of Mathematics and Computer Science
University of Bielsko-Biała, ul. Willowa 2, 43–309 Bielsko-Biała, Poland
e-mail: jkolodziej@ath.bielsko.pl

**Department of Languages and Informatics Systems
Polytechnic University of Catalonia, Campus Nord, Ed. Omega, C/Jordi Girona 1–3, 08034 Barcelona, Spain
e-mail: fatos@lsi.upc.edu

Tasks scheduling and resource allocation are among crucial issues in any large scale distributed system, including Computational Grids (CGs). These issues are commonly investigated using traditional computational models and resolution methods that yield near-optimal scheduling strategies. One drawback of such approaches is that they cannot effectively tackle the complex nature of CGs. On the one hand, such systems account for many administrative domains with their own access policies, user privileges, etc. On the other, CGs have hierarchical nature and therefore any computational model should be able to effectively express the hierarchical architecture in the optimization model. Recently, researchers have been investigating the use of game theory for modeling user requirements regarding task and resource allocation in grid scheduling problems. In this paper we present two general non-cooperative game approaches, namely, the *symmetric non-zero sum* game and the *asymmetric Stackelberg* game for modeling grid user behavior defined as user requirements. In our game-theoretic approaches we are able to cast new requirements arising in allocation problems, such as asymmetric users relations, security and reliability restrictions in CGs. For solving the games, we designed and implemented GA-based hybrid schedulers for approximating the equilibrium points for both games. The proposed hybrid resolution methods are experimentally evaluated through the grid simulator under heterogeneity, and large-scale and dynamics conditions. The relative performance of the schedulers is measured in terms of the makespan and flowtime metrics. The experimental analysis showed high efficiency of meta-heuristics in solving the game-based models, especially in the case of an additional cost of secure task scheduling to be paid by the users.

Keywords: computational grids, scheduling, non-cooperative games, user behavior, security, meta-heuristics.

1. Introduction

In recent years, grid computing systems have become popular for the resolution of large-scale complex problems in science, engineering, and industry. Such systems enable the virtualization of a wide range of resources; thus, various types of Grid systems, such as Computational Grids (CGs), desktop, enterprise, data grids, etc., can be design and studied. Among them, computational grids were primarily concerned with the development of efficient resolution methods for solving complex problems of high performance computing and more generally, problems in the domain of e-science.

Since CGs focus on high system scalability, and thus on large-scale resource sharing, an efficient resource management system is crucial for the efficacy of the system. However, providing effective scheduling and resource allocation mechanisms in CGs is a complex undertaking due to their scale and the fact that resource owners and consumers may have different goals, preferences, and policies. Differently from traditional distributed computing systems, in which users and owners of the computational resources usually belong to the same administrative domain, in CGs the security and reliability of the resources are among crucial criteria in scheduling problems. Thus, an important research issue in this domain is to achieve efficient assignment of tasks to trustful resources.

The approaches proposed in the literature so far have

shown limitations in effectively translating the complex nature of CGs and may not overcome some limitations, like different access rights of grid users, conflicting objectives of users and administrative owners, different local resource access policies or security barriers.

Heuristic-based approaches can be very efficient in solving the traditional job-shop problems (Mesghouni *et al.*, 2004), but they usually just partially cover the requirements of grid-enabled applications. Recently, researchers have been investigating the usefulness of game theory in solving the task and resource allocation problems in CGs. Using game-theoretic models enables including more requirements and features into the computational optimization model for the problem. Meta-heuristics can then be used for solving the game to more effectively tackle the resolution of the resulting computationally hard problem of finding equilibrium points of the resulting games.

In this paper, we present two general non-cooperative game models, namely, the *symmetric non-zero sum* game and the *asymmetric Stackelberg* game for modeling grid user behavior. We show that, in our game-theoretic approach, we are able to cast new requirements of allocation problems, such as symmetric and asymmetric user relations and security and reliability restrictions in CGs. We use GA-based hybrid algorithms for approximating the equilibrium points for both games. The proposed hybrid resolution methods are experimentally evaluated through the grid simulator under heterogeneity, and large-scale and dynamics scenarios. The relative performance of the grid schedulers is measured by the makespan and flowtime metrics. We also survey the most relevant research proposals in the literature for using economical game-based models for real-life resource allocation problems.

The organization of this paper is as follows. In Section 3, we introduce preliminary concepts on grid architecture, scheduling and resource allocation and game-theoretical models. In Section 4, we define the general game models for grid scheduling, and then we present the concepts of two non-cooperative symmetric and asymmetric games. Resolution meta-heuristic methods for solving such games are defined in Section 5. The experimental evaluation of the proposed hybrid meta-heuristics is presented in Section 6. In Section 7 we survey the most relevant market-based approaches with game-based models for real-life resource allocation problems. We end the paper in Section 8 with some conclusions.

2. Related work

Effective resource and task allocation in grid systems under security and resource reliability conditions has attracted recently a lot of attention in the literature. There have been proposed several extensions and enhancements of the grid security infrastructure for the verification of the security conditions and resource availability (Lin *et al.*, 2004). An architectural framework for access control to grid resources can be found in the work of Laccetti and Schmidb (2007), where the extension of the authentication and authorization features the OS layer is proposed. Hwang and Kesselman (2003) proposed a failure detection service and a failure handling mechanism, which enable the detection of both task failures and user secure requirements without the need for updating the grid protocol and the local policy at the Grid node.

Brandic *et al.* (2006) addressed the security aspect of a Quality of Service (QoS) requirement defined by grid users at workflow specification time. The authors propose the location affinity model, in which the user for security reasons may express the location affinity regarding grid resources where certain workflow of tasks may be executed.

The integration of security and resource reliability as joint scheduling criteria requires fast, scalable and effective resolution methods to solve the multi-objective scheduling problem in CGs. Heuristic methods, due to their effectiveness and robustness, have been successfully applied to solve scheduling problems in the dynamic grid environment. In Section 5.1, we present a simple classification of the most popular meta-heuristic grid schedulers. Many of them can be effectively applied for solving grid users' decision process, modeled by using game theory, under various conditions and constraints, including time deadlines and budget constraints, and security and machines reliability requirements.

An interesting application of heuristic methods in the non-cooperative game of grid users in the commodity market model is presented by Garg *et al.* (2009). The authors defined two heuristics, namely, Min-Min Cost Time Trade-off (MinCTT) and Max-Min Cost Time Trade-off (Max-CTT) algorithms, for jointly optimizing cost and execution time of user application in utility grids. Both approaches are based on min-min and max-min methods, which were adapted to user-application time slot pairs. This approach is an example of grid users' game, in which task execution and resource utilization costs are defined as bi-objective players' cost functions.

Song *et al.* (2005; 2006) considered risky and insecure conditions in online scheduling in grids caused by software vulnerability and distrusted resources. They applied a game-theoretical model introduced by Kwok *et al.* (2007) for modeling resource owners selfish behavior in the hierarchical grid structure. A Space Time Genetic Algorithm (STGA) was implemented as the main mechanism of three risk-resilient meta-heuristics, named as risky, preemptive and replicated STG algorithms. The individuals in populations evaluated in the STGA framework are encoded as messy chromosomes, which are expressed as a list of ordered pairs (job ID, site ID). The length of each chromosome can vary during the scheduling in the advent of task failures. Furthermore, a gene's value may

be over-specified, i.e., it may appear more than once in a chromosome with different values. The "cut and splice" operator has been used to replace the crossover operation.

The results presented by Song *et al.* (2006) were extended by Wu and Sun (2010) by considering the heterogeneity of the fault-tolerance mechanism in security-assured grid job scheduling. The authors defined four types of GA-based online schedulers for the simulation of some fault-tolerance mechanisms, including job retry, job migration (with and without checkpointing) and job replication mechanisms.

Our work differs from all of the above in the following aspects:

(i) It provides a new hierarchical grid resource management system, in which the resource provider plays an additional role of the trust manager.

(ii) It proposes various game-theoretical resource allocation techniques that fit well to the defined hierarchical infrastructure and do not require any special synchronization procedures.

(iii) It proposes efficient, fast and scalable resolution methods for solving large-scale complex decision problems of grid users.

3. Preliminaries

3.1. Hierarchical computational grid. The computational grid, hierarchical by nature, is usually modeled as a multi-level system, which allows efficient management of geographically distributed resources and tasks scheduling under various criteria, including security and resource reliability requirements. The model can be seen as a hybridization of centralized and decentralized modules. In the *centralized module*, there is a central authority (meta-scheduler or meta-broker), who has complete knowledge of the system by monitoring the resources and interacts with local job dispatchers in order to define optimal schedules. In the *decentralized module*, the local schedulers interact with each other to manage the task pool. They have the knowledge about the resource clusters, but they cannot monitor the whole system.

The hierarchical model addresses scalability and fault-tolerance issues, but it does not provide site-autonomy. The hierarchy in such a system usually consists of two or three levels. A Meta-Broker (MB) model is an example of the hierarchical bi-level grid system. In this model, grid users submit their tasks/applications to the meta-broker, which uses also the information supplied by the resource owners to match the users' tasks to appropriate resources. In recent approaches, security and resource reliability are defined as additional scheduling criteria. The MB structure can be then adapted for designing the security-assured grid system (Kołodziej and Xhafa, 2010).

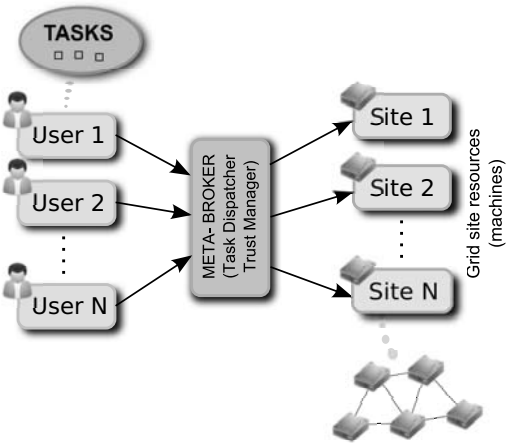

Fig. 1. Model of a security-aware grid system.

In Fig. 1 we present a simple example of a security MB model restricted to a grid site.

In this model the meta-broker plays additionally the role of a *trust manager*, who is responsible for the verification of a *secure-assurance condition* for any task-machine matching. He controls resource allocation and communication between grid users and service resource owners. It is assumed that the meta-broker is a central authority, who has access to all resources distributed in a wide grid area network. The resources inside each grid site are connected with each other under a local network structure.

Security awareness can be extended to higher level structures. Kwok *et al.* (2007) presented a hierarchical grid model with three main levels: the global, the inter-site and the intra-site level. At the intra-site level, there is a federation of autonomous resources. The information about computational capacities of the machines is sent by the resource owners to the local job dispatcher, who defines the "grid site reputation index" and sends it to the global scheduler. The participating grid sites (machine clusters) form another federation at the inter-site level. The global scheduler performs task scheduling according to a certain scheduling algorithm. He just uses the "site reputation indexes" supplied by the grid sites as inputs to the scheduling algorithm.

The grid hierarchical systems presented above can be easily transformed to game-theoretical models in order to push the concept of CGs into mainstream computing (market-based approaches) and to efficiently solve the scheduling problem under various criteria defined jointly as a scheduling objective.

3.2. Independent job batch scheduling problem. Unlike traditional distributed systems, in CGs, users and distributed resources belong to different autonomous domains. Thus, having full control over grid resources is in practice impossible. In this case, a fundamental problem

is the difficulty in transparent access to the resources from users of different administrative domains. There are also some special local usage policies specified by resource owners, which should be taken into account during the scheduling computation process. While grid users require assurance on the level, type, and quality of service provided by the resources, resource owners are usually concerned about maintaining local control on how resources are being utilized. Thus, an effective mapping of computational tasks or data transfers to resources that meet the requirements (cost, performance, security and other service metrics) remains challenging.

Due to the different demands imposed by grid-enabled applications, scheduling can be defined as a family of global optimization problems. Its complexity comes from the number of objectives to optimize (single vs. multi-objective), the type of environment (static vs. dynamic), the processing mode (immediate vs. batch), tasks interrelations, etc. Among several versions of scheduling problems highlighted by Xhafa and Abraham (2010), independent job batch scheduling is one of the simplest and fundamental scheduling problems. It usually arises in data intensive computing such as data mining and massive processing applications. For this class of applications, the batch mode is appropriate, given that such an application manage voluminous data (transfer and replication) and can run periodically. In such a case, jobs or applications are grouped in *batches* and scheduled as a group.

The independent scheduling problem can be formalized using the Expected Time To Compute (ETC) matrix model (Ali *et al.*, 2000). In this model, the following input data have to be specified:

- the number of independent tasks to be allocated to Grid resources in non-preemptive mode,
- the number of machines candidates to participate in the allocation of tasks,
- the workload (in millions of instructions) of each task,
- the computing capacity of each machine (in Mips),
- the ready times indicating when machines will have finished the previously assigned tasks,
- the ETC matrix of size $nb\_tasks \times nb\_machines$, where $ETC[j][m]$ is the value of the expected time to compute task j in machine m.

4. Game-theoretic models for scheduling and resource management

Game theory plays an important role in computer science, where it is used as a means of modeling interactive computations or multi-agent systems. Nowadays, Internet computing is seen as a new domain of applications of game theory, which, in combination with economic theory, can develop algorithms for finding equilibria in computational markets, computational auctions, grid and P2P systems as well as security and information markets. An important challenge in using game-theoretic models for grid scheduling and resource management is the large size scale of the Grid system and the fact that resources cross different administrative domains. The basic assumptions that underlie game-based models are that game players are rational and pursue well-defined objectives (cost or pay-off functions), and they take into account their knowledge or expectations of other players' behavior.

There are three main types of game models applicable to scheduling and allocation in CGs:

- **Non-cooperative game,** where the players act independently of each other. This model is based on the premise about the users' behavior in a realistic grid, in which cooperation is difficult to happen on a large scale and grid users submit their tasks independently. Also the resource owners act selfishly in order to maximize resource utilization and to execute tasks from local users.
- **Cooperative game,** in which players can form a coalition to plan their actions in advance. This model is useful for intra-site grid negotiations, where the local job dispatchers can define the joint "execution capabilities" parameters for the clusters of the grid sites and declare them to the global scheduler.
- **Semi-cooperative game,** where each player can choose (randomly) another player for cooperation. This game is usually proposed as a multi-round auction to incorporate task rescheduling.

The solution of each of those games is an equilibrium state, in which each player holds correct expectations concerning the other players' behavior (see the work of Khan and Ahmad (2006) for detailed analysis).

One of the main benefits of game-theoretical scheduling and resource management in CGs is that they enable a highly scalable system since decision-making process is distributed across all grid users and resource owners. Due to the large scale nature of grid systems, the non-cooperative game is a potential model for integrating security and resource reliability requirements in grid scheduling. We can consider two different general scenarios of the non-cooperative games: *symmetric* and *asymmetric* games.

4.1. Symmetric game. Let us denote by N the number of users (players). The total number of tasks n in a given batch can be calculated as the sum of the tasks submitted by all users, i.e., $n = \sum_{l=1}^{N} k_l$, where k_l is the number of

tasks of the user $l = 1, \ldots, N$. A *schedule* x is encoded by the following vector:

$$x = [x^1, \ldots, x^N], \tag{1}$$

where

$$x^l = [x_{\widehat{k_{(l-1)}}+1}, \ldots, x_{\widehat{k_{(l-1)}}+k_l}] \tag{2}$$

denotes the strategy vector of the user l (k_l is the number of the l-user's tasks, and $\widehat{k_{(l-1)}} = k_1 + \ldots + k_{(l-1)}$). The coordinates of vector x^l are the decision variables of the user l and indicate the numbers of the machines to which his tasks are assigned. The *symmetry* in the non-cooperative game expresses the situation where the privileges to the resources are the same for all users. Each user tries to choose an optimal strategy of mapping his tasks to machines in order to minimize the total cost of scheduling his pool of tasks. To express the symmetry of the grid users, we can assume that each player submits an equal amount of tasks, i.e., $k = k_1 = k_2 = \cdots = k_N$. This means that the total number of tasks in the batch can be calculated as $n = N \cdot k$.

Definition 1. The *symmetric grid user non-cooperative game* can be defined as the triple

$$G_N = (N; \{J_l\}_{l=1,\ldots N}; \{Q_l\}_{l=1,\ldots,N}),$$

where

- N is the number of grid users;
- $\{J_1, \ldots, J_N\}, l = 1, \ldots, N$ are the sets of the users' strategies;
- $\{Q_1, \ldots, Q_N\}, Q_l : J_1 \times \ldots \times J_N \rightarrow \mathbb{R}$, $\forall l = 1, \ldots, N$ is the set of the users' cost functions.

The users' strategy vectors x^l are the elements of the strategy spaces $J_l = J_{((l-1)\cdot k+1)} \times \ldots \times J_{(l\cdot k)}$ specified for each user $l\,(l = 1, \ldots, N)$. The cost of playing the game calculated for a particular user l is defined as the cost of scheduling his or her tasks and is denoted by Q_l. The players try to minimize simultaneously their cost functions Q_l during the game.

Definition 2. An N-dimensional multi-vector

$$(\widehat{x^1}, \ldots, \widehat{x^N})$$

of strategies is called an *equilibrium point* of the game if

$$\begin{aligned} &Q_l(\widehat{x^1}, \ldots, \widehat{x^N}) \\ &\quad = \min_{x^1 \in J_1} Q_l(\widehat{x^1}, \ldots, \widehat{x^{(l-1)}}, x^l, \widehat{x^{(l+1)}}, \ldots, \widehat{x^N}) \end{aligned} \tag{3}$$

for all $l = 1, \ldots, N$.

The equilibrium point can be interpreted as a steady state of the strategic game, in which each player holds correct expectations concerning other players' behavior[1]. If the strategies chosen by all players are equilibrium points, no player is interested in changing his strategy, because of the increase in the values of his cost function.

To be a solution of grid users' game, the equilibrium point should be additionally Pareto optimal (Straffin, 1996; Pavlidis *et al.*, 2005). In this paper we consider non-zero sum games[2], for which the equilibrium points are the results of minimization of a multi-cost game function.

Let us denote by $\min Q_l, (l = 1, \ldots N)$ the minimal value of the function Q_l calculated as follows:

$$\min Q_l = \min_{x^l \in J_l} \{Q_l(x^1, \ldots, x^N)\}. \tag{4}$$

The result of the global minimization of the following *game multi-cost* function $Q : J_1 \times \cdots \times J_N \rightarrow \mathbb{R}$:

$$Q(x^1, \ldots, x^N) = \sum_{l=1}^{N} \left([Q_l(x^1, \ldots, x^N) - \min Q_l] \right) \tag{5}$$

is an equilibrium state of a non-cooperative non-zero sum symmetric game of grid users, satisfying the condition of Pareto optimality (Pavlidis *et al.*, 2005).

The problem of the minimization of the function Q can be defined as a hierarchical procedure presented in Fig. 2. It is composed of two cooperating modules: *Global Module*, in which the values of the function Q are calculated and optimized, and *Players' Module*, which solves local level problems of the minimization of the users' cost functions Q_l.

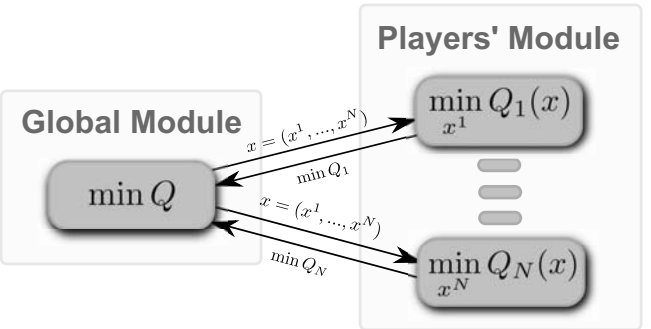

Fig. 2. Hierarchical procedure of solving a non-cooperative N-person game.

In order to specify the interaction mechanism of Global and Players' Modules let us denote by $x_{(0)}$ an initial schedule generated in Global Module, i.e., $x_{(0)} =$

[1] In the case of continuous players' cost functions, the equilibrium state of the game is called the *Nash equilibrium* (Edlefsen and Millham, 1972).

[2] In this scenario, the strategies of the players are not opposite, i.e., the sum or the values of all players cost functions Q_l is nonzero.

$[x^1_{(0)}, \ldots, x^N_{(0)}]$, where $x^l_{(0)}$ denotes the part of the schedule $x_{(0)}$ (sub-schedule of $x_{(0)}$) composed of the values of the decision variables of the user l (Eqn. (2)).

Vector $x_{(0)}$ is replicated and N copies of it are sent to Players' Module—one copy per user. Then each user independently optimizes his game cost function[3] by changing the allocations of just his own tasks. As a result of this minimization, the optimal values of the Q_l cost functions are calculated:

$$\begin{cases} \min Q_{1;(0)} = \min\limits_{x^1 \in J_1} Q_1(x^1, x^2_{(0)}, \ldots, x^N_{(0)}), \\ \vdots \\ \min Q_{N;(0)} = \min\limits_{x^N \in J_N} Q_N(x^1_{(0)}, \ldots, x^{N-1}_{(0)}, x^N). \end{cases} \tag{6}$$

These values are sent back to Global Module, where the objective function for the whole game Q is calculated for the schedule $x_{(0)}$.

4.2. Asymmetric scenario: Stackelberg game. For illustrating the asymmetric scenario in a non-cooperative grid users game, we define the Stackelberg game, in which one user acts as a *Leader* and the rest of players (users) are his *Followers*.

Stackelberg games have been well-studied in the game theory literature (e.g., Baçsar and Olsder, 1995). Roughgarden (2004) defined a Stackelberg game model for scheduling tasks on a set of machines with load-dependent latencies in order to minimize the total latency of the system.

The following examples illustrate some real-life grid scenarios, in which the Stackelberg game model can characterize situations where there is a Leader who acts first:

- There is a privileged grid user (Leader), who can have full access to resources as opposed to the other users with limited access to resources.
- Some tasks could have critical deadlines (especially in online scheduling) and they can be sent by the Leader to the meta-broker with a request to allocate them first.
- Considering a pool of tasks, the Leader could be the owner of a large portion of the tasks in the pool; it might then seem reasonable to allocate his tasks in the best resources in the system.
- Some tasks could have security requirements. Thus the Leader can send the information on these tasks to the meta-broker, requesting to allocate them in the most trustful resources (secure machines).
- Tasks arriving into a grid system could be disparate in their needs for computational resources. Some of them could be atomic tasks generated by compound tasks while others could be just monolithic applications. The high degree of heterogeneity of tasks is a crucial factor conditioning the grid system's performance. In such a scenario, the Leader could create a small batch of the most time consuming tasks, out the task pool, in order to "balance" the disparity. These tasks would then be sent to the meta-broker requesting to allocate them first.

Formally the Stackelberg game for grid users can be defined as a two-level game in the following way:

- **Leader's level: Leader's action I.** The Leader chooses his initial strategy $\widetilde{x^1} = [\widetilde{x_1}, \ldots, \widetilde{x_{k_1}}]$, where k_1 denotes the number of tasks submitted by the Leader.
- **Followers' level: Followers' action.** The followers minimize simultaneously their cost functions relative to the Leader's choice:

$$\begin{cases} x^2_F = \arg \min\limits_{x^2 \in J_2} \{Q_2(\widetilde{x^1}, x^2, \ldots, x^N)\} \\ \vdots \\ x^N_F = \arg \min\limits_{x^N \in J_N} \{Q_N(\widetilde{x^1}, \ldots, x^N)\}, \end{cases} \tag{7}$$

  where J_1 is the set of the Leader's strategies. Let us denote by $x_F = [\widetilde{x^1}, x^2_F, \ldots, x^N_F]$ the vector which is interpreted as the result of the Followers' action.
- **Leader's level: Leader's action II.** The Leader updates his strategy by minimizing his cost function Q_l taking into account the result of the Followers' actions. The vector $x_G = [x_L, x^2_F, \ldots, x^N_F]$, where

$$x_L = \arg \min_{x^1 \in J_1} Q(x^1, x^2_F, \ldots x^N_F), \tag{8}$$

  is a solution of the whole game.

It has to be noted that the Followers can play an "ordinary" non-cooperative symmetric game, but they must know the Leader's action first. The game muti-cost function Q in this case can be defined in the following way:

$$Q := Q_1 + Q_F, \tag{9}$$

where Q_1 is the Leader's cost function and

$$Q_F := \sum_{l=2}^{N} Q_l \tag{10}$$

is the *Followers' cost function*. An optimal solution of the whole game is called the *Stackelberg equilibrium*.

[3]Note that user cost optimization in Players' Module can be implemented as a parallel multi-threaded procedure, which can speed-up the whole process.

Table 1. Heuristic and meta-heuristic methods in grid scheduling.

Meta-heuristic class	Class characteristic	Scheduler type	Methods	Main references
Ad-hoc	– used for single-objective optimization – with low computational cost – useful in generating initial – solutions for population-based schedulers	Immediate mode	Minimum Completion Time (MCT) Minimum execution time Switching Algorithm (SA) Opportunistic load balancing k-percent best	(Braun *et al.*, 2001) (Xhafa *et al.*, 2008)
		Batch mode	min-min, max-min, sufferage relative cost, LJFR-SJFR	(Xhafa *et al.*, 2008)
Local search	– explore the solution space starting from an initial solution – construct a path in the solution space		Hill climbing Simulated annealing Tabu search	(Abraham *et al.*, 2000)
Population-based	– explore of the search space by the populations of individuals – require large running time – effective in finding near-optimal solutions	Single population	Genetic Algorithms (GAs) Memetic Algorithms (MAs) Particle Swarm Optimization (PSO) Ant Colony Optimization (ACO)	(Abraham *et al.*, 2000) (Liu *et al.*, 2009)
		Multi-population	Hierarchic Genetic Strategy (HGS) Grid-Enabled Hierarchical Parallel Genetic Algorithm (GE-HPGA)	(Kołodziej *et al.*, 2009) (Lim *et al.*, 2007)

5. Solving grid users' games

Due to multiple constraints and different optimization criteria in a dynamic environment, resource allocation in grids still remains a challenging, complex and computationally hard global optimization problem. Thus, heuristic approaches are candidates for effective design grid schedulers. Meta-heuristic schedulers could achieve robustness and are appropriate for tackling various scheduling attributes, like immediate and batch scheduling, multi-objectivity, decentralized and hierarchical grid architectures, etc. (Xhafa and Abraham, 2010).

5.1. Meta-heuristic methods. Meta-heuristic methods are usually classified into three main groups, namely *calculus-based methods* (greedy algorithms and ad-hoc methods), *stochastic methods* (guided and non-guided methods) and *enumerative methods* (dynamic programming and the branch-and-bound algorithm). The most popular and efficient meta-heuristics in grid scheduling are ad-hoc, local search-based and population-based methods. We briefly review them in Table 1. For a more detailed survey on meta-heuristic approaches in grid scheduling, the reader is referred to the work of Xhafa *et al.* (2009).

Another important feature of meta-heuristics which is useful in grid scheduling is that they can be easily hybridized with other approaches. It makes grid schedulers better adapted to various grid types and specific types of applications. Abraham *et al.* (2000) present a model for hybridization of GA, SA and TS heuristics; each GA-based hybrid, namely, GA+SA and GA+TS, improves the efficiency of the genetic scheduler. A GA+TS hybrid approach was proposed by Xhafa *et al.* (2009). Ritchie and Levine (2003) combine an ACO with a TS algorithm for the problem. In the work of Xhafa *et al.* (2007), a basic unstructured MA is combined with several local search algorithms in order to identify the best performance of the resulting memetic algorithm.

The hybridization ability and dealing with various conflicting optimization criteria are at the same time crucial meta-heuristics features, which makes them appropriate methods for solving grid users' games. In such games, many scheduling and resource management criteria, like security and resource utilization and reliability, are integrated into users' cost functions. This is in contrast to most of other approaches, where those criteria are addressed separately, which contradicts in fact the complex nature of grid systems. In Section 5.2, we present the main concepts of four hybrid GA-based schedulers, previously defined by Kołodziej *et al.* (2009) as well as Kołodziej and Xhafa (2010), which are well-suited for optimizing the game multi-cost function Q (see Eqns. (5) and (9)) for symmetric and asymmetric grid user games characterized in Sections 4.1 and 4.2.

5.2. GA-based hybrid approach. A general concept of hybridization of four GA-based approaches for solving symmetric and asymmetric grid user games is presented in Table 2.

The GA-based meta-heuristics act as global and local optimizers in Global and Players' Modules. Each hybrid is defined as a combination of two methods: the *Risky Genetic Algorithm (RGA)* and *Secure Genetic Algorithm (SGA)*—in Global Module—and two local level optimi-

Table 2. Hybrid meta-heuristics for risky and secure-assured task scheduling.

Hybrid meta-heuristic	Main unit	Subordinate unit
RGA-GA	RGA	PGA
RGA-PMCT	RGA	PMCT
SGA-GA	SGA	PGA
SGA-PMCT	SGA	PMCT

zers: *Player's Genetic Algorithm (PGA)* and a modified *MCT* heuristic, namely, *Player's Minimum Completion Time (PMCT)*, in Players' Module. The main difference between the *RGA* and the *SGA* is the method of evaluating the population by the users' cost functions Q_l, which can be calculated differently in the case of considering the security condition and in the case of ignoring all security requirements (we will call these two scenarios the *secure* and *risky* modes and define the appropriate players cost functions formulas in Section 6.1).

The general flowchart of the hybrid GA-based schedulers in the case of the symmetric game is presented in Fig. 3.

The algorithm implemented in Players' Module runs sequentially in order to minimize the users' cost functions. Once the population of schedules has been sent to Players' Module, each player tries to optimize his tasks allocation and compute the $\min Q_l$ values for the schedules. It can be observed that, in fact, it is not necessary to replicate the whole population for each user because for each of them independently just the changes in machine completion times must be calculated.

For the Stackelberg game, the general template of GA-based schedulers at the Leader's level is given in Algorithm 1.

The process of initialization of the population in hybrid GAs is defined as a two-steps procedure. In the first step, we define P^0 as a candidate initial population. It consists of the incomplete schedules computed by the Leader using one of the initialization methods for GA-based schedulers (see, e.g., Xhafa *et al.*, 2007). Each schedule from this set contains just the values of the Leader's decision variables. All those "incomplete" chromosomes are sent to the Followers, who fill in the respective parts of each schedule by using one of the ad-hoc heuristics. The updated P^0 is then evaluated under the game cost function $Q = \sum_{l=1,\ldots,N} Q_l$ defined as fitness. The crossover and mutation operations are performed separately on the Leader's and Followers' decision variables. Thus in each generation the Followers can update their own decision (including the initial choices) due to possible changes on the availability of resources introduced by the Leader.

Algorithm 1 A hybridized GA-based scheduler template: Leader's level.

1: Generate P^0 containing μ "incomplete" schedules; $t = 0$;
2: Send P^0 to the **Followers** to complete the respective parts of all schedules in P^0 *(using ad-hoc heuristics)*; $P^0(F)$ is created;
3: Update the population P^0 according to the Followers' solutions; $P^0 := P^0(F)$;
4: Evaluate P^0;
5: **while** not termination-condition **do**
6: Select the parental pool T^t of size λ; $T^t := Select(P^t)$;
7: Perform crossover procedures separately on the Leader's and Followers'variables on pairs of individuals in $T^t(F)$ with probability p_c; $P_c^t := Cross(T^t)$;
8: Perform mutation procedures separately to the Leader's and Followers' variables on individuals in P_c^t with probability p_m; $P_m^t := Mutate(P_c^t)$;
9: Evaluate P_m^t ;
10: Create a new population P^{t+1} of size μ from individuals in P^t and P_m^t ; $P^{t+1} := Replace(P^t; P_m^t)$
11: $t := t + 1$;
12: **end while**
13: **return** Best found individual as solution;

Local schedulers in Players' Module. We used two modifications of the well-known grid schedulers for optimizing the users' cost function in Players' Module.

The first one, named as the *player's genetic algorithm*, is a simple extension of the classical GA-based scheduler (Xhafa *et al.*, 2007) applied independently for each user with the cost function Q_l as fitness. The GA operations are executed on sub-schedules of the length k_l labeled just by the tasks submitted by user l. The GA procedure is run sequentially for the "queue" of users.

The second method, the *Player's Minimum Completion Time*, is a modification of *Minimum Completion Time (MCT)*. In this method, a task is assigned to the machine yielding the earliest completion time (defined as the sum of $ready\_time$ for the machine and the time of computing all tasks assigned there). The process is repeated until there remain tasks to be assigned. The template of the main mechanism of the *PMCT* procedure is defined in Algorithm 2.

Algorithm 2 *PMCT* algorithm template.

1: Receive the population of schedules and $ready\_times$ of the machines from Global Module;
2: **for all** Schedule in the population **do**
3: Calculate the completion times of the machines in a given schedule;
4: **for all** Individual user/Follower **do**
5: **for all** User's task/Follower's task **do**
6: Find the machine that gives minimum completion time;
7: Assign task to its best machine;
8: Update the machine completion time;
9: **end for**
10: Calculate the $\min Q_l$ value for a given schedule;
11: **end for**
12: Send the $\min Q_l$ values to Global Module;
13: **end for**

A simple case study on the four hybrid GA-based schedulers approaches is presented in the following section.

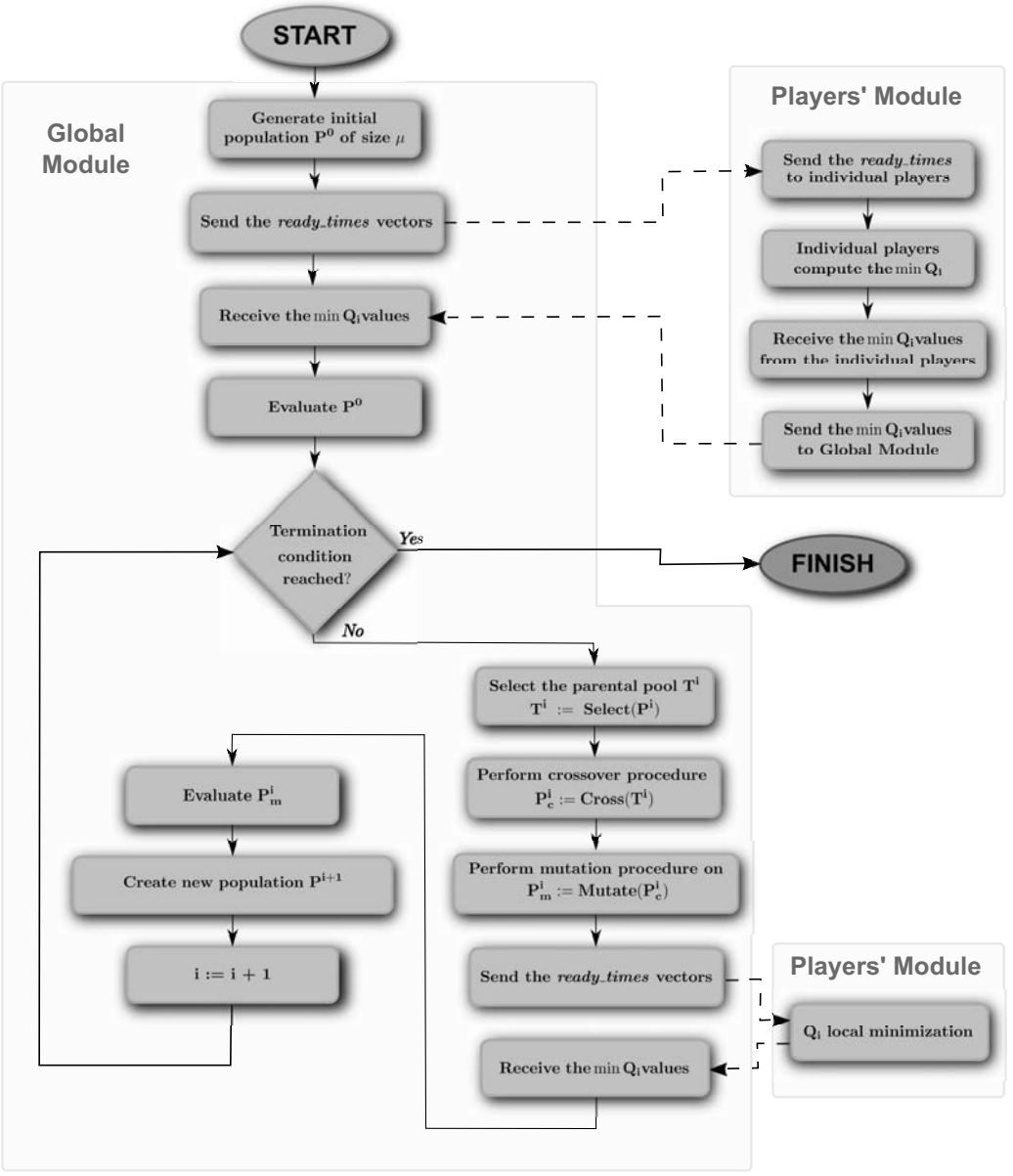

Fig. 3. Hybrid GA-based schedulers general flowchart.

6. Case study: Non-cooperative grid users games for independent batch scheduling

6.1. Specification of the players' cost functions. In our case study, we considered symmetric and Stackelberg games defined in Sections 4.1 and 4.2, for which the players' cost functions are composed of the following three factors:

$$Q_l = Q_l^{(e)} + Q_l^{(s)} + Q_l^{(u)}, \tag{11}$$

where $Q_l^{(e)}$ is the user's task execution cost, $Q_l^{(s)}$ indicates the cost of security-assured allocation of the user's tasks, and $Q_l^{(u)}$ denotes a resource utilization cost.

The total cost of the execution of the user's tasks expressed as $Q_l^{(e)}$ can be calculated as an average completion time of his tasks on the machines to which they are assigned[4]. Let us introduce the following notation: $Machines(l)$ denotes the set of machines to which all tasks of the user l are assigned, and $Tasks(l)$ is the set of tasks of the user l. The function $Q_l^{(e)}$ can be defined using the following formulae:

$$Q_l^{(e)} = \frac{\sum_{j \in Tasks(l)} completion[j][i]}{completion_{m(l)} \cdot k_l}, \tag{12}$$

where $completions[j][i]$ denotes the completion time of a

[4]The values of all components of the users' cost functions, i.e., $Q_l^{(e)}$, $Q_l^{(u)}$ and $Q_l^{(s)}$ functions, are scaled to obtain values in the same range.

task j on a machine i and it is calculated in the following way:

$$completions[j][i] = ETC[j][i] + ready[i]. \quad (13)$$

In Eqn. (13), $ETC[j][i]$ denotes the elements of the *ETC* matrix and $ready[i]$ is the finishing time of the execution of tasks previously assigned to the machine i.

In Eqn. (12) we denoted by $completion_{m(l)}$ the maximal completion time of the user's tasks, i.e.,

$$completion_{m(l)} = \max_{\substack{j \in Tasks(l) \\ i \in Machines(l)}} completion[j][i]. \quad (14)$$

The values of the function $Q_l^{(s)}$ depend on the scheduling strategy. In this work we consider two scheduling strategies:

- **Risky mode**, in which all risky and failing conditions are ignored by the users. In this case, $Q_l^{(s)} = 0$, $l = 1, \ldots, N$.
- **Secure mode**, in which the $Q_l^{(s)}$ function is defined as follows:

$$Q_l^{(s)} = \sum_{j=(k_1+\cdots+k_{l-1}+1)}^{(k_1+\cdots+k_l)} \frac{P_f[j][x_j] \cdot ETC[j][x_j]}{(ETC)_{m(l)} \cdot k_l}, \quad (15)$$

$(ETC)_{m(l)}$ is the (expected) maximal computation time of the tasks of the user l in a given schedule.

In Eqn. (15) we denoted by $P_f[j][x_j]$ the probability of the failure of machine x_j during the execution of task j. This probability is usually modeled by the negative exponential distribution:

$$P_f[j][x_j] = \begin{cases} 0 & \text{if} \quad sd_j \leq tl_{x_j}, \\ 1 - e^{-\lambda(sd_j - tl_{x_j})} & \text{if} \quad sd_j > tl_{x_j}, \end{cases} \quad (16)$$

where λ is interpreted as a failure coefficient and is a global parameter of the model, and sd_j and tl_{x_j} are the elements of *security demand* SD and *trust level* TL vectors (Song *et al.*, 2006).

The resource utilization cost $Q_l^{(u)}$ is calculated for each grid user as an average idle time of machines on which his tasks are executed:

$$Q_l^{(u)} = \sum_{x_j \in machines(l)} \left(1 - \frac{Completion_{(l)}[x_j]}{makespan}\right) \cdot \frac{\sum_{j \in Tasks_{(l)}[x_j]} ETC[j][x_j]}{Completion_{(l)}[x_j]}, \quad (17)$$

where $Completion_{(l)}[x_j]$ is the completion time of a given machine $x_j \in Machines(l)$, and $Tasks_{(l)}[x_j]$ is the set of the tasks of the user l assigned to the machine x_j.

It can be noted that the problem of risk-resilient resource allocation is defined in our approach from the grid users' perspective, and not that of the resource owners.

6.2. Experiments setting. In the experimental evaluation of the proposed four hybrid meta-heuristics in two games scenarios, we integrated the schedulers with the discrete event-based grid simulator *HyperSim-G* (Xhafa *et al.*, 2009). The experiments were conducted on two benchmarks composed of a set of static and dynamic instances. In both static and dynamic cases, four grid, size scenarios are considered: small (32 hosts/512 tasks), medium (64 hosts/1024 tasks), large (128 hosts/2048 tasks), and very large (256 hosts/4096 tasks). The settings for the simulator are presented in Table 3.

There are 16 players in the symmetric game and 15 Followers in the Stackelberg game, and the number of the Leader's tasks is a half of the whole task batch. The coefficients of SD and TL vectors as well as machines reliability probabilities P_{x_j} are defined as uniformly generated fractions in the ranges [0.6,0.9], [0.3,1] and [0.85,1], respectively. The value of the failure coefficient λ is 3.

In Table 4, we present the key parameters for the GA engines in Global and Players' Modules for the symmetric game. The parameters are the same for Leader's and Followers' Modules in the Stackelberg game.

Table 4. Settings of GAs key parameters in Global/Leader's and Player's/Followers' Modules.

Parameter	SGA and RGA	PGA
Population size	60	20
Intermediate pop. size	48	14
Crossover prob.	0.8	0.8
Mutation prob.	0.2	0.2
Stopping criterion	1000 iterations	500 iterations

A combination of genetic operators for GAs in the case of symmetric and asymmetric scenarios was selected based on the results of the tuning process performed by Xhafa *et al.* (2007). We used linear ranking selection, cycle crossover (CX), re-balancing mutation and elitist generational replacement as the main evolutionary mechanism in Global/Leader's and Players'/Followers' Modules. We also applied *LJFR-SJFR (Longest Job to Fastest Resource–Shortest Job to Fastest Resource)* as an initialization procedure to introduce more diversity to the initial population.

Performance measures. To evaluate the scheduling performance we used the *makespan* and *flowtime* metrics (see Xhafa *et al.*, 2009) defined in the following way:

- *Flowtime:* Let F_j denote the time when task j finalizes. The flowtime can be calculated using the following formulae:

$$Flowtime = \sum_{j \in Task} F_j. \quad (18)$$

Table 3. Setting for the grid simulator for static and dynamic cases.

	Static setting			
	Small	**Medium**	**Large**	**Very Large**
Nb. of hosts	32	64	128	256
Resource cap. (in MIPS)	$N(1000, 175)$			
Total nb. of tasks	512	1024	2048	4096
Workload of tasks	$N(250000000, 43750000)$			
	Dynamic setting			
Init. hosts	32	64	128	256
Max. hosts	37	70	135	264
Min. hosts	27	58	121	248
Resource cap. (in MIPS)	$N(1000, 175)$			
Add host	$N(625000, 93750)$	$N(562500, 84375)$	$N(500000, 75000)$	$N(437500, 65625)$
Delete host	$N(625000, 93750)$			
Total tasks	512	1024	2048	4096
Init. tasks	384	768	1536	3072
Workload	$N(250000000, 43750000)$			
Interarrival	$E(7812.5)$	$E(3906.25)$	$E(1953.125)$	$E(976.5625)$

The flowtime is usually considered a QoS criterion as it expresses the response time to the submitted task execution requests of grid users.

- *Makespan:* Makespan is one of the basic metrics of grid systems performance—the smaller the value of the makespan, the faster the execution of tasks in the grid system. The makespan can be calculated by the following formulae:

$$Makespan = \max_{j \in Tasks} F_j. \tag{19}$$

6.3. Experimental results. Each experiment was repeated 30 times under the same configuration of parameters and operators. In Figs. 4 and 5 we present the averaged values of makespan and flowtime achieved by four hybrid meta-heuristics when solving the symmetric and Stackelberg games within static and dynamic grid scenarios.

In the symmetric game, the best results for makespan and flowtime in all grid scenarios considered were achieved by the *SGA-GA* scheduler. Especially in static small-size grids, this method is very effective in makespan reduction. The differences in the flowtime results achieved by all hybrid meta-heuristics are not so significant, while in the case of makespan both *SGA* hybrids significantly outperform risky hybrids in all grid scenarios.

In the case of the Stackelberg game, two *PMCT* hybrids outperform the *RGA-GA* and *SGA-GA* algorithms. For makespan values, the differences in the results achieved by *PMCT* and *GA* hybrids are significant, while in the case of flowtime, all values are at the same level, except those obtained for very large grid sizes. The best results in all instances are achieved by the *SGA-PMCT* algorithm. However, in the case of static scheduling scenario, the efficiencies of *RGA-PMCT* and *SGA-PMCT* are very similar, while in the dynamic case, especially for makespan values, the differences in both schedulers' performance are significant.

Although the security requirements would imply some additional cost to the users of the grid system, it is worth assuming this cost in order to allocate tasks to trustful resources.

Our experimental analysis shows that hybrid GA-based schedulers can be effective in solving user games; however, the main drawback of using such methods is their high computation complexity. The game scenarios presented in Section 4 are very general, which makes them useful in supporting the user decision process in various situations. In some real-life approaches particular game scenarios can be applied for efficient resource allocation. Most of them are based on economical models. In the following section, we briefly highlight the most popular game models used in grid scheduling.

7. Game-theoretical models and market-based approaches

Computational economy is a popular mechanism for the design of the resource management architecture of grid systems. It allows the resource owners, acting as sellers, to earn money by letting others (mainly grid users, acting as buyers) use their (idle) computational resources. The pricing of the resources is driven by demand and supply.

A number of economic models for grid resource management has been proposed in the literature so far. We present in Table 5 three most popular approaches found in grid computing, namely, *commodity market*, *auction* and *bargaining* models. Each model can be easily modified to meet users or resource owners requirements. For example, the posted price model (Buyya and Bubendorfer, 2009) is an extension of commodity market by considering posted prices (usually lower than regular ones) as a special offer to grid users.

The auction mechanism can be also defined in many ways (e.g., English, Dutch, second price auctions). All of them differ in terms of whether they are performed as

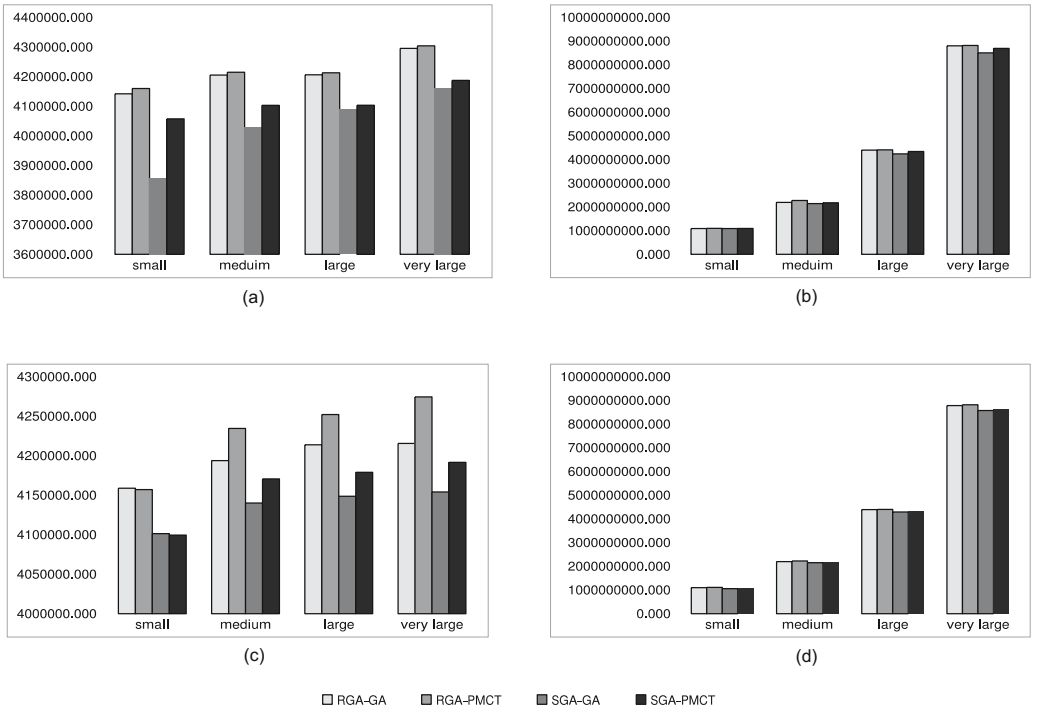

Fig. 4. Experimental results for the non-cooperative symmetric game: static case—average makespan (a), average flowtime (b); dynamic case—average makespan (c), average flowtime (d).

open or closed auctions and the offer price for the highest bidder. Wolski *et al.* (2001) proposed a model called G-Commerce in which computational resources among different grid sites are traded in a barter manner. This model can be interpreted as a combination of the commodity market and auction approaches.

Market-oriented approaches are suitable to exploit the interaction of different scheduling layers. However, grid users and resource owners are likely to behave in different (selfish or/and cooperative) manners and their behavior cannot be characterized using conventional techniques. Game-theoretical models are quite natural tools for solving this problem, because each market-based scenario can be easily translated into the game framework. This kind of approaches has recently attracted a lot of attention of researchers.

The mechanism of first price bidding auctions was applied by Kwok *et al.* (2007) to define the game-based resource management and global scheduling policy at the intra- and inter-site levels in the 3-levels hierarchical grid structure. In the intra-site bidding, each machine owner in the site declares the “execution capability" of the resource. The local job dispatcher moderates these amounts and sends a single value to the global scheduler. In the inter-site bidding, the global scheduler should allocate tasks according to the values sent by the local dispatchers. The authors showed that the cooperation of the players at both levels are optimal strategies for both games. In fact, there is also a third game defined for the resource owners, who behave selfishly. The objective of this non-cooperative game is to maximize resource utilization. The game scenarios at each level are very simple, and the authors focused in fact on the optimization of two scheduling criteria: minimization of the task deadlines (user’s requirement) and maximization of resource utilization (resource owner’s requirement). However, for successful execution of all those games, a synchronization mechanism must be introduced, which can make the whole system inefficient in a large-scale dynamic environment.

An early approach to modified auction mechanisms can be found in the work of Regev and Nisan (2000). The authors defined POPCORN market for trading online CPU times. In their system, a virtual currency called “popcoin" was used between buyers and sellers communicating via the Internet. Social efficiency and price stability were studied using the Vickrey auction game. In this scenario, cooperation between players to form a coalition and win the auction is possible, but the players usually behave selfishly. It should be noted that in this approach some form of concrete currency is needed to play the game, and for that reason the system would not be practicable in a real-life situation where there are huge numbers of machines involved.

Ghosh *et al.* (2005) studied the load balancing issues in a mobile CG, in which there is a Wireless Access Point (WAP) which mediates the requests from different mobi-

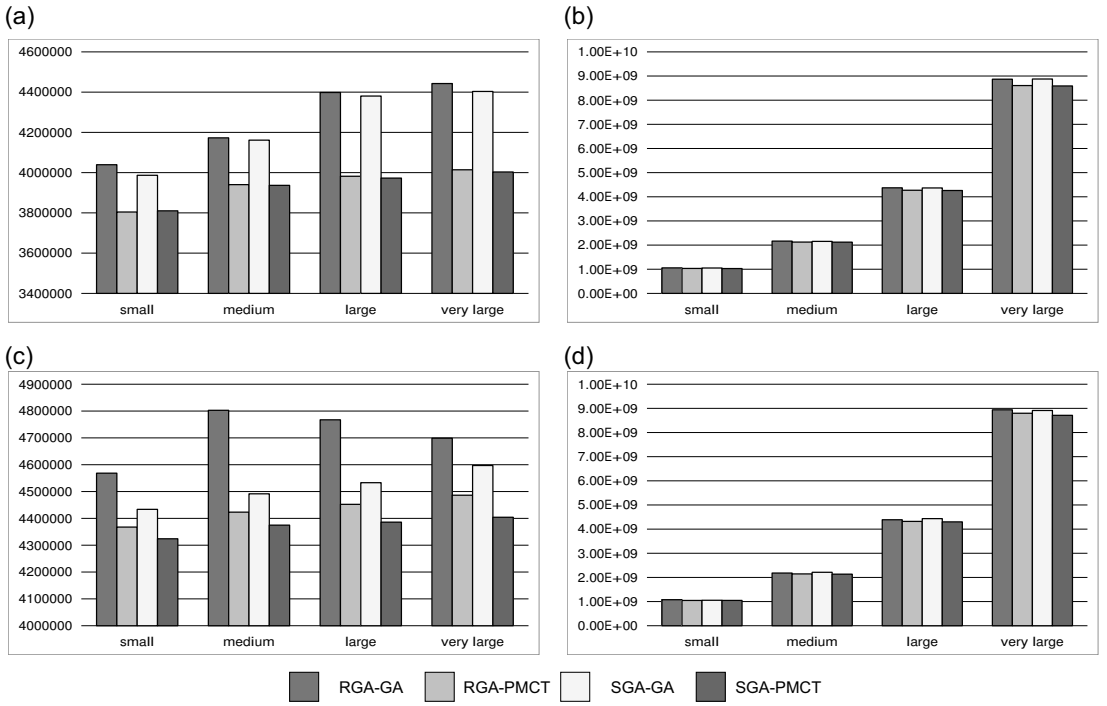

Fig. 5. Experimental results for the Stackelberg game: static case—average makespan (a), average flowtime (b); dynamic case—average makespan (c), average flowtime (d).

le devices. The problem is modeled as a bargaining cooperative game between each mobile device and the WAP server. Assuming that there are n mobile devices under a single WAP server, this server has to play n such games with the corresponding devices. The solution of the whole game is the Nash Bargaining Solution (NBS). In this approach, an explicit payment scheme must be enforced in the system. A recent study on the bargaining cooperative game application in optimizing energy consumption in grids is proposed in by Subrata *et al.* (2010).

8. Conclusions and future work

In this paper we addressed the need for using new computational paradigms to efficiently solve scheduling and resource allocation problems in computational grids. By surveying the most important approaches in the literature, we observed that the proposed methods usually fail to effectively cast additional requirements of grid scheduling such as security and reliability of resources.

Game-theoretic models proved to be useful approaches for supporting grid users' decisions, where different scheduling criteria, including security and resource reliability, must be considered at the same time. Users' behavior can be effectively translated into the computational model linked to grid scheduling. Due to a large-scale of the grid, non-cooperative games seem to be a potential model for integrating various requirements in grid scheduling.

We presented two general non-cooperative game scenarios, namely, the symmetric non-zero sum game and the Stackelberg game, and defined simple GA-based hybrid schedulers for approximating their equilibrium points. The procedure of solving the proposed non-cooperative games is complex because of the need for integration and synchronization of two cooperating modules. However, experimental analysis shows high efficiency of using the meta-heuristics as resolution methods for game-based models, especially in the case of additional security costs paid by the users. Because of the variety of real-life game scenarios, we believe that the game-based models concept can be successfully implemented also in cloud computing, where secure scheduling and information management remain challenging problems.

References

Abraham, A., Buyya, R. and Nath, B. (2000). Nature's heuristics for scheduling jobs on computational grids, *Proceedings of the 8th IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), New Delhi, India*, pp. 45–52.

Ali, S., Siegel, H., Maheswaran, M. and Hensgen, D. (2000). Task execution time modeling for heterogeneous computing system, *Proceedings of the Heterogeneous Computing Workshop, Cancun, Mexico*, pp. 185–199.

Baçsar, T. and Olsder, G. (1995). *Dynamic Non-cooperative Game Theory*, 2nd Edn., Academic Press, London.

Table 5. Market-based models for resource management in computational grids

Model	Characteristic	Optimization criteria	Grid architecture model	References
Commodity market	– service providers primarily charge the end user for the resources they consume (CPUs), – pricing policies are based on the demand from the users and the supply of resources	– minimizing the overall task execution time – minimizing the cost of the resource utilization (paid by the user under budget constraints)	Meta-broker	(Garg *et al.*, 2009) (Buyya *et al.*, 2002)
Auctions	– offering buying and selling items for bids – managing the bids (auctioneers) – two groups of participant: sellers (resource owners) and buyers (grid users)	– minimizing the highest bidder price (first price auction) – minimizing the second highest bidder price (Vickrey auction) – maximizing the resource sellers pay-offs	– all hierarchical models – decentralized models	(Ghosh *et al.*, 2005) (Buyya *et al.*, 2000)
Bargaining	– resource brokers bargain with resource providers for lower access price and higher usage duration – the negotiation is guided by user requirements (e.g., deadline) – the negotiation can be provided directly between buyers and sellers	- minimizing the cost of the resource utilization – maximizing the resource sellers pay-offs	– all hierarchical models – decentralized models	(Regev and Nisan, 2000) (Subrata *et al.*, 2010)

Brandic, I., Pllana, S. and Benkner, S. (2006). An approach for the high-level specification of qos-aware grid workflows considering location affinity, *Scientific Programming* **14**(3–4): 231–250.

Braun, T., Siegel, H.J., Beck, N., Boloni, L.L., Maheswaran, M., Reuther, A.I., Robertson, J.P., Theys, M.D., Yao, B., Hensgen, D. and Freund, R.F. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems, *Journal of Parallel and Distributed Computing* **61**(6): 810–837.

Buyya, R., Abramson, D. and Giddy, J. (2000). An economy driven resource management architecture for global computational power grids, *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA 2000), Las Vegas, NV, USA*, pp. 517–525.

Buyya, R., Abramson, D., Giddy, J. and Stockinger, H. (2002). Economic models for resource management and scheduling in grid computing, *Journal of Concurrency and Computation: Practice and Experience* **14**(13–15): 1507–1542.

Buyya, R. and Bubendorfer, K. (2009). *Market Oriented Grid and Utility Computing*, Wiley Press, New York, NY.

Edlefsen, L. and Millham, C. (1972). On a formulation of discrete n-person non-cooperative games, *Metrika* **18**(1): 31–34.

Garg, S., Buyya, R. and Segel, H. (2009). Scheduling parallel aplications on utility grids: Time and cost trade-off management, *Proceedings of the Thirty-Second Australasian Conference on Computer Science*, Vol. 91, Australian Computer Society, Inc., Darlinghurst, Australia, pp. 139–147.

Ghosh, P., Roy, N., Basu, K. and Das, S. (2005). A game theory based pricing strategy for job allocation in mobile grids, *Journal of Parallel and Distributed Computing* **65**(11): 1366–1383.

Hwang, S. and Kesselman, C. (2003). A flexible framework for fault tolerance in the grid, *Journal of Grid Computing* **1**(3): 251–272.

Khan, S. and Ahmad, I. (2006). Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation, *Proceedings of the International Parallel and Distributed Proceedings Symposium (IPDPS 2006), Rhodes Island, Greece*, pp. 101–104.

Kołodziej, J. and Xhafa, F. (2010). A game-theoretic and hybrid genetic meta-heuristic model for security-assured scheduling of independent jobs in computational grids, *in* L. Barolli, F. Xhafa and S. Venticinque (Eds.) *Proceedings of CISIS 2010, Kraków, Poland,* IEEE Press, Los Alamitos, CA, pp. 93–100.

Kołodziej, J., Xhafa, F. and Kolanko, Ł. (2009). Hierarchic genetic scheduler of independent jobs in computational grid environment, *in* J. Otamendi, A. Bargieła, J.L. Montes and L.M. Doncel Pedrera (Eds.), *Proceedings of ECMS 2009, Madrid, Spain*, IEEE Press, Los Alamitos, CA, pp. 108–115.

Kwok, Y.-K., Hwang, K. and Song, S. (2007). Selfish grids: Game-theoretic modeling and nas/psa benchmark evaluation, *IEEE Transactions on Parallel and Distributing Systems* **18**(5): 1–16.

Laccetti, G. and Schmidb, G. (2007). A framework model for grid security, *Future Generation Computer System* **23**(5): 702–713.

Lim, D., Ong, Y.-S. and Jin, Y. (2007). Efficient hierarchical parallel genetic algorithms using grid computing, *Future Generation Computer System* **23**(4): 658–670.

Lin, C., Wang, V.V.Y. and Pruthi, V. (2004). Enhancing grid security with trust management, *Proceedings of the 2004 IEEE international Conference on Services Computing, (SCC 2004,), Sanghai, China*, pp. 303–310.

Liu, H., Abraham, A. and Hassanien, A. (2009). Scheduling jobs on computational grids using a fuzzy particle swarm optimization algorithm, *Future Generation Computer System* **26**(8): 1336–1343.

Mesghouni, K., Hammadi, S. and Borne, P. (2004). Evolutionary alogorithms for job-shop scheduling, *International Journal of Applied Mathematics and Computer Science* **14**(1): 91—103.

Pavlidis, N., Parsopoulos, K. and Vrahatis, M. (2005). Computing nash equilibria through computational intelligence methods, *Journal of Computational and Applied Mathematics* **175**(1): 113–136.

Regev, O. and Nisan, N. (2000). The popcorn market—Online markets for computational resources, *Decision Support Systems* **28**(1–2): 177–189.

Ritchie, G. and Levine, J. (2003). A fast effective local search for scheduling independent jobs in heterogeneous computing environments, *Technical report*, Centre for Intelligent Systems and Their Applications, School of Informatics, University of Edinburgh, Edinburgh.

Roughgarden, T. (2004). Stackelberg scheduling strategies, *SIAM Journal on Computing* **33**(2): 332–350.

Song, S., Hwang, K. and Kwok, Y. (2005). Trusted grid computing with security binding and trust integration, *Journal of Grid Computing* **3**(1–2): 53–73.

Song, S., Hwang, K. and Kwok, Y.-K. (2006). Risk-resilient heuristics and genetic algorithms for security-assured grid job scheduling, *IEEE Transactions on Computers* **55**(6): 703–719.

Straffin, P. (1996). *Game Theory and Strategy*, Mathematical Association of America Textbooks, Washington, DC.

Subrata, R., Zomaya, A.Y. and Landfeldt, B. (2010). Cooperative power-aware scheduling in grid computing environments, *Journal of Parallel and Distributed Computing* **70**(2): 84–91.

Wolski, R., Plank, J., Bryan, T. and Brevik, J. (2001). G-commerce: Market formulations controlling resource allocation on the computational grid, *Proceedings of the 15th International Parallel and Distributed Processing Symposium (IPDPS'01), San Francisco, CA, USA*.

Wu, C. and Sun, R.-Y. (2010). An integrated security-aware job scheduling strategy for large-scale computational grids, *Future Generation Computer Systems* **26**(2): 198–206.

Xhafa, F. and Abraham, A. (2010). Computational models and heuristic methods for grid scheduling problems, *Future Generation Computer Systems* **26**(4): 608–621.

Xhafa, F., Barolli, L. and Durresi, A. (2008). An experimental study on genetic algorithms for resource allocation on grid systems, *Journal of Interconnection Networks* **8**(4): 427–443.

Xhafa, F., Carretero, J. and Abraham, A. (2007). Genetic algorithm based schedulers for grid computing systems, *International Journal of Innovative Computing, Information and Control* **3**(5): 1–19.

Xhafa, F., Carretero, J., Alba, E. and Dorronsoro, B. (2009). Tabu search algorithm for scheduling independent jobs in computational grids, *Computer and Informatics Journal* **28**(2): 237–249.

Xhafa, F., Gonzalez, J., Dahal, K. and Abraham, A. (2009). A GA(TS) hybrid algorithm for scheduling in computational grids, *in* E. Corchado, X. Wu, E. Oja, Á. Herrero and B. Baruque (Eds.), *Hybrid Artificial Inteligence Systems*, Lecture Notes in Computer Science, Vol. 5572, Springer, Berlin/Heidelberg, pp. 285–292.

Joanna Kołodziej graduated in mathematics from Jagiellonian University in Cracow in 1992, where she also obtained a Ph.D. in computer science in 2004. She joined the Department of Mathematics and Computer Science of the University of Bielsko-Biała as an assistant professor in 1997. She has served and is currently serving as a PC co-chair, general co-chair and IPC member of several international conferences and workshops including *PPSN 2010, ECMS 2011, CISIS 2011, 3PGCIC 2011, CISSE 2006, CEC 2008, IACS 2008-2009, ICAART 2009-2010*. Dr. Kołodziej is the managing editor of the *International Journal of Space-Based and Situated Computing (IJSSC)* and serves as an editorial board member and guest editor of several peer-reviewed international journals.

Fatos Xhafa is an associate professor (with tenure) at the Technical University of Catalonia, Spain. His research interests include parallel and distributed algorithms, combinatorial optimization, distributed programming, grid and P2P computing. He has widely published in international journals, books and conference proceedings covering his research area. Dr. Xhafa is the editor-in-chief of the *International Journal of Space-Based and Situated Computing (IJSSC)* and the *International Journal of Grid and Utility Computing (IJGUC)*, Inderscience. He serves as an editorial board member of nine peer-reviewed international journals and has also guest co-edited in several international journals. He has served and is currently serving as a PC co-chair/general co-chair of several international conferences and workshops.

Received: 29 June 2010
Revised: 29 December 2010
Re-revised: 4 January 2011