Paper

Secure-Sim-G: Security-Aware Grid Simulator – Basic Concept and Structure

Grzegorz Gębczyński[a], Joanna Kołodziej[a], and Samee Ullah Khan[b]

[a] Faculty of Mechanical Engineering and Computer Science, University of Bielsko-Biała, Bielsko-Biała, Poland
[b] Department of Electrical and Computer Engineering, North Dakota State University, USA

Abstract— Task scheduling and resource allocation are the key issues for computational grids. Distributed resources usually work at different autonomous domains with their own access and security policies that impact successful job executions across the domain boundaries. In this paper we present a security-aware grid simulator Secure-Sim-G, which facilitates the evaluation of the different scheduling heuristics under various scheduling criteria in several grid scenarios defined by the security conditions, grid size and system dynamics. The simulator allows the flexible activation or inactivation of all of the scheduling criteria and modules, which makes the application well adapted to the proper illustration of the different realistic scenarios and avoids the possible restriction to the specific scheduling resolution methods. The simulation results and traces may be graphically represented and stored at the server and can retrieved in different formats such as spreadsheets or pdf files.

Keywords—computational grid, grid simulator, scheduling, security.

1. Introduction

Grid computing has emerged as a wide area distributed platform for solving the large-scale problems in science and engineering. Computational Grid (CG) involves the combination of many computing resources into a network for the execution of computational tasks. The resources are distributed across multiple organizations, administrative domains with their own access and usage policies. The effective task scheduling and the management of the grid resources are the complex key issues for CGs. They usually demand sophisticated tools for analyzing the algorithms performances before applying them to the real systems, which is in fact necessary in the case when the main scheduling objectives, such as the maximization of the resource utilization and profits of the resource owners, may conflict with grid users' security requirements and system reliability. The grid resource may not be accessible if the grid network or grid cluster is under an external attack or the grid users' security priorities in scheduling may not cope with the offer of the resource providers. Therefore, it is desirable to have a prior knowledge about the security demands from submitted applications and the trust level assured by a resource provider at the grid cluster. An effective grid scheduler must be then security-driven and resilient in response to all scheduling and risky conditions. It means that in order to achieve the successful tasks executions according the specified users' requirements, the relations between the assurance of secure computing services by a grid site or by a cluster node (security) and the behavior of a resource node (trust) must be defined, analyzed and simulated in all possible scenarios.

Simulation seems to be the most effective solution for the comprehensive analysis of the security-aware scheduling algorithms in large-scale distributed dynamic systems such as grid or cloud environments. It simplifies the study of schedulers performances and avoids the overhead of coordination of the resources, which usually happens in the real-life grid or cloud scenarios. Simulation is also effective in working with very large problems that require the involvement of a large number of active users and resources, which is usually very hard for the management in real-life approaches. In such cases a considerable number of independent runs is needed to ensure significant statistical results, that can be easily realized with the system simulator.

In this work, we present the main concept of a security-aware grid simulator Secure-Sim-G for independent batch scheduling, which is an extension and modification of the HyperSim-G framework [1]. Secure-Sim-G is an event-based application, which facilitates the evaluation of the different scheduling heuristics under various scheduling criteria in several grid scenarios defined by the security conditions, grid size and system dynamics. The simulator allows the flexible activation or inactivation of all of the scheduling criteria and modules, which makes the application well adapted to the proper illustration of the different realistic scenarios and prevents the possible restriction to the specific scheduling resolution methods. The simulation results and traces may be graphically represented and stored at the server and can retrieved in different formats such as spreadsheets or pdf files. The simulator structure enables an easy association with the external or internal embedded database systems for storing the historical executions, which allows a comprehensive study of the use cases for different types of grid schedulers. In order to illustrate the impact of the security conditions on the scheduling results, we provided a simple evaluation analysis of the simulator by using two risk-resilient metaheuristic-based schedulers under the varying heterogeneity and large-scale system dynamics.

Table 1
Main attributes of grid scheduling

Attribute	Type	Brief description
Environment	Static	The number of resources is fixed and all of them are available
	Dynamic	The availability of the resources can dynamically change
Grid architecture	Centralized	The schedulers have a full knowledge and control over resources
	Decentralized	No central entity controlling the resources, the local schedulers are responsible for managing and maintaining the tasks
	Hierarchical	The coordination of different schedulers at certain levels, the full knowledge of resources available for the schedulers at the lowest level
Task processing policy	Immediate	Tasks are scheduled as soon as they enter the system
	Batch	Available tasks are grouped into batches and the scheduler assign the batch to the resources
Tasks interrelations	Independency	Tasks are scheduled independently of each other
	Dependency	There are precedence constraints among tasks
Security conditions	Risky mode	All risky and failing conditions are ignored
	Secure mode	All security and resource reliability conditions are verified for the possible task-machine pairs

The rest of the paper is organized as follows. Related work is discussed in Section 2. The main types of the grid scheduling problems are defined in Section 3. The concept of the Secure-Sim-G and its main modules and parameters are presented in Section 4. We report the results of simple experimental analysis in Section 5. We summarize and conclude our work in Section 6.

2. Related Work

Using the simulators for an evaluation of the grid schedulers is feasible, mainly because of high complexity of the grid environment. Many simulation packages, useful in the design and analysis of scheduling algorithms in grid systems, have been recently proposed in the literature. Among many others, MicroGrid [2], ChicSim [3] and Grid-Sim [4] seem to be the most popular in the domain. Some of them are integrated with the grid portals in order to provide the users with an easy access to the simulation packages as well as the online monitoring of the scheduling process. A web-based platform for simulating scheduling methods in grid computing with Grid-Sim package was proposed in [5].

The security aspects in grid scheduling in risky environments are explored in numerous research. Song et al. ([6] and [7]) developed a security aware model in online grid scheduling, where security demand and trust levels are expressed as scalar parameters. Humphrey and Thompson presented [8] a classification of security-aware grid models for an immediate job execution mode. They define a job control system for accessing grid information services through authentication. However, they did not elaborate on how a scheduler should be designed to address the security concerns in collaborative computing over distributed cluster environment. An extensive survey of the research endeavors in this domain is presented in [9].

In [10] the authors present an approach on fault-tolerance method in CG scheduling. They provided a failure detection service, which enables the detection of both task failures and user secure requirements, and a flexible failure handling framework as a fault-tolerant mechanism on the grid. Abawajy [11] developed a model, in which jobs are replicated at multiple grid sites to improve the probability of the satisfaction of the security requirements and successful job executions. Resource reliability and security have been defined as additional scheduling criteria in independent grid scheduling [12], [13], [14]. Matching grid users' security requirements and the "reputation" of the grid clusters impact the behavior and strategies of users, task managers, and resource brokers. A game-theoretical support to the users' decision making activities was presented in our previous work [15].

3. Scheduling problems in CGs

The main purpose of the schedulers is an efficient and optimal allocation of tasks originated by applications to a set of available resources. In dynamic large-scale heterogeneous environments both tasks and resources could be dynamically added/dropped to/from the system. Additionally, various system's entities such as the system's users, managers, and resource providers may operate in different autonomous domains with incoherent local policies. Therefore scheduling in grids is usually considered as a family of highly parametrized problems. The type of the scheduling problem in CGs is specified by setting up the main scheduling attributes presented in Table 1.

In this paper, we focus on an Independent Batch Scheduling in Hierarchical CG problem, where it is assumed that the tasks are grouped into batches and can be executed independently in a hierarchical multi-level grid system in both static and dynamic modes. We consider two possible security scenarios: risky and secure modes. Due to the massive capacity of parallel computation in CGs, this kind of scheduling is very useful in illustrating a lot of realistic scenarios, where the users, independent of each other, submit their jobs to the system, all grid-enabled applications run periodically, and large amounted of data are simultaneously transferred, replicated, and accessed by those applications.

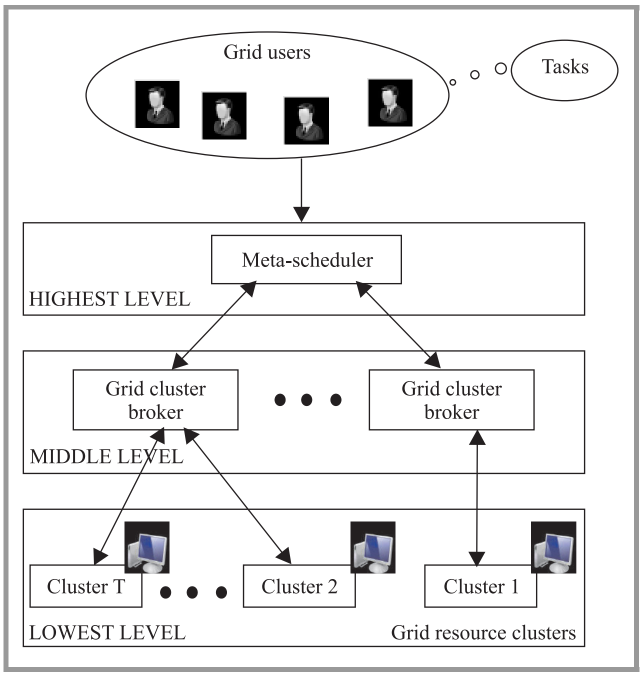


Fig. 1. The model of hierarchic grid architecture

The multi-level large-scale hierarchical structure of CG usually consists of two or three levels as it is shown in Fig. 1.

A central meta-scheduler is the main module of the system that works at the highest level. In today's grid applications a region of activity of the meta-scheduler is in fact restricted to a wide grid cluster, so all global network is managed by few fully cooperated meta-schedulers. The meta-scheduler interacts with local task dispatchers (brokers) and CG users in order to generate the optimal schedules. If "security" is considered as an additional scheduling criterion, the meta-scheduler must analyze the security requirements for the execution of tasks and requests of the CG users for trustful resources available in the system. The system brokers collect information about the "computing capacities" of the resources supplied by the resource owners within the clusters, and additionally analyze the "reputation" indexes of the machines received from the resource managers. They moderate the resources, and send all of the data to the meta-scheduler. The brokers also control the resource allocation and communication between CG users and resource owners.

4. Security Aware Grid Simulator – Basic Concept

To simulate the secure independent batch scheduling we developed a Secure-Sim-G simulator by extending the HyperSim-G framework [1]. HyperSim-G simulator is based on a discrete event model. The sequence of events and the changes in the state of the system capture the realistic grid dynamics. The simulator provides the full simulation trace by indicating a parameter for the trace generation. The main concept of the Secure-Sim-G simulator is presented in Fig. 2.

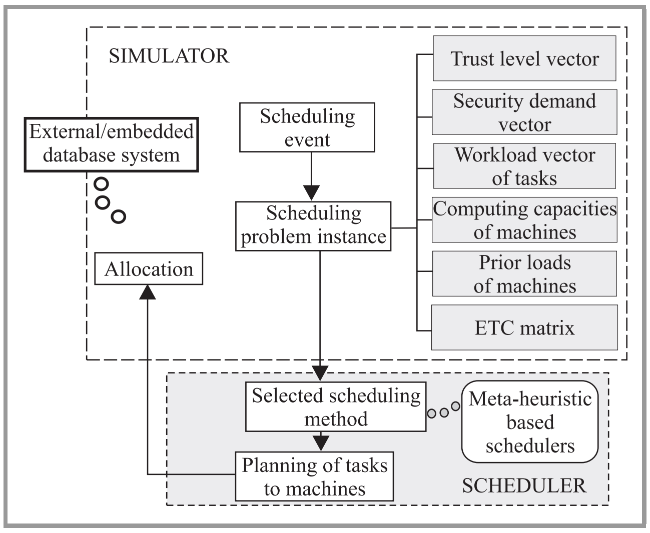

SIMULATOR
External/embedded database system
Scheduling event
Scheduling problem instance
Allocation
Trust level vector
Security demand vector
Workload vector of tasks
Computing capacities of machines
Prior loads of machines
ETC matrix
Selected scheduling method
Meta-heuristic based schedulers
Planning of tasks to machines
SCHEDULER

Fig. 2. General flowchart of the Secure-Sim-G simulator linked to scheduling.

There are two main modules in the system, namely Simulator module and Scheduler module. The main simulation flow can be defined as follows. When a scheduling event is triggered, the simulator creates an instance of the scheduling problem, based on the current task batch and the pool of available machines. The simulator computes an instance of the scheduling and passes it on to a given scheduler which computes the planning of tasks to machines. Finally, the scheduler sends the schedules back to the simulator, which makes the allocation and re-schedules any tasks assigned to machines not available in the system.

The main structure of the Secure-Sim-G application is based on the 2-module HyperSim-G architecture. We modified the Simulator module used in HyperSim-G by defining a security submodule, which allows to specify the security conditions and to define the settings of the risky and secure scheduling scenarios. The secure Simulation submodule is also equipped with the resource failure monitoring system, which reports the unsuccessful resource allocation results due to too strong security requirements. The scheduling methods in Scheduler module defined in HyperSim-G are extended in Secure-Sim-G by the verification of the security conditions and recalculating the values of the objective functions. The software is written in C++ for Linux

Ubuntu 10.10. In the following subsections we briefly characterize the main simulator modules and parameters.

4.1. Input Data and Scheduling Instance

The Secure-Sim-G simulator generates an instance of the scheduling problem by using the following input data:

- the trust level vector of the machines,
- the security demand vector of tasks,
- the workload vector of tasks,
- the computing capacity vector of machines,
- the vector of prior loads of machines,
- the ETC matrix of estimated execution times of tasks on machines.

The number of tasks in a given batch and the number of machines must be also specified. These parameters are constant in the static scheduling and may vary in the dynamic case. For the dynamic scheduling we defined the probability distributions in order to estimate the changes in the system states. We implemented the Constant, Triangle, Normal, Exponential, Trace, Zipf and Uniform distributions for this purpose.
The task in our system are defined as monolithic applications or metatasks with no dependencies among the components. Each task j is characterized by the following parameters:

- wl_j is a computational load of j expressed in millions of instructions per second (MIPS), we denote by $WL = [wl_1,\dots,wl_n]$ a *workload vector* for all tasks in the batch,
- sd_j is a security demand parameter, which is a component of a *security demand vector* $SD = [sd_1,\dots,sd_n]$.

Each machine $i, (i \in M_l)$ in the system is characterized by the following three parameters:

- cc_i – is a computing capacity of i expressed in millions of instructions per second (MIPS), we denote by $CC = [cc_1,\dots,cc_m]$ a *computing capacity vector*,
- $ready_i$ – is a ready time of i, which expresses the time needed for the reloading of the machine i after finishing the last assigned task, a *ready times vector* for all machines is denoted by $ready\_times = [ready_1,\dots,ready_m]$,
- tl_i – is a trust level parameter, which specifies how much a grid user can trust a given resource manager and is the component of a *trust level vector* $TL = [tl_1,\dots,tl_m]$.

The trust level and security demand parameters are expressed as scalar quantities, which are generated by the aggregation of several scheduling and system attributes at users' and resource owners' sites. We base our approach on the fuzzy-logic trust model developed by Song *et al.* [7]. In this model the task security demand is supplied by the user programs as a single parameter. The demand may appear as request for authentication, data encryption, access control, etc.
The values of the sd_j and tl_i parameters are real fractions within the range [0,1] with 0 representing the lowest and 1 the highest security requirements for a task execution and the most risky and fully trusted machine, respectively. A task can be successfully completed at a resource when a *security assurance condition* is satisfied. That is to say that $sd_j \le tl_i$ for a given (j,i) task-machine pair.
SD and TL vectors are used for generation of a *Machine Failure Probability* matrix P_f, the elements of which, are interpreted as the probabilities of failures of the machines during the tasks executions due the high security restrictions. These probabilities, denoted by $P_f[j][i]$, are calculated by using the negative exponential distribution function as follows:

$$P_f[j][i] = \begin{cases} 0, & sd_j \le tl_i \\ 1 - e^{-\alpha(sd_j - tl_i)}, & sd_j > tl_i \end{cases} \tag{1}$$

where α is interpreted as a failure coefficient and is a global parameter of the model.
For estimating the execution times of tasks on machines we used the *Expected Time to Compute (ETC)* matrix model [16]. The elements of the *ETC* matrix, $ETC = [ETC[j][i]]_{n\times m}$ are defined as the expected (estimated) times needed for the completion of the tasks on machines.
In the simplest case, these times can be computed as the ratios of the proper coordinates of WL and CC vectors. That is to say:

$$ETC[j][i] = \frac{wl_j}{cc_i}. \tag{2}$$

All of the values of wl_j and cc_i are generated by using the Gamma or simple Gaussian probability distributions for the expression of tasks and machines heterogeneities in the grid system. In cases when: (a) meta-tasks are submitted by the users and (b) multiprocessor machines are proposed by the resource providers, the values of *ETC* matrix can be computed by using some special local scheduling policies and resolution methods.

4.2. Resolution Methods

The *Secure-Sim-G* simulator allows and facilitates integration of different scheduling implementations. The design of the simulator enables scheduling algorithms to be decoupled from the simulator main body. Various types of the evolutionary based grid schedulers are plugged in the simulator by using and *Adapter* pattern as it is presented in Fig. 3.

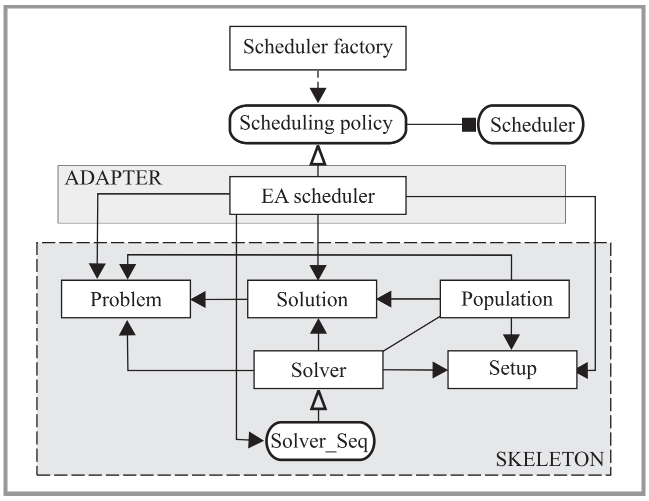


Fig. 3. The simulator adapter pattern used for different evolutionary based grid schedulers

We divided the implemented scheduling heuristic into three main classes, namely *ad hoc*, *local search-based*, *population-based* heuristics.

Ah hoc methods. These methods are usually used for single-objective optimization. They are simple and distinguished from their low computational cost, thus, they are also very useful in generating the initial solutions for population-based schedulers. The ad hoc heuristics could be grouped into an immediate mode heuristics and batch mode heuristics.
The *Immediate Mode Heuristics* group includes, among others, the following schedulers:

- *Opportunistic Load Balancing (OLB)*, where a task is assigned to the earliest idle machine without taking into account its execution time in the machine.
- *Minimum Completion Time (MCT)*, in which a task is assigned to the machine yielding the earliest completion time.
- *Minimum Execution Time (MET)*, in which a task is assigned to the machine having the smallest execution time for that task.

The *Batch Mode Heuristics* group contains, among others, the following methods:

- *Min-Min*: In this method for each task the machine yielding the earliest completion time is computed, then the task with the shortest completion time is selected and mapped to the corresponding machine.
- *Max-Min*: This method differs to the Min-Min in the final selection of the task with the latest completion time.
- *Sufferage*: The main idea of this method is to assign to a given machine a task, which would "suffer' more if it were assigned to any other machine.
- *Relative Cost*: In allocating tasks to machines, this method takes into account both the load balancing of machines and the execution times of tasks in machines.
- *Longest Job to Fastest Resource - Shortest Job to Fastest Resource (LJFR-SRFR)*: This method tries to simultaneously minimize both makespan and flowtime values: LJFR minimizes makespan and SJFR minimizes flowtime.

Local search methods. These methods explore the optimization domain by starting from an initial solution and constructing a path in solution space. The most effective local-based grid scheduler is Tabu Search (TS) due to its mechanisms of tabu lists, aspiration criteria, intensification and diversification (see [17]). TS can be easily hybridized with more sophisticated schedulers (like GAs) to improve their efficiency.

Population-based heuristics. In this methods a population of individuals, which is evaluated, crossed over and mutated, is used to explore the solution space for the problem along a number of generations. The most popular in this group are Genetic Algorithms (GA), proposed by many authors [18], [19], [20]. Recently, a multi-population hierarchical GA-based scheduler has been defined in [21], [22]. In this method a set of dependent genetic processes is executed simultaneously. Each process creates a branch in the tree structure of the whole strategy, by using the GA-based scheduler with different settings. The search accuracy in a given branch (expressed as the branch degree parameter) depends on the mutation probability set for the scheduler activated in this branch (the higher mutation prob. – the lower accuracy).

4.3. Schedule Representation and Scenarios

We use in our approach two different encoding methods of schedules, namely *direct encoding* and *permutation-based encoding.* In the direct encoding the coordinates of a schedule vector $x = [x_1,\ldots,x_{nb\_task}]^T$ are defined as the indexes of machines to which the particular tasks are assigned. In permutation-based encoding we define for each machine a sequence of tasks assigned to that machine. The tasks in the sequence are increasingly sorted with respect to their completion times. Then, all of the task sequences are concatenated into one global vector $u = [u_i,\ldots,u_{nb\_task}]^T$; $u_i \in Tasks$, which is in fact the permutation of tasks to machines. In this representation some additional information about the numbers of tasks assigned to each machine is required. We defined then the vector $v = [v_1,\ldots,v_{nb\_machines}]^T$, in which the numbers of tasks assigned to the following machines are specified as its coordinates.

4.3.1. Optimization Criteria and Objective Function

The problem of scheduling tasks in CG is multiobjective in its general setting as the quality of the solutions can be measured under several criteria. In this work, for the purpose of a simple experimental evaluation of the simulator, we consider the scheduling in CGs as a bi-objective global optimization problem with the hierarchical procedure of the minimization of makespan and flowtime objectives with makespan as a privileged criterion.
Let us denote by F_j the time of finalizing task j and let *Schedules* be a set of directly encoded schedules in a given batch.

- A *makespan* is defined as the finishing time of the latest task in the batch. That is to say:

$$makespan = \min_{s \in Schedules} \max_{j \in N_l} F_j, \tag{3}$$

- We define a *flowtime* as the sum of the finalization times of all the tasks in the batch in the following way:

$$flowtime = \min_{s \in Schedules} \sum_{j \in N_l} F_j\,. \tag{4}$$

Both makespan and flowtime are expressed in arbitrary time units. In fact, the numerical values are in incomparable ranges: flowtime has a higher magnitude order over makespan and its values increase as more jobs and machines are considered. Therefore, in this approach we use $mean\_flowtime = flowtime/m$ for the evaluation of the flowtime criterion.
Using the *ETC* matrix model we can express the makespan and flowtime in terms of the completion times of the machines. The time of finishing the last task can be interpreted as the maximal completion time of the machines. Let us denote by *completion* a vector of the size *nb_machines*, which indicates the time that machine i finalizes the processing of the previously assigned and planned tasks. That is to say:

$$completion[i] = ready_i + \sum_{\substack{j \in N_l: \\ s[j]=i}} ETC[j][i]. \tag{5}$$

The makespan can be now expressed as:

$$makespan = \max_{i \in M_l} completion[i]. \tag{6}$$

We calculate the flowtime of the sequence of tasks on a given machine i by using the following formula:

$$\begin{aligned} flowtime[i] = \quad & ready_i + \\ & + \textstyle\sum_{\substack{j \in Sort[i]: \\ s[j]=i}} ETC[j][i] \end{aligned} \tag{7}$$

where $Sort[i]$ denotes the set of tasks assigned to the machine i sorted in ascending order according to their ETC values.

Having makespan and flowtime as two main scheduling criteria, we define the objective of the scheduling problem as the following function:

$$obj = \lambda \cdot makespan + (1-\lambda) \cdot mean\_flowtime. \tag{8}$$

The weight coordinate λ is used in fact for the specification of the priority of the considered scheduling criteria. Following the experimental tuning results presented in [23] for a classical independent scheduling problem we set the λ value as 0.75. That is to say that in our approach the makespan is the preferred schedulers performance measure. The Secure-Sim-G simulator allows the users to define and integrate the other scheduling criteria, such as the resource utilization and matching proximity [1].

4.3.2. Scheduling Scenarios

To express the impact of the verification of security condition to the scheduling results, we consider two scheduling scenarios, namely *secure* and *risky* modes. In security mode the scheduler analyzes the *Machine Failure Probability* matrix in order to minimize the failure probabilities for task-machine pairs. We assume that additional "cost" of the verification of security assurance condition for a given task-machine pair: (a) may delay the predicted execution time of the task on the machine and (b) is proportional to the probability of failure of the machine during the task execution. We define this "cost" as a product $P_f[j][i] \cdot ETC[j][i]$ and the completion time of the machine i can be calculated as follows:

$$\begin{aligned} & completion[i] = ready\_time[i] + \\ & + \sum_{\{j \in Tasks_i\}} (1 + P_f[j][i]) ETC[j][i]\,, \end{aligned} \tag{9}$$

where $Tasks_i$ denotes a set of tasks assigned to the machine i in a given batch.
In risky mode the scheduler performs as an "ordinary" scheduler without any prior analysis of the security conditions. It aborts the task scheduling in the case of machine failure, and reschedules this task at another resource. It means that the scheduling is performed just by analyzing the *ETC* matrix. If failures are observed during some tasks executions, then the unfinished tasks are temporarily moved into the backlog set. This set is defined as a considered batch supplement and the tasks are re-scheduled as in the secure mode. The total completion time of machine i in this case can be defined as follows:

$$\begin{aligned} & completion^r[i] = \\ & completion_{(I)}[i] + completion_{(II)}[i]\,, \end{aligned} \tag{10}$$

where $completion_{(I)}$ is the completion time of machine i calculated by using the Eq. (5) for tasks primarily assigned to the machine, and $completion_{(II)}$ is the completion time of machine i calculated by using the Eq. (9) for rescheduled tasks, i.e., the tasks moved to the machine i from the other resources.

5. Experimental Analysis

In this section we present the results of a simple experimental evaluation of two variants of genetic-based grid schedulers working in risky and secure modes in static and dynamic grid environments. The setting and configuration of GA scheduler are presented in Table 2.

Table 2
GA settings for large static and dynamic benchmarks

Parameter	Value
Evolution steps	$5 \times (nb\_jobs)$
Population size (pop_size)	$\lceil(\log_2(nb\_jobs))^2 - \log_2(nb\_jobs)\rceil$
Intermediate pop.	$pop\_size - 2$
Selection method	LinearRanking
Crossover method	Cycle Crossover
Cross probab.	0.9
Mutation method	Rebalancing
Mutation probab.	0.2
replace_only_if_better	false
replace_generational	false
Initialization	LJFR-SJFR + MCT + Random
max_time_to_spend	40 s $(static)$ / 25 s $(dynamic)$

The detailed definition of the genetic operators used in GA configuration can be found in [24].

The Secure-Sim-G simulator is highly parametrized to reflect the various realistic grid scenarios. The values of key input parameters[1] for the simulator are presented in Table 3.

We considered the following four grid size scenarios in our study: (a) small grid (32 hosts/512 tasks), (b) medium grid (64 hosts/1024 tasks), (c) large grid (128 hosts/2048 tasks), and (d) very large grid (256 hosts/4096 tasks).

The user can specify his own scenario by changing the number of tasks and machines. The capacity of the resources and the workload of tasks are randomly generated by a normal distribution. It is also assumed that all tasks submitted to the system must be scheduled and all machines in the system can be used.

The number of hosts initially activated in the grid environment is defined by the parameter *Init. number of hosts*. The parameters *Max.hosts* and *Min.hosts* specify the range of changes in the number of active hosts during the simulation process[2]. The frequency of appearing and disappearing resources is defined by *Add host* and *Delete host*, according to constant distributions for the static case, and normal distributions in dynamic case. The initial number of tasks is given by *Init. tasks*, which is kept constant in the static case. New tasks in the dynamic scheduling can arrive at the system with the frequency *Interarrival* until *Total tasks* is reached. The *Activation* parameter establishes the activation policy (it is usually modeled by an exponential distribution in the dynamic case). The assigned tasks which have not been executed yet cannot be rescheduled if the value of the boolean parameter *Reschedule* is false. The *Scheduler strategy* parameter denotes the Scheduler type. Its value $GA\_Scheduler(25,s)$ means that the simulator runs the GA-based scheduler for 25 s in simultaneous optimization mode[3].

We used the following three metrics to evaluate the scheduling performance:

- *Makespan* (see Eq. 3) for secure and risky scenarios,
- $Flowtime$ (see Eq. 4) for secure and risky scenarios,
- *FailureRate* F_r parameter defined as follows:

$$F_r = \frac{n_{failed}}{n} \cdot 100\%, \tag{11}$$

where n_{failed} denotes the number of unfinished tasks, which must be rescheduled.

Each experiment was repeated 30 times under the same configuration of operators and parameters.

5.1. Results

The results of the experiments expressed by the averaged flowtime and makespan values achieved by two variants of security-aware GA-based schedulers in static and dynamic cases are presented in Fig. 4.

It can be observed that in both static and dynamic cases the secure version of GA-based scheduler, namely *GA-Secure* algorithm, outperforms its risky variant. The differences are significant especially for the makespan values. The results suggest that it is more resilient for the grid users to pay some additional scheduling cost due to verification of the security conditions instead of taking a risk and allocating them at untrustful resources. It can be also noted that as the instance size is doubled, the flowtime values increase considerably for all applied schedulers, while the makespan is almost at the same level. The good results in secure scenario are confirmed by the low failure rates achieved by the *GA-Secure* scheduler presented in Table 4.

In all instances the values achieved by secure scheduler are approximately two times lower than in risky scenario,

[1] We use the notation $U[x,y]$, $N(a,b)$ and $E(c,d)$ for uniform, Gaussian and exponential probability distributions respectively.

[2] In the case of dynamic scheduling, they are different from the initial number of hosts.

[3] Similarly, the parameter h can be used to indicate hierarchic mode optimization, e.g., $GA\_Scheduler(25,h)$.

Table 3
Values of key parameters of the grid simulator in static and dynamic cases

Parameter	Small	Medium	Large	Very large
Static case				
Number of hosts	32	64	128	256
Resource cap. (in MHz CPU)	$N(5000, 875)$			
Total nb. of tasks	512	1024	2048	4096
Workload of tasks	$N(250000000, 43750000)$			
Security demandssd$_j$	$U[0.6; 0.9]$			
Truest levels tl$_i$	$U[0.3; 1]$			
Failure coefficient α	3			
Dynamic case				
Init. number of hosts	32	64	128	256
Max.hosts	37	70	135	264
Min.hosts	27	58	121	248
Resource cap. (in MHz CPU)	$N(5000, 875)$			
Add host	$N(625000, 93750)$	$N(562500, 84375)$	$N(500000, 75000)$	$N(437500, 65625)$
Delete host	$N(625000, 93750)$			
Init. tasks	384	768	1536	3072
Total tasks	512	1024	2048	4096
Interarrival	$E(7812.5)$	$E(3906.25)$	$E(1953.125)$	$E(976.5625)$
Workload	$N(250000000, 43750000)$			
Security demandssd$_j$	$U[0.6; 0.9]$			
Trust levels tl$_i$	$U[0.3; 1]$			
Failure coefficient α	3			

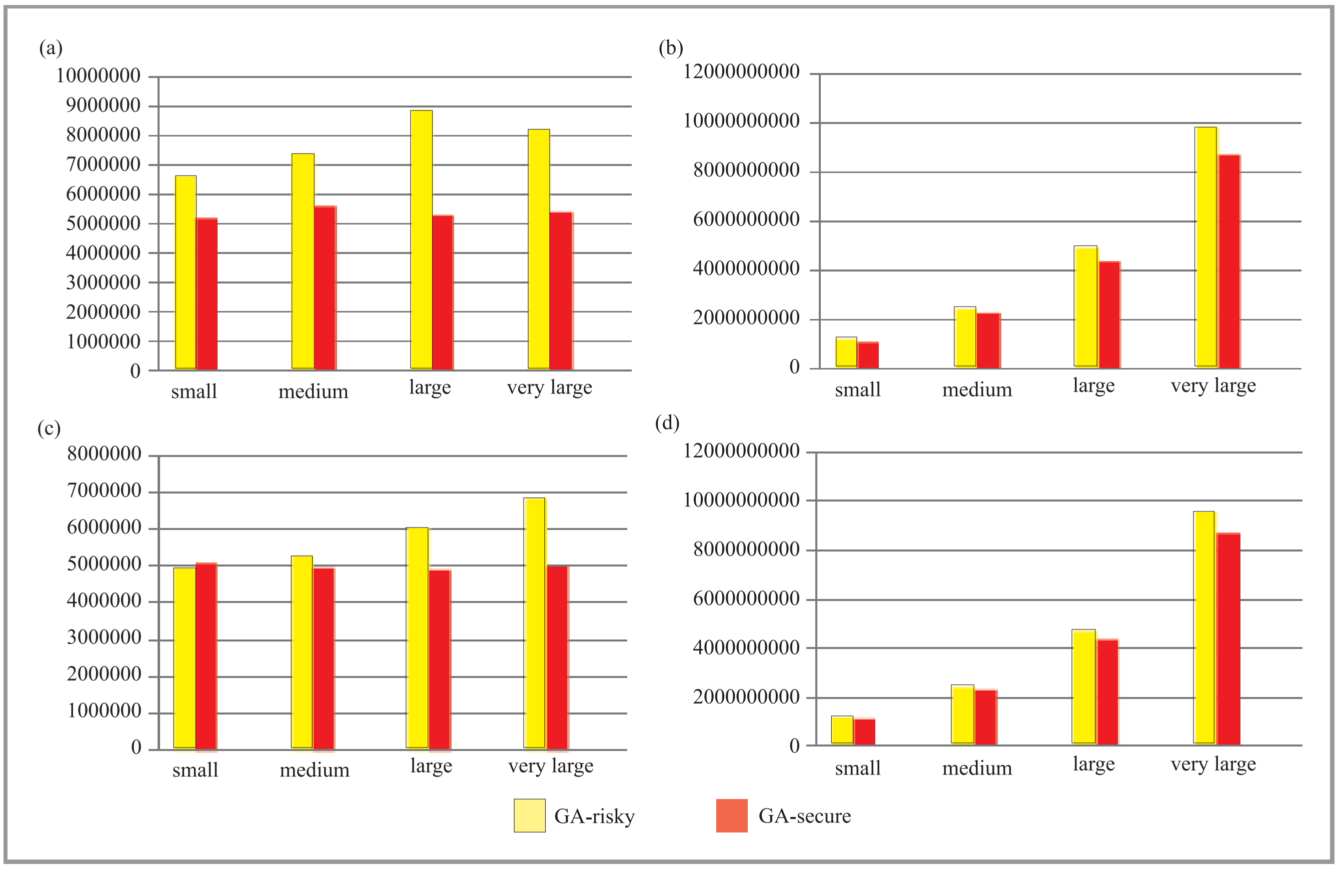


Fig. 4. Experimental results achieved by security-aware GA-schedulers: in static case – (a) average makespan, (b) average flowtime; in dynamic case – (c) average makespan, (d) average flowtime.

what may explain the additional scheduling costs in this case, i.e., many tasks have been re-scheduled.

Table 4
Average values of failure rate parameter
for six GA-based schedulers

Strategy	Small	Medium	Large	Very large
Static Instances				
GA-Risky	15.632	18.405	9.351	20.345
GA-Secure	5.682	9.415	6.134	8.435
Dynamic Instances				
GA-Risky	19.522	24.265	28.563	25.455
GA-Secure	10.223	12.635	11.546	10.535

6. Conclusions

In this paper we presented the main concept, architecture and parameters of the *Secure-Sim-G* grid simulator for independent batch scheduling, which allows the evaluation of the different scheduling heuristics under various scheduling criteria in several grid scenarios defined by the security conditions, grid size and system dynamics. The simulator is an extension and modification of the event-based *HyperSim-G* framework [1] With *Secure-Sim-G* the user can flexibly activate and inactivate all the scheduling criteria and modules, which makes the application well adapted to the proper illustration of the different realistic scenarios and reduces the possible restriction to the specific scheduling resolution methods.

We provided a simple evaluation analysis of the simulator by using two risk-resilient metaheuristic-based schedulers under the varying heterogeneity and large-scale system dynamics. The results suggest that it is more resilient for the grid users to pay some additional scheduling cost due to verification of the security conditions instead of taking a risk and allocating them at untrustful resources.

The presented software package can be easily extended by plugging in additional scheduling methods and scheduling criteria, like energy consumption, which a hot research topic in intelligent green networking ([25]–[32]). It can be also adapted to the scheduling simulation in cloud systems, which will be our next research effort.

References

[1] F. Xhafa, J. Carretero, L. Barolli, and A. Durresi, "Requirements for an event-based simulation package for grid systems". *J. Interconnection Netw.*, vol. 8, no 2, pp. 163–178, 2007.

[2] H. J. Song, X. Liu, O. Jakobsen, R. Bhagwan, X. Zhang, K. Taura, and A. Chien, "The microgrid: a scientific tool for modeling computational grids", *J. Sci. Program.*, vol. 8, no. 3, pp. 127–141, 2000.

[3] K. Ranganathan and I. Foster, "Simulation studies of computation and data scheduling algorithms for data grids", *J. Grid Comput.*, vol. 1, no. 1, pp. 53–62, 2003.

[4] R. Buyya, and M. M. Murshed, "Grid-Sim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing", *Concurrency and Comput.: Practice and Experience*, vol. 14, no. 13–15, pp. 1175–1220, 2002.

[5] O. Kang and S. Kang, "Web-based dynamic scheduling platform for grid computing", *Intern. J. Comp. Sci. Netwo. Secur.*, vol. 6, no. 5, pp. 67–75, 2006.

[6] S. Song, K. Hwang, and Y. K. Kwok, "Risk-resilient heuristics and genetic algorithms for security- assured grid job scheduling", *IEEE Trans. Comput.*, vol. 55, no. 6, pp. 703–719, 2006.

[7] S. Song, K. Hwang, and Y. K. Kwok, "Trusted grid computing with security binding and trust integration", *J. Grid Comput.*, vol. 3, no. 1-2, pp. 53–73, 2005.

[8] M. Humphrey and M. R. Thompson, "Security implications of typical grid computing usage scenarios", in *Proc. 10th IEEE Int. Symp. High Performa. Distrib. Comput.*, San Francisco, CA , USA, 2001.

[9] E. Cody, R. Sharman, R. H. Rao, and S. Upadhyaya, "Security in grid computing: a review and synthesis", *Decision Supp. Sys.*, vol. 44, pp. 749–764, 2008.

[10] S. Hwang and C. Kesselman, "A flexible framework for fault tolerance in the grid", *J. Grid Comput.*, vol. 1, no. 3, pp. 251–272, 2003.

[11] J. Abawajy, "An efficient adaptive scheduling policy for high performance computing", *Future Gener. Comp. Sys.*, vol. 25, no. 3, pp. 364–370, 2009.

[12] J. Kołodziej and F. Xhafa, "Meeting security and user behaviour requirements in grid scheduling", *Simul. Modell. Practice and Theory*, vol. 19, no. 1, pp. 213–226, 2011.

[13] J. Kołodziej, F. Xhafa, and M. Bogdański, "Secure and task abortion aware ga-based hybrid metaheuristics for grid scheduling", in *PPSN XI*, Schaefer *et al.*, Eds. LNCS, vol. 6238, 2010, pp. 526–535.

[14] J. Kołodziej and F. Xhafa, "Integration of Task Abortion and Security Requirements in GA-based Meta-Heuristics for Independent Batch Grid Scheduling", *Computers and Mathematics with Applications*, DOI: 10.1016/j.camwa.2011.07.038, 2011.

[15] J. Kołodziej and F. Xhafa, "A game-theoretic and hybrid genetic meta-heuristic model for security-assured scheduling of independent jobs in computational grids", in *Proc. CISIS 2010*, USA: IEEE Press, 2010, pp. 93–100.

[16] S. Ali, H. J. Siegel, M. Maheswaran, and D. Hensgen, "Task execution time modeling for heterogeneous computing systems", in *Proc. 9th Heterogen. Comput. Worksh. HCW 2000*, Cancun, Mexico, 2000, pp. 185–199.

[17] F. Xhafa, J. Carretero, E. Alba, and B. Dorronsoro, "Tabu search algorithm for scheduling independent jobs in computational grids", *Comp. Informatics J.*, special issue on *Intelligent Computational Methods*, J. Burguillo-Rial, J. Kołodziej, and L. Nolle, Eds., vol. 28, no. 2, pp 237–249, 2009.

[18] F. Xhafa and A. Abraham, "Meta-heuristics for grid scheduling problems", in *Meta-heuristics for Scheduling in Distributed Computing Environments*, Chapter 1, Series *Studies in Computational Intelligence*, Springer, 2009, pp. 1–37.

[19] F. Xhafa, J. Carretero, and A. Abraham, "Genetic algorithm based schedulers for grid computing systems", *Int. J. Innovative Comput. Informa. Control*, vol. 5, pp. 1–19, 2007.

[20] F. Pinel, J. E. Pecero, P. Bouvry, and S. U. Khan, "A two-phase heuristic for the scheduling of independent tasks on computational grids", in *Proc. ACM/IEEE/IFIP Int. Conf. High Performa. Comput. Simulation HPCS 2011*, Istanbul, Turkey, 2011.

[21] J. Kołodziej, F. Xhafa, and Ł. Kolanko, "Hierarchic genetic scheduler of independent jobs in computational grid environment", in *Proc. 23rd Euro. Conf. Modell. Simulation ECMS 2009*, Madrid, Spain, 2009, J. Otamendi, A. Bargieła, J. L. Montes and L. M. Doncel Pedrera, Eds. Dudweiler, Germany: IEEE Press, 2009, pp. 108–115.

[22] J. Kołodziej and F. Xhafa, "Enhancing the genetic-based scheduling in computational grids by a structured hierarchical population", *Future Generation Computer Systems*, vol. 27 (2011), pp. 1035–1046, DOI: 10.1016/j.future.2011.04.011, 2011.

[23] F. Xhafa, J. Carretero, and A. Abraham, "Genetic algorithm based schedulers for grid computing systems", *Int. J. Innovative Comput. Informa. Control*, vol. 3, no. 5, pp. 1053–1071, 2007.

[24] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Springer, 1992.

[25] J. Kołodziej, S. U. Khan, and F. Xhafa, "Genetic algorithms for energy-aware scheduling in computational grids", in *Proc. 6th IEEE Int. Conf. P2P, Parallel, Grid, Cloud, Internet Comput. 3PGCIC*, Barcelona, Spain, 2011.

[26] D. Kliazovich, P. Bouvry, Y. Audzevich, and S. U. Khan, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers", in *Proc. 53rd IEEE Global Commun. Conf. Globecom 2010*, Miami, FL, USA, 2010.

[27] S. U. Khan and I. Ahmad, "A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids", *IEEE Trans. Parallel and Distrib. Sys.*, vol. 20, no. 3, pp. 346–360, 2009.

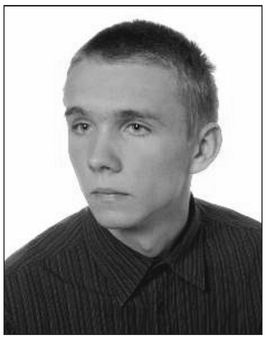
[28] S. U. Khan, "A goal programming approach for the joint optimization of energy consumption and response time in computational grids", in *Proc. 28th IEEE Int. Performa. Comput. Commun. Conf. IPCCC 2009*, Phoenix, AZ, USA, 2009, pp. 410–417.

[29] S. U. Khan and I. Ahmad, "Non-cooperative, semi-cooperative, and cooperative games-based grid resource allocation", in *Proc. 20th IEEE Int. Parallel Distrib. Process. Symp. IPDPS 2006*, Rhodes Island, Greece, 2006.

[30] G. L. Valentini *et al.*, "An overview of energy efficiency techniques in cluster computing systems", *Cluster Computing*, DOI: 10.1007/s10586-011-0171-x, 2011.

[31] L. Wang and S. U. Khan, "Review of performance metrics for green data centers: a taxonomy study", *J. Supercomput.*, pp. 1–18. DOI:10.1007/s11227-011-0704-3, 2011.

[32] S. Zeadally, S. U. Khan, and N. Chilamkurti, "Energy-efficient networking: past, present, and future", *J. Supercomput.*, pp. 1–26. DOI:10.1007/s11227-011-0632-2, 2011.



Grzegorz Gębczyński is an undergraduate student of Computer Science at the Faculty of Mechanical Engineering and Computer Science at the University of Bielsko-Biała. The topics of his engineering diploma thesis are the dynamic routing, trust and decentralized management in IPv6 networks.

E-mail: grzegorz.gebczynski@gmail.com
Faculty of Mechanical Engineering and Computer Science
University of Bielsko-Biała
Willowa st 2
43–309 Bielsk-Biała, Poland

Joanna Kołodziej graduated in Mathematics from the Jagiellonian University in Kraków in 1992, where she also obtained the 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 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 Managing Editor of IJSSC Journal and serves as a EB member and guest editor of several peer-reviewed international journals.

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

Samee Ullah Khan is an Assistant Professor of Electrical and Computer Engineering at the North Dakota State University, Fargo, ND, USA. He has extensively worked on the general topic of resource allocation in autonomous heterogeneous distributed computing systems. As of recent, he has been actively conducting cuttingedge research on energy-efficient computations and communications. A total of 115 (journal: 40, conference: 54, book chapter: 12, editorial: 6, technical report: 3) publications are attributed to his name. For more information, please visit: http://sameekhan.org/.

E-mail: samee.khan@ndsu.edu
Department of Electrical and Computer Engineering
North Dakota State University
ND 58108; USA