Genetic Algorithms for Energy-aware Scheduling in Computational Grids

Joanna Kołodziej
Department of Mathematics and Computer Science
University of Bielsko-Biała
Bielsko-Biała, Poland
Email: jkolodziej@ath.bielsko.pl

Samee Ullah Khan
Department of Electrical and Computer Engineering
North Dakota State University
ND 58108, USA
Email: samee.khan@ndsu.edu

Fatos Xhafa
Department of Languages and Informatics Systems
Technical University of Catalonia
Barcelona, Spain
Email: fatos@lsi.upc.edu

***Abstract*—Because of its sheer size, Computational Grids (CGs) require advanced methodologies and strategies to efficiently schedule users tasks and applications to resources. Scheduling becomes even more challenging when energy-efficiency, classical makespan criterion and user perceived Quality of Service (QoS) are treated as first-class objectives in CG resource allocation methodologies. In this paper we approach the independent batch scheduling in CG as a bi-objective minimization problem with makespan and energy consumption as the scheduling criteria. We use the Dynamic Voltage Scaling (DVS) methodology for reducing the cumulative power energy utilized by the system resources. We develop two Genetic Algorithms (GAs) with elitist and struggle replacement mechanisms as energy-aware schedulers. The proposed algorithms were experimentally evaluated for four CG size scenarios in static and dynamic modes. The simulation results showed that our proposed GA-based schedulers fairly reduce the energy usage to a level that is sufficient to maintain the desired quality level(-s).**

***Keywords*-Computational Grids, Scheduling, Multi-objective Optimization, Energy-awareness, Makespan**

I. Introduction

Computational Grids (CGs) are large-scale distributed systems that virtually combine geographically distributed IT resources from many different administrative domains into one single customized computational infrastructure. Grid resources may belong to different owners or individuals who can impose various usage policies on CG users based on different sets of rules and configuration directives.

While the CGs have been widely promoted for high performance computations and as cheap alternative to super-computers, we are witnessing a disproportion of resource availability and resource provisioning. A severe increase in energy consumption to match resource provisioning in light of the computational demands [Khan and Ahmad, 2009] is observed. Therefore, the current efforts in the grid computing research is to design new effective grid schedulers, which can simultaneously minimize not only *old* parameters, such as makespan, flowtime and resource usage but also the energy consumption between all the different resource providers in the system. That is, while keeping the main purpose of the grid schedulers to efficiently and optimally allocate tasks originated by applications to a set of available resources, one should consider a series of requirements including energy efficiency. Energy-efficient scheduling in CGs becomes thus a complex endeavor due to the multi-constraints and different optimization criteria and different priorities of the resource owners.

Related work

In the recent past, heuristic methods, such as Simulated Annealing (SA), Tabu Search (TS), and Genetic Algorithms (GA), have been successfully applied to solve scheduling problems in CGs, mainly focusing to system performance optimization. Braun et al. [Braun et al., 2001] compare the efficiency of a simple GA-based scheduler and methods from the set of ten static meta-task mapping heuristics from the literature, including Min-Min, Min-Max, Minimum Completion Time algorithms. The experimental study were performed for the static benchmark for Independent Job Scheduling in distributed heterogeneous computing environment. The instances in this benchmark were defined using the ETC matrix model [Ali et al., 2000]. The same type of scheduling problem is considered by Xhafa et al. [Xhafa et al., 2007], where the authors examine several variations of GAs operators in order to identify a configuration of operators and parameters that works best for the problem. Then, the efficiency of GA-based scheduler with the appropriate combination of operators is compared to the effectiveness of the GAs approach presented in [Braun et al., 2001]. The GA scheduler has been plugged into a Grid simulator to perform the experiments in a dynamic environment to capture realistic features of CGs.

Energy efficiency is an emerging research issue, recently addressed by several researchers. Khan and Ahmad [Khan and Ahmad, 2009] were the first to use game theoretical methodologies to simultaneously optimize system performance and energy consumption. Since then several research works have used similar models and approaches, which have addressed a mix of research problems related to large-scale computing systems, such as energy proportionality, memory-aware computations, data intensive computations, energy-efficient, and grid scheduling [Khan, 2009], [Guzek et al., 2010], [Pinel at al., 2010], [Kliazovich et al., 2010],

[Zomaya, 2009], [Lee and Zomaya, 2009]. (The reader is referred to [Beloglazov et al., 2011] for a recent survey.)

Our contribution

In this paper we consider the *Independent Batch Job Scheduling* problem in CGs, in which tasks are processed in a batch mode and there are no dependencies among tasks. This is a simplest scheduling scenario in CGs, which is useful to illustrate many real-life approaches, mainly due the parallel nature of CG enabling independent computation of tasks on grid resources. The paper contributed in three aspects:

- Specification of the grid scheduling as a bi-objective discrete global optimization problem aiming to minimize the makespan and the system energy consumption.
- Design of two new energy-aware GA-based schedulers that are integrated within the HyperSim-G simulator.
- Evaluation of the proposed schedulers in static and dynamic grid scenarios and measurement of their performance by observing the makespan and relative energy consumption improvement rate.

The remainder of this paper is structured as follows. We define the scheduling problem in Section II. The GA framework and genetic operators are presented in Section III. Section IV presents a brief characteristics of the HyperSim-G simulator extended by the dynamic voltage scaling mechanism and the experimental evaluation of the proposed genetic schedulers. Finally, Section V concludes this paper.

II. Statement of the Problem

A. System model

CGs are usually modeled as a multi-level large-scale hierarchical system, which is a compromise between centralized and decentralized task and resource management methodologies. There is a central meta-scheduler in such systems, who interacts with local task dispatchers (brokers) to define the optimal schedules. The local brokers have limited knowledge about grid resource clusters. Therefore, the brokers cannot monitor the whole system and just collect the information about the "computing capacity" of the resource supplied by the resource owners, moderate the resources, and send the information to the scheduler. The global scheduler must schedule tasks using the information received from the local brokers. Replicas of the defined schedule are sent back to the brokers for the resource allocation. The CG hierarchy usually consists of three levels (see Fig. 1).

The hierarchical grid model is well suited to capture realistic administrative features of a real-life large-scale CGs, in which many complex scheduling criteria, such as security, resource reliability, users preferences, specific budget and time constraints, and resource and energy utilization are concerned. In this work, we focus on two levels on the grid system: level I, in which meta-scheduler operates, and level III with multiple sites of grid resources. The transmission

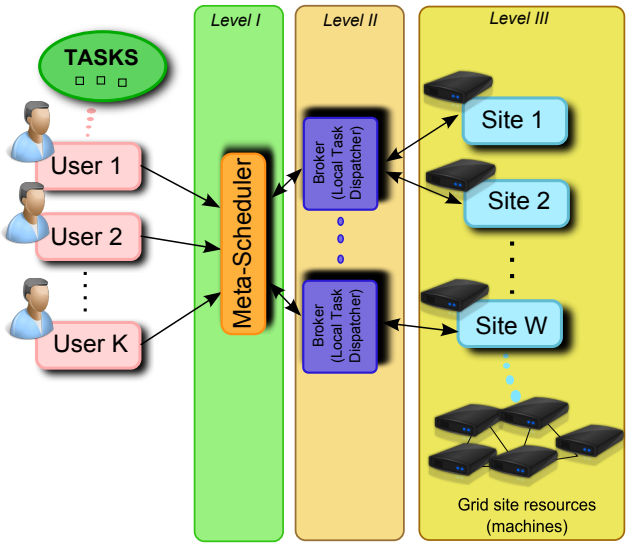

Figure 1. The model of hierarchic grid architecture

processes between scheduler and local brokers and local brokers and resource providers are not analyzed in this paper. However the proposed methodology can easily be extended to the general grid scenario. We assume that the tasks submitted by the users are processed in a batch mode and batches are created at level I.

The following notation is used throughout the paper:

- n — the number of tasks in a batch;
- m — the number of machines available in the system for an execution of a given batch of tasks;
- $N = \{t_1, \ldots, t_n\}$ — set of tasks in a batch;
- $M = \{x_1, \ldots, x_m\}$ — set of available machines for the task batch;
- $N_l = \{1, \ldots, n\}$ — a set of tasks labels;
- $M_l = \{1, \ldots, m\}$ — a set of machines labels.

Task characteristics: We define tasks as monolithic applications with no dependencies among them. Each task j is characterized by its computational load parameter wl_j expressed in Millions Instructions Per Second (MIPS). Let us denote by $WL = [wl_1, \ldots, wl_n]$ a *workload vector* for all of the tasks in the batch.

Machine representation: For energy-assured scheduling, we assume that each machine is equipped with a Dynamic Voltage Scaling (DVS) module [Lorch and Smith, 2001], which allows to modulate its supply voltage and operating frequency (that is approximately proportionate to the processing speed (see [Mejia-Alvarez et al., 2004])). The decrease in the supply voltage and frequency reduces the energy consumed by the machine. We consider that grid resources can be classified into the various 'power supply' categories according to the maximal supply voltage values. For each of the aforementioned classes, we define a vector of DVS levels with reduced values of the supply voltage and

the corresponded relative machine frequency parameters. We assume that the supply voltage cannot be changed during the task execution, but can be different for different tasks.

Formally, a machine $m_k \in M$ in our Grid system is represented as a triplet $m_k =< cc_k; ready_k; s^k >$, where:

- cc_k is the computing capacity parameter of the machine m_k expressed in MIPS, we denote by $CC = [cc_1, \ldots, cc_m]$ a *computing capacity vector*;
- $ready_k$ is the ready time value for the machine m_k, which estimates the time needed for the reloading of the machine m_k after finishing the last assigned job; we denote by $ready\_times = [ready_1, \ldots, ready_m]$ a *ready times vector*;
- s^k denotes its 'power supply' category (class) for the machine m_k.

For each s^k class we define a column meta-vector $Vr_{(k)}$ of possible DVS levels in the following way:

$$Vr_{(k)} = \left[(v_{s_0}(k), f_{s_0}(k)); \ldots; (v_{s_{l(max)}}(k), f_{s_{l(max)}}(k))\right]^T \quad (1)$$

It should be noted that for lower supply voltage the operation frequency of the machine decreases, which means that $f_{s_l}(k)$ coefficients are within the range of [0,1].

Estimation of the tasks computational times on machines: For estimating the execution times of tasks on machines, we based our methodology on the *Expected Time to Compute (ETC)* matrix model [Ali et al., 2000], adopted to the independent batch scheduling. In this model it is assumed that: (i) a task can only be executed on one grid node in each batch; (ii) no preemptive process is allowed within tasks or resources; (iii) when a machine fails, its tasks will be re-scheduled in the next batch; (iv) when a machine processes its tasks, there is no priority distinctions between the tasks assigned in the previous batches and those assigned in the current batch; and (v) a machine cannot remain idle when tasks have been assigned to it. The main structure in this model is an *ETC* matrix, $ETC = [ETC[j][k]]_{n\times m}$, in which the term $ETC[j][k]$ denotes an expected time needed for the completion of the task j on machine k. For instance, the entries of the *ETC* matrix can be computed as the ratio of the coordinates of WL and CC vectors, i.e.:

$$ETC[j][k] = \frac{wl_j}{cc_k}. \quad (2)$$

All of the wl_j and cc_k values are generated by using the Gamma probability distribution (or its special case — standard Gauss distribution) for the expression of tasks and machines heterogeneities in the grid system. A possible decreasing of the machine frequency and supply voltage leads to the increased computational times of the tasks executed on the machine. Because the time to complete the task is inversely proportional to the machine frequency, we assume that the inversions of the $f_{s_l}(k)$ coefficients approximately estimates the growth coefficients of the completion times of tasks on machine of the class s^k. Assuming $l(max)$ number of DVS levels in class s^k, for each $(t_j; m_k)$ 'task-machine' pair we can define a vector of estimated task completion times at the DSV levels as follows:

$$\widehat{ETC}[j][k] = \\ = \left[\frac{1}{f_{s_0}(k)} \cdot ETC[j][k], \ldots, \frac{1}{f_{s_{l(max)}}(k)} \cdot ETC[j][k]\right] \quad (3)$$

where $ETC[j][k]$ times are calculated according to the Eq. (2) and $\{f_{s_0}(k), \ldots, f_{s_{l(max)}}(k)\}$ are the relative frequencies of the machine k, specified for the machine category s^k at the $s_0, \ldots, s_{l(max)}$ DVS levels.

We can construct for the energy-aware scheduling model an *ETC* meta-matrix by replacing for each of the pair $(t_j; m_k)$ the $ETC[j][k]$ values, i.e.:

$$\widehat{ETC} = \left[\widehat{ETC}[j][k]\right]_{j\in N_l; k\in M_l}. \quad (4)$$

Energy model: Our energy model is based on the power consumption model in complementary metal-oxide semiconductor (CMOS) logic circuits. The power consumption of a CMOS-based microprocessor is defined as a sum of the capacitive, short-circuit and leakage power. The most significant factor is the capacitive power, which can be interpreted as the dynamic power consumption (see [Ge et al., 2005]). The power consumption P_{kj} of machine m_k during the execution of the task t_j is calculated in the following way:

$$P_{kj} = A \cdot C \cdot v^2 \cdot f, \quad (5)$$

where A is the number of switches per clock cycle, C is the total capacitance load, v is the supply voltage and f is the machine frequency.

The energy consumed by machine m_k for the completion of the task t_j can be defined as follows:

$$E_{jk} = \gamma \cdot (f_{s_l}(k))_j \cdot f \cdot [(v_{s_l}(k))_j]^2 \cdot \widehat{ETC}[j][k][s_l], \quad (6)$$

where $\gamma = A \cdot C$ is assumed constant for a given machine; $(v_{s_l}(k))_j$ is a voltage supply value for the class s^k of the machine m_k at the DVS level s_l for computing the task t_j; $(f_{s_l}(k))_j$ is a corresponding relative frequency of machine m_k at the level s_l set for the computing task t_j; and $\widehat{ETC}[j][k][s_l]$ is an s_l-th coordinate of the $\widehat{ETC}[j][k]$ vector. Using Eq. (3) for estimating the growth of the computation times of t_j at particular DVS levels within the considered machine class s^k, the Eq. (6) is written as follows:

$$E_{\{j,k,l\}} = \\ = \gamma \cdot (f_{s_l}(k))_j \cdot f \cdot [(v_{s_l}(k))_j]^2 \cdot (f_{s_1}(k))_j \cdot ETC[j][k] \\ = \gamma \cdot f \cdot [(v_{s_l}(k))_j]^2 \cdot ETC[j][k] \quad (7)$$

Let $T(k)$ denote a set of tasks assigned to the machine m_k. The cumulative energy consumed by this machine for the completion of all tasks from $T(k)$ is defined as:

$$E_k = \sum_{\substack{j\in T(k)\\ l\in L_j}}\{E_{\{j,k,l\}}\} + \gamma\cdot f\cdot[v_{s_{max}}]^2\cdot ready_k + \\ +\gamma\cdot f_{s_{min}}(k)\cdot f\cdot[v_{s_{min}}(k)]^2\cdot Idle_k = \gamma\cdot f\cdot \\ \cdot\left[\begin{array}{l}\sum_{\substack{j\in T(k)\\ l\in L_k}}([(v_{s_l}(k))_j]^2\cdot ETC[j][k]) + \\ +[v_{s_{max}}(k)]^2\cdot ready_k + f_{s_{min}}(k)\cdot[v_{s_{min}}(k)]^2\cdot Idle_k\end{array}\right] \quad (8)$$

where $ready_k$ is the ready time of the machine m_k, $Idle_k$ is an idle time of the machine m_k and L_k denotes a subset of DVS levels for used for the tasks assigned to machine m_k. We assume that each machine starts the execution of a set of assigned tasks at the maximal voltage supply level $v_{s_{max}}$ and maximal relative frequency which is 1 in each machine class s^k. In the idle time the machine turns into the sleep mode and thus the voltage supply $v_{s_{min}}(k)$ and relative frequency $f_{s_{min}}(k)$ are at the lowest level. Because the machine frequency transition overheads take usually a negligible amount of time (e.g., 10ms- 150ms, see [Min et al., 2000]), these overheads are not considered in this paper and the inclusion of such an overhead will have no bearing on the overall model of the proposed study.

An average cumulative energy utilized by the grid system for completion of the whole task batch can be defined as follows:

$$E_{batch} = \frac{\sum_{k\in M} E_k}{m} \quad (9)$$

B. Scheduling problem definition

In the multi-objective optimization methodologies, two basic models are utilized: hierarchical and simultaneous modes. In the *simultaneous mode* all objectives are optimized simultaneously and defined as a cumulative weighed function with the weight coordinates, which express the priorities in the optimization criteria. In the *hierarchical* case, the objectives are sorted *a priori* according to their importance in the model. The process starts by optimizing the most important objective. When further improvements are not possible, the second objective is optimized, without worsening the value of the first one. The method proceeds until all objectives are optimized or some stopping criteria is met.

In this work we approach the energy-aware scheduling in grids as a bi-objective combinatorial global optimization problem, in which *makespan* and *energy consumption* are optimized in **hierarchical** mode and the makespan is the **primary** objective. We start the formalization of the scheduling criteria from the definition of schedules encoding methods. Each schedule x is represented as a column *schedule vector* encoded by the machines' labels in the following way:

$$x = [i_1, \ldots, i_n]^T, \quad (10)$$

where $i_j \in M_l$ is the number of machine on which the task t_j is executed. This encoding method is called a *direct representation* of the schedule. Schedules are defined as the elements of a *Schedules* space expressed as the Cartesian product of n copies of the M_l label sets, i.e:

$$Schedules = \underbrace{M_l \times \ldots \times M_l}_{n}. \quad (11)$$

The direct representation can easily be transformed into a *permutation-based representation*, in which, for each machine, a sequence of tasks assigned to that machine is specified. The tasks in the sequence are increasingly sorted with respect to their completion times. Then all task sequences are merged (concatenated) into a vector u, which is in fact a permutation of tasks to machines. In this representation, an additional information about the numbers of tasks assigned to each machine is required. We define the vector v of the size m, in which the numbers of tasks assigned to the following machines are specified as its coordinates. A schedule in this representation is then defined as a pair of vectors:

$$x = (u; v), u = [u_1, \ldots, u_n]^T, v = [v_1, \ldots, v_m]^T \quad (12)$$

where $u_i \in N_l$, $1 \le v_j \le card(N_l)$ and $card(N_l)$ denotes the cardinality of the N_l set.

We use the permutation-based representation for implementing mutation and crossover operators. The scheduling in our approach is considered to be a discrete time process and is realized in two main stages: (i) makespan minimization, and (ii) total energy consumption minimization.

***Makespan optimization**:* Makespan is expressed as a finishing time of the latest task as follows:

$$Makespan = \min_{x\in Schedules}\max_{j\in N} F_j, \quad (13)$$

where F_j denotes the time when task j is finalized.

Using the ETC matrix model the makespan can be defined in terms of the completion times of the machines. The time of finishing the last task is specified as the maximal completion time of the machines available for the batch of tasks. Let us denote by $completion$ a vector of size m, in which $completion[k]$ indicates the total time needed for reloading the machine m_k after finalizing the previously assigned tasks and for completing the tasks actually scheduled to the machine. We assume that for optimizing the makespan each machine execute the assigned tasks with the maximal frequency and voltage supply. Thus, the completion time $completion[k]$ of the machine m_k is calculated in the following way:

$$completion[k] = ready_k + \sum_{j \in T(k)} ETC[j][k]. \quad (14)$$

The makespan can be now expressed as:

$$makespan = \max_{k \in M} completion[k]. \quad (15)$$

The idle time for machine m_k can be calculated in the following way:

$$Idle_k = makespan - completion[k] \quad (16)$$

For the machine with the maximal completion time (makespan) the idle factor is zero.

Energy optimization: We assume that with optimizing the makespan defined in Eq. 15 the average cumulative energy utilized by the system is defined as follows:

$$E_I = \frac{\sum_{k \in M} \gamma \cdot completion[k] \cdot f \cdot [v_{s_{max}}(k)]^2}{m} + \frac{\sum_{k \in M} \gamma \cdot f_{s_{min}}(k) \cdot [v_{s_{min}}(k)]^2 \cdot Idle_k}{m} \quad (17)$$

It means that the system is working at the maximal voltage supply level during the tasks computation and at the minimal level – in the idle periods. After optimizing the makespan we try to reduce the E_I energy amount consumed by the system, and consider that the voltage and machine relative frequency can be decreased for some tasks, which will increase their computational times, however the updated completion time of machine machine shouldn't exceed the makespan value optimized in the first stage. The scheduling objective in this case is to minimize the average cumulative energy given by the Eq. (9) subject to the following restriction:

$$\sum_{l \in L_k} \left[\frac{1}{f_{s_l}(k)} \cdot ETC[j][k] \right] \leq \ makespan; \ \forall k \in M, \quad (18)$$

where L_k denotes a subset of DVS levels used for the tasks assigned to a machine m_k.

III. Genetic-based energy-aware schedulers

For designing the GA-based schedulers, we used the classical framework of the the $(\mu + \lambda)$ evolutionary strategy (see e.g. [Michalewicz, 1992]), adapted to the scheduling problem through the implementation of specific schedule encoding methods and genetic operators.

An initial population is generated by using the *MTC + LJFR-SJFR* method, in which two solutions are created by using the *Longest Job to Fastest Resource - Shortest Job to Fastest Resource (LJFR-SJFR)* and *Minimum Completion Time (MCT)* heuristics, and the rest of the individuals – randomly. In LJFR-SJFR method the number of m tasks with the highest workload are assigned to the available machines sorted under the computing capacity criterion. The remaining unassigned tasks are allocated in the remaining resources. In the MCT heuristic, a given task is assigned to the machine yielding the earliest completion time.

Based on the tuning results for the genetic operators of the GA-based schedulers presented in [Xhafa et al., 2007] we use in our study the following configuration of the genetic procedures:

- **Selection method:** Linear Ranking Selection;
- **Crossover operator:** Partially Mapped Crossover (PMX);
- **Mutation operator:** Rebalancing;
- **Replacement operators:** Elitist Generational and Struggle.

The *Rebalancing* mutation method makes use of the load balancing technique. The idea is to first improve the schedule quality by rebalancing the machine load and then mutate it. Rebalancing is done in two steps: in the first step, a machine m_i, from the pool of the most overloaded resources is chosen at random; further, two tasks t_j and t_r are selected according the following rule: task t_r is assigned to machine m_l ($m_l \neq m_i$) and $ETC[j][s] \leq ETC[j][i]$. Then tasks t_j and t_r are interchanged (swapped). If such procedure is impossible, a classical 'move' mutation is applied.

In the *Elitist Generational* replacement method, a new generation of the individuals is created by the newly generated off springs and two best adapted parental chromosomes. This method can be very fast in the makespan reduction. However, the high probability of the premature convergence in the case of keeping the 'elitist' solutions in genetic generations can be a main reason of the low effectiveness of the scheduler in the exploration of the new regions of the optimization landscape and very fast unification of the populations. A *Struggle* replacement mechanism can be an effective tool for avoiding too fast scheduler's convergence to the local optima. In Struggle GAs (hereafter, StGAs) a new generation of individuals is created by replacing a part of the population by the most similar individuals, if this replacement optimizes the fitness value.

The definition of the struggle replacement procedure requires a specification of the appropriate *similarity measure*, which indicates the degree of the similarity among two GA's chromosomes. We use in this work a *Normalized Euclidean Metrics* as a similarity indicator in the struggle mechanism, which is calculated as follows:

$$sim_e(s_1; s_2) = \sqrt{\sum_{j=1}^{n} \frac{(s_1[j] - s_2[j])^2}{\sigma_j^2}} \quad (19)$$

where σ_j is the standard deviation of the $s_1[j]$ over the set of all schedules (the GA population).

StGA strategy has shown to be very effective in solving several large-scale multi-objective problems (see [Grueninger et al., 1997]). However the computational cost of the replacement strategy can be very high. To reduce the

execution time of the similarity indication procedures in struggle framework we use a *hash technique*, in which the hash table with the *task-resource allocation* key is created. The value of this key, is calculated as the sum of the absolute values of the subtraction of each position and its precedent in the direct representation of the schedule vector (reading the schedule vector in a circular way). The hash function f_{hash} is defined as follows:

$$f_{hash}(K) = \begin{cases} 0, & K < K_{min} \\ \left\lfloor N \cdot \left(\frac{K - K_{min}}{K_{max} - K_{min}} \right) \right\rfloor & K_{min} \leq K < K_{max} \\ N - 1, & K \geq K_{max} \end{cases} \quad (20)$$

where K_{min} and K_{max} correspond respectively to the smallest and the largest value of the key K in the population, and N is the population size. The other hash techniques for the Struggle-based grid schedulers are presented in [Xhafa et. al., 2008].

IV. Experimental evaluation

In this section we present the results of a simple experimental analysis of two GA-based energy-aware scheduler implementations for static and dynamic versions of the scheduling problem in grid. We used the modified *HyperSim-G* grid simulator (see e.g. [Xhafa et al., 2007]) extended by integrating the machines categories specification mechanism, DVS levels matrix and energy objective. We present briefly the main idea of the simulator in Sec. IV-A

Our GA-based schedulers were evaluated on two benchmarks composed by a set of static and dynamic instances generated by the grid simulator.

A. Energy-aware HyperSim-G grid Simulator - basic concept

The main concept and general flow of our energy-aware version of the HyperSim-G simulator can be briefly described as follows When a scheduling event is triggered, the simulator creates an instance of the scheduling problem, based on the current tasks and available machines pools. The instance contains: (a) workload vector of tasks; (b) computing capacity of machines; (c) prior load of machines; (d) Machine categories specification parameters (number of classes, maximal computational capacity value, computational capacity ranges interval for each class, machine operational speed parameter for each class, etc.); (e) DSV levels matrix for machine categories; and (g) the ETC matrix. The defined instance is then passed on to the selected scheduler which computes the planning of tasks to machines. Finally, the scheduler sends the planning back to the simulator, which makes the allocation. It should be noted that while computing the planning of the batch, some machine might have dropped from the system. In that case, all tasks planned for those machines are re-scheduled in the next batch. The sequence of events and the changes in the state of the system in the simulator capture the grid system dynamics. The simulator provides the full simulation trace by indicating a parameter for the trace generation.

B. Experimental settings

The performance of GA-based meta-heuristics was analyzed in two types of grid environment: static and dynamic. In both cases four Grid size scenarios: small (32 hosts/512 tasks), medium (64 hosts/1024 tasks), large (128 hosts/2048 tasks), and very large (256 hosts/4096 tasks). The capacity of the resources and the workload of tasks are randomly generated by a normal distribution. It is assumed that all tasks submitted to the system must be scheduled and all machines in the system can be used.

The GA parameters values are presented in Table I.

Table I
GA setting for static and dynamic benchmarks.

Parameter	Elitist Generational	Struggle
evolution steps	$5 * m$	$20 * m$
pop. size (pop_size)	$\lceil (\log_2(m))^2 - \log_2(m) \rceil$	$4 * (\log_2(m) - 1)$
intermediate pop.	$pop\_size - 2$	$(pop\_size)/3$
selection method	LinearRanking	
crossover method	PMX	
cross probab.	0.9	1.0
mutation method	Rebalancing	
mutation probab.	0.2	
initialization	LJFR-SJFR + MCT + Random	
$max\_time\_to\_spend$	40 secs $(static)$ / 25 secs $(dynamic)$	

The parameter setting for the simulator for static and dynamic grid scenarios are presented in Tables II and III.

Table II
Parameter setting for the grid simulator for generating static instances

	Small	Medium	Large	Very Large
Number of hosts	32	64	128	256
Resource capacities (in MIPS)	$N(1000, 175)$			
Total number of tasks	512	1024	2048	4096
Workload of tasks	$N(250000000, 43750000)$			
Host selection	All			
Task selection	All			
Number of runs	30			

In the dynamic scenario tasks and machines could be dynamically added/dropped to/from the system. The number of hosts initially activated in the grid environment is defined by the parameter *Init.hosts*. The parameters *Max.hosts* and *Min.hosts* specify the range of changes in the number of active hosts in the simulation process. The frequency of appearing and disappearing resources is defined by the normal distributions given by *Add host* and *Delete host*, while the initial number of tasks is given by *Init. tasks*. New tasks can arrive at the system with the frequency *Interarrival* until *Total tasks* is reached. The *Activation* parameter establishes the activation policy according to an

Table III
SETTINGS FOR THE DYNAMIC GRID SIMULATOR

	Small	Medium	Large	Very Large
Init. hosts	32	64	128	256
Max. hosts	37	70	135	264
Min. hosts	27	58	121	248
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	$E(7812.5)$	$E(3906.25)$	$E(1953.125)$	$E(976.5625)$
Activation	Resource_and_time_interval(250000)			
Reschedule	True			
Host select	All			
Task select	All			
Number of runs	30			

exponential distribution. The assigned tasks which have not been executed yet cannot be rescheduled if the value of the boolean parameter *Reschedule* is false.

We consider 3 machine classes – Class I, Class II and Class III – and 16 DVS levels. The class identifier was selected randomly for each machine in the system. The values of supply voltages and relative machine frequencies at all DVS levels are specified in Table IV.

Table IV
DVS LEVELS FOR THREE MACHINE CLASSES

	Class I		Class II		Class III	
Level	**Volt.**	**Rel.Freq.** f_{s_1}	**Volt.**	**Rel.Freq.** f_{s_2}	**Volt.**	**Rel.Freq.** f_{s_3}
0	1.5	1.0	2.2	1.0	1.75	1.0
1	1.4	0.9	1.9	0.85	1.4	0.8
2	1.3	0.8	1.6	0.65	1.2	0.6
3	1.2	0.7	1.3	0.50	1.9	0.4
4	1.1	0.6	1.0	0.35		
5	1.0	0.5				
6	0.9	0.4				

Schedulers performance measures: The relative performance of the proposed schedulers is measured through the two following metrics:

- makespan defined in Eq. (15);
- a relative energy consumption improvement rate expressed as follows:

$$Im(E) = \frac{E_I - E_{batch}}{E_{batch}} \cdot 100\%, \qquad (21)$$

where E_{batch} and E_I are defined in Eq. (9) and Eq. (17) respectively;

C. Computational results

Each experiment was repeated 30 times under the same configuration of operators and parameters. The averaged makespan and energy saving rate values are presented in Tables V and VI.

As can be seen from the Tables V and VI, both versions of GA algorithms achieve a considerable reduction in energy consumption (up to 33%). However, Struggle Strategy GA outperformed the base Elitist GA for all instances in energy consumption. The reduction on the energy consumption is in fact noticeable if we have into account that energy was considered a secondary objective in the optimization model.

Table V
MAKESPAN VALUES (± %C.I.) FOR STATIC INSTANCES (C.I.: CONFIDENCE INTERVAL)

Scheduler	Small	Medium	Large	Very Large
Makespan values (in arbitrary time units)				
GA	**3971630.27**	3986741.95	4006153.30	4038090.10
	(±)0.5714%	(±)0.7950%	(±)0.9351%	(±)1.1843%
StGA	3972614.96	**3979528.76**	**3986350.17**	**3999442.95**
	(±)0.6421%	(±)0.7714%	(±)1.1750%	(±)1.5132%
Energy consumption reduction rates				
GA	26.4 %	24.7 %	18.6 %	16.2 %
	(±)0.7442%	(±)0.8810%	(±)0.8781%	(±)1.0614%
StGA	**32.9 %**	**33.2 %**	**27.3 %**	**23.8 %**
	(±)0.7250%	(±)0.8905%	(±)1.0833%	(±)1.3244%

Table VI
MAKESPAN VALUES (± % C.I.) FOR DYNAMIC INSTANCES (C.I.: CONFIDENCE INTERVAL)

Scheduler	Small	Medium	Large	Very Large
Makespan values (in arbitrary time units)				
GA	**4048152.90**	**3988204.13**	4015066.05	4041820.13
	(±)0.7560%	(±)0.8501%	(±)1.0724%	(±)1.7805%
StGA	4062331.15	3999261.61	**3981408.54**	**3978104.73**
	(±)0.8109%	(±)1.4350%	(±)1.9363%	(±)1.8390%
Energy consumption reduction rates				
GA	21.5%	23.0%	17.2%	16.8%
	(±)0.8102%	(±)0.9240%	(±)1.2912%	(±)1.7805%
StGA	**24.1 %**	**26.8%**	**24.8%**	**21.3%**
	(±)0.8225%	(±)0.8905%	(±)1.3773%	(±)1.9150%

V. CONCLUSIONS AND FUTURE WORK

In this work we have presented Genetic Algorithms for the scheduling problem in Computational Grids aiming to minimize both the makespan and the energy consumption. We have first formalized the problem as a bi-objective optimization, which includes makespan and flowtime. Then, two versions of GAs are presented, namely Elitist GA and Struggle GA, for solving the bi-objective problem in hierar-

chic mode, with makespan being the primary objective and energy consumption as a secondary one. The experimental study conducted using a Grid simulator showed that both GAs are effective methods for achieving energy reduction, although Struggle GA performed best.

This study has also revealed the complexity of the scheduling problem when energy consumption is considered as optimization criteria. In our future work, we would like to properly formulate the problem in a way that trade-offs among optimizing the system performance and assuring QoS to the Grid users can be efficiently computed.

REFERENCES

[Ali et al., 2000] Ali, S., Siegel, H.J., Maheswaran, M., and Hensgen, D.: "Task execution time modeling for heterogeneous computing systems", *Proceedings of Heterogeneous Computing Workshop*, pp. 185–199, 2000.

[Beloglazov et al., 2011] Beloglazov, A., Buyya, R., Lee, Y., C., and Zomaya, A. Y.: "A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems," *Advances in Computers*, 82, pp. 47–111, 2011.

[Braun et al., 2001] Braun, T.D., 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.: "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), pp. 810-837, 2001.

[Ge et al., 2005] Ge, R., Feng, X., and Cameron, K.: "Performance-constrained distributed dvs scheduling for scientific applications on power-aware clusters", in *Proceedings of the 2005 ACM/IEEE Conference on Supercomputing*, IEEE Computer Society, USA, 2005.

[Grueninger et al., 1997] Grueninger, T.: "Multimodal optimization using genetic algorithms", *Technical report, Department of Mechanical Engineering, MIT, Cambridge, MA*, 1997.

[Guzek et al., 2010] Guzek, K., Pecero, J. E., Dorrosoro, B., Bouvry, P., and Khan, S. U.: "A Cellular Genetic Algorithm for Scheduling Applications and Energy-aware Communication Optimization," in *ACM/IEEE/IFIP International Conference on High Performance Computing and Simulation (HPCS)*, Caen, France, June 2010, pp. 241–248.

[Khan, 2009] Khan, S. U.: "A Self-adaptive Weighted Sum Technique for the Joint Optimization of Performance and Power Consumption in Data Centers", in *22nd International Conference on Parallel and Distributed Computing and Communication Systems (PDCCS)*, USA, 2009, pp. 13–18.

[Khan and Ahmad, 2009] Khan, S. U., and Ahmad, I.: "A Cooperative Game Theoretical Technique for Joint Optimization of Energy Consumption and Response Time in Computational Grids", *IEEE Transactions on Parallel and Distributed Systems*, 20(3), pp. 346–360, 2009.

[Kliazovich et al., 2010] Kliazovich, D., Bouvry, P., and Khan, S. U.: "DENS: Data Center Energy-Efficient Network-Aware Scheduling," in *ACM/IEEE International Conference on Green Computing and Communications (GreenCom)*, Hangzhou, China, December 2010, pp. 69–75.

[Lee and Zomaya, 2009] Lee, Y. C. and Zomaya, A. Y.: "Minimizing Energy Consumption for Precedence-Constrained Applications Using Dynamic Voltage Scaling," in *9th IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid)*, Shanghai, China, 2009, pp. 92–99.

[Lorch and Smith, 2001] Lorch, J.R., and Smith, A.J.: "Improving dynamic voltage scaling algorithms with pace", In *2001 ACM SIGMETRICS international conference on measurement and modeling of computer systems*, pp. 50-61, 2001.

[Mejia-Alvarez et al., 2004] Mejia-Alvarez, P., Levner, E., Mossé, D.: "Adaptive scheduling server for power-aware real-time tasks', *ACM Trans Embed Comput Syst*, 3(2), pp. 284-306, 2004.

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

[Min et al., 2000] Min, R., Furrer, T., and Chandrakasan, A.: "Dynamic voltage scaling techniques for distributed microsensor networks', In *Proc. IEEE Workshop on VLSI*, pp. 43-46, 2000.

[Pinel at al., 2010] Pinel, F., Pecero, J., Bouvry, P., and Khan, S. U.: "Memory-aware Green Scheduling on Multi-core Processors," in *39th IEEE International Conference on Parallel Processing (ICPP)*, pp. 485–488, USA, 2010.

[Xhafa et al., 2007] Xhafa, F., Carretero, J.,and Abraham, A.: "Genetic Algorithm Based Schedulers for Grid Computing Systems", *International Journal of Innovative Computing, Information and Control*, 3(5), pp. 1053–1071, 2007.

[Xhafa et. al., 2008] Xhafa, F., Duran, B., Abraham, A., and Dahal, K. P.: "Tuning Struggle Strategy in Genetic Algorithms for Scheduling in Computational Grids", *Neural Network World*, 18(3), pp. 209–225, 2008.

[Xhafa et al., 2010] Xhafa, F., Abraham, A.: "Computational models and heuristic methods for grid scheduling problems", *Future Generation Computer Systems*, vol. 26 , pp. 608–621, 2010.

[Zomaya, 2009] Zomaya, A. Y.: "Energy-Aware Scheduling and Resource Allocation for Large-Scale Distributed Systems", in *11th IEEE International Conference on High Performance Computing and Communications (HPCC)*, Seoul, Korea, 2009.