Chapter 8
Energy-Aware Scheduling of Independent Tasks in Computational Grids

Abstract. This chapter introduces the application of the Hierarchical Genetic Strategy-based Grid scheduler (HGS-Sched) to the energy-aware independent batch scheduling problem in Computational Grids (CGs). The *Dynamic Voltage Scaling (DVS)* methodology is used for both scaling the power supply of the grid resources and reducing the cumulative power energy utilized by the grid computing machines. Two implementations of HGS-Sched—with elitist and struggle replacement mechanisms respectively—are defined and empirically evaluated. The effectiveness of the hierarchical schedulers are compared with the quality of single-population Genetic Algorithms (GAs) and Island GA models for four CG significant scenarios in static and dynamic modes. The simulation results show that meta-heuristic grid schedulers can significantly reduce the energy consumption in the system as well as be easily adapted to various scheduling scenarios.

8.1 Introduction

The main issues related to power consumption and effective thermal management in high performance computing have been induced by the sheer scale of enterprise computing environments and data centers. In large supercomputer centers and next generation distributed systems such as 'green' grid clusters and clouds, the growing operating, power and cooling rates have become the dominant part of the users' and system managers' budgets. Novel innovative green computing technologies are mainly devoted to the optimization of system thermodynamics [108]. Profiles of hardware energy consumption and application energy consumption are gathered in order to be correlated with workload distribution and energy consumption of the different power and cooling systems [43].

While CGs have been widely promoted as affordable alternatives to supercomputers, a significant disproportion of resource availability and resource provisioning has been empirically observed [76]. Therefore, a significant deal of research in grid computing is devoted to design novel, effective grid schedulers, which can simultaneously optimize the key grid objectives—such as makespan, flowtime, and

J. Kołodziej: Evolutionary Hierarchical Multi-Criteria Metaheuristics, SCI 419, pp. 155–175.
springerlink.com

resource utilization [35], as well as the energy consumed by all system components and users.

That is to say, while the main purpose of grid schedulers is to efficiently and optimally allocate application tasks to a set of available resources, one should also consider a series of requirements including energy efficiency. Energy-efficient scheduling in CGs has therefore become a relevant yet complex endeavor due to the multitude of constraints and the different optimization criteria and priorities of the resource owners. Heuristic approaches have demonstrated to be effective for designing energy-aware grid schedulers by keeping a balance among various preferences and goals of the grid users, resource and service managers, and resource owners.

This chapter addresses the problem of energy optimization for *Independent Batch Scheduling* in CGs. The *average energy consumption* is considered as a complementary scheduling criterion along with the *makespan* as the primary objective. According to the notation introduced in Sec. 1.4.2, an instance of the independent batch grid scheduling problem with energy optimization criterion is expressed in the following way:

$$Rm\left[\{b, indep, (stat, dyn), hier\}\right](C_{max}, E_I(E_{II})) \tag{8.1}$$

where:

- C_{max} – denotes a makespan as the primary scheduling objective
- $E_I(E_{II})$ – denotes total energy consumption as the second scheduling criterion (E_I or E_{II} is selected depending on the scheduling scenario (see Sec. 8.3.2))

This chapter extends the empirical analysis presented in [82] by the implementation and the comparative analysis of the effectiveness of the multi-population and single-population GA-based Grid schedulers. The term 'green' used in this chapter refers not just to the low-power system devices, but also to the energy-aware schedulers.

8.2 Energy Model

The main module in the energy-aware grid scheduling model presented in this chapter is the *Dynamic Voltage and Frequency Scaling (DVFS)* technique. Used for adjusting the voltage supplies and frequencies of the grid computational nodes, the DVFS technique is primarily based on the power consumption model employed in complementary metal-oxide semiconductor (CMOS) logic circuits [13]. In this model, the capacitive power Pow_{ji}–utilized by the machine i for computing the task j–depends on the voltage supply and machine frequency, and it is calculated as follows:

$$Pow_{ji} = A \cdot C \cdot v^2 \cdot f, \tag{8.2}$$

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 frequency of the machine (see also Chapter 7, Sec. 7.3.1, Eq. (7.1)). The energy consumed per machine i for the computation of task j can then be derived using the following formula:

$$E_{ji} = \int_0^{completion[j][i]} Pow_{ji}(t)dt \tag{8.3}$$

where $completion[j][i]$ is a completion time of the task j on machine i.

Each machine in the grid has been equipped with a DVFS module [97] for scaling its supply voltage and operating frequency. It has been assumed that the frequency of the machine is proportional to its processing speed (see [104]). It follows from the Eq. (8.2) that the reduction of the supply voltage and frequency is directly correlated to the reduction of the energy utilization. Table 8.1 shows the parameters for 16 DVFS levels and three main 'energetic' categories for machines defined for the grid system employed in this study.

Table 8.1 DVFS levels for three machine classes

	Class I		Class II		Class III	
Level	**Volt.**	**Rel.Freq.**	**Volt.**	**Rel.Freq.**	**Volt.**	**Rel.Freq.**
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				

The energetic class of machine i, ($i \in M$), denoted by s^i and represented by the meta-vector $Vr_{(i)}$ of DVFS levels, can be specified as:

$$Vr_{(i)} = \left[(v_{s_0}(i), f_{s_0}(i)); \ldots; (v_{s_{l(max)}}(i), f_{s_{l(max)}}(i))\right]^T \tag{8.4}$$

where $v_{s_l}(i)$ refers to the voltage supply for machine i at level s_l, $f_{s_l}(i)$ is a scaling parameter for the frequency of the machine at the same level s_l, and l_{max} is the number of levels in the class s^i. The parameters $\{f_{s_0}(i), \ldots, f_{s_{l(max)}}(i)\}$ are in the [0,1] range and should be interpreted as the relative frequencies of the machine i of class s^i at the $s_0, \ldots, s_{l(max)}$ DVFS levels.

The reduction of the machine frequency and its supply voltage can lead to the extension of the computational times of the tasks executed on that machine. For a given 'task-machine' pair (j,i), the completion times for the task j on machine i at

different DVFS levels in the class s^i can be interpreted as the coordinates of a vector $\widehat{ETC}[j][i]$ which is defined as:

$$\widehat{ETC}[j][i] = \left[\frac{1}{f_{s_0}(i)} \cdot ETC[j][i], \ldots, \frac{1}{f_{s_{l(max)}}(i)} \cdot ETC[j][i]\right] \tag{8.5}$$

where $ETC[j][i]$ are the expected completion times for task j on machine i calculated by using the conventional ETC matrix model (see Chapter 2, Sec. 2.2).

The ETC matrix model defined in Chapter 2 can be directly adapted to the energy-aware scheduling of independent tasks in grids. The completion times calculated for each pair (j,i) of task–machine labels in conventional ETC matrix (see Eq. (2.2)) should be replaced by the $\widehat{ETC}[j][i]$ vectors, that is:

$$\widehat{ETC} = \left[\widehat{ETC}[j][i][s_l]\right]_{n \times m \times s_{l(max)}} \tag{8.6}$$

where $\widehat{ETC}[j][k][s_l]$ is the time necessary for the completion of the task j on machine i at the level s_l.

Based on Equations (8.2), (8.3) and (8.6) the energy utilized for completing task j on machine i at level s_l can be defined as a scalar product of the number of switches per clock cycle, the total capacitance load, the frequency and the squared voltage at level s_l, and the estimated completion time of task j on machine i. That is to say:

$$E_{ji}(s_l) = \gamma \cdot (f_{s_l}(i))_j \cdot f \cdot [(v_{s_l}(i))_j]^2 \cdot \widehat{ETC}[j][i][s_l], \tag{8.7}$$

where $\gamma = A \cdot C$ is a constant parameter for a given machine class, $(v_{s_l}(i))_j$ is a voltage supply value for class s^i and machine i at level s_l for computing task j, and $(f_{s_l}(i))_j$ is a corresponding relative frequency for machine i.

Based on the Equations (8.6), (8.7) and (8.5) the computational times for each possible pair (j,i) at the level s_l can be calculated as:

$$\begin{aligned} Tim_{\{j,i,s_l\}} &= \gamma \cdot (f_{s_l}(i))_j \cdot f \cdot [(v_{s_l}(i))_j]^2 \cdot (f_{s_1}(i))_j \cdot ETC[j][i] = \\ &= \gamma \cdot f \cdot [(v_{s_l}(i))_j]^2 \cdot ETC[j][i] \end{aligned} \tag{8.8}$$

The cumulative energy utilized by the machine i for the completion of all tasks from the batch that are assigned to this machine, is defined in the following way:

$$\begin{aligned} E_i = \sum_{\substack{j \in Tasks(i) \\ l \in \hat{L}_j}} \{Tim_{\{j,i,s_l\}}\} + \gamma \cdot f \cdot [v_{s_{max}}]^2 \cdot ready_i + \gamma \cdot f_{s_{min}}(i) \cdot f \cdot \\ \cdot [v_{s_{min}}(i)]^2 \cdot Idle[i] = \gamma \cdot f \cdot \sum_{\substack{j \in Tasks(i) \\ l \in \hat{L}_i}} ([(v_{s_l}(i))_j]^2 \cdot ETC[j][i]) + \\ + [v_{s_{max}}(i)]^2 \cdot ready_i + f_{s_{min}}(i) \cdot [v_{s_{min}}(i)]^2 \cdot Idle[i] \end{aligned} \tag{8.9}$$

where $Tasks(i)$ is a set of tasks assigned to machine i, $ready_i$ is the ready time of machine i, $Idle[i]$ denotes an idle time of machine i, and $\hat{L}_i$ denotes a subset of DVFS levels used for the tasks assigned to machine i. All additional machine frequency transition overheads are ignored. These overheads take usually a negligible amount of time (e.g., 10ms- 150ms, see [109]) and do not bear down on the overall ETC model with an active 'energetic' module.

Finally, an average cumulative energy utilized by the grid system for completion of all tasks in the batch is defined as follows:

$$E_{batch} = \frac{\sum_{i=1}^{m} E_i}{m} \tag{8.10}$$

This model is used in the following section for specification of two scheduling scenarios and the definition of the scheduling criteria.

8.3 Scheduling Scenarios and Objectives

Two main scheduling scenarios are considered in this study, namely:

I. **I – Max-Min Mode**, in which each machine works at the **maximal** DVFS level during the execution and computation of tasks and enters into idle mode after the execution of all tasks assigned to this machine;
II. **II – Modular Power Supply Mode**, in which each machine can work at **different** DVFS levels during the task executions and can then enter into idle mode.

In the former, the consumption of the of the energy depends on the 'energetic' class of the system devices or services, defined as 'machines' (resources) in the system. No modifications of the conventional scheduling procedures and standard scheduling objectives—such as makespan, flowtime, tardiness, etc. (see Chapters 1 and 2)—are needed. In the latter, the optimal power supply levels can be specified for each machine, and the energy consumption can subsequently be reduced by diminishing the power supply in the machines while preserving the deadline constraints for the main tasks.

The procedures for calculation and optimization of the two scheduling objective functions, makespan and cumulative energy utilized by the system, are different in the aforementioned scheduling scenarios. The details are discussed in the two following subsections.

8.3.1 Makespan Optimization

The minimization of the makespan is the first step of the optimization procedure in the scheduling objectives. Based on the ETC matrix model and denoted by C_{max}, the makespan can be defined in terms of the completion times of the machines (see Chapter 2, Sec. 2.2.2). The finishing time for the last task in the batch is specified as the maximal completion time of all machines available in that batch. Denoted by $completion[i]$, the completion time of machine i is the cumulative time required for

both reloading the machine i after finalizing the previously assigned tasks and for completing the tasks currently assigned to the machine.

In **Max-Min Mode** such completion time can be defined as:

$$completion_I[i] = ready_i + \sum_{j \in Tasks(i)} ETC[j][i]. \tag{8.11}$$

The makespan in this mode is calculated in the following way:

$$(C_{max})_I = \max_{i=1}^{m} completion_I[i]. \tag{8.12}$$

The idle time for machine i working in **Max-Min Mode** can be expressed as the difference between the makespan and $completion_I[i]$, i.e.:

$$Idle_I[i] = (C_{max})_I - completion_I[i] \tag{8.13}$$

It should be clear that for the machine with the maximal completion time (makespan) the idle factor is zero.

In **Modular Power Supply Mode**, for each task-machine pair, the DSV level s_l must be specified. The formulae for computing the completion time, makespan, and idle time at the level s^i can be defined as:

$$completion_{II}[i] = ready_i + \sum_{j \in Tasks(i)} \frac{1}{f_{s_l}(i)} \cdot ETC[j][i]. \tag{8.14}$$

$$(C_{max})_{II} = \max_{i=1}^{m} completion_{II}[i]. \tag{8.15}$$

$$Idle_{II}[i] = (C_{max})_{II} - completion_{II}[i] \tag{8.16}$$

8.3.2 Energy Optimization

The second step of the scheduling optimization procedure is the minimization of the total energy consumed in CG for scheduling a given batch of tasks.

The average energy consumed in the system in **Min-Max Mode** is defined as:

$$\begin{aligned} E_I = {} & \tfrac{1}{m} \cdot \textstyle\sum_{i=1}^{m} \gamma \cdot completion_I[i] \cdot f \cdot [v_{s_{max}}(i)]^2 + \\ & + \tfrac{1}{m} \cdot \textstyle\sum_{i=1}^{m} \gamma \cdot f_{s_{min}}(i) \cdot [v_{s_{min}}(i)]^2 \cdot Idle_I[i] \end{aligned} \tag{8.17}$$

In **Modular Power Supply Mode** the average cumulative energy is given by Eq. (8.10):

$$E_{II} = E_{batch} = \frac{\sum_{i=1}^{m} E_i}{m} \tag{8.18}$$

where[1]

$$E_i = \gamma \cdot f \cdot \sum_{\substack{j \in T(i) \\ l \in L_i}} ([(v_{s_l}(i))_j]^2 \cdot ETC[j][i]) + \\ + [v_{s_{max}}(i)]^2 \cdot ready_i + f_{s_{min}}(i) \cdot [v_{s_{min}}(i)]^2 \cdot Idle[i] \tag{8.19}$$

In both cases E_I and E_{II} are minimized and subject to the following constraint:

$$\sum_{l \in \hat{L}_i} \left[\frac{1}{f_{s_l}(i)} \cdot ETC[j][i] \right] \leq C_{max}; \forall i \in \{1, \ldots, m\}, \tag{8.20}$$

where $\hat{L}_i$ denotes a subset of DVFS levels specified for tasks assigned to machine i.

8.4 Empirical Analysis

The implementation of the energy management model based on the DVFS method typically produces an improvement in the load balancing of machines. However, DVFS itself does not change the task assignment. While machines are kept in use for a longer time, they work in a low-cost mode in terms of energy consumption. Conventional load-balancing methods, such as goal programming, are typically effective just for the static scheduling case. It is arguable that, for a successful implementation of dynamic load-balancing and dynamic programming schedulers, a knowledge of all possible states in the system is needed, which is not feasible for large-scale grids.

Metaheuristic approaches are the most promising solution for 'green' scheduling. The effectiveness of single-population GAs for the energy optimization in grid scheduling has been presented in [82]. This chapter substantially extends our initial analysis by introducing an empirical evaluation of multi-population grid schedulers. Within this section, the implementations of the *HGS-Sched* algorithm for energy-aware scheduling problem is referred by *Green-HGS-Sched*.

Similarly to previous chapters, several grid scenarios for static and dynamic scheduling are modeled using the grid simulator. In this case the general architecture of *Sim-G-Batch* has been extended through an energy module as presented in Fig. 8.1.

The simulator generates benchmarks for the problem based on the following input data:

- workload vector of tasks;
- computing capacities of machines;
- prior machines loads;
- 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.);
- DVFS levels matrix for machine categories; and
- the ETC matrix.

[1] See Equation (8.9).

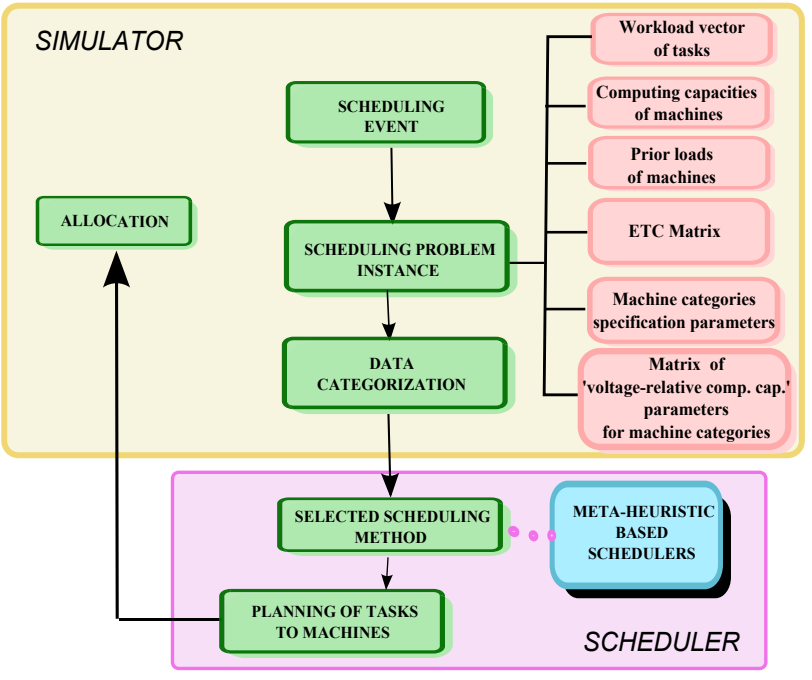

Fig. 8.1 General model of the 'energy-aware' implementation of *Sim-G-Batch* simulator

The machines can work at 16 DVFS levels and can be categorized into three 'energetic' resource classes, Class I, Class II, and Class III. The class identifiers have been selected randomly for the machines. The values of supply voltages and relative machine frequencies at all DVFS levels are specified in Table 8.1. The general settings of the simulator for four grid scenarios—*Small*, *Medium*, *Large* and *Very Large* grids—are the same as in the empirical analysis presented in Chapter 4, and defined in Table 4.1.

The configurations of key parameters for both implementations of single-population GA, *IGA* and *Green-HGS-Sched* meta-heuristics are presented in Tables 8.3, 8.4 and 8.5. The size of the initial and intermediate populations in *IGA* depends on the implementation of the genetic engine in islands and are the same as for the single-population *GA-Elit* and *GA-St* algorithms. The parameters for single-population GA schedulers and the energy-aware implementation of *HGS-Sched* are similar to the settings defined in Chapters 4 and 5 for GAs and *HGS-Sched*.

8.4.1 Energy Aware Genetic-Based Batch Schedulers

Six genetic-based meta-heuristics have been developed for minimizing the makespan and energy consumption in the **Max-Min** and **Modular Power Supply** scheduling

Table 8.2 Six GA-based grid schedulers evaluated in the empirical analysis

Scheduler	Type of algorithm	Replacement method
GA-Elit	Single-population GA	Elitist Generational
GA-St	Single-population GA	Struggle
IGA-Elit	Island GA	Elitist Generational
IGA-St	Island GA	Struggle
HGS-Elit	*Green-HGS-Sched*	Elitist Generational
HGS-St	*Green-HGS-Sched*	Struggle

modes defined in the previous section. The configuration of the genetic operators in those meta-heuristics is presented in Table 8.2.

The aforementioned methodologies differ in the implementation of the replacement mechanism in the main genetic framework. The *Elitist Generational* replacement is used in *xxx-Elit* algorithms and the *Struggle* procedure in *xxx-St* algorithms. Both single-population GAs—*GA-Elit* and *GA-St*—are implemented as the main genetic mechanism in *IGA-Elit*, *HGS-Elit*, *IGA-St*, and *HGS-St* respectively.

The concept of IGA algorithm with the specification of all key parameters for this strategy, was presented in Chapter 4 (Sec. 4.4.2.1).

The template of the main genetic engine in all schedulers is defined in Alg. 1 in Chapter 3, and the encoding methods for the schedulers are the same as in Sec. 3.3. The combination of the main operators for all schedulers is similar to the optimal configuration of the genetic mechanism in the HGS-Sched generated in Chapter 4. The *Linear Ranking* is used as the selection scheme, and the *Cycle Crossover (CX)* and *Move* mutation are selected as the main genetic operators.

The configurations of key parameters for both implementations of single-population GA, *IGA* and *Green-HGS-Sched* meta-heuristics are presented in Tables 8.3, 8.4 and 8.5. The size of the initial and intermediate populations in *IGA* depends on the implementation of the genetic engine in islands and are the same as for the single-population *GA-Elit* and *GA-St* algorithms. The parameters for single-population GA schedulers and the energy-aware implementation of *HGS-Sched* are similar to the settings defined in Chapters 4 and 5 for GAs and *HGS-Sched*.

The relative performance of all six schedulers has been quantified with the following two metrics:

- minimal makespan defined as follows:

$$makespan = min\{Makespan_I, Makespan_{II}\} \tag{8.21}$$

Table 8.3 GA setting for static and dynamic benchmarks

Parameter	GA-Elit	GA-St
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.	$pop\_size - 2$	$(pop\_size)/3$
cross probab.	1.0	1.0
mutation probab.	0.2	
$max\_time\_to\_spend$	30 secs $(static)$ / 45 secs $(dynamic)$	

Table 8.4 *HGS-Sched* settings for static and dynamic benchmarks

Parameter	
period_of_metaepoch	$20 * n$
nb_of_metaepochs	10
degrees of branches (t)	0 and 1
population size in the core	$3 * (\lceil 4 * (\log_2 n - 1)/(11.8) \rceil)$
population size in the sprouted branches (b_pop_size)	$(\lceil (4 * (\log_2 n - 1)/(11.8) \rceil)$
intermediate pop. in the core	$abs((r\_pop\_size)/3)$
intermediate pop. in the sprouted branch	$abs((b\_pop\_size)/3)$
cross probab.	0.9
mutation probab. in core	0.4
mutation probab. in the sprouted branches	0.2
$max\_time\_to\_spend$	40 secs $(static)$ / 70 secs $(dynamic)$

- a relative energy consumption improvement rate expressed as follows:

$$Im(E) = \frac{E_I - E_{batch}}{E_{batch}} \cdot 100\%, \tag{8.22}$$

where E_{II} and E_I are defined in Eq. (8.10) and Eq. (8.17) respectively;

Table 8.5 Configuration of *IGA* algorithm

Parameter	
it_d	$20 * n$
mig	5 %
number of islands (demes)	10
cross probab.	1.0
mutation probab.	0.2
max_time_to_spend	40 secs (*static*) / 70 secs (*dynamic*)

8.4.2 Results

Each experiment has been repeated 30 times under the same configuration of operators and parameters. The box-plots of the first, mean, and the third quantiles (confidence level - 95 %) for the makespan and relative energy consumption rate $Im(E)$ are presented in Fig. 8.2–8.5.

Makespan optimization results

Fig. 8.2–8.5 depicts the box-plots of the makespan values for six considered schedulers. The makespan is measured and expressed in arbitrary time units defined for the execution of tasks.

Both implementations of the *Green-HGS-Sched* have achieved the best results in all instances but *Large grid* in the static case and *Small* and *Large* instances in the dynamic case, where they are outperformed by the *IGA* algorithm.

On the one hand, a simple comparison of the impact of the replacement method on the algorithms performance provided for all pairs of the *xxx-Elit* and *xxx-St* schedulers shows that *Struggle* replacement is much more effective that *Elitist Generational* method in the case of single-population *GA* and *IGA* schedulers. It also confirms the results of the preliminary study on the effectiveness of single-population genetic schedulers in CGs presented in [82].

On the other hand, for the *Green-HGS-Sched* the situation is completely different. In most of the scheduling instances the effectiveness of both hierarchical schedulers are at comparative levels, with little advantage for the elite technique in the dynamic cases. It seems to indicate that the replacement mechanism does not play a crucial role in the fast exploration of the search space by the *Green-HGS-Sched*. Such exploration process can be construed as very slow when using the conventional GA and IGA schedulers. The core of *Green-HGS-Sched* can activate the more accurate processes in the neighborhoods of the partial solutions of the problem. Those solutions

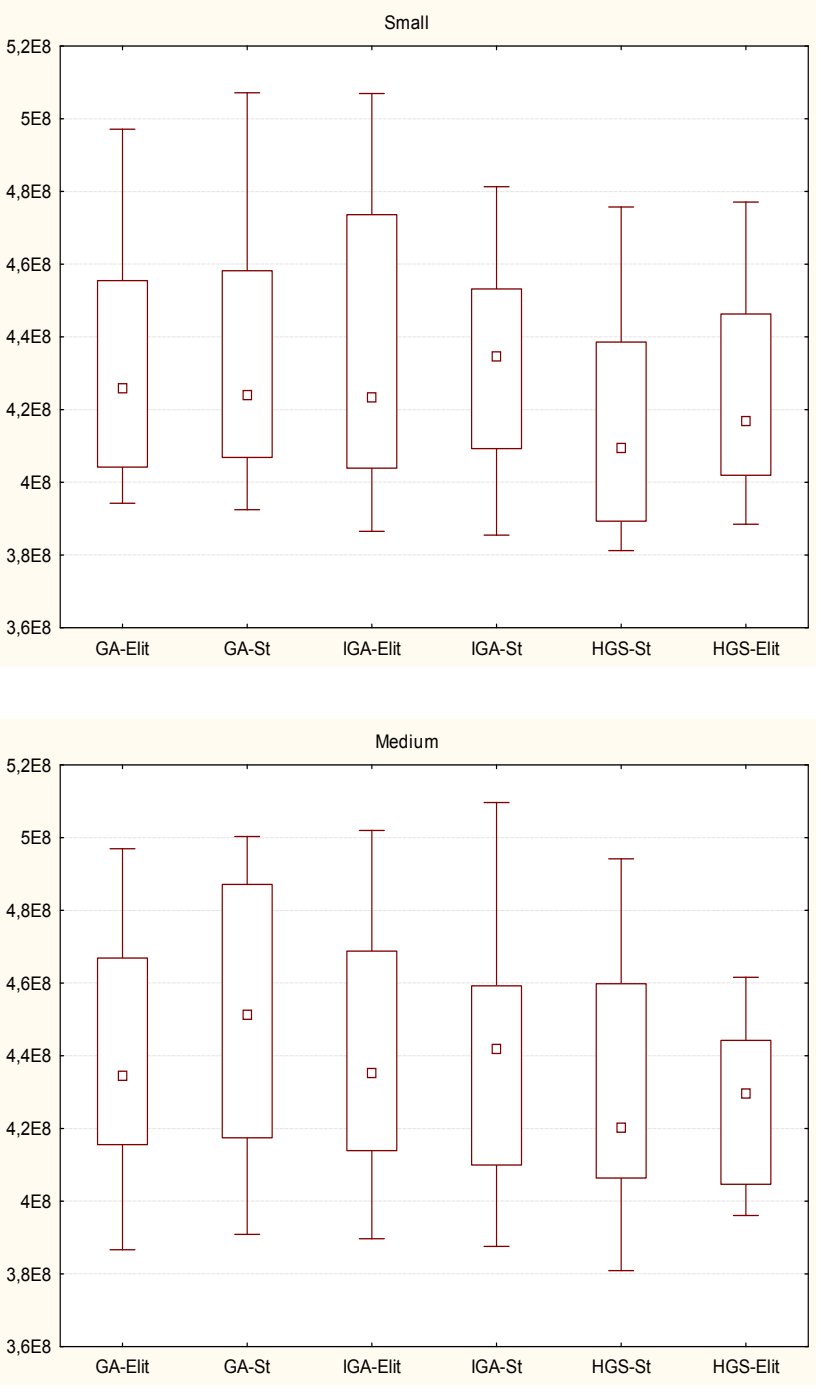

Fig. 8.2 The box-plot of the results for makespan in static case: Small and Medium grids

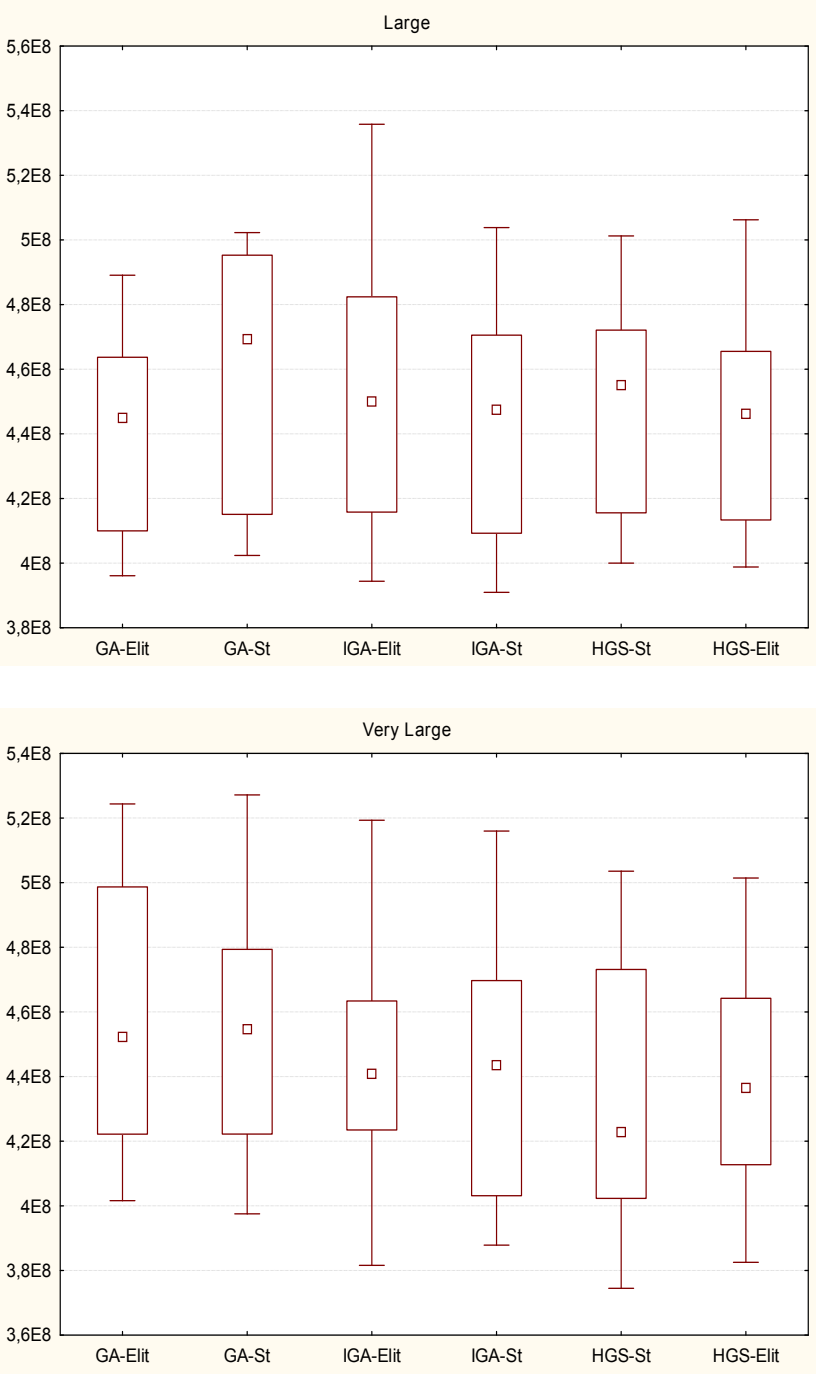

Fig. 8.3 The box-plot of the results for makespan in static case: Large and Very Large grids

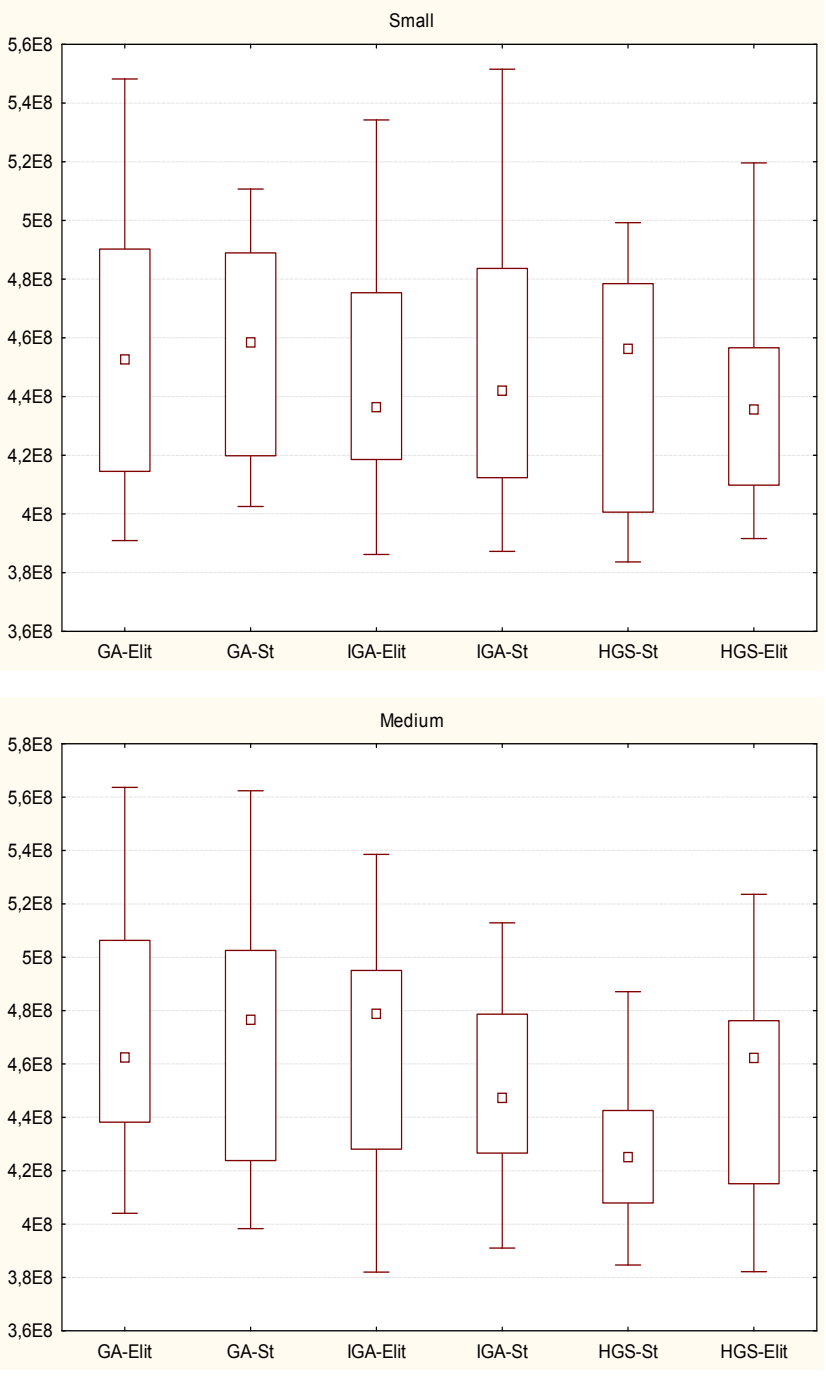

Fig. 8.4 The box-plot of the results for makespan in dynamic case: Small and Medium grids

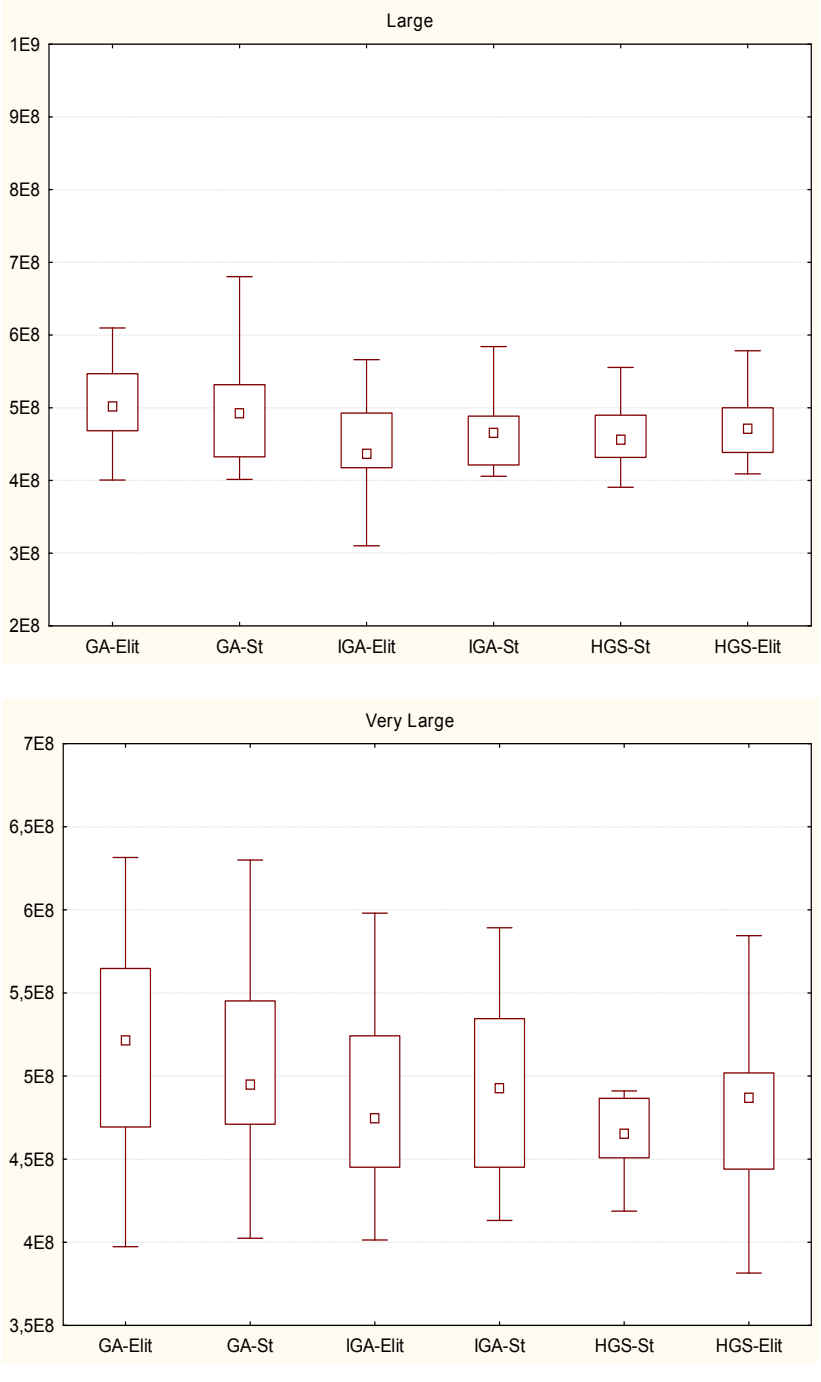

Fig. 8.5 The box-plot of the results for makespan in dynamic case: Large and Very Large grids

may be not detected by the other schedulers, which makes the *Green-HGS-Sched* very effective in the exploration of new regions in the optimization domain and in escaping the basins of attraction of the local solutions.

The complexity of the hierarchic system is, in fact, not a drawback of the *Green-HGS-Sched*, because the constraints of the execution time for HGS and IGA are exactly the same. The ranges in the achieved makespan values for all considered meta-heuristics are not greater than $30-35$ % of the mean makespan values, which means that the stability of all schedulers in all cases is at an acceptable level. The distributions of the makespan results are asymmetric: the skewness in the static case is positive—for GA and IGA and negative for *Green-HGS-Sched* in most of the static instances—and it is negative in the dynamic grids for almost all schedulers. It also implies that the reduction of the average makespan in this case is more difficult than in the static case, which confirms the complexity of the problem in the realistic dynamic grid scenarios.

Energy optimization results

The main effect of the makespan minimization is arguably the balance of the loads in grid resources. The application of the DVFS technique typically leads to a significant reduction of the energy consumption in the system, especially in the case of substantial differences in the loads of particular machines. The box-plots for the energy saving rates $Im(E)$ are presented in Fig. 8.6–8.9.

The results of the energy optimization are slightly different in comparison with the makespan ones. In this case, each of the *IGA-Elit* and *GA-Elit* algorithms outperforms the rest of the schedulers in five instances, and the single-population GAs are the best in the three cases. *Green-HGS-Sched* is not as effective in energy optimization as in the makespan minimization. It means that this algorithm works quite well in **Min-Max** scenario: the makespan is relatively short, and the scaling of the voltage supply may lead to not so significant energy conservation.

In the case of single population and island models, the extensions of the completion times of the tasks in **Modular Power Supply** mode allow to keep the machines busy for a longer time than in the **Min-Max** mode. However, the average difference in energy saving rates achieved by the *Green-HGS-Sched* scheduler and the remaining meta-heuristics does not exceed 10 %, which is smaller than in the makespan case (15 %). This signifies that the average cumulative energy utilization achieved by *Green-HGS-Sched* is lower than the island GA and conventional GAs. The range of the average energy saving rate values is 10%–35% for most of the schedulers. It can also be observed that the skewness of the distribution of the results is positive or neutral for the worst 'energy optimizers' and negative for the best ones.

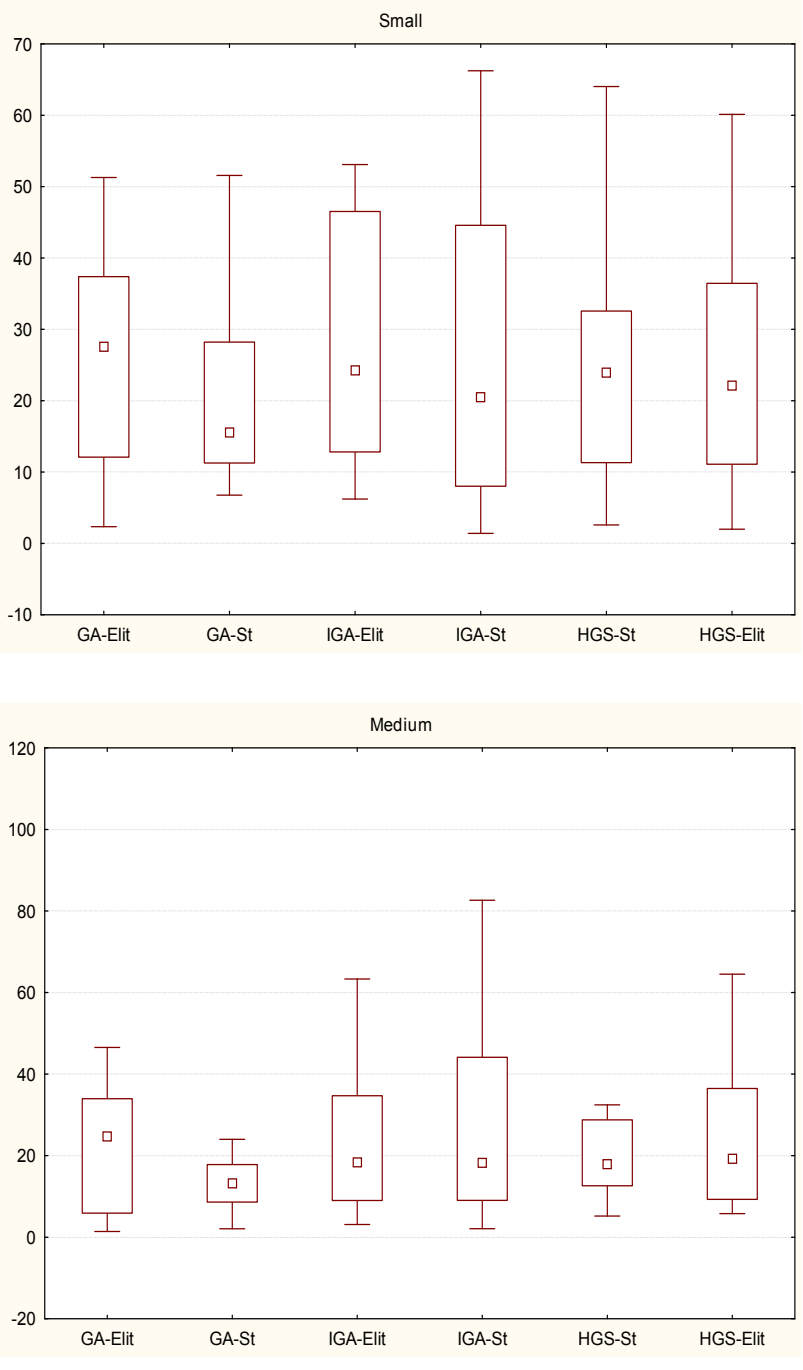

Fig. 8.6 The box-plot of the results for relative energy saving rate in static case (in %): Small and Medium grids

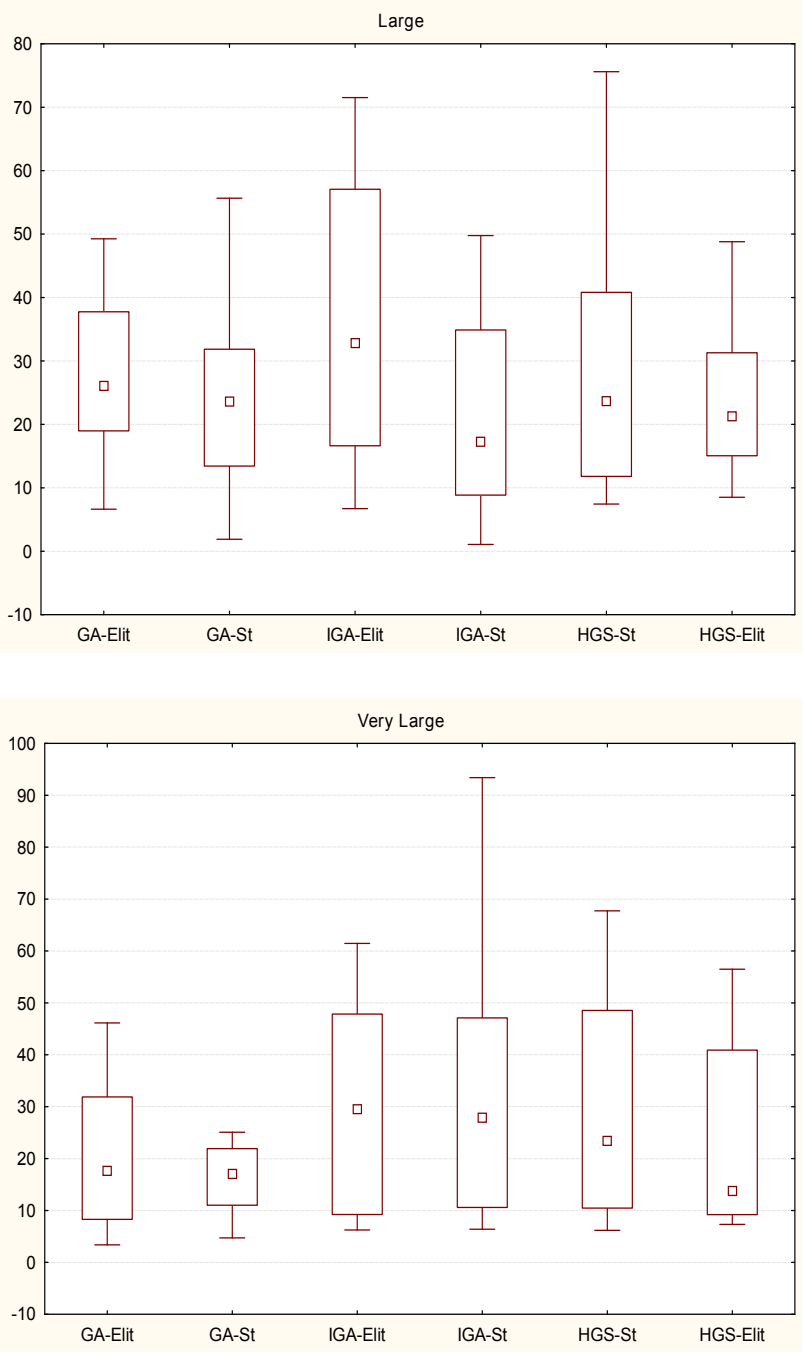

Fig. 8.7 The box-plot of the results for relative energy saving rate in static case (in %): Large and Very Large grids

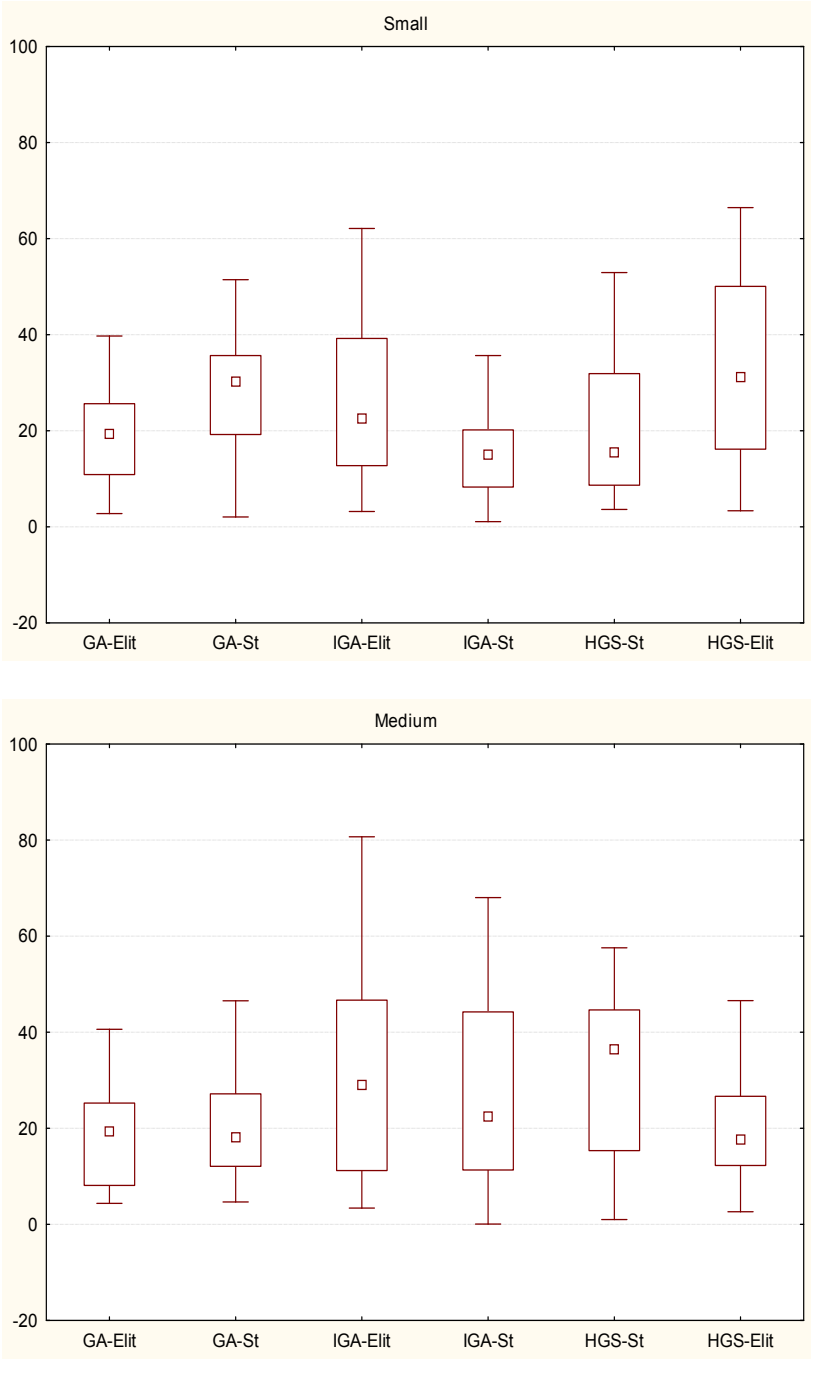

Fig. 8.8 The box-plot of the results for relative energy saving rate in dynamic case (in %): Small and Medium grids

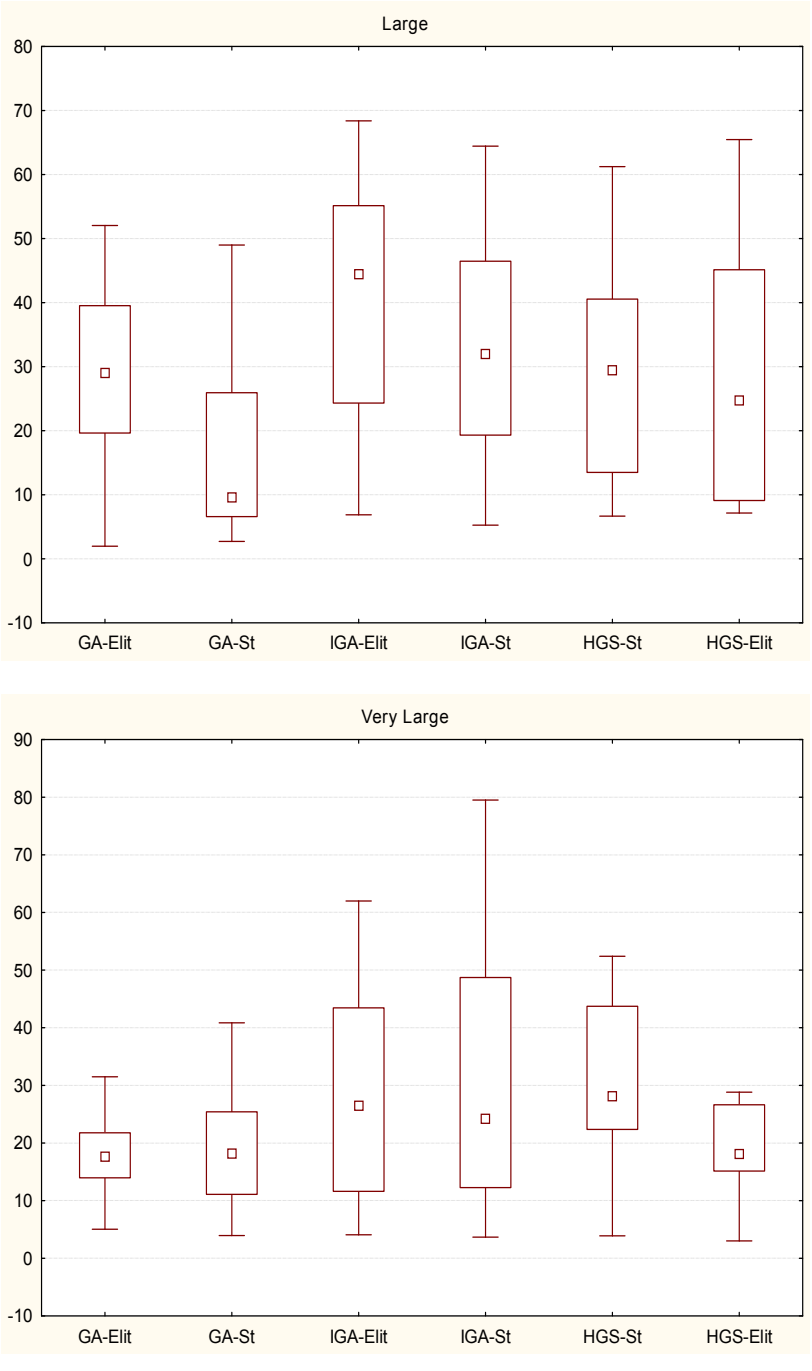

Fig. 8.9 The box-plot of the results for relative energy saving rate in dynamic case (in %): Large and Very Large grids

8.5 Summary

This chapter addressed the problem of optimizing the energy utilized in CGs in independent batch scheduling. The energy management model is based on *Dynamic Voltage Scaling (DVFS)* technique adapted to the dynamic Grid environment. The energy-aware Grid scheduling was formalized as a bi-objective optimization problem with makespan and average cumulative energy consumption as the main objectives.

For solving the addressed Grid scheduling problem, two implementations of an energy-efficient Hierarchical Grid Scheduler *HGS-Sched* were developed evaluated in two 'energetic' scheduling modes in static and dynamic Grid scenarios under the makespan and relative energy consumption improvement rate criteria. The efficiencies of the hierarchical schedulers were compared with the results achieved by four single-population Genetic Algorithm (GA) and Island GA schedulers. The simulation results confirmed the effectiveness of the proposed schedulers in the reduction of the energy consumed by the whole system and in dynamic load balancing of the resources in Grid clusters, which is sufficient to maintain the desired quality level(-s).