J Supercomput (2017) 73:354–369
DOI 10.1007/s11227-016-1866-9

Blacklist muti-objective genetic algorithm for energy saving in heterogeneous environments

Eloi Gabaldon[1] · **Josep Lluis Lerida**[1] · **Fernando Guirado**[1] · **Jordi Planes**[1]

Published online: 20 September 2016

Abstract Reducing energy consumption in large-scale computing facilities has become a major concern in recent years. Most of the techniques have focused on determining the computing requirements based on load predictions and thus turning unnecessary nodes on and off. Nevertheless, once the available resources have been configured, new opportunities arise for reducing energy consumption by providing optimal matching of parallel applications to the available computing nodes. Current research in scheduling has concentrated on not only optimizing the energy consumed by the processors but also optimizing the makespan, i.e., job completion time. The large number of heterogeneous computing nodes and variability of application-tasks are factors that make the scheduling an NP-Hard problem. Our aim in this paper is a multi-objective genetic algorithm based on a weighted blacklist able to generate scheduling decisions that globally optimizes the energy consumption and the makespan.

Keywords Weighted blacklist · Energy aware scheduling · Multi-objective optimization · Genetic algorithm · Federated clusters · Co-allocation

✉ Eloi Gabaldon
eloigabal@diei.udl.cat

Josep Lluis Lerida
jlerida@diei.udl.cat

Fernando Guirado
f.guirado@diei.udl.cat

Jordi Planes
jplanes@diei.udl.cat

[1] Department of Computer Science, Universitat de Lleida, Lleida, Spain

1 Introduction

The computing requirements of modern scientific applications are growing continuously, as is the amount of data to process. The use of more sophisticated and scalable infrastructure becomes necessary to cover these requirements. Different architectural environments, such as Federated Clusters, Grid infrastructures or Cloud computing, are examples of federated resource environments, where computing resources in different administrative domains work together to solve a problem [1,2].

In this work, we focus on scheduling in Federated Cluster environments. These environments allow the execution of parallel applications in which the computing resource requirements exceed the resources available in a single cluster. Thus, parallel applications can be co-allocated to different clusters, sharing computation and communication resources. Co-allocation favor the reduction of internal cluster fragmentation, thus improving resource usage and increasing job throughput as the applications can reduce its waiting times [2,3]. However, mapping jobs across cluster boundaries can result in rather poor overall performance when co-allocated jobs contend for inter-cluster network bandwidth.

Beside this, the high energy consumption from the large-scale use of computing resources translates into high energy cost and carbon emissions which are not environmentally sustainable. Hence, there is an urgent need for energy-efficient solutions that redefine our way of using computing resources. Recent research has been able to determine the quantity of resources to be used based on load predictions, QoS requirements, etc., or dynamically redefine the voltage supply based on the workload. Nonetheless, a major factor in efficient resource utilization is the proper mapping and scheduling of parallel jobs among the computing processors.

The problem of scheduling parallel jobs within a heterogeneous distributed resource environment is known to be NP-hard [4,5], therefore, the use of heuristics and meta-heuristics is the best approach to cope with its difficulty. Current research is focused on minimizing the energy consumed by the processors together with the minimization of job completion time, flowtime, resource usage, etc, [6]. The motivation of this work is to provide a meta-heuristic technique able to obtain scheduling decisions efficiently considering resource heterogeneity and parallel application requirements within federated environments. The contribution of this paper is a novel scheduling approach based on a multi-objective Genetic algorithm (GA) which uses a weighted blacklist to determine dynamically, for each application, a clusterization of forbidden resources. The algorithm not only generates an optimal mapping of applications to resources but also a scheduling to complete the workload in a minimum period of time as well as utilizing the resources in an efficient way.

The remainder of this paper is organized as follows. Section 2 presents related work. The proposed genetic algorithm is elaborated in Sect. 3. Section 4 demonstrates the performance analysis and simulation results for real workload traces. Finally, the conclusions are presented in Sect. 5.

2 Related work

Scheduling strategies in heterogeneous distributed environments have generated great interest in recent years due to the growth of resources in organizations. The potential benefit of sharing computing resources among federated sites has been widely discussed in previous research [1,2,7]. Bucur et al. [2] carried out a performance evaluation of different scheduling strategies using co-allocation based on job queues. Their results show that unrestricted co-allocation is not recommended and performance is improved by correctly adjusting the component size of the co-allocated jobs. Other studies used co-allocation to develop load balancing techniques [8,9] or optimize the application execution time by selecting the most suitable resources [4,10].

Traditional scheduling techniques in the literature treat the jobs in the waiting queue individually without considering the remaining jobs in the batch queue [11], thus limiting the scheduling opportunities for future allocations and decreasing overall system performance. Shmueli et al. [12] proposed a backfilling technique in which later jobs are packaged to fill in holes and increase utilization without delaying the earlier jobs. Tsafrir et al. [13] proposed a method to select the most suitable jobs to be moved forward based on system-generated response time predictions. These techniques, however, are based on the job arrival order, only moving jobs forward that accomplish specific deadline requirements. Blanco et al. [14,15] proposed diverse techniques for determining the best scheduling of sets of job packages, proposing a new job execution order to minimize their overall execution time, based on a Mixed-Integer programming model. Due to the intractable nature of the problem, it is desirable to explore other avenues for developing good heuristic algorithms for the problem.

Some nature-inspired meta-heuristics, such as Simulated Annealing (SA), Genetic Algorithms (GA) and Particle Swarm Optimization (PSO), have been presented as effective schedulers in complex large-scale environments in attempts to obtain pseudo-optimal solutions in practical times for large-scale environments. GAs are well known for their robustness and have been applied successfully to solve scheduling problems in a variety of fields. Zomaya and Teh [16] used GAs in dynamic load balancing problems. Braun et al. [11] compared the efficiency of a simple GA-based scheduler and the MinMin, MinMax, Minimum Completion Time (MTC) algorithms. Carretero and Xhafa presented in [17] an extensive study of GAs for designing efficient Grid schedulers where makespan and flowtime are minimized to include QoS in the solutions, but considering independent jobs without inter-cluster communications. Gabaldon et al. [18] presented a GA-based scheduling meta-heuristic able to optimize the makespan together with the flowtime, thus providing a certain level of QoS from the users point of view.

Beside this, energy consumption has become a great challenge in the field of high performance computing, because of various concerns such as cost, environmental sustainability, and system performance. Two primary methods are commonly used: switching off underutilized resources [19–21] or using voltage and frequency scaling (VFS) techniques [6,22–24]. Cocaña et al. [19] presented a software tool that predicts the future node requirements using a machine-learning approach, and then stopping those that will not be required in the near future. Chae et al. [20] illustrate a method of determining the aggregate system load and the minimal set of computational resources

that can process the workload. Orgerie et al. in [21] present a three-step strategy based on a framework able to control the computing requirements by switching the unused nodes off, predicting their usage to switch on them again and finally aggregating some reservations to avoid frequent on/off cycles. Christobel et al. in [6] proposed an energy-aware scheduling approach for scientific workflows based on Particle Swarm Optimization and Dynamic Voltage Scaling. Kolodziej et al. in [22] and Kim et al. in [23] address independent batch scheduling in computational grids as a bi-objective global minimization problem with makespan and energy consumption criteria, using a VFS model directed by a GA-based grid scheduler.

Our proposal differs from research works [6,19–23] in two main aspects. We propose optimizing makespan and energy consumption at the level of the scheduler. Once the required computational nodes are identified, we treat the scheduling process not only to determine the job co-allocation to the computing resources but also to define the best execution order for parallel applications in the batch queue. We also deal with federated clusters environments, which are distinguished from Grid environments by the use of a dedicated interconnection network between cluster resources with a known topology and predictable performance characteristics.

In this paper, the main contribution is the implementation of a Multi-Objective GA-based scheduling algorithm, named MOGA, based on a weighted blacklist to determine the resource unavailability during the mapping process for a set of parallel jobs to be scheduled. Our proposal avoids the systematic allocation of jobs to resources with minimum consumption or maximum computational capacities, as is usual in the literature, providing new scheduling opportunities for the remaining jobs in the system queue, thus optimizing the energy consumption and the makespan.

3 Multi-objective genetic algorithm (MOGA)

A GA is a search heuristic to find near-optimal solutions using nature-based techniques. It starts by creating an initial population of solutions known as individuals, each one encoded using a chromosome. Four steps are carried out to create a new generation: ranking the individuals driven by a fitness function, selection, crossover of the selected individuals and mutation. The algorithm is motivated by the hope that after several generations, the new population will be better than the older ones.

In an earlier work [18], the authors presented *GA-MF*, a GA-based technique with the aim of increasing the system throughput for batch workloads, using the makespan and flowtime as the optimization criteria. Due to the increasing importance of developing energy-aware systems to reduce the environmental footprint, the authors propose a new multi-objective GA, named *MOGA*, focused on reducing both energy consumption and makespan.

The makespan (Eq. 1) is defined as the elapsed time between the submission of the first job until the finalization of the last one:

$$makespan = \max_{i \in \mathcal{J}}(F_i) - \min_{j \in \mathcal{J}}(S_j) \tag{1}$$

where F_i is the finish time of the job i, calculated as $F_i = S_i + T_i$; T_i is the execution time of the job i; and S_i is the starting time of the job i. The execution time of a job in an heterogeneous federated cluster is calculated using the model presented in [18], where resource heterogeneity and network saturation is considered.

The energy consumption is modeled by Eq. 2.

$$energy = \sum_{n} (C_n * CT_n + I_n * IT_n) \quad \forall n \in N \tag{2}$$

where C_n is the energy consumed by node n when it is computing, I_n when it is idle, CT_n is the computing time of the node n and IT_n is its idle time. Both models are related because of their dependency on the job execution time, thus trying to minimize one of them could have negative effects on the other one.

The first decision in developing a GA is the chromosome design used to represent the individuals in the population. The individual representation we designed is made up of two parts: the first part represents the order the jobs are going to be executed in the system; the second part is a weighted blacklist representing the resource unavailability for each job. The blacklist (BL) is implemented for each job as a list of real numbers, where each value is in the range [0, 1]. This value represents the percentage of cluster nodes forbidden from the allocation of the corresponding job, providing a reservation mechanism for further jobs on the queue. To illustrate our proposed individual design, we present the following example:

Example 1 Given a set of jobs $\mathcal{J} = \{J1, J2, J3, J4, J5\}$, where every job is made up of a set of tasks as follows: $J1 = \{T1, T2, T3, T4\}$, $J2 = \{T5\}$, $J3 = \{T6, T7, T8\}$, $J4 = \{T9, T10, T11, T12, T13\}$ and $J5 = \{T14\}$; the execution environment corresponds to a set of clusters $\mathcal{C} = \{C1, C2, C3\}$, where each cluster has a set of computing nodes as follows: $C1 = \{N1, N2\}$, $C2 = \{N3, N4, N5\}$, and $C3 = \{N6, N7\}$; a possible individual from the MOGA population is displayed in Fig. 1.

For this example, the corresponding scheduling and Gantt diagram of the job execution is shown in Fig. 2. Observe that job $J4$ is the last job to be scheduled and it cannot use any of the nodes in cluster $C3$, since the individual has a value of 1.0 in its node allocation spot. This means that 100 % of the nodes in this cluster are forbidden for this job. Also notice that the cluster $C3$ consumption is much higher than for the rest of the clusters and thus, it is a good choice not to use it to save energy. $J4$ will wait for the moment when all computing nodes from cluster $C1$ are free because its forbidden access to cluster $C3$.

The method to create the job scheduling is described in Algorithm 1. For every job in the set of jobs, and for every task in the job, a node in the list of free nodes is selected, taking into account the availability given by the blacklist, which is configured by the MOGA based on the optimization of both criteria energy and makespan. The method first searches for the most powerful nodes (lines 2–5), where $Power(n)$ is the computational power for node n. Note that for parallel jobs, the execution time is given by the slowest computational node selected. Then, the method tries to shift the tasks from powerful nodes to available slow nodes if the execution time remains the same, which may free up the assigned powerful nodes (lines 6–12). The solution is a list of assignments of every task to a node.

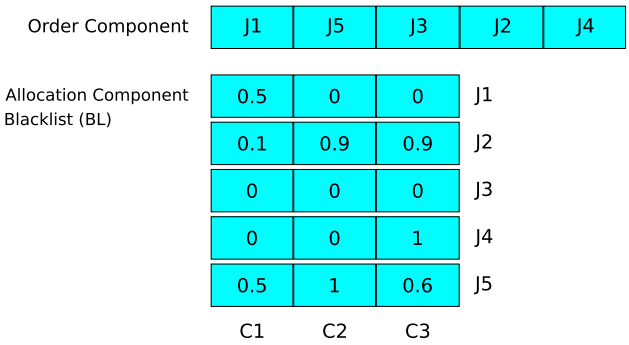
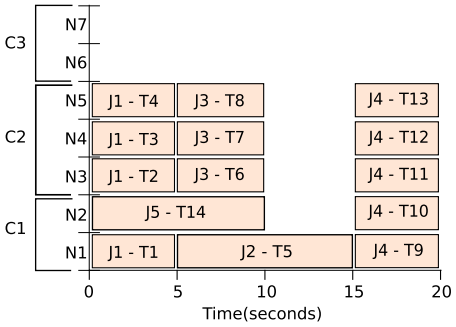


Fig. 1 MOGA individual representation for Example 1



Cluster	Energy Consumption	Computing Power
C1	200	1000
C2	200	1000
C3	400	1000

Fig. 2 Gantt diagram of Example 1

Algorithm 1 Allocation method

Require: $\mathcal{J}$: Set of jobs
Ensure: $\mathcal{A}$: Set of $(Task, Node)$
1: **for** $Job \in \mathcal{J}$ **do**
2: **for** $Task \in Job$ **do**
3: $N = FreeNodes - ForbiddenNodes$
4: $\mathcal{A} \leftarrow \mathcal{A} \cup (Task, \text{argmax}_{n \in N}(Power(n)))$
5: **end for**
6: **for** $(Task, Node) \in \mathcal{A} : Node = \text{argmax}_{(t,n) \in \mathcal{A}}(Power(n))$ **do**
7: $N = FreeNodes - ForbiddenNodes$
8: $Node' \in N : Power(Node') \geq \min_{(t,n) \in \mathcal{A}}(Power(n))$
9: **if** $\exists Node'$ **then**
10: $\mathcal{A} \leftarrow \mathcal{A} \setminus (Task, Node) \cup (Task, Node')$
11: **end if**
12: **end for**
13: **end for**

If we run out of computational nodes in the allocation process, MOGA predicts the first job to finish using the job execution model presented in [18] and then releases the allocated nodes for the subsequent jobs.

At each iteration, MOGA selects the best individuals placed on the non-dominated Pareto fronts using the NSGA-II algorithm [25]. To evolve to a new population, the

crossover affects the two components of the individual representation maintaining some similarities with the parents:

- *Order component* A mask of random binary values is generated. For every position with value 1, the job of the first parent is placed in the offspring. For the missing jobs, value 0, the order of the second parent is chosen.
- *Allocation component* A real value intermediate crossover is used. A random value α, in the range $[-0.2, 1.2]$, is selected to determine the new value in the blacklist matrix (BL) by applying Eq. 3. This range was chosen to avoid a premature convergence on a local minimum.

$$BL_{child} = \alpha * BL_{parent1} + (1 - \alpha) * BL_{parent2} \tag{3}$$

Finally, a *mutation* operator is applied to maintain the diversity across the generations. This process is applied in both components of the chromosome with a probability defined by the *Mutation Frequency* parameter: for the *Order component*, two randomly selected jobs from the list are swapped; for the *Allocation component*, a value from the blacklist is exchanged for a new random value.

4 Experimentation

In this section, we conducted an experimental study with the aim of analyzing the reduction in makespan and energy consumption obtained by applying the proposed MOGA scheduling technique, which was implemented using the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [25] and also prove its robustness and efficiency. The experimentation was carried out by simulation, using the Gridsim simulator [26] configured to emulate a Federated Cluster environment made up of 4 different clusters. The characteristics of each cluster are shown in Table 1. The consumptions of the computing nodes were defined using the study done by Orgerie et al. in [21].

To process the experimental study, 30 different job-sets corresponding to one month of execution time were selected, and each was processed 30 times. These job-sets were extracted from real cluster traces in the workloads HPC2N, RICC and CEA CURIE described by Feitelson [27].

The most relevant characteristics of these traces are:

Table 1 Federated cluster characteristics

Cluster	Nodes	Effective power MIPS	Energy idle KW/h	Energy computing KW/h
1	64	1000	130	300
2	64	1200	150	320
3	64	1300	250	400
4	64	1800	280	400

– Half of the jobs in the *HPC2N* are made up of 1–2 tasks and the rest, evenly distributed in 4, 8 and 16 tasks. The average execution time of this workload is around 1 h.
– 80 % of jobs in the *RICC* workload are made up of 1 task while the rest can contain up to 32, and the execution times of each job can vary, greatly from several seconds to 10 h.
– In the *CEA CURIE* workload, approximately 50 % of the jobs are made up of 32 tasks, with the execution time of 80 % of the jobs being extremely low, ranging from 1 s to 1 min.

4.1 Convergence of MOGA

In this section we conducted an experimentation to evaluate the MOGA convergence, check its robustness and identify the optimal number of iterations. This test was performed by selecting a job-set from each workload. Figure 3 displays the makespan and

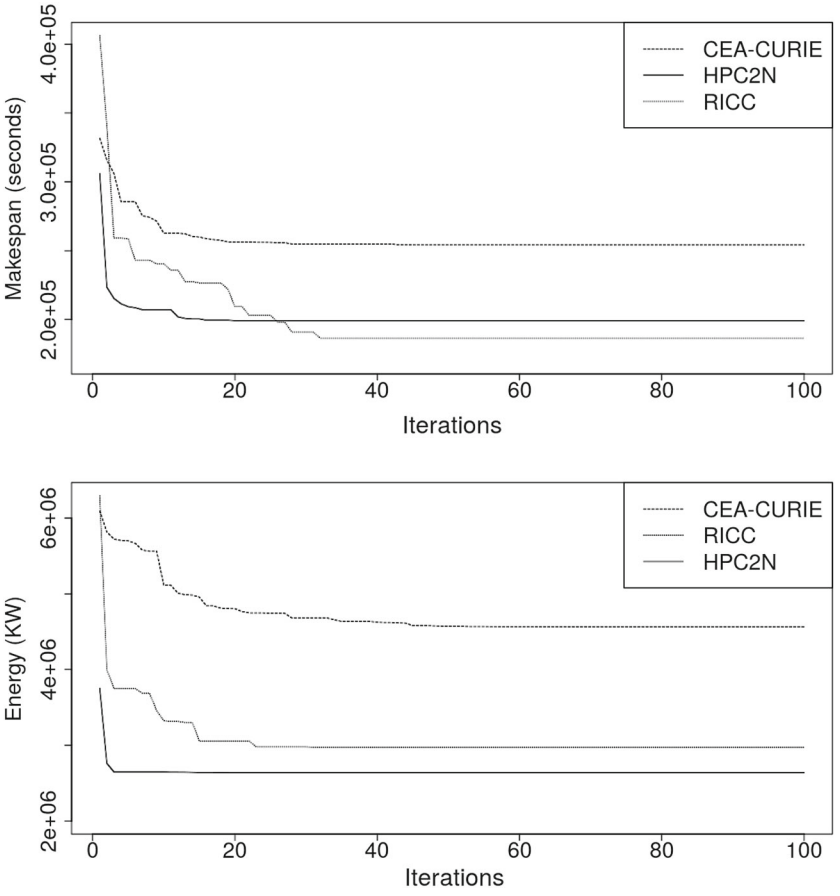


Fig. 3 Convergence study for the MOGA meta-heuristic for the makespan and energy criteria

Table 2 MOGA parameters

Parameter	Value
Num. iterations	60
Population size	80
Mutation frequency	10 %

Table 3 Heuristic characteristics

Technique	Modifies queue	Mapping objective
FCFS	No	Select available nodes
CBS	No	Avoid inter-cluster link saturation
Min-Min	Yes	Energy
JPR	No	Makespan or energy
PSO	No	Makespan and energy
HILL	Yes	Makespan and energy
MOGA	Yes	Makespan and energy

energy consumption (Y-axis) of the best individual for each generation (X-axis). Both parameters improve as more iterations are evaluated. Near 50 iterations, the algorithm converges in all experiments. In further experimentation, we set the number of iterations to 60. The set of parameters that guide our genetic algorithm proposal is shown in Table 2.

4.2 MOGA performance comparison

In the evaluation process, MOGA was compared with other well-known techniques in the literature that can be used in Federated Cluster environments. Table 3 shows the principal characteristics and differences among the techniques; the ability to modify the job execution order and the main goal of the mapping.

- *FCFS* is a technique that schedules the tasks of the first job in the scheduling queue to the available nodes, Schwiegelshohn et al. [28].
- The *CBS* heuristic was proposed by Jones et al. [29] and tries to allocate as much as possible the tasks of the first job in the queue to a single cluster in an attempt to avoid inter-cluster link saturation.
- *Min-Min* is a heuristic presented by Li et al. on [30] that is able to consider a set of jobs with the aim of minimizing energy consumption.
- *JPR* is a variant of the Naik et al. heuristic [10], where the jobs are matched with the best available resources depending on just only one of the criteria selected, makespan or energy.
- *PSO* is a particle swarm optimization presented by Oukfif et al. in [31] focused on minimizing both parameters.
- *HILL* uses the hill-climbing technique, a local search method [32] focused on minimizing both the makespan and the energy.

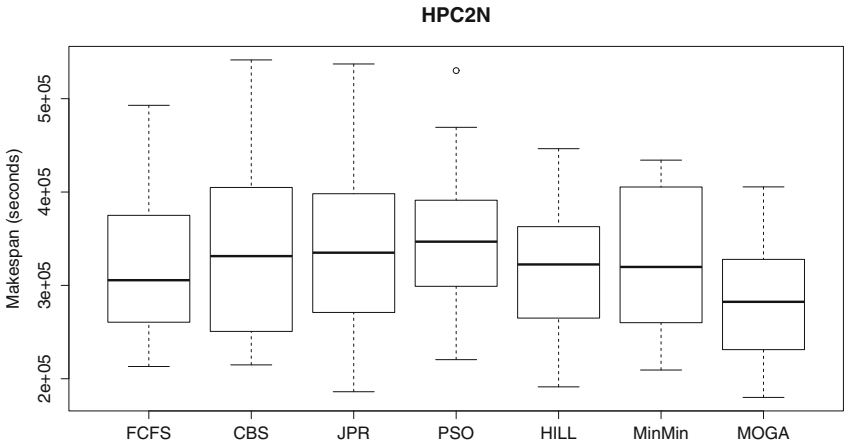


Fig. 4 Makespan study for the HPC2N workload

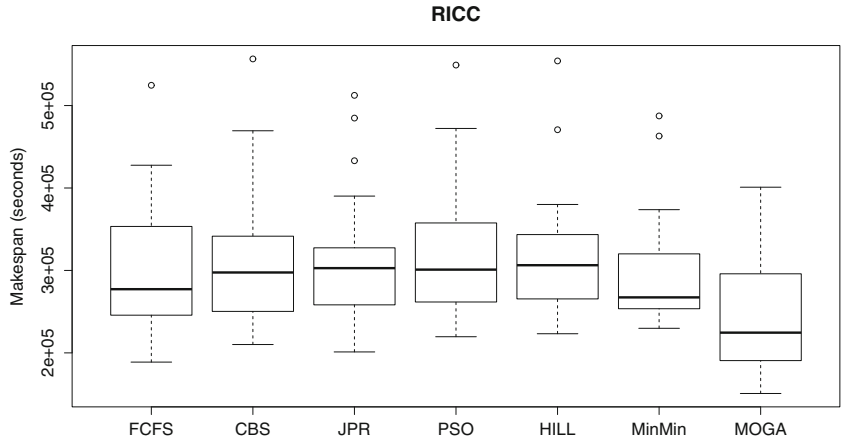


Fig. 5 Makespan study for the RICC workload

The PSO and HILL methods optimize both objectives based on the Eq. 4.2.

$$fitness = \alpha \times makespan + (1 - \alpha) \times energy$$

In this experimentation, the objectives were compared individually, first makespan and then energy. It is important to point out that the JPR, PSO and HILL techniques are able to optimize both objectives. To obtain the best solution for the previous techniques depending on the studied objective, they were set to optimize only one objective (makespan or energy, i.e., α set to 0 or 1). Meanwhile, MOGA optimizes both objective functions individually, finding the optimal solution.

We used a boxplot that synthesizes the most relevant information to present the results: these being the minimum, median and maximum consumption values as well

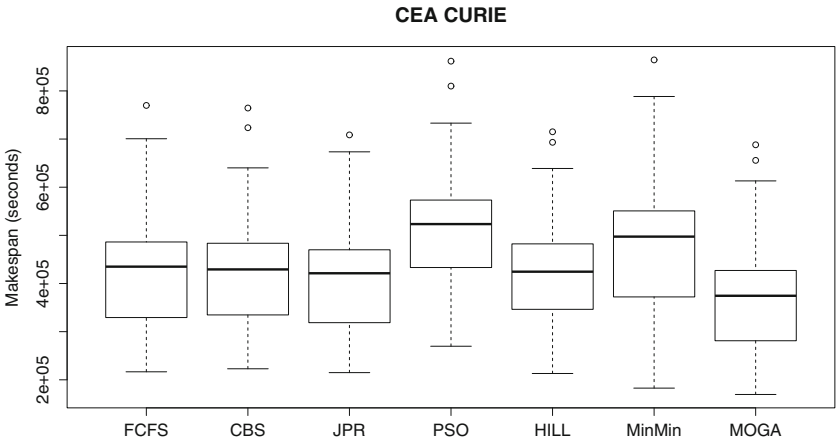


Fig. 6 Makespan study for the CEA CURIE workload

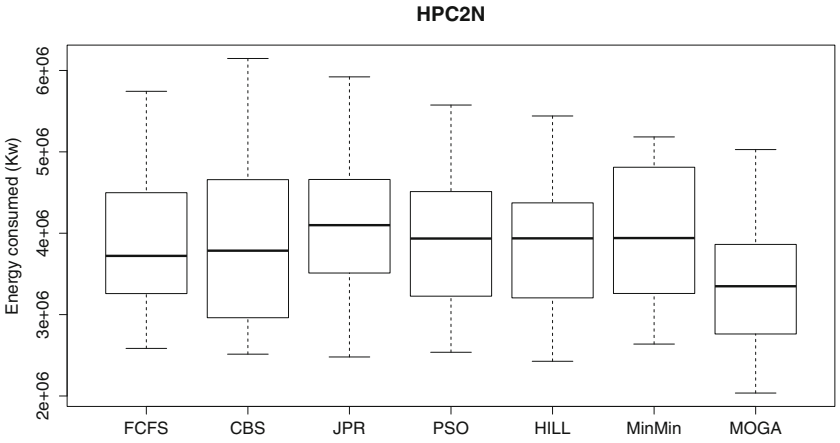


Fig. 7 Energy consumption study for the HPC2N workload

as the frequency of jobs in the first and third quartile. The outliers are displayed as dots.

First, we analyzed the makespan obtained by the techniques. Figure 4 shows the results obtained for the HPC2N workload. As we can see, MOGA obtained lower minimum and maximum values than the others, and the median was 10 % lower than the FCFS heuristic, the best of the literature techniques.

The results for the RICC workload can be seen in Fig. 5. The behavior of MOGA was similar to the previous experiment. It gave the best results among the other techniques, obtaining an even greater improvement than in the previous experiment.

In the case of the CEA CURIE, Fig. 6, the MinMin and PSO, which are meta-heuristic techniques, differ significantly from the others and produce the worst results, which is unexpectedly bad behavior. On the contrary, the MOGA meta-heuristic again obtained the best results.

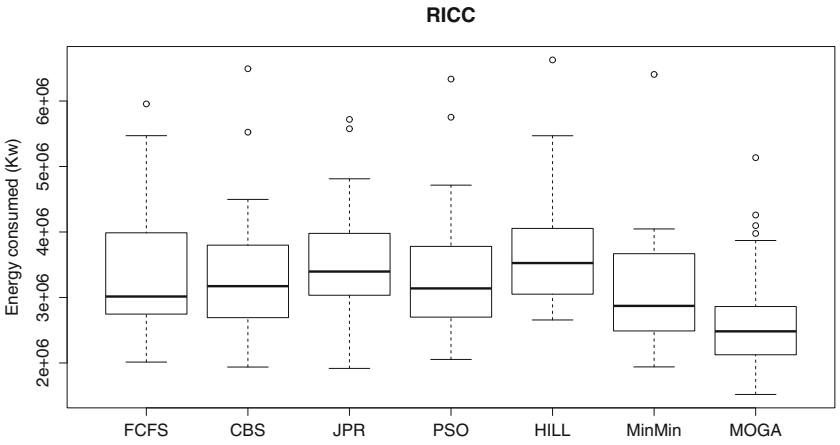


Fig. 8 Energy consumption study for the RICC workload

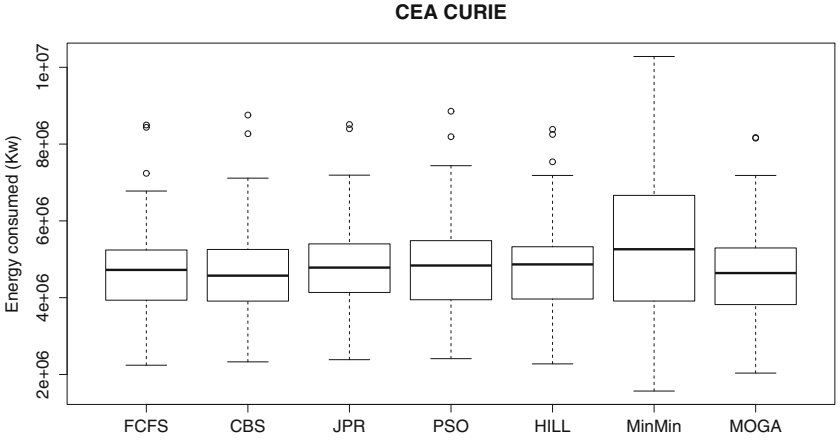


Fig. 9 Energy consumption study for the CEA CURIE workload

These results come from the fact that on evaluating the whole set of jobs, MOGA has the ability to forbid access to some nodes by means of the blacklist, providing a reservation mechanism for those jobs with computational requirements that can provide better benefits in the future.

The same analysis was done to observe the energy savings achieved by the techniques. For the HPC2N workloads Fig. 7, when using MOGA, the system consumed 10 % less energy when using the other techniques. This means that it reduced not only the makespan but also the energy consumptions.

In RICC workloads in Fig. 8, MOGA obtained also the better results, maintaining the reduction in both objectives. For the CEA-CURIE workloads Fig. 9, we can observe that all the techniques obtained very similar result. In this case, only MinMin differed from the others and obtained the worst results. It is important to point out that the jobs in this workload have very low execution times. This means that the dif-

Table 4 Summary of experimentation results

Policy	HPC2N	RICC	CEA-CURIE
FCFS			
Makespan	305,553	277,264	434,979
Energy	3,721,573	3,014,554	4,723,073
Compute time	0.65	0.64	0.68
CBS			
Makespan	331,362	297,525	429,279
Energy	3,785,578	3,172,691	**4,573,592**
Compute time	0.65	0.67	0.70
JPR			
Makespan	335,038	302,794	421,282
Energy	4,099,127	3,395,906	4,783,408
Compute time	0.69	0.68	0.73
PSO			
Makespan	346,849	300,998	523,415
Energy	3,934,970	3,138,362	4,837,743
Compute time	121.86	99.41	58.36
HILL			
Makespan	322,434	306,340	424,535
Energy	3,937,109	3,525,591	4,866,540
Compute time	39.05	68.24	40.12
MinMin			
Makespan	319,751	267,316	497,380
Energy	3,940,691	2,871,167	5,261,294
Compute time	1.24	1.22	0.95
MOGA			
Makespan	**282,396**	**224,534**	**374,517**
Energy	**3,348,483**	**2,482,854**	4,641,108
Compute time	8.9	10.21	6.92

ferences in scheduling decisions about resources allocation or job ordering have little effect on the consumption and makespan results. Nonetheless, it should be noted that MOGA presents a slight reduction on the makespan while obtaining the same energy consumption results as the other techniques.

To sum up the experimentation, the results in Table 4 show the median value of the makespan and energy consumption, for each workload and technique. We also add the computational cost in seconds for each heuristic. The table shows in bold the best results obtained for each workload. We can see that MOGA mainly obtained the best results in both objectives. In the CEA-CURIE workload, CBS obtained a slightly better energy consumption than MOGA, but with a much higher makespan.

We can observe that the computational costs of JPR, CBS , FCFS and MinMin are very low in comparison with the other meta-heuristics, however, MOGA obtains the best results and also lower computational times than PSO and HILL meta-heuristics.

We can conclude that the workloads with higher variability in the execution times (HPC2N and RICC) produce better scheduling chances. Our proposal, the MOGA technique, using a blacklist of computational resources, has proved to be effective for energy saving and makespan reduction and also shows a low sensitivity to workloads variations. However, for situations where the nature of the workloads does not provide scheduling opportunities, as in the case of the CEA CURIE, all the techniques gave similar results. Nevertheless, our proposal managed to obtain lower makespan while maintaining the energy consumption.

5 Conclusions

In the present paper, a Multi-Objective Genetic Algorithm (MOGA) is presented to efficiently obtain an efficient scheduling process taking into account not only optimization of the global makespan, but also reduction of energy consumption. Our proposal was developed using a novel technique based on a resource allocation blacklist that has the ability to forbid access to some computational nodes, providing a reservation mechanism for those jobs with computational requirements that can provide higher benefits.

The experimental study was carried out by simulation and was conducted with real workload traces in a heterogeneous federated cluster environment. The MOGA results were compared with other techniques in the literature and demonstrated its capability to obtain better results while optimizing both objectives, makespan and energy efficiency. Furthermore, the solutions provided had lower dispersion results allowing us to conclude that the robustness of MOGA does not depend on the nature of the workloads.

MOGA is focused on the low-level scheduling process, which relates the job to resources allocation. Accordingly, in future work, we aim to combine our proposal with high-level strategies that predict the minimal set of computational resources to deal with a future workload.

Acknowledgements This research is partly supported by the European Union FEDER (CAPAP-H5 network TIN2014-53522-REDT) and MEyC-Spain under contract TIN2014-53234-C2-2-R and TIN2015-71799-C2-2-P.

References

1. Ernemann C, Hamscher V, Uwe S, Ramin Y, Achim S (2002) On advantages of grid computing for parallel job scheduling. In: CCGRID, IEEE, pp 39–39
2. Bucur AID, Epema DHJ (2007) Scheduling policies for processor coallocation in multicluster systems. Trans Parallel Distrib Syst IEEE 18(7):958–972
3. Blanco H, Montañola A, Guirado F, Lerida JL (2010) Fairness scheduling for multi-cluster systems based on linear programming. CMMSE 1:227–239
4. Jones WM, Ligon WB, Pang LW, Stanzione D (2005) Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters. J Supercomput 34(2):135–163

5. Abawajy JH, Dandamudi SP (2003) Parallel job scheduling on multicluster computing system. In: Cluster Computing, 2003. In: Proceedings. 2003 IEEE International Conference on, pp 11–18
6. Christobel M, Tamil Selvi S, Benedict S (2015) Efficient scheduling of scientific workflows with energy reduction using novel discrete particle swarm optimization and dynamic voltage scaling for computational grids. Sci World J
7. Muthuvelu N, Vecchiola C, Chai I, Chikkannan E, Buyya R (2013) Task granularity policies for deploying bag-of-task applications on global grids. Future Gener Comput Syst 29(1):170–181
8. Heien EM, Fujimoto N, Hagihara K (2008) Static load distribution for communication intensive parallel computing in multiclusters. In: Parallel Distributed and Network-Based Processing, IEEE, pp 321–328
9. Yang CT, Tung HY, Chou KY, Chu WC (2008) Well-balanced allocation strategy for multi-cluster computing environments. Future Trends Distrib Comput Syst IEEE 0:178–184
10. Naik VK, Liu C, Yang L, Wagner J (2005) Online resource matching for heterogeneous grid environments. In: CCGRID, pp 607–614
11. Braun TD, Siegel HJ, Beck N, Bölöni LL, Maheswaran M, Reuther AI, Robertson JP, Theys MD, Yao B, Hensgen D, Freund RF (2001) A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. J Parallel Distrib Comput 61(6):810–837
12. Shmueli E, Feitelson DG (2005) Backfilling with lookahead to optimize the packing of parallel jobs. J Parallel Distrib Comput 65(9):1090–1107
13. Tsafrir D, Etsion Y, Feitelson DG (2007) Backfilling using system-generated predictions rather than user runtime estimates. Trans Parallel Distrib Syst IEEE 18(6):789–803
14. Blanco H, Lerida JL, Cores F, Guirado F (2011) Multiple job co-allocation strategy for heterogeneous multi-cluster systems based on linear programming. J Supercomput 58(3):394–402
15. Blanco H, Guirado F, Lerida JL, Albornoz VM (2012) Mip model scheduling for bsp parallel applications on multi-cluster environments. In: 3PGCIC, IEEE, pp 12–18
16. Zomaya AY, Teh YH (2001) Observations on using genetic algorithms for dynamic load-balancing. Trans Parallel Distrib Syst IEEE 12(9):899–911
17. Carretero J, Xhafa F, Abraham A (2007) Genetic algorithm based schedulers for grid computing systems. Int J Innov Comput Inf Control 3(6):1–19
18. Gabaldon E, Lerida JL, Guirado F, Planes J (2015) Multi-criteria genetic algorithm applied to scheduling. J Simul 9(4):287–295
19. Cocaña A, Ranilla J, Sánchez L (2015) Energy-efficient allocation of computing node slots in HPC clusters through parameter learning and hybrid genetic fuzzy system modeling. J Supercomput 71(3):1163–1174
20. Chase JS, Anderson DC, Thakar PN, Vahdat AM, Doyle RP (2001) Managing energy and server resources in hosting centers. SIGOPS Oper Syst Rev 35(5):103–116
21. Orgerie AC, Lefevre L, Gelas JP (2008) Save watts in your grid: Green strategies for energy-aware framework in large scale distributed systems. In: Parallel and Distributed Systems, 2008. ICPADS '08. 14th IEEE International Conference on, pp 171–178
22. Koodziej J, Khan SU, Wang L, Zomaya AY (2015) Energy efficient genetic-based schedulers in computational grids. Concurrency Comput Pract Exp 27(4):809–829
23. Hoon KK, Rajkumar B, Jong K (2007) Power aware scheduling of bag-of-tasks applications with deadline constraints on dvs-enabled clusters. CCGRID 7:541–548
24. Lin YC, You YP, Huang CW, Lee JK, Shih WK, Hwang TT (2007) Energy-aware scheduling and simulation methodologies for parallel security processors with multiple voltage domains. J Supercomput 42(2):201–223
25. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: Nsga-ii. IEEE Trans Evol Comput 6(2):182–197
26. Garg SK (2009) Gridsim simulation framework. http://www.buyya.com/gridsim. Accessed 21 June 2015
27. Feitelson D (2005) Parallel workloads archive. http://www.cs.huji.ac.il/labs/parallel/workload. Accessed 15 Jan 2016
28. Schwiegelshohn U, Yahyapour R (1998) Analysis of first-come-first-serve parallel job scheduling. In: SODA, vol 98, pp 629–638. Citeseer
29. Jones WM, Ligon III WB, Pang LW, Stanzione DC Jr (2005) Characterization of bandwidth-aware meta-schedulers for co-allocating jobs across multiple clusters. J Supercomput 34(2):135–163

30. Li Y, Liu Y, Qian D (2009) A heuristic energy-aware scheduling algorithm for heterogeneous clusters. In: Parallel and Distributed Systems (ICPADS), 2009 15th International Conference on, pp 407–413
31. Oukfif K, Bouali L, Bouzefrane S, Boumghar F (2015) Energy-aware dpso algorithm for workflow scheduling on computational grids. In: Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on, pp 651–656. IEEE
32. Balas E, Vazacopoulos A (1998) Guided local search with shifting bottleneck for job shop scheduling. Manag Sci 44(2):262–275