Multiobjective Energy-Aware Workflow Scheduling in Distributed Datacenters

Sergio Nesmachnow[1(✉)], Santiago Iturriaga[1], Bernabé Dorronsoro[2], and Andrei Tchernykh[3]

[1] Universidad de la República, Montevideo, Uruguay
sergion@fing.edu.uy
[2] Universidad de Cádiz, Cádiz, Spain
[3] CICESE Research Center, Ensenada, Mexico

Abstract. This article presents a multiobjective approach for scheduling large workflows in distributed datacenters. We consider a realistic scheduling scenario of distributed cluster systems composed of multi-core computers, and a multi-objective formulation of the scheduling problem to minimize makespan, energy consumption and deadline violations. The studied schedulers follow a two-level schema: in the higher-level, we apply a multiobjective heuristic and a multiobjective metaheuristic, to distribute jobs between clusters; in the lower-level, specific backfilling-oriented scheduling methods are used for task scheduling locally within each cluster, considering precedence constraints. A new model for energy consumption in multi-core computers is applied. The experimental evaluation performed on a benchmark set of large workloads that model different realistic high performance computing applications demonstrates that the proposed multiobjective schedulers are able to improve both the makespan and energy consumption of the schedules when compared with a standard Optimistic Load Balancing Round Robin approach.

1 Introduction

Datacenters are large supercomputing facilities hosting computing resources that provide multiple services, including computing power, networking, storage, etc. in different application domains, including science, industry and commerce [29].

New paradigms for computation that propose using geographically distributed infrastructures to deal with complex problems (i.e. *grid* and *cloud* computing) have gained notorious interest due to the emergence of modern datacenter facilities and parallel computing methodologies and libraries. Indeed, a federation of distributed datacenters provides a significantly large amount of computing power to be used in modern supercomputing applications. Each datacenter in a federation is typically composed by a large number of computational resources, including high performance clusters, large storage systems, and/or components of large grids or cloud systems [30].

Energy efficiency has become a major issue when using large computing infrastructures. The energy consumption of datacenters should be kept as low

© Springer International Publishing Switzerland 2016
I. Gitler and J. Klapp (Eds.): ISUM 2015, CCIS 595, pp. 79–93, 2016.
DOI: 10.1007/978-3-319-32243-8_5

as possible, for both economic and environmental reasons. However, energy efficiency is in conflict with the performance of the system, since increasing the performance requires using more energy, and reducing the energy consumption will negatively affect the Quality of Service (QoS) that the computing system provides to the users. Thus, a multi-objective analysis is needed for finding accurate solutions of the datacenter planning problem, providing different trade-offs between energy consumption and performance.

Different techniques for reducing the energy consumption in datacenters have been proposed, ranging from ad-hoc hardware solutions to more general software methods adapted for specific infrastructures [1,24,25,28].

This article presents a hierarchical multi-objective approach for energy-aware scheduling of large workloads into a federation of distributed datacenters, composed by a number of clusters that might be geographically distributed, which is indeed the architecture of modern high performance and distributed computing systems, including big supercomputers, high performance computing centers, and cloud infrastructures, among others. We extend the greedy list scheduling heuristic approach for multi-core heterogeneous computing systems presented in our previous works [6,17] to consider: (i) a hierarchical model that uses two levels for assigning jobs to resources; (ii) the scheduling of large workflows having tasks with dependencies; and (iii) the utilization of a mutiobjective evolutionary algorithm to decide the best assigning of jobs to distributed cluster nodes.

The hierarchical two-level approach [7,20,21] divides the scheduling problem into a number of simpler and smaller sub-problems to be solved in each component of the datacenter infrastructure, and a specific ad-hoc backfilling heuristic based on combining the makespan, the energy consumption, and the QoS of solutions is presented for scheduling within each cluster. In this work, we measure the QoS of each schedule using a simple metric that accounts for the jobs whose deadlines are not met.

The experimental evaluation of the studied schedulers is performed over a benchmark set of 75 workloads with large jobs that model typical high performance computing applications over realistic distributed infrastructures. Three classes of workloads are considered: Series-Parallel, Heterogeneous-Parallel, and Mixed. Each problem instance contains 1000 jobs, with up to 132 tasks each, to be scheduled in a federation of datacenters with up to 1500 computational resources. The experimental results demonstrates that accurate solutions are computed by the best performing schedulers, allowing the planner to achieve improvements of up to **17.9** % in makespan, **20.7** % in energy consumption, and **36.4** % in deadline violation penalization over a traditional optimistic load balancing round-robin strategy.

The article is organized as follows. The problem formulation and review of the related work are presented in Sect. 2. The scheduling approach and the proposed methods ares described in Sect. 3. The experimental evaluation is reported in Sect. 4, including a comparison against a traditional optimistic load balancing round robin approach. Finally, Sect. 5 formulates conclusions and main lines for future work.

2 Energy-Aware Scheduling in a Federation of Datacenters

This section introduces the problem model and discusses the related work about energy-aware scheduling in datacenters.

2.1 Problem Model and Formulation

The energy-aware scheduling problem addressed in this article considers the following elements:

- A distributed infrastructure (datacenter federation) formed by k heterogeneous *Cluster Nodes* (the datacenters) $CN = \{CN_0, CN_1, \ldots, CN_k\}$. Each CN is a collection of NP_r multi-core processors, which is characterized by five values $(NP_r, ops_r, c_r, E^r_{IDLE}, E^r_{MAX})$, defining the number of processors, their performance (in FLOPS) and number of cores, and the energy consumption of each processor at idle and peak usage, respectively.
- A set of n independent heterogeneous jobs $J = \{j_0, j_1, \ldots, j_n\}$. Each job j_q has an associated deadline D_q. Each job j_q is a parallel application that is decomposed into a (large) set of tasks $T_q = \{t_{q0}, t_{q1}, \ldots t_{qm}\}$ with dependencies among them. Typically, each task has different computing requirements.
- Each task $t_{q\alpha}$ is characterized by two values $(o_{q\alpha}, nc_{q\alpha})$ defining its length (number of operations), and the number of resources (cores) required for the parallel execution, respectively.

Each job is represented as a *Directed Acyclic Graph* (DAG), i.e. a precedence task graph $j_q = (V, E)$, where the set of nodes V contains each task $t_{q\alpha}$ $(0 \leq \alpha \leq m)$ of the parallel program j_q. The set of (directed) edges E represents the dependencies between tasks, a partial order $t_{q\alpha} \prec t_{q\beta}$ that models the precedence constraints: an edge $e_{\alpha\beta} \in E$ means that task $t_{q\beta}$ cannot start before task $t_{q\alpha}$ is completed. We consider negligible communication costs, as they only occurs between servers within the same *CN*.

We are dealing with large workloads, so the problem instances are composed of thousands of jobs (this means hundreds of thousands of tasks) to be scheduled onto a number of CN (hundreds to thousands computing resources).

The described scheduling scenario is modeled with the multi-objective problem min (f_M, f_E), that proposes the simultaneous optimization of the *makespan* f_M and the *energy consumption* f_E.

The makespan evaluates the total time to execute a set of jobs, according to the expression in Eq. 1, where $\boldsymbol{x}$ represents an allocation, k is the number of available cluster nodes, and CT_r is the completion time of cluster node r (CN_r). The *energy consumption* function for a set of jobs executed in certain cluster nodes is defined in Eq. 2, using the energy model for multi-core architectures by Nesmachnow et al. [17], where f_1 is the higher-level scheduling function, and f_2 is the lower-level scheduling function. Both the energy required to execute the tasks assigned to each computing resource within a CN, and the energy that each

resource consumes in idle state are taken into account. The deadline violation penalization is defined in Eq. 3. A penalty function $Penalty_q(F_q)$ is associated with every application j_q, where F_q is the additional amount of time required to finish the execution of j_q after its deadline D_q is met. If j_q is finished before its deadline, then F_q is 0. Three different penalization functions are used in this work, a simple identity function ($Penalty_q(F_q) = F_q$), a square root function ($Penalty_q(F_q) = \sqrt{F_q}$), and a square function ($Penalty_q(F_q) = F_q^2$).

$$f_M(\boldsymbol{x}) = \max_{0 \leq r \leq k} CT_r \tag{1}$$

$$f_E(\boldsymbol{x}) = \sum_{r \in CN} \sum_{\substack{j_q \in J: \\ f_1(j_q) = CN_r}} \sum_{\substack{t_{qi} \in T_q: \\ f_2(t_{qi}) = p_j}} EC(t_{qi}, p_j) + \sum_{p_j \in CN} EC_{IDLE}(p_j) \tag{2}$$

$$f_P(\boldsymbol{x}) = \sum_{j_q \in J} Penalty_q(F_q) \tag{3}$$

In this article, we study the optimization problem from the point of view of the computing system (i.e. the infrastructure administration), thus we use two system-related objectives. Additionally, we consider a QoS-related objective such as the number of job deadlines violated, taking into account the point of view of the customer/user in the problem formulation.

2.2 Related Work

Many works in the literature have dealt with energy-aware scheduling in computing systems. Two main optimization approaches are established: independent and simultaneous. In the *independent* approach, energy and performance are assumed independent, so scheduling algorithms that optimize classic performance metrics are combined with a slack reclamation technique, such as dynamic voltage scaling (DVS)/dynamic voltage and frequency scaling (DVFS) [3,22]. In the *simultaneous* approach, performance and energy are simultaneously optimized, and the problem is modeled as a multi-constrained, bi-objective optimization one. The algorithms are oriented to find Pareto optimal schedules; where no scheduling decision can strictly dominate the other ones with better performance and lower energy consumption at the same time.

In this article, we follow the simultaneous approach. Below we briefly review the main related works about simultaneous optimization of energy and performance metrics.

Khan and Ahmad [9] applied the concept of Nash Bargaining Solution from game theory for scheduling independent jobs, simultaneously minimizing makespan and energy on a DVS-enabled grid system. Lee and Zomaya [11] studied several DVS-based heuristics to minimize the weighted sum of makespan and energy. A makespan conservative local search technique is used to slightly modify scheduling decisions when they do not increase energy consumption for

executing jobs, in order to escape from local optima. Later, Mezmaz et al. [15] improved the previous work by proposing a parallel bi-objective hybrid genetic algorithm (GA) for the same problem, using the cooperative island/multi-start farmer-worker model, significantly reducing the execution time of the scheduling method. Pecero et al. [18] proposed a two-phase bi-objective algorithm based on the Greedy Randomized Adaptive Search Procedure (GRASP) that applies a DVS-aware bi-objective local search to generate a set of Pareto solutions.

Kim et al. [10] studied the priority/deadline constrained scheduling problem in ad-hoc grids with limited-charge DVS-enabled batteries, and proposed a resource manager to exploit the heterogeneity of tasks while managing the energy. Luo et al. [14] showed that batch mode dynamic scheduling outperforms online approaches, though it requires significantly more computation time too.

Li et al. [12] introduced a MinMin-based online dynamic power management strategy with multiple power-saving states to reduce energy consumption of scheduling algorithms. Pinel et al. [19] proposed a double minimization approach for scheduling independent tasks on grids with energy considerations, first applying a MinMin approach to optimize the makespan, and then a local search to minimize energy consumption. Lindberg et al. [13] proposed six greedy algorithms and two GAs for solving the makespan-energy scheduling problem subject to deadline and memory requirements.

In our previous work [17], we introduced an energy consumption model for multi-core computing systems. Our approach did not use DVS nor other specific techniques for power/energy management. Instead, we proposed an energy consumption model based on the energy required to execute tasks at full capacity, the energy when not all the available cores of the machine are used, and the energy that each machine on the system consumes in idle state. We proposed twenty fast list scheduling methods adapted to solve a bi-objective problem, by simultaneously optimizing both makespan and energy consumption when executing tasks on a single cluster node. Using the same approach, Iturriaga et al. [8] showed that a parallel multi-objective local search based on Pareto dominance outperforms deterministic heuristics based on the traditional Min-Min strategy.

In [8,17], we tackled the problem of scheduling independent Bag-of-Tasks (BoT) applications. In this article, we extend the previous approach to solve a more complex multi-objective optimization problem, by considering large jobs, whose tasks have precedences, modeled by DAGs. In addition, here we propose a fully hierarchical scheduler that operates in two levels for efficiently planning large jobs in distributed datacenters.

3 The Proposed Hierarchical Energy-Aware Schedulers for Federations of Datacenters

We propose a hierarchical two-level scheduling approach, which fits properly to our problem model and the considered nowadays distributed infrastructures.

The higher-level scheduler (executing in a service front-end) applies a cluster assignment optimization, adapting a combined heuristic from our previous

work [17], in order to distribute jobs to cluster nodes. Within each cluster node, the lower-level scheduler applies a local scheduler specifically conceived for multi-core architectures and managing idle times (we called them *holes*) due to core availability. Both methods are described in the next section.

3.1 Lower-Level Scheduler

The proposed low-level scheduling heuristics are based on the Heterogeneous Earliest Finish Time (HEFT) strategy [27]. HEFT is a successful scheduler for DAG-modeled applications that works by assigning priorities to tasks, taking into account the *upward rank* metric, which evaluates the expected distance of each task to the last node in the DAG (the end of computation). The upward rank is recursively defined by $UR_i = t_i + \max_{j \in succ(i)} c_{ij} + UR_j$, where t_i is the execution time of task i in the computing resources, *succ* is the list of successors of task i, and c_{ij} is the communication cost between tasks i and j. After sorting all tasks of the job by taking into account the upward rank metric, HEFT assigns the task with the highest upward rank to the computing element that computes it at the earliest time.

The proposed heuristic for low-level scheduling in datacenters is *Earliest Finish Time Hole* (EFTH). It follows the schema of HEFT, using a backfilling technique and adapting the algorithm to work with multi-core computing resources, by taking into account the "holes" that appear when a specific computing resources is not fully used by a single task.

EFTH sorts the tasks according to the upward rank values, then gives priority to assign the tasks to existing holes rather than using empty machines in the CN. When a given task fits on more than one hole, the heuristic selects the hole that can complete the task in the earliest time, disregarding the hole length or other considerations. As a consequence, this variant targets the reduction of deadline violations and the improvement of the QoS for the users of the datacenter. When no holes are available to execute the task, the heuristic chooses the machine with the minimum finish time for that task.

The rationale behind this strategy is to use available holes and left unoccupied large holes and empty machines for upcoming tasks. Ties between holes as well as between machines are decided lexicographically, as the method searches sequentially (in order) both holes and machines.

3.2 Higher-Level Scheduler

The higher-level scheduler assigns jobs to cluster nodes. In this work, we study two algorithms: a specific version of the two-phase combined heuristic MaxMIN [17] and a multiobjective evolutionary algorithm, NSGA-II.

MaxMIN. The class of combined heuristics is a set of specific greedy list scheduling methods, which combine the makespan and energy consumption optimization criteria for scheduling in multi-core computers. Originally proposed to schedule independent tasks following the Bag-of-Task model [17,26], in this work

we extend the greedy approach in order to schedule large workflows having tasks with dependencies. MaxMIN operates in two phases. First, it builds a set of pairs (job, cluster node), by associating every job to the cluster node that can complete it with less energy use, taking into account all previous assignments already performed for each CN. After that, among all these pairs, it chooses the one with the maximum completion time among feasible assignments (i.e., the servers of the cluster node have enough cores to execute the job). Therefore, larger tasks are allocated first in the most suitable cluster nodes and shorter tasks are mapped afterward, trying to balance the load of all cluster nodes and making use of available results. When deciding where to assign a given job, MaxMIN first checks which CNs are able to execute the job, meaning that their servers have enough number of cores to execute any task in the job. In order to guide the search of the MaxMIN scheduler, we use heuristic functions to estimate the execution time and the energy required to execute each jobs. We approximate the completion time of a job in the assigned CN as the sum of the expected time to compute all tasks in the job, if they were executed sequentially, divided by the total number of cores available in the CN. To estimate the energy consumption when executing the job j_q in CN_r, we multiply the execution time estimation by the number of processors in CN_r and the energy consumption of such processors at peak power, and add it to the time the CN_r remains idle after finishing its assigned jobs until the last CN executes all jobs (i.e., the makespan value).

NSGA-II. Evolutionary algorithms (EAs) are non-deterministic methods that emulate the evolution of species in nature to solve optimization, search, and learning problems [2]. In the last thirty years, EAs have been successfully applied for solving many high-complexity optimization problems. Multiobjective evolutionary algorithms (MOEAs) [4,5] have been applied to solve hard optimization problems, obtaining accurate results when solving real-life problems in many research areas. Unlike many traditional methods for multiobjective optimization, MOEAs are able to find a set with several solutions in a single execution, since they work with a population of tentative solutions in each generation. MOEAs must be designed taking into account two goals at the same time: (*i*) approximating the Pareto front, usually applying a Pareto-based evolutionary search and (*ii*) maintaining diversity instead of converging to a reduced section of the Pareto front, usually accomplished by using specific techniques also used in multimodal function optimization (sharing, crowding, etc.).

In this work, we apply the *Non-dominated Sorting Genetic Algorithm, version II* (NSGA-II) [5], a popular state-of-the-art MOEA that has been successfully applied in many application areas. NSGA-II includes features to deal with three criticized issues on its predecessor NSGA, to improve the evolutionary search: (*i*) an improved non-dominated elitist ordering that diminishes the complexity of the dominance check; (*ii*) a crowding technique for diversity preservation; and (*iii*) a new fitness assignment method that considers the crowding distance values. Next we present the main characteristics of the proposed NSGA-II algorithm.

Solution Encoding. Each solution is encoded as a set of lists of integers. Each list represents the job execution queue for each data center and contains the

identifiers of its assigned jobs. The execution order of the jobs in each data center is given by the order of the job identifiers in each list.

Fitness Function. The fitness function is computed using the EFTH algorithm. Given a higher-level schedule, EFTH computes the lower-level scheduling and calculates the makespan, energy consumption, and violation penalization metrics.

Population Initialization. The initial population is created randomly using an uniform distribution function.

Selection Operator. Selection is performed using the binary tournament method. This method randomly selects two solutions from the population. If one of the selected solutions is dominated, then it is discarded and the non-dominated solution is selected. If both solutions are non-dominated, then the solution which is in the most crowded region is discarded and the remaining solution is selected.

Crossover Operator. The well-known Partially Matched Crossover (PMX) method is used as the crossover operator. To apply this method, a single job list is constructed for each parent by concatenating the job list of every data centres. Two jobs are randomly selected from this list as cutting points. All jobs in between these two points are swapped. The remaining jobs are rearranged using position wise exchanges, maintaining its original ordering information. Finally, the resulting list is disaggregated to reconstruct a job list for each data centre.

Mutation Operator. A simple exchange method is used as the mutation operator. This method works by randomly selecting a job and swapping it with another randomly selected job from any job list.

Repair Operator. This special operator repairs an infeasible solution turning it into a feasible solution. It is applied right after the Crossover and Mutation operators in order to repair any infeasibility introduced by these operators.

3.3 Baseline Scheduler for the Comparison

In order to compare results computed by the proposed schedulers, we consider a typical scenario as a baseline reference, applying a load balancing method and a backfilling technique such as the ones traditionally used in current cluster, grid, and cloud management systems.

Both methods are described next:

- In the higher-level, *Optimistic Load Balancing Round Robin* (OLB-RR) [6] assigns every job to a cluster node trying to balance the load between them. If the job can not be executed in the selected cluster node (because some task in it requires more cores than the number of cores of the servers in the cluster node), then the heuristic continues the iteration to the next ones until a suitable cluster node is found.

– In the lower-level, *NOUR Best Fit Hole* (NOUR) [6] applies a "best fit hole" strategy, i.e. selecting the hole with the closest length to the execution time of the task, but without taking into account the task sorting using the upward rank metric. Instead, the heuristic simply sorts the list of tasks lexicographically (from task #0 to task #N), but it obviously takes into account the precedence graph. This heuristic is intended to produce simple and compact schedules by not sticking to the importance given by the upward rank metric.

4 Experimental Analysis

This section reports the experimental analysis of the proposed hierarchical scheduling methods.

4.1 Problem Instances

A benchmark set of 75 different workflows batches was generated for the experimental evaluation of the proposed energy-aware hierarchical scheduler. The number of tasks in workflows ranges from 3 to 132. Workflows were generated using the SchMng application [23].

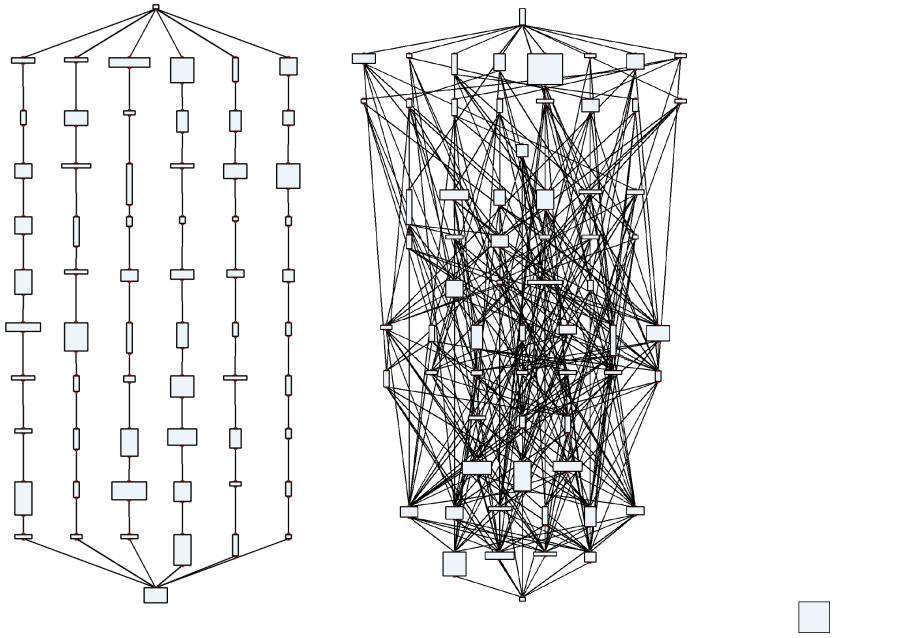
We use three different workflow models to consider different problem scenarios: (1) *Series-Parallel* (2) *Heterogeneous-Parallel*, and (3) *Mixed*. The Series-Parallel model represents jobs that can be split into concurrent threads/processes running in parallel. Heterogeneous-Parallel represent a generic job composed of non-identical computational tasks with arbitrary precedences. The Mixed workflow category combines Series-Parallel, Heterogeneous-Parallel and single-task jobs. Figure 1 shows the overall shape of the different workflow types, aimed to reflect real high performance computing applications. Each block represents a computational task, the execution time of a task is represented by the height of the block, and the number of cores is represented by the width of the block. Dependencies are represented by the edges in the graph.

In the benchmark set of 75 batch of workflows, 25 correspond to 1000 Series-Parallel workflows (25000 workflows altogether), 25 are composed of 1000 Heterogeneous-Parallel workflows (25000 workflows altogether), and the remaining 25 are Mixed, including a combination of different workflow types (300 Heterogeneous-Parallel workflows, 300 Series-Parallel workflows, and 400 Single-Task applications). A total number of **75000** workflows are studied in the experimental analysis. The benchmark set of workflows is publicly available at https://www.fing.edu.uy/inco/grupos/cecal/hpc/EAWSDD-2015.tar.gz.

Regarding the computational infrastructure, we consider scenarios with five cluster nodes, with up to 100 processors each. We take into account combinations of nowadays Intel processor with one to six cores, listed in Table 1.

4.2 Development and Execution Platform

Both proposed schedulers (higher- and lower-level) were implemented in the C programming language, using the `stdlib` library and the GNU gcc compiler.



(a) Series-Parallel (b) Heterogeneous-Parallel (c) Single-Task

Fig. 1. Workflow types used in the experimental evaluation of the proposed energy-aware hierarchical scheduler

Table 1. Characteristics of the processors considered for the CN infrastructures

Processor	*Frequency*	*Cores*	*GFLOPS*	E_{IDLE}	E_{MAX}	*GFLOPS/ core*
Intel Celeron 430	1.80 GHz	1	7.20	75.0 W	94.0 W	7.20
Intel Pentium E5300	2.60 GHz	2	20.80	68.0 W	109.0 W	10.40
Intel Core i7 870	2.93 GHz	4	46.88	76.0 W	214.0 W	11.72
Intel Core i5 661	3.33 GHz	2	26.64	74.0 W	131.0 W	13.32
Intel Core i7 980 XE	3.33 GHz	6	107.60	102.0 W	210.0 W	17.93

The experimental evaluation was performed on a Dell Power Edge server, Quad-core Xeon E5430 processor at 2.66 GHz, 8 GB RAM and Gigabit Ethernet, from the Cluster FING high performance computing facility (Universidad de la República, Uruguay, website http://www.fing.edu.uy/cluster) [16].

4.3 NGSA-II Parameter Configuration

We configured a number of 100 solutions for the NSGA-II population. The crossover operator is applied with a probability $p_c = 1.0$ and the mutation

operator with a probability of $p_m = 0.2$. Finally, for the stopping condition, we considered a fixed number of 20000 fitness function evaluations, which provides an adequate convergence behaviour for the population of solutions.

4.4 Results and Discussion

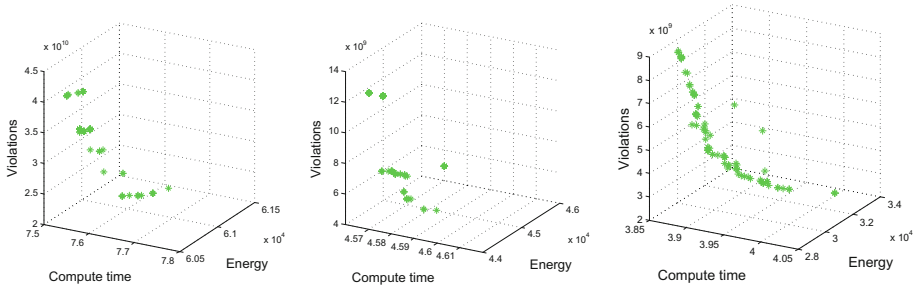
Table 2 reports the best, average, and standard deviation values for the makespan and energy consumption objectives, obtained in 25 executions of the proposed scheduler for different batches of each workflow type. We compare the MaxMIN-EFTH results with those computed by two schedulers combinations: MaxMIN-NOUR and RR-NOUR. This way, we study the capability of the proposed scheduler to improve the results in both (higher and lower) scheduling levels.

Table 2. Makespan and energy comparison for the studied schedulers

	MaxMIN-NOUR			**MaxMIN-EFTH**			**RR-NOUR**		
Series-Parallel workflows									
metric	f_M	f_E	f_P	f_M	f_E	f_P	f_M	f_E	f_P
avg.	8782.7	70998.9	1.11×10^8	**7658.0**	**62003.4**	8.73×10^7	8847.6	71351.8	$\mathbf{6.56\times10^7}$
σ	237.4	1881.0	0.97×10^7	203.6	1603.4	0.72×10^7	230.8	1833.4	0.71×10^7
best	8457.3	68393.3	9.29×10^7	7352.8	59680.2	7.38×10^7	8452.7	68291.2	5.19×10^7
Heterogeneous-Parallel workflows									
metric	f_M	f_E	f_P	f_M	f_E	f_P	f_M	f_E	f_P
avg.	5060.9	50305.1	3.45×10^7	**4616.7**	**45940.3**	2.91×10^7	5130.1	50774.8	$\mathbf{2.21\times10^7}$
σ	148.0	1412.2	0.41×10^7	124.2	1176.3	0.34×10^7	152.9	1519.7	0.25×10^7
best	4842.4	48300.2	2.54×10^7	4407.6	43966.7	2.20×10^7	4881.5	48296.4	1.63×10^7
Mixed workflows									
metric	f_M	f_E	f_P	f_M	f_E	f_P	f_M	f_E	f_P
avg.	3112.6	28722.0	9.71×10^6	**2961.6**	**28535.7**	8.68×10^6	3607.0	32855.9	$\mathbf{6.05\times10^6}$
σ	747.6	5160.5	5.73×10^6	601.8	3804.6	4.52×10^6	641.1	4240.1	3.50×10^6
best	2381.3	22998.0	5.35×10^6	2288.0	23458.6	4.89×10^6	2677.0	25633.9	3.09×10^6

The Kruskal-Wallis statistical test was applied to study the statistical confidence of the results, by analyzing the distributions of the results computed by each scheduler for each problem instance class. The best results for each metric and problem instance are marked in bold (gray background) in Tables 2 and 3 when the p-value computed in the correspondent pair-wise Kruskal-Wallis test is below 10^{-2} (meaning a statistical confidence of the results greater than 99 %).

The results in Table 2 demonstrate that the proposed MaxMIN-EFTH scheduler computes the best makespan and energy results for all problem classes. Overall, MaxMIN-EFTH computes the best makespan values in all **75** scheduling scenarios, and the best energy values in **58** out of 75. Although its accuracy regarding the makespan and energy objectives, the penalization is neglected by MaxMIN-EFTH. This is shown in Table 2 where the RR-NOUR baseline schedulers are able to compute the best penalization values for all the problem classes.



(a) Series-Parallel (b) Heterogeneous-Parallel (c) Mixed

Fig. 2. Example Pareto fronts computed by NSGA-II-EFTH when solving a Series-Parallel, a Heterogeneous-Parallel, and a Mixed problem instance.

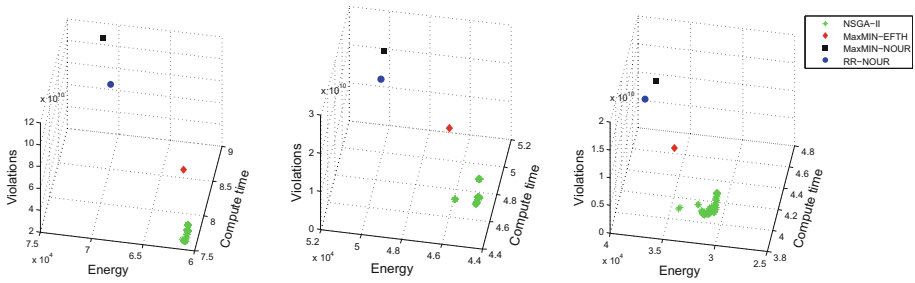
Next we evaluated the NSGA-II-EFTH algorithm considering a total of 30 independent executions for each problem instance. Figure 2 presents the Pareto front computed by a single NSGA-II-EFTH execution when solving a problem instance of each workload class.

To compare the schedules computed by NSGA-II-EFTH and RR-NOUR, we chose from each Pareto front computed by NSGA-II-EFTH the *compromise solution*, i.e. the *closest* to the one computed by RR-NOUR for each instance using a normalized Euclidean distance Table 3 presents the average improvements of the solutions computed by MaxMIN-NOUR, MaxMIN-EFTH, and the compromise solution computed by NSGA-II-EFTH, over the reference baseline schedulers for each workload class and objective function.

Table 3. Average makespan, energy consumption, and penalization improvements over RR-NOUR

workflow type	**MaxMIN-EFTH**			**NSGA-II-EFTH**		
	f_M	f_E	f_P	f_M	f_E	f_P
Series-Parallel	13.4%	13.1%	*-33.0%*	13.8%	13.7%	**36.4%**
Heterogeneous-Parallel	10.0%	9.5%	*-31.4%*	10.5%	**11.8%**	**34.2%**
Mixed	17.9%	13.1%	*-43.5%*	17.2%	**20.7%**	**19.3%**

The results demonstrate that MaxMIN-EFTH computes better schedules than RR-NOUR in terms of makespan and energy consumption, but RR-NOUR computes better penalization improvements than MaxMIN-EFTH. This is because MaxMIN considers makespan and energy consumption but not task's deadlines, while RR does not consider any objective but favors meeting deadlines by evenly distributing tasks among datacenters. NSGA-II-EFTH is able to compute more accurate schedules than MaxMin-EFTH for nearly all objetives and all problem instances, improving RR-NOUR schedules on all objetives. MaxMIN-EFTH computes competitive solutions when considering the makespan objective,



(a) Series-Parallel (b) Heterogeneous-Parallel (c) Mixed

Fig. 3. Example solutions computed when solving a Series-Parallel, a Heterogeneous-Parallel, and a Mixed problem instance.

but it is outperformed by NSGA-II-EFTH in the remaining objectives. NSGA-II-EFTH computes up to a **7.6 %** improvement in energy consumption, and up to **69.4 %** improvement in the penalization function over MaxMIN-EFTH. On the other hand, the execution time of NSGA-II-EFTH ranges from 12 h when solving problem instances of the Heterogeneous-Parallel and Mixed workload classes, up to 45 h when solving problem instances of the Series-Parallel workload class. These execution time requirements turn NSGA-II-EFTH unsuitable for tackling online scheduling problems.

Figure 3 graphically shows the solutions computed by RR-NOUR, MaxMIN-NOUR, MaxMin-EFTH, and a single NSGA-II-EFTH execution when solving a problem instance of each workload class.

5 Conclusions and Future Work

We introduced a multiobjective formulation of a two-level scheduling problem in datacenters using multi-core computers and considering makespan, energy consumption, and deadline violation penalization. The EFTH backfilling-oriented scheduler is used as a lower-level algorithm to schedule tasks locally within each cluster, while the MaxMIN heuristic and NSGA-II metaheuristic are both adapted to work with distributed datacenters and used as higher-level schedulers.

The experimental evaluation of the MaxMIN-EFTH and NSGA-II-EFTH schedulers compares the makespan, energy, and deadline violation penalization results against those computed by a traditional RR, and the MaxMIN heuristic both combined with a simple backfilling technique. The evaluation is performed over a set of 75 instances consisting of 1000 jobs each considering a total of 30 independent executions. From the experimental results, we conclude that MaxMIN-EFTH is able to obtain significant improvements in makespan and energy consumption objectives over the references baseline schedulers, but sacrificing accuracy in the deadline violation penalization objective. On the other hand, NSGA-II-EFTH obtains improvements in all three objectives while sacrificing efficiency by requiring execution times not suitable for online scheduling.

MaxMIN-EFTH is a promising shceduler for modern distributed datacenter infrastructures. Nevertheless, the results computed by NSGA-II-EFTH show

that the solutions computed by MaxMIN-EFTH could be greatly improved specially for the deadline violation penalization objective.

The main lines for future work are focused on improving the scheduling approach by studying different combinations of higher-level heuristics and lower-level backfilling schedulers.

References

1. Ahmad, I., Ranka, S.: Handbook of Energy-Aware and Green Computing. Chapman & Hall/CRC, Boca Raton (2012)
2. Bäck, T., Fogel, D., Michalewicz, Z.: Handbook of Evolutionary Computation. Oxford University Press, New York (1997)
3. Baskiyar, S., Abdel-Kader, R.: Energy aware DAG scheduling on heterogeneous systems. Cluster Comput. **13**, 373–383 (2010)
4. Coello, C., Van Veldhuizen, D., Lamont, G.: Evolutionary Algorithms for Solving Multi-objective Problems. Kluwer, New York (2002)
5. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms. Wiley, Chichester (2001)
6. Dorronsoro, B., Nesmachnow, S., Taheri, J., Zomaya, A., Talbi, E.G., Bouvry, P.: A hierarchical approach for energy-efficient scheduling of large workloads in multicore distributed systems. Sustain. Comput. Inf. Syst. **4**(4), 252–261 (2014)
7. Hirales-Carbajal, A., Tchernykh, A., Yahyapour, R., González-García, J., Röblitz, T., Ramírez-Alcaraz, J.: Multiple workflow scheduling strategies with user run time estimates on a grid. J. Grid Comput. **10**(2), 325–346 (2012)
8. Iturriaga, S., Nesmachnow, S., Dorronsoro, B., Bouvry, P.: Energy efficient scheduling in heterogeneous systems with a parallel multiobjective local search. Comput. Inf. J. **32**(2), 273–294 (2013)
9. Khan, S., Ahmad, I.: A cooperative game theoretical technique for joint optimization of energy consumption and response time in computational grids. IEEE Trans. Parallel Distrib. Syst. **20**, 346–360 (2009)
10. Kim, J.K., Siegel, H., Maciejewski, A., Eigenmann, R.: Dynamic resource management in energy constrained heterogeneous computing systems using voltage scaling. IEEE Trans. Parallel Distrib. Syst. **19**, 1445–1457 (2008)
11. Lee, Y., Zomaya, A.: Energy conscious scheduling for distributed computing systems under different operating conditions. IEEE Trans. Parallel Distrib. Syst. **22**, 1374–1381 (2011)
12. Li, Y., Liu, Y., Qian, D.: A heuristic energy-aware scheduling algorithm for heterogeneous clusters. In: Proceedings of the 15^{th} International Conference on Parallel and Distributed System, pp. 407–413 (2009)
13. Lindberg, P., Leingang, J., Lysaker, D., Khan, S., Li, J.: Comparison and analysis of eight scheduling heuristics for the optimization of energy consumption and makespan in large-scale distributed systems. J. Supercomputing **59**(1), 323–360 (2012)
14. Luo, P., Lü, K., Shi, Z.: A revisit of fast greedy heuristics for mapping a class of independent tasks onto heterogeneous computing systems. J. Parallel Distrib. Comput. **67**(6), 695–714 (2007)
15. Mezmaz, M., Melab, N., Kessaci, Y., Lee, Y., Talbi, E.G., Zomaya, A., Tuyttens, D.: A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems. J. Parallel Distrib. Comput. **71**, 1497–1508 (2011)

16. Nesmachnow, S.: Computación científica de alto desempeño en la Facultad de Ingeniería, Universidad de la República. Revista de la Asociación de Ingenieros del Uruguay 61, pp. 12–15 (2010). (text in Spanish)
17. Nesmachnow, S., Dorronsoro, B., Pecero, J.E., Bouvry, P.: Energy-aware scheduling on multicore heterogeneous grid computing systems. J. Grid Comput. **11**(4), 653–680 (2013)
18. Pecero, J., Bouvry, P., Fraire, H., Khan, S.: A multi-objective grasp algorithm for joint optimization of energy consumption and schedule length of precedence-constrained applications. In: International Conference on Cloud and Green Computing, pp. 1–8 (2011)
19. Pinel, F., Dorronsoro, B., Pecero, J., Bouvry, P., Khan, S.: A two-phase heuristic for the energy-efficient scheduling of independent tasks on computational grids. Cluster Comput. **16**(3), 421–433 (2013)
20. Quezada-Pina, A., Tchernykh, A., González-García, J.L., Hirales-Carbajal, A., Ramírez-Alcaraz, J.M., Schwiegelshohn, U., Yahyapour, R., Miranda-López, V.: Adaptive parallel job scheduling with resource admissible allocation on two-level hierarchical grids. Future Gener. Comput. Syst. **28**(7), 965–976 (2012)
21. Ramírez-Alcaraz, J., Tchernykh, A., Yahyapour, R., Schwiegelshohn, U., Quezada-Pina, A., González-García, J., Hirales-Carbajal, A.: Job allocation strategies with user run time estimates for online scheduling in hierarchical grids. J. Grid Comput. **9**(1), 95–116 (2011)
22. Rizvandi, N., Taheri, J., Zomaya, A.: Some observations on optimal frequency selection in DVFS-based energy consumption minimization. J. Parallel Distrib. Comput. **71**(8), 1154–1164 (2011)
23. Taheri, J., Zomaya, A., Khan, S.: Grid Simulation Tools for Job Scheduling and Datafile Replication in Scalable Computing and Communications: Theory and Practice. Wiley, Hoboken (2013). Chap. 35, pp. 777–797
24. Tchernykh, A., Lozano, L., Bouvry, P., Pecero, J., Schwiegelshohn, U., Nesmachnow, S.: Energy-aware online scheduling: ensuring quality of service for iaas clouds. In: Proceedings of the International Conference on High Performance Computing Simulation, pp. 911–918 (2014)
25. Tchernykh, A., Lozano, L., Schwiegelshohn, U., Bouvry, P., Pecero, J., Nesmachnow, S.: Bi-objective online scheduling with quality of service for iaas clouds. In: Proceedings of the 3rd International Conference on Cloud Networking, pp. 307–312 (2014)
26. Tchernykh, A., Pecero, J.E., Barrondo, A., Schaeffer, E.: Adaptive energy efficient scheduling in peer-to-peer desktop grids. Future Gener. Comput. Syst. **36**, 209–220 (2014)
27. Topcuouglu, H., Hariri, S., Wu, M.Y.: Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Trans. Parallel Distrib. Syst. **13**(3), 260–274 (2002)
28. Valentini, G., Lassonde, W., Khan, S., Min-Allah, N., Madani, S., Li, J., Zhang, L., Wang, L., Ghani, N., Kolodziej, J., Li, H., Zomaya, A., Xu, C.Z., Balaji, P., Vishnu, A., Pinel, F., Pecero, J., Kliazovich, D., Bouvry, P.: An overview of energy efficiency techniques in cluster computing systems. Cluster Comput. **16**(1), 3–15 (2013)
29. Zomaya, A., Khan, S.: Handbook on Data Centers. Springer, New York (2014)
30. Zomaya, A.Y., Lee, Y.C.: Energy Efficient Distributed Computing Systems. Wiley-IEEE Computer Society Press, New York (2012)