Computing and Informatics, Vol. 30, 2011, 1201–1223

A MULTI-CRITERIA META-FUZZY-SCHEDULER FOR INDEPENDENT TASKS IN GRID COMPUTING

Antonio Javier Sanchez Santiago, Antonio Jesus Yuste
Jose Enrique Muñoz Exposito, Sebastian García Galán
Rocio Perez de Prado

Telecommunication Engineering Department
Alfonso X el Sabio, 28
Linares, SPAIN
e-mail: {ajsantia, ajyuste, jemunoz, sgalan, rperez}@ujaen.es

Communicated by Bogdan Wiszniewski

Abstract. The paradigm of distributed computation in heterogeneous resources, grid computing, has given rise to a large amount of research on resource scheduling. This paper presents a Meta-Scheduler for grid computing that does not need any given information about tasks length or tasks arrival time unlike traditional dynamic heuristics. Our Meta-Scheduler is of multi-criteria type, because it solves two conflicting objectives: minimize the makespan of a set of tasks and distribute these tasks in a balanced way among the resources of the Grid. Experimental results using fuzzy scheduler show that, through our proposal, we achieve these two objectives and improve dynamic heuristics presented in prior literature.

Keywords: Grid computing, scheduling, multi-criteria, fuzzy logic, makespan, workload balancing

Mathematics Subject Classification 2000: 68T99, 68W15

1 INTRODUCTION

A Grid System (Figure 1) is a large scale, heterogeneous collection of resources, geographically distributed and interconnected by high speed networks [1]. Each set of heterogeneous computing resources are interconnected through a Grid Node

(GN). Each GN has its own access policy, cost, design and faces particular constrains according to the particular organizational setting each GN belongs to. Therefore, a Grid System is a set of shared heterogeneous resources that can be used by different organizations.

Grid computing entails some new challenges. One of these challenges consists of adaptation of parallel programs developed for homogeneous resources to dynamic and heterogeneous grid resources with minimal interference with resource scheduling, or load balancing. Grids have a series of specific features, like heterogeneity, autonomy, scalability and adaptability, which make the load balancing problem even more difficult. A load balancing algorithm attempts to improve the response time of the tasks by ensuring maximal utilization of available resources.

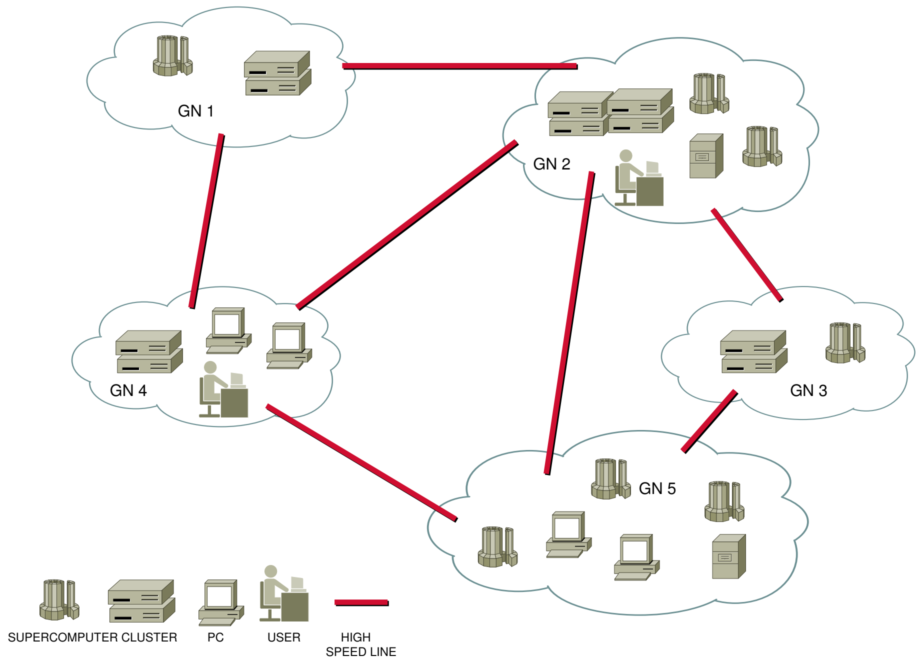


Fig. 1. A Grid system with five GNs

In a Grid environment, users essentially interact with the Grid Resource Center (GRC) that provides the functionality for reservation, discovery and publishing of resources [2]. Scheduling, submission and monitoring of tasks on the resources is managed by the Meta-Scheduler, located in the GRC. The Meta-Scheduler makes decisions on resources use and execution orders through minimizing (or maximizing) an objective function, like global time of execution of a set of tasks (makespan) or workload distribution, to optimize the use of the resources. A Grid system can be considered as a dynamic and heterogeneous distributed system. Here, by dynamic, as opposed to static, the Meta-Scheduler must adapt the scheduling strategy according to the changes in the Grid System.

In this work, our aim, focused on computational grid [3], consists of developing a Meta-Fuzzy-Scheduler (MFS) that using fuzzy logic would be able to adapt to the changes in the system so that it can make decisions on when executing a set of given tasks through the Grid System. To do so, two confronted objectives will be faced; firstly, to obtain a low makespan; secondly, a better distribution of tasks across the GNs that form the Grid System. Therefore, our MFS can be considered as a multi-criteria scheduler [4].

The MFS is developed to be applied in a Grid that uses the idle resources of a Small and Medium Sized Enterprise (SME). Under these circumstances, it is important to obtain a lower makespan and make a good load balancing distribution among resources, thus not overloading the resources from the same Grid Node. In order to avoid this effect, our fuzzy logic controller contains a set of inference rules that prevent overloading of the Grid Nodes that have executed several tasks.

Analyzing the new MFS requires a large number of tests involving many resources of the SME. We will also provide a solution to this problem since our tests are based on simulation techniques which do not need "real resources". To validate this proposal, our MFS is developed using GridSim [5] and we will take into account several scenarios, by changing key parameters of the Grid system. Experimental results show that, through the use of our Meta-scheduler, we are able to improve traditional dynamic heuristics.

This document is organized as follows: In Section 2 we analyze heuristics in grid resource scheduling. In Section 3 we describe the proposed scheduling model. In Section 4 we show the simulation results with GridSim. Finally, in Section 5 we present some conclusions and future directions.

2 BACKGROUND

Task scheduling is one of the most difficult works in a computational Grid environment and is the key technology in Grid resource allocation [6].

In this section we carry out a review of the most relevant traditional heuristics for Grid computing. Firstly, tasks scheduling can be divided into static and dynamic approach [7]. Static algorithms assume *a priori* information about all of the characteristics of the computing nodes and scheduling decisions are made prior to application execution [8], while dynamic algorithms study the grid state before mapping a task and scheduling decisions are made during application execution [9].

Bearing this classification in mind, it is possible to take into account two kinds of task-scheduling problems [10], namely the on-line mode and the batch mode:

- In the on-line mode, tasks are assigned to resources immediately upon their arrival [11]. When the arrival rate is low, the on-line mode obtains a better performance because tasks do not need to wait to the next mapping event [12].

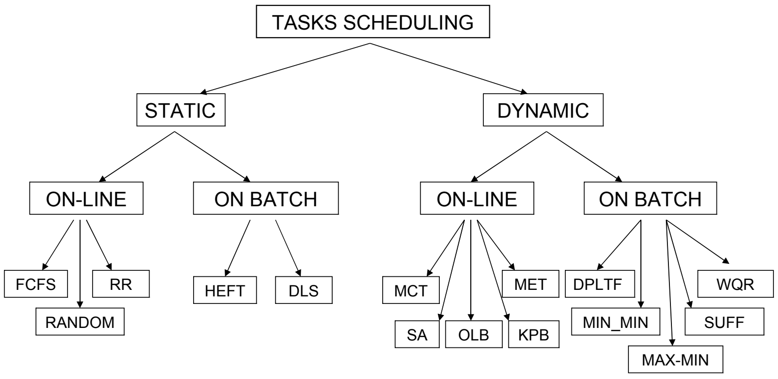


Fig. 2. Classification of tasks scheduling heuristics

- In the batch-mode, when tasks arrive to the Meta-Scheduler, they are not mapped onto a resource, but are collected in a Bag-of-Tasks (BoT) [13] and they are mapped at next mapping event. These mapping events start on a given time or when the Bag-of-Tasks is full. The batch-mode obtains better results when the arrival rate is high because there is a high number of tasks being executed in the resources in order to keep them busy.

While on-line mode heuristics considers a task for mapping only once, batch mode heuristics considers a task for mapping at each mapping event until the task begins execution. For online mode, there is no mapping delay between mapping events, the tasks are mapped right after they arrive, but their performance is not as good as in the batch-mode when the arrival of tasks is very intensive. For the batch mode there is higher throughput and utilization, but their real time is relatively worse.

According to prior literature, there are many traditional algorithms for both modes. In the on-line mode there are five traditional algorithms. Minimum Completion Time (MCT) [14] assigns a task to the resource that offers a lower completion time; Minimum Execution Time (MET) [15] assigns a task to the resource available that offers lower total execution time; Switching Algorithm (SA) [16] uses MCT and MET in a cyclic way depending on the resources workload; Oportunistic Load Balancing (OLB) [17] maps the task to the resource that becomes ready next. To do so, the algorithm examines all resources to find the resource that becomes ready next. Finally, K-Percent Best (KPB) [18] considers for each task a subset of resources in order to map the task. This subset is made up by the best resources according to the earliest completion time.

In the batch-mode, there is a group of three dynamic heuristics for Grid computing that are focused on minimizing the overall makespan of a Bag-of-Tasks [19]. These three algorithms are very similar and are based on the calculation of MCT [14].

These heuristics need a Bag-of-Tasks, for each task of the bag MCT is calculated and, according to the algorithm, a task to be executed is selected. The newly mapped task is removed from the bag, and the process is repeated until all tasks are mapped. Min-Min algorithm [20] gives priority to the tasks that can be completed earlier, so tasks with lower MCT are executed first. The percentage of tasks assigned to their first choice is likely to be higher for Min-Min than for Max-Min (defined next). A smaller makespan can be obtained if more tasks are assigned to the machines that complete them the earliest and also execute them the fastest. Max-Min algorithm [21] selects the task with high overall MCT from the bag and the task is assigned to the corresponding machine. Max-Min attempts to minimize the penalties incurred from performing tasks with longer execution times. Mapping the task with the longest execution time to its best machine first allows this task to be executed concurrently with the remaining tasks (with shorter execution times). For this case, this would be a better mapping than a Min-Min mapping, where all of the shorter tasks would be executed first, and then the longer running task would be executed while several machines sit idle. Thus, in similar cases to the one of this example, the Max-Min heuristics may give a mapping with a more balanced load across machines and a better makespan. In Sufferage algorithm [20], tasks with high sufferage value take precedence. For each task, the sufferage value is defined as the difference between the best MCT and the second MCT for a task.

There are papers that describe various extensions of these heuristics. In [22] a simple alternative of the Min-Min algorithm by scheduling large tasks first was proposed. The proposed algorithm retains the advantage of the Min-Min algorithm and achieves good load balance at the same time. In task scheduling new challenges appear, based on properties such as security or quality of service (QoS). A novel algorithm based on a general adaptive scheduling heuristics that includes QoS guidance was developed in [23]. In Xsufferage [24] an improvement to Sufferage heuristic was created. The sufferage value is computed not with MCTs, but with cluster-level MCTs, i.e. by computing the minimum MCTs over all hosts in each cluster.

Task scheduling problem is known to be NP-complete [25]. Therefore the use of meta-heuristics of soft computing copes with this difficulty. There are some works in prior literature that use single heuristics. In [26, 27, 28] tabu search (TS) is used to find optimal solutions to schedule tasks. [29, 30] use simulated annealing to obtain a satisfactory high-performance scheduling. Some works use ant colony [31, 32, 33] for job scheduling in the Grid environment. Some hybrid heuristic approaches have also been reported for this problem. In [34] an ant colony optimization algorithm with a TS algorithm was combined. Other approaches use genetic algorithms to develop scheduling strategies for jobs in Grid systems [35, 36]. Particle swarm optimization is also used for scheduling in grids [37, 38].

Some of these approaches use meta heuristics of soft computing to sort the Bag-of-Tasks [31, 39]. These same works use a simple scheduling algorithm FCFS [40] or EASY Blackfilling [41] in order to assign tasks to a resource. In this work, we classify the Bag-of-Tasks in a simple way, because our goal does not rely on how to classify the bag, but in identifying, for each task, the best resource that minimizes the makespan and obtains a better workload balancing distribution.

Min-Min, Max-Min and Sufferage are the heuristics commonly used for comparison in Grid computing to execute tasks in batch mode. Therefore, they are ideal to be compared to our Meta-Scheduler (Min-Min; Max-Min; Sufferage and Segmented Min-Min) because these heuristics use the MCT, which aims to minimize the makespan. These algorithms need a prediction of the length of the tasks. However, our Meta-Scheduler does not need this prediction.

3 PROPOSED GRID SCHEDULING MODEL

Local scheduling is a typical problem of mapping tasks to local CPUs in homogeneous environments; however Meta-Scheduling regards mapping (i.e., discovery, matching and selection) higher-level resources to tasks. Scheduling onto a Grid has three main phases [42]. Phase one is resource discovery [43], which generates a list of potential resources. Phase two involves the publication of information about those resources and choosing the best to send a task. In phase three the job is executed. These phases are showed in Figure 3.

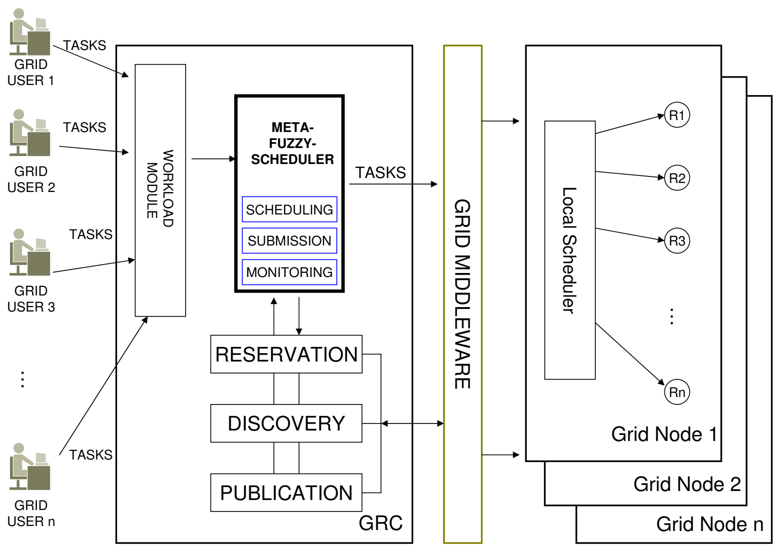


Fig. 3. Grid system with scheduling in two levels

The GRC is responsible for phase one, discovering resources, its reservation and publication, hiding to the user the complexity of a Grid System. In phase two the Meta-Scheduler submits tasks onto Grid Nodes registered in GRC using scheduling strategies, and interacts with Local Schedulers (LS) for task deployment on resources, and with other components of GRC, such as monitoring.

Our Grid System Model is made up by a set of Grid Nodes; each GN is made up by of several computational heterogeneous nodes or resources. The MFS does not have any explicit information about the resources, therefore does not have any control over them, and only interacts with the Local Scheduler from each GN. This interaction is provided for the GRC (Figure 3) which is based on a two-level scheduler [44], a Meta-Scheduler and a Local Scheduler for each GN.

To formulate the problem, we consider T_j as a set of tasks from independent users to be assigned to G_i heterogeneous GN for a dual purpose: minimizing the makespan and using the Grid Nodes effectively to balance resource utilization. These objectives are conflicting, because to obtain a better load balancing it is recommended to use resources of nodes that have a minor computing power, therefore this implies worse makespan for a Bag-of-Tasks.

$$T_j = (j \in \{1, 2, 3, \ldots, m\}) \tag{1}$$
$$G_i = (i \in \{1, 2, 3, \ldots, n\}) \tag{2}$$

The first objective to improve is the makespan of a Bag-of-Tasks. The makespan of T is defined as the time from the start of the first task to the end of the last task. Equation (3) shows how to evaluate the makespan of T, where M is the makespan, $Time_{s1}$ is the time of start for the first task and $Time_{\text{end}}$ is the end of the time of the task T_m.

$$M = \max(\text{Time}_{\text{end}} - \text{Time}_{s1}) \tag{3}$$

Load balancing has been defined in prior literature as the distribution of tasks which is able to use as many resources being available to maximize throughput and minimize response time, maximizing the use of all available resources. In a Grid system that uses this kind of load balancing, the GN with more and better computing resources will be responsible to execute many tasks from a Bag-of-Tasks, leaving other GNs unused.

Our scheduler is designed to be implemented in a real Grid system, that uses idle resources. The use of these resources is shared between the owners and the Grid system, so it is not advisable to use always the same resources for executing tasks. Therefore it is relevant to introduce a new concept of load balancing on Grid nodes, that gives option to the worst nodes to be used for tasks execution. Hence a balanced task execution is obtained, with the participation of all nodes of the Grid. With this new definition of load balancing, Grid Nodes with more resources and more computing power execute less tasks than when using common vision of load balancing, so they are not overloaded. This approach ensures that Nodes with less computing power are also used to execute tasks.

The second objective, Workload Balancing (WLB), is defined in Equation (4). With this equation it is checked that Grid Nodes execute tasks in a balancing way. Thus, some resources of the best node remain available. WLB is evaluated for each

Grid Node when the last task in the Bag-of-Tasks has been executed. Values near to zero are optimal.

$$\mathrm{WLB} = \frac{\mathrm{ResourcesNode}}{\mathrm{TotalResources}} - \left|\frac{\mathrm{NodeLoad}}{\mathrm{TotalLoad}} - \frac{1}{n}\right| \tag{4}$$

Several parameters, related to a Grid Node, are used to calculate this objective:

- The number of resources of a node (ResourcesNode) with respect to total resources (TotalResources) of the Grid System.
- The number of tasks executed in this node (NodeLoad) with respect to the total of tasks executed (TotalLoad) on the Grid System.
- The number of Grid Nodes (n) of the Grid System.

To execute a Bag-of-Tasks, some information of the current state of each Grid Node is needed. With this information given by each LS to the GRC, our MFS decides, which GN the task should be sent to in order to accomplish two objectives: to minimize the makespan and to compute a WLB value close to zero. In the GN selected, the LS sends the task to the best resource free at any given moment. The structure of the proposed Meta Fuzzy Scheduler is shown in Figure 4.

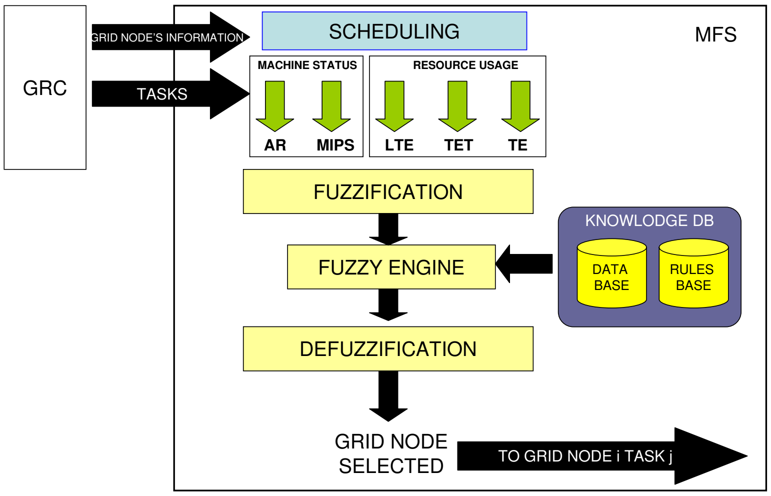


Fig. 4. Structure of the Meta-Fuzzy-Scheduler

Grid Nodes state may change for various reasons at any time. Before the scheduler submits a new task to the Grid System, our MFS checks the current status of all nodes in the Grid (Figure 4). According to this status each node has a Factor of Selection (FS). FS of a node is obtained with five measures concerning the current and historical status of the node. These five parameters rate between 0 and 1 and they are related to the status of a node. They are defined into two groups. Parameters related to the current computing state of the node are defined and grouped as Machine Status. Resource Usage groups parameters related with tasks are executed by this node for a Bag-of-Task.

Machine Status parameters are highly related with the first objective (makespan) since they symbolize the computing power of a Grid Node. A node with high computing capacity power can finish tasks in a shorter time. Two parameters are defined in this group:

- Rate of avaliable resources (AR): a resource avaliable in a Grid Node is a computing element (cluster, server, PC, ...) with sufficient computing power to execute the next task in the Bag-of-Task. This input shows the number of free avaliable resources in a Grid Node. The normalization equation for this input is

  $$FR(i) = \frac{\text{FreeResources}(i)}{\text{TotalResources}(i)} \tag{5}$$

  where i is a Grid Node of the Grid System.
- Rate of idle MIPS (MIPS): in a Grid Node, each available resource has a computing power for executing tasks. This input indicates the computing power for idle resources of a Grid Node. The normalization equation for this input is

  $$\text{MIPS}(i) = \frac{\text{IdleMips}(i)}{\text{TotalMips}(i)} \tag{6}$$

  where i is a Grid Node of the Grid System.

Parameters grouped in Resource Usage are related with the second objective WLB. These parameters show historical information of tasks previously executed in this Grid Node for the current Bag-of-Task. These parameters allow to distribute tasks on nodes with less number of tasks executed for the current Bag-of-Tasks. Thus, Grid Nodes with more computing power are not overloaded with new tasks to execute. Two parameters with information related with the tasks executed are used to check if a node is overloaded: length of the tasks executed and the utilization time of the Node. Also, the number of tasks executed by a node during the execution of a Bag-of-Tasks is counted:

- Length of tasks executed (LTE): in the Bag-of-Tasks each task has different length, so this parameter is necessary to study the historical workload state of a Grid Node. This input shows the length of tasks executed in a Grid Node, in relation to all tasks that have been executed of the current Bag-of-Tasks in all nodes. The normalization equation for this input is

  $$\text{LTE}(i) = \frac{\text{LengthTaskExecuted}(i)}{\sum_{i=1}^{n} \text{TotalLengthTaskExecuted}} \tag{7}$$

  where i is a Grid Node and n is the number of Grid Nodes of the Grid.

- Tasks executing time (TET): each node is different and heterogeneous, so the execution time of a task varies from each node. This parameter is related to the total time spent by a node to execute tasks assigned in current Bag-of-Tasks or to the total time spent by all tasks executed in all nodes. The normalization equation for this input is

$$\mathrm{TET}(i) = \frac{\mathrm{TimeSpent}(i)}{\sum_{i=1}^{n} \mathrm{TotalTimeSpentAllNodes}} \tag{8}$$

  where i is a Grid Node and n is the number of Grid Nodes of the Grid.

- Tasks executed (TE): the number of tasks executed and completed for a Grid Node for the current Bag-of-Tasks, in relation with the total tasks executed by all the nodes that form the Grid System. The normalization equation for this input is

$$\mathrm{TE}(i) = \frac{\mathrm{Count}(\mathrm{TasksExecuted}(i))}{\sum_{i=1}^{n} \mathrm{Count}(\mathrm{TotalTasksExecuted})} \tag{9}$$

  where i is a Grid Node and n is the number of Grid Nodes of the Grid.

For these five inputs the degree of membership for each fuzzy set is calculated. Membership functions are defined by a triangle-shaped fuzzy set since they have been extensively used in real time applications due to their simple formulas and computational efficiency. The membership equation is shown in Equation (10), with values in the interval [0, 1]. The output fuzzy variable of the system is defined as a Factor of Selection of a GN. The membership function of the inputs and the output fuzzy variable are defined in Figure 5.

$$\mu(x) = \begin{cases} 0, & x \le a \\ \frac{(x-a)}{(m-a)}, & x \in (a, m] \\ \frac{(b-x)}{(b-m)}, & x \in (m, b] \\ 0, & x \ge b \end{cases} \tag{10}$$

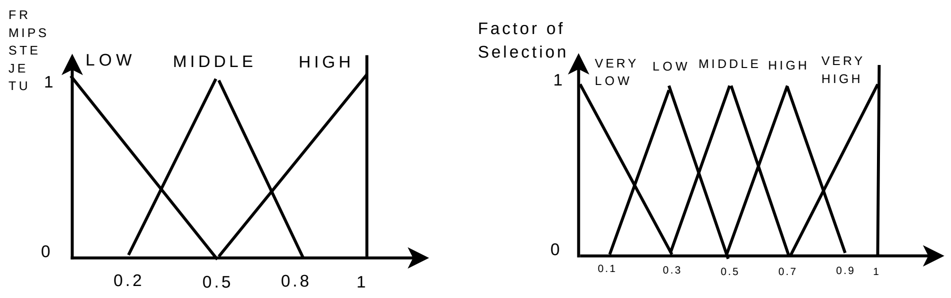


Fig. 5. Membership functions for the input and the output

Fuzzy inference rules, Mamdani type [45], are constructed using the input and output variables that have been defined previously. These rules have the same weight and all rules use the connector AND. According to our experience in Grid computing, we develop some rules for fuzzy controller, which consider both objectives showed previously. Two samples of these rules are:

Rule 1: if **FR** is HIGH and **MIPS** is HIGH and **LTE** is HIGH and **TET** is HIGH and **TE** is HIGH then **FS** is MIDDLE

Rule 2: if **FR** is HIGH and **MIPS** is HIGH and **LTE** is LOW and **TET** is LOW and **TE** is LOW then **FS** is VERYHIGH.

In rule 1 the membership functions for the five inputs are *HIGH*. Using a Meta Scheduler with common load balancing, Factor of Selection for this rule will be *VERYHIGH*, because this node has a high computing power to execute tasks. However, with our proposal of WLB, Factor of Selection for this rule is *MIDDLE*. This Grid Node has executed a large number of tasks, so it is advised to choose another Grid Node to execute the task.

Although this node has a high computational power for executing tasks, it executed a large number of length tasks already and much time has been spent for the current Bag-of-Tasks. If this node is selected to execute the task, the WLB of the node would worsen, because now this node has a high value for resource usage inputs. Using our approach of WLB, as the three inputs related with tasks (LTE, TET, TE) are *HIGH*, so this Grid Node should not receive more tasks to execute.

This is otherwise in rule two. In this rule, the Grid Node has the membership functions for resource usage *LOW* and for Machine Status the membership functions are *HIGH*. With these values a Grid Node is excellent to receive and execute tasks, thus this node does not worsen its WLB and it is able to execute a task in minimum time.

There are many methods to defuzzificate and obtain the output: centroid, bisector, middle of maximum, largest of maximum, etc. In this work, we use the most popular defuzzification method which is the centroid method. This method returns the centre of area under the curve.

4 EXPERIMENTAL RESULTS

In this work, an approach is proposed to achieve a lower makespan and minimal workload balancing. In this study simulation tasks are collected in a Bag-of-Tasks. We consider scheduling Bag-of-Tasks applications, which are those parallel applications whose tasks are independent of one another. BoT applications are used in a variety of scenarios, including parameter sweeps, computational biology [46], computer imaging [47] and [48], data mining, fractal calculations and simulations. Furthermore, because of the independence of tasks, BoT applications can be successfully executed over geographically distributed computational grids, as demonstrated by SETI@home [49].

In this work, the size of the Bag-of-Tasks ranges between 500–1 500 tasks. The Meta-Scheduler dispatches the tasks once. To simplify the scenario for analysis, this study only simulated cases with a fixed number of Grid Nodes in five nodes and 240 resources.

The following presumptions are detailed in this study: the tasks of a job have been previously divided by a logic unit and all the tasks are independent from each other and parameters passing between tasks are not necessary. However, our MFS can be used in scenarios with task dependencies, e.g., workflows,.

In this study, proposed simulation model is derived from the scenarios proposed in [14]. Task heterogeneities are categorized into two types:

1. high, ranging at $[1, 300\,000]$ and
2. low, ranging at $[1, 10\,000]$.

Resource heterogeneities are also classified into two types:

1. high, ranging at $[1, 10\,000]$ and
2. low, ranging at $[1, 1\,000]$.

Therefore, the four scenarios are showed in Table 1.

Heterogeneity	Task	Resource
HH	High $[1, 300\,000]$	High $[1, 10\,000]$
HL	High $[1, 300\,000]$	Low $[1, 1\,000]$
LH	Low $[1, 10\,000]$	High $[1, 10\,000]$
LL	Low $[1, 10\,000]$	Low $[1, 1\,000]$

Table 1. Four scenarios in the simulation model

As an example, when both tasks and resources heterogeneities are high, the length of tasks belongs to $[1, 300\,000]$ Millions Instruction (MI) and Million Instructions per Second (MIPS) of resources of a GN belongs to $[1, 10\,000]$ MIPS. According to the Central Limit Theorem in statistics, when the number of samples exceeds or is equal to 30, the diagram for sample dispatching would seem very similar to that of normal distribution. Therefore, in this experiment each scheduling heuristics includes four scenarios, each of which has 30 different samples for each size of the Bag-of-Tasks.

Due to the difficulties associated to tests in real settings, the benefits of this Meta-Scheduler have been verified through the use of simulations. Therefore, it was necessary to develop our Meta-Scheduler using [5]. GridSim is a simulation toolkit widely used for resource modeling and application scheduling in parallel and distributed computing systems.

Our approach is compared with three dynamic heuristics on batch mode for Grid computing (Max-Min, Min-Min and Sufferage) and one extension of Min-Min:

Segmented Min-Min, with sub-policy 1 (Average). Also a modification of our proposal is compared. This modification is a simple pre-processing of the BoT. The Bag-of-Tasks is sorted by length of the tasks from highest to lowest, so tasks with greater computing power requirement are executed first. In MFS with no Pre-processing, tasks are mapped following the order obtained by the random uniform distribution.

The Grid system for these simulations is made up by a set of Grid Nodes, each GN having the following parameters: resources range between 6 and 144; each resource has one processor and bandwidth between GN's is 2.5 Gb/s.

The simulation parameters are applied to each heuristic to obtain the experimental results. In addition, the properties of task and resource are categorized by the degree of heterogeneity into four scenarios:

- Task and resources are characterized by high heterogeneity (HH).
- Task is high heterogeneity and resource is low heterogeneity (HL).
- Task is low heterogeneity and resource is high heterogeneity (LH).
- Task and resources are characterized by low heterogeneity (LL).

In order to minimize makespan it is ideal to use the best nodes available, thus tasks will be finished in a minor time. To minimize WLB objective it is needed that all available nodes in the Grid system are used, including those with low computing power. If nodes with low computing power are included to execute a Bag-of-Tasks, makespan will be penalized. Then makespan and Workload Balancing become contradictory objectives

In the following tables, the performance measurements of dynamic heuristics and MFS are defined by two metrics: makespan and WLB. We addressed ourselves to observing how well these algorithms performed in terms of such performance metrics under different scenarios.

4.1 High-High Scenario

In the experiments of this section, the performance of our algorithm is observed under the High(Tasks)-High(Resources) scenario of the Grid system.

Figure 6 shows makespan for the HH scenario. According to the results, for a less combination of size of the BoT (500–600 tasks) our approach obtains a similar time as dynamic heuristics. Increasing the size of the BoT (700–1 500 tasks) our Meta-Scheduler performs better than dynamic heuristics. Using simple pre-processing, it obtains an improvement over no pre-processing of some 2 % in makespan. Max-Min achieves a better makespan for dynamic heuristics. Improvement between Min-Min and Segmented Min-Min is 1 % for the Min-Min extension.

From Table 2 values for the second objective in a HH scenario can be observed. For our MFS, the value of WLB is close to optimal. With no pre-processing our MFS obtains a better WLB. High sizes of BoT for both MFS penalize WLB. Dynamic heuristics get worse results and they do not make a good WLB through Grid system

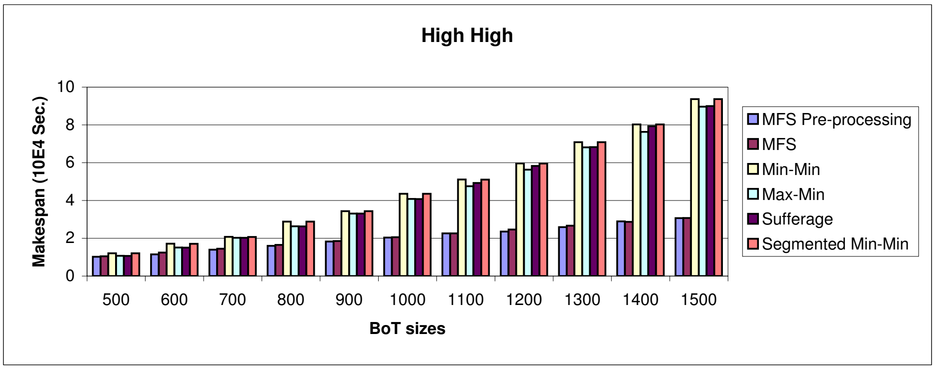


Fig. 6. Makespan for High-High scenario

nodes. Segmented Min-Min obtains the best WLB for dynamic heuristics, but this value is far from the optimum.

	Meta-Schedulers					
BoT	MFS Pre-Pro	MFS	Max-Min	Min-Min	Suff	Seg Min-Min
500	0.0091	0.007	0.2671	0.2626	0.2626	0.2471
600	0.0089	0.0071	0.26	0.26	0.26	0.251
700	0.0074	0.0065	0.2646	0.2646	0.2646	0.2246
800	0.0073	0.0061	0.2618	0.2618	0.2618	0.2515
900	0.0073	0.0064	0.2602	0.2602	0.2602	0.2236
1 000	0.0071	0.0062	0.2634	0.2634	0.2634	0.2543
1 100	0.0125	0.0113	0.2641	0.2641	0.2641	0.2569
1 200	0.0115	0.0108	0.2609	0.2609	0.2609	0.2496
1 300	0.0114	0.0108	0.2657	0.2657	0.2657	0.2531
1 400	0.012	0.0111	0.2593	0.2593	0.2593	0.2427
1 500	0.0121	0.0111	0.2637	0.2637	0.2637	0.2584

Table 2. Workload balancing for High-High scenario

In this scenario (HH), for any size of the BoT, our algorithm always achieves a better time for makespan and a lower WLB regarding the results obtained by traditional heuristics.

4.2 High-Low Scenario

In this section the performance of the Meta-Schedulers is showed for a high tasks heterogeneity and a low resources heterogeneity. In this scenario the maximum value of computing power for a resource is 1 000 MIPS, so makespan is increased with regard to the above scenario.

For small BoT sizes, makespan is parallel in all policies. With a greater size of BoT, the difference between our proposal and dynamic heuristics increases. For this scenario, when the size of BoT is increased our algorithm achieves a low makespan than the dynamic heuristics. Makespan for HL scenario is showed in Figure 7. With pre-processing, an improvement over the no pre-processing of 1.5 % is obtained. Sufferage gets the third best time for makespan.

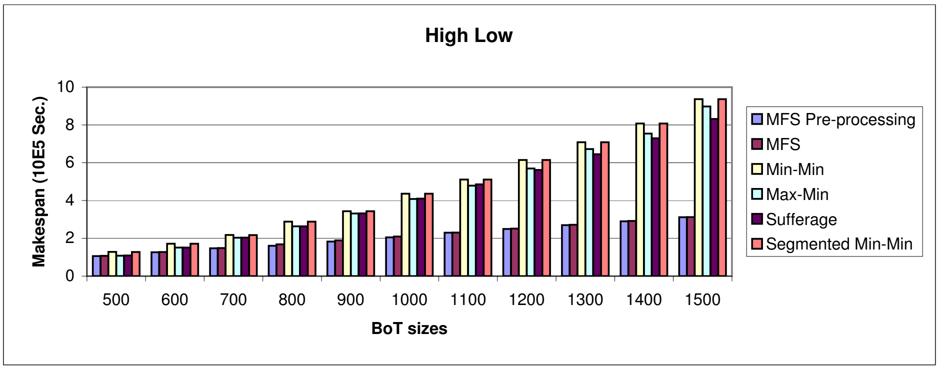


Fig. 7. Makespan for High-Low scenario

Table 3 illustrates WLB for HL scenario. In this scenario, the performance for MFS is worse than in the previous scenario. Thus with the combination of large tasks and small capacity of computing resources, a resource spent more time to execute a task, overloading the nodes. Despite this situation, our approach gets a lower WLB than dynamic heuristics. MFS penalizes WLB with high heterogeneities for tasks and low heterogeneities for resources.

	Meta-Schedulers					
BoT	MFS Pre-Pro	MFS	Max-Min	Min-Min	Suff	Seg Min-Min
500	0.035	0.0274	0.2746	0.256	0.2613	0.2512
600	0.0325	0.0272	0.2556	0.2556	0.2625	0.2493
700	0.0317	0.0254	0.2658	0.2618	0.2713	0.2584
800	0.0312	0.0256	0.2597	0.2636	0.2577	0.2515
900	0.0307	0.0251	0.2608	0.2687	0.2626	0.2594
1 000	0.0301	0.0257	0.2643	0.2556	0.26	0.2534
1 100	0.0305	0.0255	0.2612	0.2659	0.2646	0.2558
1 200	0.0306	0.0247	0.2638	0.2597	0.2618	0.2562
1 300	0.0311	0.0256	0.264	0.2608	0.2602	0.2578
1 400	0.0309	0.0252	0.2625	0.2643	0.2634	0.2523
1 500	0.0308	0.025	0.2601	0.2632	0.2623	0.2565

Table 3. Workload balancing for High-Low scenario

4.3 Low-High Scenario

Results for the scenario which low tasks heterogeneity and high resources heterogeneity are present in this subsection. In this scenario the lowest makespan is obtained for all scenarios.

For this scenario, our proposal obtains a lower makespan for all sizes of the Bag-of-Tasks. Figure 8 shows that our proposed algorithm performs better makespan than dynamic heuristics. The gap for a low size of the BoT is low, but using a BoT with more than 700 tasks the gap is increased. Dynamic heuristics obtain a similar time in all experiments of this scenario. The gap between pre-processing and no pre-processing is 2 %.

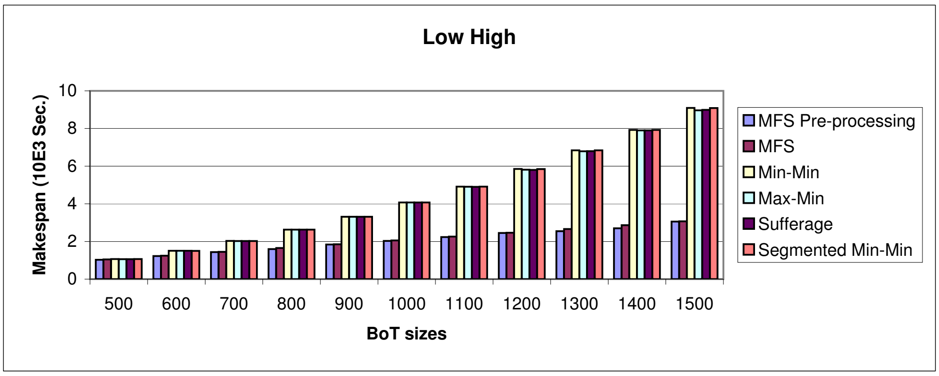


Fig. 8. Makespan for Low-High scenario

Once again, our proposed Meta-Scheduler enjoys better performance in terms of WLB as compared with its counterparts (Table 4). For this scenario, our algorithm obtains a Workload Balance value close to zero. The WLB is better with a smaller number of tasks of the BoT. Ideal length for the BoT in this scenario ranges between 500 and 1 000 tasks.

4.4 Low-Low Scenario

In the next set of results makespan and WLB for a Low (Tasks)-Low (Resources) scenario are shown. Figure 9 shows the makespan for LL scenario.

For a low size of the BoT (500 tasks) our approach obtains a similar time as dynamic heuristics. With a greater size of BoT, it can be observed that the gap between our proposal and dynamic heuristics increases. Makespans for traditional heuristics are parallel for any size of the BoT. For this scenario the gap between MFS with pre-preprocessing and MFS with no pre-processing is 1 %.

Table 5 illustrates WLB for the LL scenario. In terms of WLB, MFS penalizes large sizes for BoT. Dynamic heuristics obtain a bad performance for this scenario.

	Meta-Schedulers					
BoT	MFS Pre-Pro	MFS	Max-Min	Min-Min	Suff	Seg Min-Min
500	0.0077	0.0073	0.2626	0.2626	0.2626	0.2545
600	0.0074	0.0067	0.26	0.26	0.26	0.2507
700	0.0072	0.0065	0.2646	0.2646	0.2646	0.2568
800	0.007	0.0063	0.2618	0.2618	0.2618	0.2496
900	0.007	0.0063	0.2602	0.2602	0.2602	0.2475
1 000	0.0069	0.0061	0.2634	0.2634	0.2634	0.2567
1 100	0.0119	0.0112	0.2641	0.2641	0.2641	0.2525
1 200	0.0121	0.0113	0.2613	0.2613	0.2613	0.2563
1 300	0.0113	0.0106	0.2657	0.2657	0.2657	0.2529
1 400	0.0116	0.0107	0.2593	0.2593	0.2593	0.2435
1 500	0.02	0.011	0.2637	0.2637	0.2637	0.2578

Table 4. Workload balancing for Low-High scenario

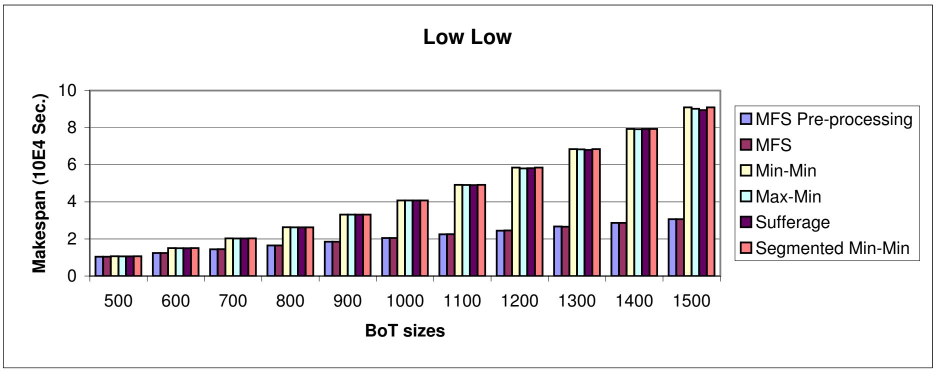


Fig. 9. Makespan for Low-Low scenario

	Meta-Schedulers					
BoT	MFS Pre-Pro	MFS	Max-Min	Min-Min	Suff	Seg Min-Min
500	0.0065	0.0056	0.2687	0.2687	0.2687	0.2523
600	0.0059	0.0051	0.2556	0.2556	0.2556	0.2421
700	0.0062	0.0056	0.2653	0.2653	0.2653	0.2523
800	0.0071	0.0059	0.2597	0.2597	0.2597	0.2479
900	0.0068	0.0057	0.2608	0.2608	0.2608	0.2486
1 000	0.0067	0.0057	0.2643	0.2643	0.2643	0.2458
1 100	0.0088	0.0076	0.2612	0.2612	0.2612	0.2505
1 200	0.0092	0.0081	0.2613	0.2613	0.2613	0.2512
1 300	0.0095	0.0084	0.264	0.264	0.264	0.2538
1 400	0.0091	0.008	0.2625	0.2625	0.2625	0.2529
1 500	0.0093	0.0081	0.2601	0.2601	0.2601	0.2536

Table 5. Workload balancing for Low-Low scenario

The best value for WLB is obtained with Low-Low scenario with a BoT of 600 tasks for MFS with no Pre-processing. With more than 1 000 tasks MFS with Pre-processing obtain worse WLB.

4.5 Performance of MFS Under Heterogeneous Scenarios

Experimental results for makespan show that time for all dynamic heuristics grows as the size of the BoT is increased. However, in our proposal, this growth is linear while for dynamic heuristics this growth is exponential. As mentioned above, our approach with pre-preprocessing obtains the best makespan for all scenarios, although the difference with dynamic heuristics is greater when the size of the BoT is high, with any combination of heterogeneity of tasks.

For future real development of our approach, MFS should be used with a BoT with more than 500 tasks and less than 1 000 tasks. These tasks can be highly heterogeneous or with low heterogeneities, but resources heterogeneities must be high to obtain a better workload balancing. The length of these tasks must not exceed 300 000 MI.

In this study, the simulation results of makespan and workload balancing were examined in the above four subsections. It is observed that the makespans of the MFS with and without pre-processing were ranked as the first and second fastest. However, WLB for MFS with pre-processing was ranked as the second position and WLB for MFS without pre-processing was ranked as the first position. Therefore, in order to obtain a lower makespan it is necessary to sort the BoT in a descending way, but with this pre-processing a worse WLB is obtained. MFS with no pre-processing gets a better workload balancing although it takes more time for makespan. So it is recommended to use pre-processing when nodes overload is no significant. If nodes overload is an important factor, not to use pre-processing is the best option.

5 CONCLUSIONS AND FUTURE WORKS

A Grid environment is heterogeneous in nature in the real world since it works with different computing speeds at different participating sites. Heterogeneity presents a challenge for effectively arranging load sharing activities in a computational grid. This paper explores tasks scheduling and allocation issues in heterogeneous computational Grids scenarios using fuzzy logic.

Compared to prior literature, this Meta-Scheduler is of a great value, since in prior literature there is no Meta-Scheduler able to manage a set of tasks with no a priori information. Conversely, previously revised heuristics need to know the tasks to be executed in advance.

At this stage of the research, we have some future directions that might prove interesting: Firstly, we will conduct additional experiments to obtain automatic rules in order to improve the fuzzy logic sets. Secondly, we will apply priorities to the tasks in the queue so that those with the highest priority can be brought forward in the queue of the Meta-Scheduler. Thirdly, we will carry out several studies using different methods to sort the Bag-of-Tasks in order to organize the queue of the Meta-Scheduler. Furthermore, we will study the behavior of our approach in online mode.

REFERENCES

[1] Foster, I.—Kesselman, C.: The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, San Francisco (USA) 1999.

[2] Buyya, R.—Krauter, R.—Maheswaran, M.: A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. J. Softw. Pract. Exper., Vol. 32, 2002, No. 2, pp. 135—164.

[3] Xhafa, F.—Abraham, A.: Computational Models and Heuristic Methods for Grid Scheduling Problems. Future Generation Computer Systems, Vol. 26, 2010, No. 4, pp. 608–621.

[4] Hoheisel, A.—Wieczoreka, M.—Prodan, R.: Towards a General Model of the Multi-Criteria Workflow Scheduling on the Grid. Future Generation Computer Systems, Vol. 25, 2009, No. 3, pp. 237–256.

[5] Venugopal, S.—Robic, B.—Sulistio, a.—Cibej, U.—Buyya, R.: A Toolkit for Modelling and Simulating Data Grids: An Extension to Gridsim. Concurrency and Computation: Practice and Experience (CCPE) 2007.

[6] Lin, P.—Huang, P.—Peng, H.—Li, X.: Static Strategy and Dynamic Adjustment: An Effective Method for Grid Task Scheduling. Future Generation Computer Systems, Vol. 25, 2009, No. 8, pp. 884–892.

[7] Da Silva Porto, S. C.–Almeida, V. A. F.—Menascé, D. A.—Saha, D.—Tripathi, S. K.: Static and Dynamic Processor Scheduling Disciplines in Heterogeneous Parallel Architectures. J. Parallel Distrib. Comput., Vol. 28, 1995, No. 1, pp. 1–18.

[8] Xhafa, F.—Abraham, A.: Metaheuristics for Scheduling in Distributed Computing Environments. Springer Berlin/Heidelberg, Vol. 146, Chapter 1, 2008, pp. 1–37.

[9] Maheswaran, M.—Siegel, H. J.: A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems. In Seventh Heterogeneous Computing Workshop, 1998, pp. 57–69.

[10] Etminani, K.—Naghibzadeh, M.: A Min-Min Max-Min Selective Algorithm for Grid Task Scheduling. In 3rd IEEE/IFIP International Conference in Central Asia, 2007, pp. 1–7.

[11] Aziz, A.—El-Rewini, H.: On the Use of Meta-Heuristics to Increase the Efficiency of Online Grid Workflow Scheduling Algorithms. Cluster Computing, Vol. 11, 2008, No. 4, pp. 373–390.

[12] Christodoulopoulos, K.—Sourlas, V.—Mpakolas, I.—Varvarigos, E.: A Comparison of Centralized and Distributed Meta-Scheduling Architectures for Computation and Communication Tasks in Grid Networks. Computer Communications, Vol. 32, 2009, No. 7–9, pp. 1172–1184.

[13] Vilar, F.–Da Silva, D. P.—Cirne, W.—Grande, C.: Trading Cycles for Information: Using Replication to Schedule Bag of Tasks Applications on Computational Grids. In Proc of Euro-Par 2003, pp. 169–180.

[14] Braun, T. D.—Siegel, H. J.—Beck, N.: A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks Onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, Vol. 61, 2001, pp. 810–837.

[15] Yu, K.—Yin Zhou, W.—Liu, Y.—Wei, J.—Xia, J.: A Unified Strategy for Dynamic Mapping in Grid Computing Environments. In International Conference on Mechatronics and Automation (ICMA) 2007, pp. 2309–2313.

[16] Siegel, H. J.—Hensgen, D.—Maheswaran, M.—Ali, S.—Freund, R. F.: Dynamic Matching and Scheduling of a Class of Independent Tasks Onto Heterogeneous Computing Systems. Eighth Heterogeneous Computing Workshop 1999, pp. 30–44.

[17] Kousalya, K.—Balasubramanie, P.: Ant Algorithm for Grid Scheduling Powered by Local Search. Int. J. Open Problems Compt. Math., Vol. 1, 2008, No. 2, pp. 30–44.

[18] Siegel, H. J.—Ali, S.: Techniques for Mapping Tasks to Machines in Heterogeneous Computing Systems. In International Conference on Parallel Processing (ICPP 2004), pp. 174–185.

[19] Weng, C.—Lu, X.: Heuristic Scheduling for Bag of Tasks Applications in Combination With Qos in the Computational Grid. Future Generation Computer Systems, Vol. 21, 2005, No. 2, pp. 271–280.

[20] Yu, K. M.—Chen, C. K.: An Adaptive Scheduling Algorithm for Scheduling Tasks in Computational Grid. In Seventh International Conference on Grid and Cooperative Computing 2008, pp. 185–189.

[21] Liang, C. H.—Lee, L. T.—Chang, H. Y.: An Adaptive Task Scheduling System for Grid Computing. In The Sixth IEEE International Conference on Computer and Information Technology (CIT 06) 2006, pp. 57–62.

[22] Min-You, W.—Shu, W.—Zhang, H.: Segmented Min-Min: A Static Mapping Algorithm for Meta-Tasks on Heterogeneous Computing Systems. Heterogeneous Computing Workshop, 2000, pp. 375–385.

[23] He, X.—Sun, X.–von Laszewski, G.: QoS Guided Min-Min Heuristic for Grid Task Scheduling. Journal of Computer Science and Technology, Vol. 18, No. 4, 2003, pp. 442–451.

[24] Casanova, H.—Legrand, A.—Zagorodnov, D.—Berman, F.: Heuristics for Scheduling Parameter Sweep Applications in Grid Environments. Procceding of the 9th heterogeneous Computing Workshop (HCW '00), 2000, pp. 349–363.

[25] Lewis, T.—El-Rewini, H.—Ali, H.: Task Scheduling in Parallel and Distributed Systems. PTR Prentice Hall, 1994.

[26] Carretero, J.—Dorronsoro, B.—Alba, E.—Xhafa, F.: A Tabu Search Algorithm for Scheduling Independent Jobs in Computational Grids. Computing and Informatics, Vol. 28, 2009, No. 2, pp. 237–250.

[27] Garibaldi, J. M.—Fayad, C.—Ouelhadj, D.: Fuzzy Grid Scheduling Using Tabu Search. In IEEE International Conference on Fuzzy Systems, 2007, pp. 1054–1059.

[28] Buyya, R.—Abraham, a.—Nath, B.: Natures Heuristics for Scheduling Jobs on Computational Grids. In IEEE International Conference on Advanced Computing and Communications (ADCOM 2000), 2000, pp. 45–52.

[29] Yarkhan, A.—Dongarra, J.: Experiments with Scheduling Using Simulated Annealing in a Grid Environment. In International Workshop on Grid Computing (GRID 2002), 2002, pp. 232–242.

[30] Paletta, M.—Herrero, P.: A Simulated Annealing Method to Cover Dynamic Load Balancing in Grid Environment. Advances in Soft Computing, Vol. 50, 2009, pp. 1–10.

[31] Sheng Chang, J.—Chang, R.-S.—Lin, P.-S.: An Ant Algorithm for Balanced Job Scheduling in Grids. Future Generation Computer Systems, Vol. 25, 2009, No. 1, pp. 20–27.

[32] Fidanova, S.—Durchova, M.: Ant Algorithm for Grid Scheduling Problem. Large Scale Computing, Lecture Notes in Computer Science, Vol. 3743, 2006, pp. 405–412.

[33] Deldari, H.—Salehi, M. A.—Dorri, B. M.: Balancing Load in a Computational Grid Applying Adaptive Intelligent Colonies of Ants. Informatica, Vol. 32, 2008, pp. 327–335.

[34] Ritchie, G.—Levine, J.: A Hybrid Ant Algorithm for Scheduling Independent Jobs in Heterogeneous Computing Environments. In 23$^{\text{rd}}$ Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2004), 2004.

[35] Kent, R. D.—Aggarwal, M.—Ngom, A.: Genetic Algorithm Based Scheduler for Computational Grids. In International Symposium on High Performance Computing Systems and Applications, Vol. 15, 2005, pp. 209–215.

[36] Rong, H.—Gao, Y.—Huang, J. Z.: Adaptive Grid Job Scheduling With Genetic Algorithms. Future Generation Computer Systems, Vol. 21, 2005, No. 1, pp. 151–161.

[37] Liao, C.-J.—Tseng, C. T.: A Discrete Particle Swarm Optimization for Lot-Streaming Flowshop Scheduling Problem. European Journal of Operational Research, Vol. 191, 2008, No. 2, pp. 360–373.

[38] Wang, H.—Jiang, C.—Kang, Q.—He, H.: A Novel Discrete Particle Swarm Optimization Algorithm for Job Scheduling in Grids. In 4$^{\text{th}}$ International Conference on Natural Computation, ICNC, 2008, pp. 401–415.

[39] Franke, C.—Hoffmann, F.—Lepping, J.—Schwiegelshohn, U.: Development of Scheduling Strategies With Genetic Fuzzy Systems. Appl. Soft. Comput., Vol. 8, 2008, No. 1, pp. 706–721.

[40] Ferrari, D.: Real-Time Communication in an Internetwork. Journal of High Speed Networks, Vol. 1, 1992, No. 1.

[41] Subramani V.—Srinivasan, S.—Kettimuthu, R.—Sadayappan, P.: Characterization of Backfilling Strategies for Parallel Job Scheduling. In International Conference on Parallel Processing Workshops (ICPPW 02), 2002, pp. 514–519.

[42] Schopf, J. M.: A General Architecture for Scheduling on the Grid. Special Issue on Grid Computing, J. Parallel and Distributed Computing 2002.

[43] Naghibzadeh, M.—Bagheri, E.: A New Approach to Resource Discovery and Dissemination for Pervasive Computing Environments Based on Mobile Agents. Electrical and Computer Engineering, Vol. 14, 2006, No. 6.

[44] Ranaldo, N.—Zimeo, E.: Time and Cost-Driven Scheduling of Data Parallel Tasks in Grid Workflows. IEEE Systems Journal, Vol. 3, 2009, No. 1, pp. 104–120.

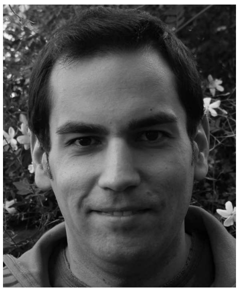
[45] Mamdani, E. H.—Assilian, S.: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. International Journal of Man-Machine Studies, Vol. 7, 1975, No. 1, pp. 1–13.

[46] Stiles, J. R.—Bartol, T. M.—Salpeter, E. E.—Salpeter, M. M.: Monte Carlo Simulation of Neuromuscular Transmitter Release Using M-Cell, a General Simulator of Cellular Physiological Processes. Computational Neuroscience, 1998, pp. 279–284.

[47] Smallen, S.—Cirne, W.—Berman, F.—Young, S.—Ellisman, M.—Frey, J.—Wolski, R.—Su, M.—Kesselman, C.: Combining Workstations and Supercomputers to Support Grid Applications: the Parallel Tomography Experience. Proceedings of the 9th Heterogeneous Computing Workshop, 2000, pp. 241–252.

[48] Smallen, S.—Casanova, H.—Berman, F.: Applying Scheduling and Tuning to On-Line Parallel Tomography. Proceedings of Supercomputing 2001, pp. 46.

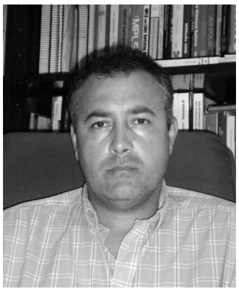
[49] Abramson, D.—Giddy, J.—Kotler, L.: High Performance Parametric Modeling With Nimrod/G: Killer Application for the Global Grid. Proceedings of Fourteenth IPDPS, Cancun 2000, pp. 520–528.



Antonio Javier Sanchez Santiago received the M. Sc. Degree in Computer Science from the University of Jaén in 2007. Nowadays, he is a researcher at the Telecommunication Engineering Department of the University of Jaén. His areas of research interests include high-distributed computing resource management and scheduling. His Ph. D. thesis is focused on artificial intelligence applied to scheduling on grids.

Antonio Jesus Yuste received the M. Sc. in Telecommunication Engineering from University of Málaga in 1994. Since 2003, he has been Associate Professor at Telecommunication Engineering Department of Jaén University. His areas of research interests include routing in ad hoc networks, grid computing and traffic analysis of computer networks. He is involved in research projects of the Spanish Ministry of Science and Education and of private companies.

Jose Enrique Muñoz Exposito received M. Sc. and Ph. D. Degrees in Telecommunication Engineering from Mlaga University and Jaén University in 1995 and 2009, respectively. Since 2003, he has been an Associate Professor at the Telecommunication Engineering Department at Jaén University. His areas of research interest include speech and audio analysis, computer networks, and soft computing. He is involved in research projects of the Spanish Ministry of Science and Education and of private companies.



Sebastian García Galán received M. Sc. and Ph. D. Degrees in Telecommunication Engineering from Málaga University (UMA) and Madrid Technical University (UPM), in 1995 and 2004, respectively. Since 1999, he has been an Associate Professor at the Telecommunication Engineering Department at Jaén University. His research areas include engineering applications, artificial intelligence, grid computing, and telecommunication systems. He is involved in research projects of the Spanish Ministry of Science and Education and of private companies.

Rocio Perez de Prado received M. Sc. Degree in Telecommunication Engineering from Seville University in 2008. At present, she is a researcher at the Telecommunication Engineering Department at Jaén University. Her research areas include highly distributed computing resource management and scheduling. Her Ph. D. thesis is focused on artificial intelligence applied to scheduling in computational grids.