Meta-Schedulers for Grid Computing based on Multi-Objective Swarm Algorithms

María Arsuaga-Ríos[a,*], Miguel A. Vega-Rodríguez[b], Francisco Prieto-Castrillo[c]

[a] *Beams Department, European Organization for Nuclear Research, CERN CH-1211, Genève 23, Switzerland*
Phone: +41-22-77235, Fax: +41-22-69654
[b] *ARCO Research Group, University of Extremadura, Dept. Technologies of Computers and Communications, Escuela Politecnica, Campus Universitario s/n, 10003, Cáceres, Spain*
Phone: +34-927-25-72-63, Fax: +34-927-25-71-87
[c] *Extremadura Research Center for Advanced Technologies, Ceta-Ciemat s/n, 10200, Trujillo, Spain*
Phone: +34-927-65-93-17, Fax: +34-927-32-32-37

Abstract

Job scheduling is a challenging task on grid environments because they must fulfill user requirements. Scientists often have deadlines and budgets for their experiments (set of jobs). But these requirements are in conflict with each other - cheaper resources are slower than the expensive ones -. In this paper, we have implemented two multi-objective swarm algorithms. One of them is based on a biological behavior - Multi-Objective Artificial Bee Colony (MOABC) - and the other on physics - Multi-Objective Gravitational Search Algorithm (MOGSA) -. Multi-objective properties enhance the optimization of execution time and cost per experiment. These algorithms are evaluated regard to the standard and well-known multi-objective algorithm - Non-dominated Sorting Genetic Algorithm II (NSGA II) - in order to prove the goodness of our multi-objective proposals. Moreover, they are compared with real meta-schedulers as the Workload Management Sys-

*Corresponding author
Email addresses: maria.arsuaga.rios@cern.ch (María Arsuaga-Ríos), mavega@unex.es (Miguel A. Vega-Rodríguez), francisco.prieto@ciemat.es (Francisco Prieto-Castrillo)

tem (WMS) from the most used European grid middleware, gLite, and the Deadline Budget Constraint (DBC) from Nimrod-G, that takes into account the same requirements. Results show us that MOABC offers better results in all the cases using diverse workflows with dependent jobs over different grid environments.

Keywords: scheduling, grid computing, swarm, multi-objective optimization

---

1. Introduction

The paradigm of grid computing is defined as a parallel and distributed system that allows to share, select and collect autonomous resources geographically distributed in a dynamic way. All these actions are carried out in execution time depending on the availability, capacity, cost and quality of the resources required by the users. This focus emerges from the synergy between the cooperation among computing resources with a decentralized control and providing them as services. Therefore, one of the most important and challenging actors, that participate in this decentralized control, are the meta-schedulers, also known as resource brokers. This service is implemented in the middleware which facilitates the grid environment management. The main function of a meta-scheduler is to assign the jobs to adequate resources following computational requirements and quality of service demanded by the users.

Grid computing is widely used in the scientific world solving complex experiments (set of interdependent jobs) that require a high performance. Scientists often have to consider deadlines and budgets of their experiments related to important projects. Because of that, the optimization of execution time and cost is a key factor to consider in the job scheduling process carried out by the meta-schedulers. However, these types of objectives are in conflict each other, because usually cheaper resources are slower than expensive ones, hence a multi-objective optimization is required.

In this paper, a study of bio-inspired algorithms is presented to solve this multi-objective problem. The implemented algorithms are based on Swarm Intelligence (SI). Swarm intelligence is a kind of intelligence that emerges from the collaboration and competition among individuals. In particular, two novel swarm algorithms from different fields - biology and physics - have been adapted and evaluated. Multi-Objective Artificial Bee Colony

(MOABC) is based on the Artificial Bee Colony [1], [2] from the biological field and its collaborating agents are represented as bees. Multi-Objective Gravitational Search Algorithm (MOGSA) is built from the Gravitational Search Algorithm [3] and the planets are its agents, according to the physical field. One of the main contributions of this research is the adaption of these algorithms to deal with multi-objective requirements that are in conflict each other (execution time and cost). Therefore, to give more reliability to this multi-objective study, an evaluation with a well-known standard multi-objective algorithm - Non-Dominated Sorting Genetic Algorithm II (NSGA II) [4] - has been accomplished. GridSim [5] is the simulator used to implement all the meta-schedulers.

GridSim [1] is a Java-based toolkit for modelling and simulating distributed resource management in Grid environment. GridSim is based on SimJava, a general purpose discrete-event simulation package implemented in Java. All components in GridSim communicate with each other through message passing operations defined by SimJava. It allows modelling of heterogeneous types of resources and the resources can be modeled operating under space or time shared mode. The resource capability can be defined in the form of MIPS (Million Instructions Per Second) and they can be located in any time zone. Moreover, applications with different parallel application models can be simulated. GridSim toolkit is suitable for application scheduling simulations in Grid Computing environment. GridSim is of great value to test new algorithms and strategies in a controlled environment. By using GridSim, it is possible to perform repeatable experiments and studies that are not possible in a real dynamic Grid environment. The main advantage of GridSim is that various allocation or scheduling policies can be implemented and integrated into GridSim easily, by extending them from one of the classes. Research students in the GRIDS Laboratory [2] are themselves heavy users of GridSim and extend it whenever necessary for their own research needs. In the last 5 years, GridSim has been continuously extended in this manner to include many new capabilities and has also received contributions from external collaborators. Therefore, it has been chosen due these advantages and overall to offer the facilities to configure complex topologies and resource features such as processing speed, MIPS (Million Instructions Per Second), or cost of resources

[1] http://www.buyya.com/gridsim/

[2] http://www.cloudbus.org/intro.html

per time unit. Furthermore, thanks to the GridSim flexibility, it has been modified to support workflows with dependent jobs, due to the importance to control the execution time in this type of workflows (child jobs have to wait until their parents are successfully executed). In particular, in this research six workflows - Gaussian, Gauss-Jordan, LU decomposition, Find-Max, Fast Fourier Transform and Stencil - have been tested with all the implemented meta-schedulers to study their behaviour in each situation. In addition, two different grid environments have been used to reinforce the study of their behaviour in different scenarios. Also, the best meta-scheduler, in this case the proposal MOABC, has been compared with two real meta-schedulers: the Workload Management System (WMS) and the Deadline Budget Constraint meta-scheduler (DBC) to show the relevance of our results.

This paper is organized as follows. Section 2 presents the related work. Section 3 exposes the problem including an introduction of the multi-objective approach. Section 4 introduces the Multi-Objective Artificial Bee Colony algorithm. In Section 5 Multi-Objective Gravitational Search Algorithm is explained. Section 6 presents the standard Non-Dominated Sorting Genetic Algorithm II. Then, several experiments are provided and analyzed in detail in Section 7. Finally the last section summarizes the main conclusions of this work.

2. Related Work

Real important meta-schedulers have been considered to evaluate the goodness of the proposals to fulfill the cost and time requirements demanded by the users.The first one is the Workload Management System (WMS) [3] due to it belongs to the most extended middleware in Europe gLite - Lightweight Middleware for Grid Computing [4] -. The second meta-scheduler selected for this study is the Deadline Budget Constraint Algorithm (DBC) [6] from Nimrod-G. This algorithm uses a greedy algorithm to attain the budget and deadline for an experiment. Moreover, a general model of workflow scheduling on the grid was presented by Wieczorek et al. [7] is applied to classify the workflow scheduling from different taxonomies taking into account the current grid systems. In that classification, only the Nimrod/G system ([8],[6]) optimizes execution time and economic cost at the same time. The study of

[3] http://web.infn.it/gLiteWMS/
[4] http://glite.cern.ch/

Xhafa and Abraham [9] reveals the complexity of the grid scheduling problem and shows the usefulness of heuristic and meta-heuristic approaches for design multi-objective grid schedulers.

Currently, multi-objective algorithms are emerging in the literature to optimize conflictive objectives. Conflictive objectives are usually resolved from different versions of multi-objective evolutionary algorithms (MOEAs) [10]. Evolutionary algorithms are commonly based on the nature behaviour as the genetic algorithms. Other bioinspired algorithms, as artificial weed colonies [11] or particle swarm [12] algorithms; have also demonstrated a good performance to optimize multiple functions in complex environments. Job scheduling in grid computing is one of the most challenging tasks due to its complexity for its dynamic behaviour and its decentralized control. Currently, this problem is tackled with multi-objective algorithms often based on genetic algorithms ([13], [14], [15], [16]) to optimize both execution time and cost for several experiments. The experimental results have proven that multi-criteria genetic algorithms offers also better solutions than classical algorithms such as SA, Duplex and min-min for resource control in Large-Scale Distributed Systems[17]. However, the test environments do not take into account specific topologies with network configuration (baud rate, delay, MTU (Maximum Transfer Unit), etc.) and resource location with specific features (operating system, number of machines, CPUs, speed, cost, etc.) being them usually homogeneous. Furthermore, most of these researches neither take into account workflows that follow a DAG (Directed Acyclic Graph) model (dependent jobs).

New multi-objective algorithms are usually compared with the standard and well-known multi-objective genetic algorithm called NSGA II [4] (Non-dominated Sorting Genetic Algorithm II) in order to assess the quality of multi-objective approaches. The NSGA II becomes a very popular algorithm mainly for its huge efficiency. For this reason, the swarm algorithms presented in this paper are evaluated with this popular genetic algorithm.

3. Problem Statement

Grid environments allow to solve computing problems using several heterogeneous resources coordinated in a decentralized way. Job scheduling is a critical problem in these environments. A proper scheduling algorithm can reduce the response time and the execution cost. Users generally have deadlines and budgets associated to their experiments, but these requirements

are in conflict each other. Faster resources usually are more expensive than slower ones. To tackle this problem, a multi-objective algorithm is expected to optimize these objectives (execution time and cost). A multi-objective optimization problem (MOP) [18] includes a set of n parameters (decision variables) and a set of k objective functions that can be subjected to determinate restrictions. Then, a MOP without restrictions can be defined as:

$$Optimize\, y = f(x) = (f_1(x), f_2(x), ..., f_k(x)) \tag{1}$$

where $x = (x_1, x_2, ..., x_n) \in X$ is the decision vector and $y = (y_1, y_2, ..., y_k) \in Y$ is the objective vector. X denotes the decision space and Y de objective space. The optimization, depending on the problem, can minimize or maximize. The optimization problem consists in founding the x that has the *best value* of *f(x)*. In general, it does not exist one only *best value*, but a set of optimum solutions. This set of optimum solutions is called *Pareto optimal*. The region of points defined by the *Pareto optimal* set in the value space of the objective functions is called *Pareto front.*

Thus, a point x^* is said to be *Pareto Optimal* for the problem if there is no other vector $x \in X$ such that $\forall\ i = 1, ..., k$: $f_i(x) \leqslant f_i(x*)$ and, for at least one $i \in 1, ..., k$: $f_i(x) < f_i(x^*)$ (considering minimization problems). This definition is based on the intuitive conviction that the point $x^* \in X$ is chosen as the optimal if no criterion can be improved without worsening at least one other criterion.

Our multi-objective approach minimizes two objectives: execution time and cost. Hence, given a set of resources $R = \{R_j\}$, $j = 1,..,n$ and a set of jobs $J = \{J_i\}$, $i = 1,..,m$, the fitness functions are described as follows:

$$Minimize\ F = (F_1, F_2) \tag{2}$$

$$F_1 = max\ time\,(J_i,\ f_j(J_i)) \tag{3}$$

$$F_2 = \sum cost\,(J_i,\ f_j(J_i)) \tag{4}$$

where $f_j(J_i)$ is a mapping function that assigns J_i onto resource R_j. Function *time* $(J_i, f_j(J_i))$ denotes the completion time and *cost* $(J_i, f_j(J_i))$ is the data transfer cost and resource cost for processing the solution.

Many workflows of applications follow a DAG (Directed Acyclic Graph) model, denoting the precedence among the jobs that compound them. This fact influences the execution time, because the child jobs need to wait to be

processed until their parents are successfully executed. Thus, these workflows have been modeled as a graph like this: *G = (V, E, l, d)* where *V* denotes the set of nodes (jobs) and *E* is the set of edges, which represents the precedence among jobs. Jobs have assigned a constant length *l(j)* that indicates its length in thousands of *MI (Million of instructions)*. The precedence among jobs is denoted as $\langle j, j' \rangle$. This means, that job *j'* cannot be executed until its parent, job *j*, has been completed successfully. Then, *j'* receives all necessary data from *j*. The label *d(j→j')* expresses the data transfer between jobs (Figure 1). This constraint is taking into account during the calculation of our fitness functions.

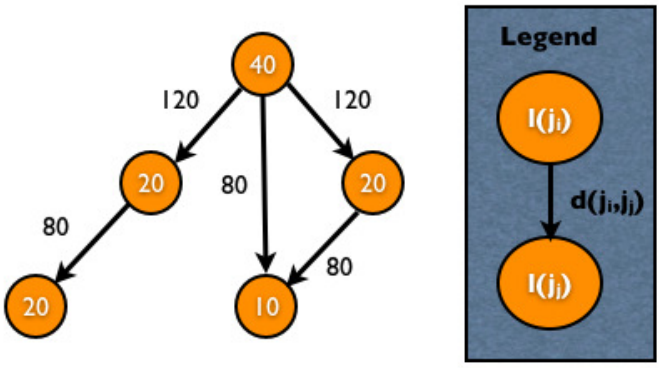

Figure 1: A simple workflow representation

Bioinspired algorithms usually work with agents (bees, planets or individuals) that often represent possible solutions for the problem. The three algorithms studied in this research use the same representation for their agents. Figure 2 depicts an example of the correspondence of the workflow to be executed in the grid with a possible solution (agent). A job scheduling solution is represented by two vectors that define an agent. These two vectors are called: allocation and order vector and are based on the Talukder's research ([16], [19]).

- Allocation vector indicates the mapping between jobs onto the resources where the jobs are going to be executed. The length of the allocation vector is denoted as |J|, such that *a(i) = j*, where $0 \leq i < |J|$

and $0 \leq j < |R|$, i.e. job J_i is assigned to resource R_j . $|J|$ indicates the total number of jobs to allocate and $|R|$the total number of available grid resources.

- Order vector decides the sent order for the jobs according to the precedence denoted by the workflow. The length of order vector is expressed as $|J|$, such $o(k) = i$, where $0 \leq i; k < |J|$, and each J_i just appears once in the vector.

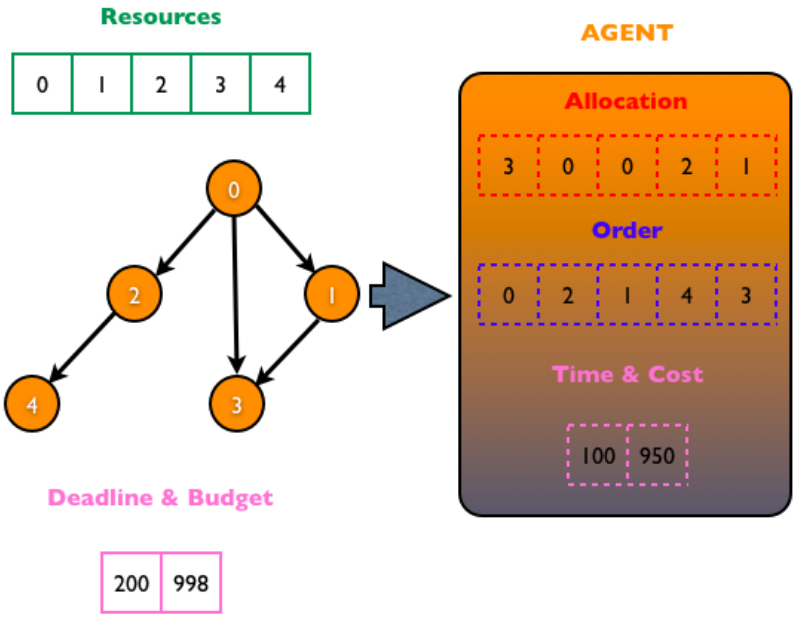

Figure 2: Agent representation for the job scheduling problem.

According to these vectors, agents obtain the execution time and cost providing possible solutions for the problem.

4. Multi-Objective Gravitational Search Algorithm (MOGSA)

Gravitational Search Algorithm [3] (GSA) is a swarm algorithm from the physical field. Its agents represent planets that have masses with different sizes exerting gravitational attractions among them through different dimensions. These attractions follow the Newtonian gravity law as a metaheuristics. Thus, biggest masses exert more force of attraction than others,

positioning themselves as best solutions. In this paper, a multi-objective version, called Multi-Objective Gravitational Search Algorithm (MOGSA), is presented to deal with the job scheduling problem in grid environments. In this new algorithm, the dimensions correspond to the combination of the two vectors: allocation and order vector and this combination is denoted as U_{A+O}. Hence, the gravitational forces are applied to each element of the vector U_{A+O} and its size is the sum of the allocation and order vector size. MOGSA manages their agents considering its multi-objective context. The main steps of MOGSA are shown in Algorithm 1.

Algorithm 1 MOGSA pseudocode

INPUT: Population Size, G_0, Min_{Kbest}, α, ϵ
OUTPUT: Set of Solutions

```
1: Initialize population of solutions;
2: Evaluate population (Time and Cost);
3: while not time limit (2 minutes) do
4:    Update Gravitational constant;
5:    Calculate size of masses;
6:    Calculate Force and Acceleration between masses;
7:    Update Velocity and Position per each mass;
8:    Select Set of Best Solutions (Pareto Front);
9:    Generate New Population;
10: end while
```

MOGSA has the same parameters of GSA [3].

- Population size is the number of agents that participate per iteration.
- G_0 is the Initial gravitational force that acts in each dimension of the agents that compose the population.
- Min_{Kbest} indicates the minimum of agents that exert their force over others. In GSA no all the agents exert their force over the others, only the best ones, that means, those that have bigger masses. Initially, *Kbest* attribute has the same value that population size and it is decreasing during the time of the algorithm until it reaches the value of Min_{Kbest}, but always keeping the best masses.
- α and ϵ are parameters used in the equations 6 and 9 respectively.

The output returned by MOGSA is the Pareto front found after completing all the time of the algorithm. This Pareto front contains the best set of solutions found that minimize the fitness functions and with the same importance.

4.1. Initialization and Evaluation

The algorithm starts initiating the population by random. Allocation vector assigns randomly available grid resources to the jobs that compound the workflow. The initialization of order vector is similar to allocation vector selecting the sent order for the jobs by random, but also considering the precedence constraints from the workflow. This initialization process is the same in the other two algorithms implemented (MOABC and NSGA II). However, to facilitate the management of the dimensional position for each agent, MOGSA joins the allocation and order vectors to form U_{A+O} in order to calculate faster the exerted force per dimension. Once allocation and order vectors are created per agent, GridSim returns the execution time and cost according to the vectors.

The evaluation begins using the ranking operator from NSGA II [4]. This operator is implemented in MOGSA to classify the agents per Pareto fronts. This ranking operator assigns a rank per agent corresponding with the front that belongs to. Then, a second operator from NSGA II is also applied, called Crowding distance, in order to calculate the multi-objective fitness (*MOFitness*) per agent. This value is calculated by equation 5.

$$MOFitness(X_i) = (2^{(X_i.r)} + \frac{1}{1 + X_i.cd})^{-}1 \tag{5}$$

where X_i denotes the agent i, r indicates the rank of Pareto front and cd represents the crowding distance. MOFitness is used to sort the agents according its goodness. Agents with more MOFitness represent better solutions.

4.2. Update Gravitational Constant

Gravitational constant G is initialized at the beginning with the value of G_0 but it is reduced as the time goes by, in order to control the search accuracy (see equation 6).

$$G = G_0 \exp^{(-\alpha)\frac{t}{T}} \tag{6}$$

where t is the current time, T is the total time and is a parameter used to measure the reduction of G.

4.3. Calculate size of masses

Heavier masses mean more efficient agents, so better agents exert higher attractions and walk more slowly. The size of masses is calculated considering their fitness and they are updated following equations 7 and 8.

$$X_i.q = \frac{X_i.MOFitness - MOFitness_{worst}}{MOFitness_{best} - MOFitness_{worst}} \tag{7}$$

$$X_i.m = \frac{X_i.q}{\sum_1^N X_j.q} \tag{8}$$

where $MOFitness_{worst}$ and $MOFitness_{best}$ are the lowest and highest values of *MOFitness*, respectively.

4.4. Calculate Force and Acceleration between masses

Acceleration is caused by the exerted force between masses in all these dimensions. Therefore, MOGSA calculates the exerted force from the *Kbest* agents on the rest of the population for each dimension. That means, only *Kbest* agents act to the whole population. At first, *Kbest* has the same value of the population size, but *Kbest* is decreasing as the time goes by until Min_{Kbest}. To calculate the corresponding force (see equation 9) per each pair of agents an Euclidean distance is calculated following equation 10. Euclidean distance is carried out from the U_{A+O} vectors of the pair of agents. Using in the first part of the vector (allocation vector), the distance between the resource numbers and for the second part (order vector) the distance between the order positions.

$$X_i.F_j^d = G \times \frac{X_i.m \times X_j.m}{R_{ij} + \epsilon} \times (X_j.U_{A+O}^d - X_i.U_{A+O}^d) \tag{9}$$

$$R_{i,j} = \|X_i, X_j\|, \forall i, 1 \leq i \leq N; \forall j, 1 \leq j \leq K_{best} \tag{10}$$

Total force that acts on Xi is the sum of a random weight in all its dimensions (equation 11). MOGSA applies this stochastic feature to consider the exploration process. In swarm algorithms, exploration and exploitation processes are required to avoid local optimums and achieve optimal solutions.

$$X_i.f^d = \sum_{j \in K_{best}, j \neq i}^{N} rand\,[0,1] \times X_i.F_j^d \tag{11}$$

Agents obtain their acceleration taking into account its total mass and the forces that are exerted on them in all their dimensions (see equation 12).

$$X_i.a^d = \frac{X_i.f^d}{X_i.m} \tag{12}$$

4.5. Update Velocity and Position per each mass

The acceleration provokes the update in the velocity and the position of the agents. However, during the calculation of the velocity not only acceleration affects its update, but a random number is applied to improve the exploration process (see equations 13 and 14).

$$X_i.v^d = rand\,[0,1] \times X_i.v^d + X_i.a^d \tag{13}$$

$$X_i.U^d_{A+O} = X_j.U^d_{A+O} + X_i.v^d \tag{14}$$

The new agent position is updated increasing the values of the U_{A+O} vector, the identifier of the resource in case of the allocation vector or the number of the position in the order vector. In case the last vector, the precedence constraint is considered.

4.6. Select Set of Best Solutions (Pareto Front)

The current best solutions are calculated from the ranking operator that classifies the population per Pareto fronts. MOGSA has an improvement to avoid soon stagnations, so a stagnation evaluation is implemented on the new population. When stagnation occurs along several iterations, a mutation process is applied to the agents that are not changed regarding the previous population. This mutation process is subdivided in two types of mutations per each vector (allocation and order). These mutations use heuristics of the job scheduling problem to do a local search. Both, allocation and order vector are generated previously by random and then heuristics are applied. The random order is compared with other built from a greedy algorithm. This greedy algorithm is based on the precedence constraint of the workflow. The method consists in executing first the parent jobs with more dependent jobs. The mutation process for the allocation vector is more complex, because more heuristics are considered: order of resources according to speed/cost, time per job in its specific resource, overhead time (prediction of execution for the entire workflow) and wait time produced by the precedence between jobs.

Finally, the resulting first Pareto front will be saved as a set of best solutions and MOGSA restarts a new generation until the time limited is expired.

5. Multi-Objective Artificial Bee Colony (MOABC)

Artificial Bee Colony (ABC) ([1], [2]) is a mono-objective swarm algorithm from the biological field. This algorithm is based on the collective behaviour of its agents - bees - to find the best nectar from the flowers. The main feature of the ABC algorithm is that its agents have different behaviours. Some bees move in a multidimensional search space by selecting nectar source considering their last experience and the experience of their hive fellows. However, other bees move randomly without experience and influence. When they find a better nectar source (flower position), they memorize it and forget it previous flower position. That means, ABC also combines exploration and exploitation processes trying to equilibrate them. These processes are collected from the behaviour of tree kind of bees: employed, onlooker and scout bees.

- Onlookers: are waiting in the work area to select a good flower, previously chosen by employed bees.
- Employed bees: initially go to the flowers and they dance according to the goodness of the nectar found. Onlookers choose them according to the dance of the best ones.
- Scouts: are those that perform random scans to find other flowers, usually are based on heuristics of the problem.

In this research, a multi-objective version, called Multi-Objective Artificial Bee Colony (MOABC), is implemented and also is adapted to solve the job scheduling problem in grid environments. The fundamental steps of the MOABC are shown in Algorithm 2.

MOABC manages the same parameters than the ABC algorithm ([1], [2]). In particular, this swarm algorithm stands out for its simplicity with a small number of parameters.

- Population size indicates the number of agents - bees - per iteration.
- Mutation probability is used in the mutation process during the algorithm execution.

Algorithm 2 MOABC pseudocode

INPUT: Population Size, Mutation Probability
OUTPUT: Set of Solutions

```
1: Initialize population of solutions;
2: Evaluate population (Time and Cost);
3: while not time limit (2 minutes) do
4:    Multi-Objective Exploitation Process (Employed and Onlooker Bees);
5:    Multi-Objective Exploration Process (Scout Bees);
6:    Select Set of Best Solutions (Pareto Front);
7:    Generate New Population;
8: end while
```

The output of the MOABC algorithm is composed of a set of solutions - bees- obtaining by the calculation of the Pareto front after completing the algorithm execution.

5.1. Initialization and Evaluation of the population

The initialization of the population is similar to the MOGSA. However, the population begins considering only the half number of the population size given by parameter. This half is for creating the employed bees. During the algorithm execution, new bees (onlookers and scouts) are added until completing the population size. These first employed bees are also composed by the allocation and order vectors, which are created by random, taking into account the workflow precedence constraint. Then, GridSim provides their execution time and cost according to the vectors. Once employed bees are created, they are classified per Pareto fronts using the ranking operator from the NSGA II algorithm. And also their Crowding distance is calculated to evaluate the goodness of the solutions found.

5.2. Multi-Objective Exploitation Process

In the multi-objective exploitation process, employed and onlooker bees search the best solutions from their last experience and the experience of their fellows. This process begins generating neighbour bees from the employed bees. This step uses two types of mutations, one per vector (allocation and order) to generate a neighbour bee.

- Order mutation is in charge of modifying the order vector considering the DAG model from the workflow. First of all, each job is selected

taking into account the mutation probability. The process identifies the last position of order vector that is occupied by one parent of the job to mutate. After that, the positions of the child jobs are searched in the order vector to select the first position of them. And finally the process chooses randomly a new position for the job to mutate among the parent last position and the child first position.

- Allocation mutation provides a neighbour allocation vector using the mutation probability. The selected jobs to mutate choose randomly a resource identification number from the list of available grid resources.

Once employed and neighbour bees are created, they compete to be selected for the next generation. This competition uses two operators from the standard NSGA II. First, the ranking operator is applied to calculate the rank for each bee. The bee with less rank is the winner. In case of the bees have the same rank, the Crowding distance operator is used to tie break. The bee with more crowding distance is the winner. And the winner substitutes the other bee.

Then, all the winner bees are ordered according to their solution goodness using the ranking and crowding distance operators. After that, a method of the roulette selection is used to generate these new bees following the bee probability equation 15 (that is, using a linear bias [20]).

$$bee.pb = \frac{\frac{1}{bee.p}}{\sum_i \frac{1}{bee_i.p}} \tag{15}$$

Once onlookers are selected by this process, also neighbours generation is applied to these bees in the same way as the employed bees. And, winner bees substitute the looser bees.

5.3. Multi-Objective Exploration Process

The multi-objective exploration process is executed by a new kind of bee: the scout bee, whose behavior considers heuristics of the job scheduling problem. In this problem, one scout bee is enough to execute this process. Scout bee generates randomly its vectors - allocation and order - and then modifies them using information from the problem but without the experience of their hive fellows. For each vector a different process is performed considering different heuristics. These processes are similar to the mentioned in the stagnation mutation in the MOGSA.

- Heuristic generation for the order vector begins generating a random order vector according to the precedence constraint as processed in the population initialization. At the same time, other order vector is generated using a greedy algorithm that considers that jobs with more dependent jobs must be executed first. Then, the random order is modified trying to achieve the greedy order without violating the job dependencies.

- Heuristic generation for the allocation vector begins generating a random allocation vector. Then, the algorithm applies as heuristics per job the processing time $\frac{MI}{MIPS}$, where MI (Millions of Instructions) denotes the job length and $MIPS$ (Million Instructions per Second) is the speed of the assigned resource. Also, the total time would take the execution of the workflow is calculated according to the dependencies between jobs. This process assumes that jobs with no dependencies between them can be executed in parallel. After that, each job is assigned to a resource that reduces the current total execution time. The available grid resources are sorted according to the value processing speed/cost. The overhead time is also considered by the competitive jobs - jobs without dependencies each other that are allocated in the same resource -.

5.4. Select Set of Best Solutions

This set of solutions is selected from the set of all types of bees. All the bees - employed, onlooker and scout bees - are ranked and the bees that compose the first Pareto front are the best solutions of the current iteration. Then, this new first Pareto front is saved and it will be compared with the Pareto front from the next iteration.

Finally, new population is selected by the ranking operator too, saving the number of employed bees beginning from the first fronts. That means, if the employed number is 50, the first 50 bees from the first Pareto fronts are chosen.

6. Non Dominated Sorting Genetic Algorithm (NSGA II)

Non-dominated Sorting Genetic Algorithm II (NSGA II) [4] is the most popular multi-objective genetic algorithm and it is widely known due to its efficiency. In this research, this algorithm is applied to the job scheduling

problem to prove the multi-objective goodness of the two proposed swarm algorithms. As a genetic algorithm, it has agents as individuals that compose the evolutionary population. The main steps of NSGA II are shown in Algorithm 3. The NSGA II input has three parameters:

Algorithm 3 NSGA II pseudocode

INPUT: Size of population, Number of Iterations or Maximum time of execution, Crossover and Mutation Probability
OUTPUT: Set of Solutions

```
1: Initialize population of solutions;
2: Evaluate population (Time and Cost);
3: while not time limit (2 minutes) do
4:    Binary Tournament Selection;
5:    Crossover;
6:    Mutation;
7:    Select Set of Best Solutions (Pareto Front);
8:    Select New Generation;
9: end while
```

- Population size indicates the number of individuals that take part of the optimization.
- Crossover probability is the probability for interchanging the gens (allocation or order vector elements) to crossover with other individuals.
- Mutation probability is the probability to mutate the gens of an individual.

The output is the same as the previous algorithms; it is the set of best solutions found during the execution of the algorithm composing a Pareto front.

6.1. Initialization and evaluation of the population

The initialization of the population follows the same process as the two other algorithms. Population size indicates the number of individuals and these individuals create randomly their allocation and order vectors respecting the precedence constraint from the workflow to execute. Moreover, GridSim provides the execution time and cost for each individual taking into

account its vectors. After that, the evaluation is done by the NSGA II operators: Ranking of Pareto fronts and Crowding distance. The next processes - tournament selection, crossover and mutation - are in charge of creating the offspring population of the same size of the initial population.

6.2. Global optimum search

In NSGA-II the descendent population Q (size N) is created from the parent population P (size N). After that, these two populations are combined forming a new population R with size 2N. Then, the population R is classified by a non-dominated sorting in different Pareto fronts. This allows a global verification of dominance between the parent and descendent populations. When the process of non-dominated sorting is finalized, the new population is generated from the non-dominated Pareto fronts. This new population starts its building from the best non-dominated front (F1) followed from the second Pareto front (F2) and so on. As the size of population R is 2N and the size of the origin population is N, not all the solutions from R will be conform the new population. Those fronts that cannot be added to the new population will be discarded. When the last front is being considered, the solutions that belong to it can exceed the size of the population, and then the crowding distance is applied to these solutions. The crowding distance permits the solutions selection located in a few crowded area (far away from other solutions) to complete the new population instead of choosing the solutions randomly. This process is not relevant for the first iterations of this algorithm, because many fronts survive to the next iteration. However, when the process advances many solutions are located in the first front and second front and they could have more than N solutions. This fact makes important the selection of quality solutions and also that these solutions ensure the diversity. The idea is to promote always those solutions that ensure more diversity from the same Pareto front. When the population converges to the optimum Pareto front (the first one), the algorithm ensures that the solutions are not closed one each other and achieves the global optimum search objective.

6.3. Binary tournament selection

Binary tournament selection takes care of choosing the parent individuals to cross. This method sorts the individuals based on non-domination and with crowding distance assigned, then, the selection in carried out using a crowded-comparison-operator ($\prec n$).

- non-domination rank for individual p indicates that individual is in front F_i, so if p is in F_i its rank will be $p_{rank} = i$.
- crowding distance $F_i(d_j)$: In this case, $p \prec n$ q if
  - $p_{rank} < q_{rank}$
  - or if p and q belong to the same front F_i then $F_i(d_p) > F_i(d_q)$ i.e. the crowing distance should be greater for p.

The individuals are selected by using a binary tournament selection with croweded-comparison-operator.

6.4. Crossover

The crossover process is divided in two crossover one per vector - allocation and order vector -. These two crossover processes are based on the Talukder's work [19]:

- Allocation crossover selects randomly a position from the allocation vector, which indicates the identifier of the job. Then, the parent vectors swap their vectors from the random position, creating two new individuals.
- Order crossover is similar to the allocation vector crossover but considering the precedence constraint. Therefore, after the swapping process, this method checks the dependencies and possible repeated order positions. To avoid this during the crossover, before doing the swapping, this method checks if there is not any position in the order vector. If it occurs, this position is stored and when there is a repeated position, one position of the stored ones is randomly selected.

These new individuals are evaluated obtaining their execution time and cost and they are added to the population set. Population size is set to $2N$.

6.5. Mutation

Mutation process is also divided in two mutation processes for the two vectors. These mutations are the same as the ones described in the MOABC algorithm in Section 5.2 and also execution time and cost are calculated for the new individuals.

6.6. Select set of best solutions

The set of best solutions are calculated by the ranking operator from the total of individuals. The first Pareto front from the current iteration will be compared with the previous iteration set following the same process as the previous algorithms.

Finally, the new population is formed by choosing the individuals that compose the first Pareto fronts until completing the new population indicated by the population size parameter.

7. Experiments and Results

In this section, results from several experiments are described. The intention of this analysis is the comparison of two swarm algorithms, MOGSA and MOABC, from different fields - physical and biological- with the standard and well-known multi-objective algorithm NSGA II to prove the multi-objective efficiency of these new proposed multi-objective algorithms. These evaluations use six different workflows over two complete and real topologies. Moreover, MOABC, our best algorithm, is compared with two real schedulers as Workload Management System (WMS) and Deadline Budget Constraint (DBC) to stands out the necessity to improve the current meta-schedulers.

7.1. Grid topologies and workflows

Test environment - topologies, workflows and algorithms - are implemented in the simulator GridSim with a complete configuration.

On the one hand, a variety of workflows based on specific numerical computational problems such as Parallel Gaussian Algorithm, Parallel Gauss-Jordan Algorithm, Parallel LU decomposition [21], Find-Max Algorithm [22], Fast Fourier Transform (FFT) and Stencil [23], are tested. All of them follows a DAG model (Figure 3) and their complex structure allows a good study of the behaviour of our algorithms.

On the other hand, two specific topologies are described based on real grid environments. The first topology is a combination between two testbeds. The topology information is based on the EU DataGrid testbed [24] and it is completed with grid resource characteristics from the WWG testbed [5]. The combination of these testbeds is shown in Figure 4 and in Table 1. Moreover, we have simulated 3 working nodes (WNs) per resource to add more complexity to this environment.

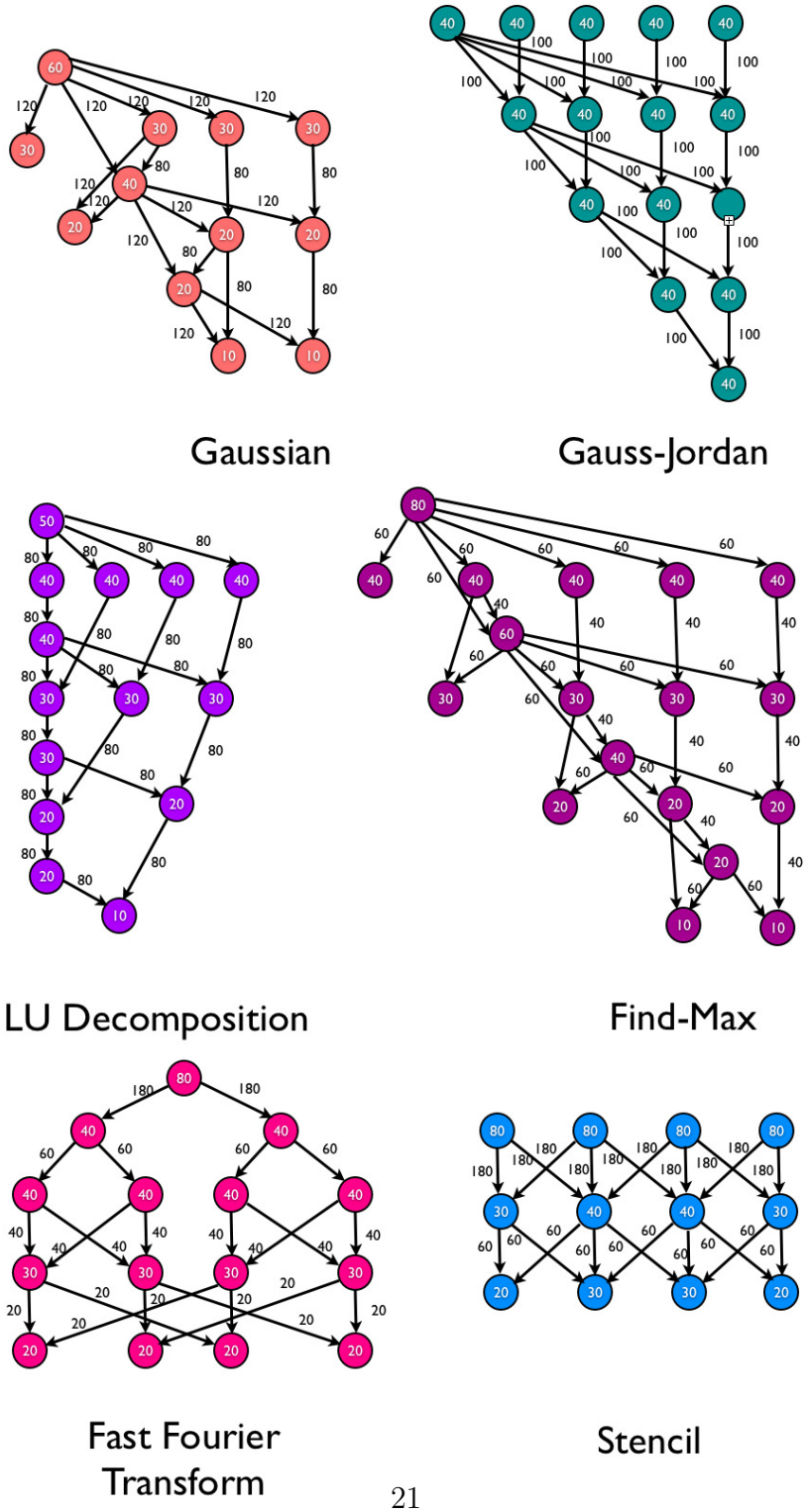

Figure 3: Workflows: Gaussian, Gauss-Jordan, LU Decomposition , Find-Max, Fast Fourier Transform (FFT) and Stencil.

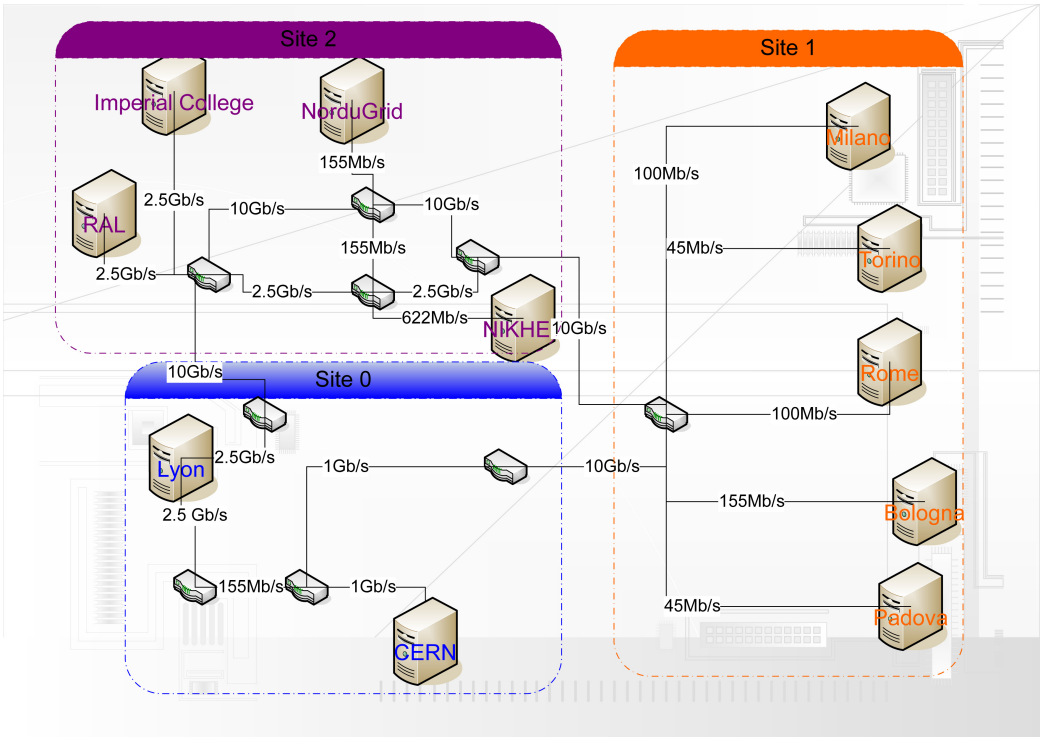

Figure 4: Testbed EU DataGrid.

Table 1: Resource Characteristics (G$ means Grid dollars)

Resource Name	Features (Vendor, Type, OS, CPUs/WN)	Resource Manager Type	MIPS per CPU	Price (G$/ CPU time)
LYON	Compaq, AlphaServer, OSF1, 4	Time-shared	515	8
CERN	Sun, Ultra, Solaris, 4	Time-shared	377	4
RAL	Sun, Ultra, Solaris, 4	Time-shared	377	3
IMPERIAL	Sun, Ultra, Solaris, 2	Time-shared	377	3
NORDUGRID	Intel, Pentium/VC820, Linux, 2	Time-shared	380	2
NIKHEF	SGI, Origin 3200, IRIX, 6	Time-shared	410	5
PADOVA	SGI, Origin 3200, IRIX, 16	Time-shared	410	5
BOLOGNA	SGI, Origin 3200, IRIX, 6	Space-shared	410	4
ROME	Intel, Pentium/VC820, Linux, 2	Time-shared	380	1
TORINO	SGI, Origin 3200, IRIX, 4 (accesible)	Time-shared	410	6
MILANO	Sun, Ultra, Solaris, 8	Time-shared	377	3

Figure 5: Site View from the IBERGRID infrastructure

The second topology is based on the Spanish Grid Initiative (NGI) [5] including the Portuguese Grid infrastructure to represent the Iberian Grid Infrastructure (IBERGRID) [6]. This testbed was carried out in the CETA-Ciemat [7] during the month of June'11. The information of the grid sites is shown in Figure 5 and Table 2. The network communication is based on the RedIris-NOVA[8] using the combination of the RedIris[9] and the RCTS[10] (Figure 6), where the fiber ring communication network allows 10Gbps. The costs of the resources are adapted following the EU Data Grid testbed. In order to take advantage from this complex topology, we have considered that

[5]http://www.e-ciencia.es/ngi/
[6]http://www.ibergrid.eu
[7]http://www.ceta-ciemat.es/
[8]http://www.redirisnova.es/caracteristicas/mapa-red.html
[9]http://www.rediris.es/lared/mapa.html
[10]http://www.umic.pt/

the size of jobs sent per each workflow is measured by Million of MI (Million of Instructions).

Table 2: Resource Characteristics (¢G$ means cent of Grid dollars). Note: Some sites have more than one resource or computing element.

Resource Name	Features (Vendor, CPU Type, OS, WNs, CPUs/WN)	Resource Manager Type	MIPS per ¢G$	Price (¢G$/ CPU time)
CESGA	AMD, Opteron 6174, Linux (Carbon), 21, 24	Space-shared	4400	56
USC	Intel, Xeon E5335, Linux (Boron), 22, 8	Space-shared	3990	18
UNICAN	Intel, Pentium D, CentOS, 241, 2	Space-shared	7505	91
CETA-Ciemat	Intel, Xeon , Linux (Beryllium), 29, 4	Space-shared	6405	19
Ciemat-TIC	AMD, Opteron 270, Linux (Boron), 74, 4	Space-shared	3608	27
Ciemat-LCG	Intel, Xeon E5450, Linux (Boron), 118, 8	Space-shared	5985	142
SGAI-CSIC	AMD, Opteron 246, Red Hat (Tikanga), 9, 2	Space-shared	3983	2
BIFI	Intel, Xeon E5650, Linux (Boron), 31, 12	Space-shared	5334	50
IFIC1	Intel, Xeon E5420, Linux (Boron), 47, 8	Space-shared	5004	47
IFIC2	Intel, Xeon E5420, Linux (Boron), 106, 8	Space-shared	4988	106
IFISC	Intel, Xeon L5520, Linux (Boron), 60, 8	Space-shared	4534	55
IAA-CSIC	Intel, Xeon X7350, Red Hat (Tikanga), 32, 16	Space-shared	5863	75
Uporto1	AMD, Opteron 250, Linux (Boron), 23, 2	Space-shared	4786	6
Uporto2	AMD, Opteron 250, Linux (Boron), 11, 2	Space-shared	4779	3
Uminho1	Intel, Xeon E5420, Linux (Beryllium), 12, 1	Space-shared	4991	2
Uminho2	Intel, Xeon E5420, Linux (Boron), 8, 1	Space-shared	4988	1
LIP-Coimbra1	Intel, Xeon E5472, Linux (Boron), 22, 8	Space-shared	5985	26
LIP-Coimbra2	Intel, Xeon E5472, Linux (Boron), 22, 8	Space-shared	5985	26
NCG-INGRID	Quad-Core AMD, Opteron 2356, Linux (Boron), 128, 8	Space-shared	4600	118
CFP-IST	Intel, Core i7 980, Linux (Boron), 4, 12	Space-shared	6747	8
LIP-LISBON	Intel, Xeon E5472, Linux (Boron), 2, 8	Space-shared	5985	2
IFCA1	Intel, Xeon E5420, Linux (Boron), 182, 8	Space-shared	6254	228
IFCA2	Intel, Xeon E5420, Linux (Boron), 182, 8	Space-shared	6254	228
IFCA3	Intel, Xeon E5420, Linux (Boron), 182, 8	Space-shared	6254	228
IEETA	Intel, Xeon E5130, Linux (Boron), 4, 2	Space-shared	4989	1

7.2. Parameterization

This section summarizes the parameter settings for each of the multi-objective algorithms - MOGSA, MOABC and NSGA II -. We have previously performed a study (each single experiment was repeated 30 times independently) to find the best parameter configuration for each algorithm in order to solve our problem:

- MOGSA: Population size = 25, G0 = 10000, $Min_{Kbest} = 5$, $\alpha = 2$, $\epsilon = 1$.

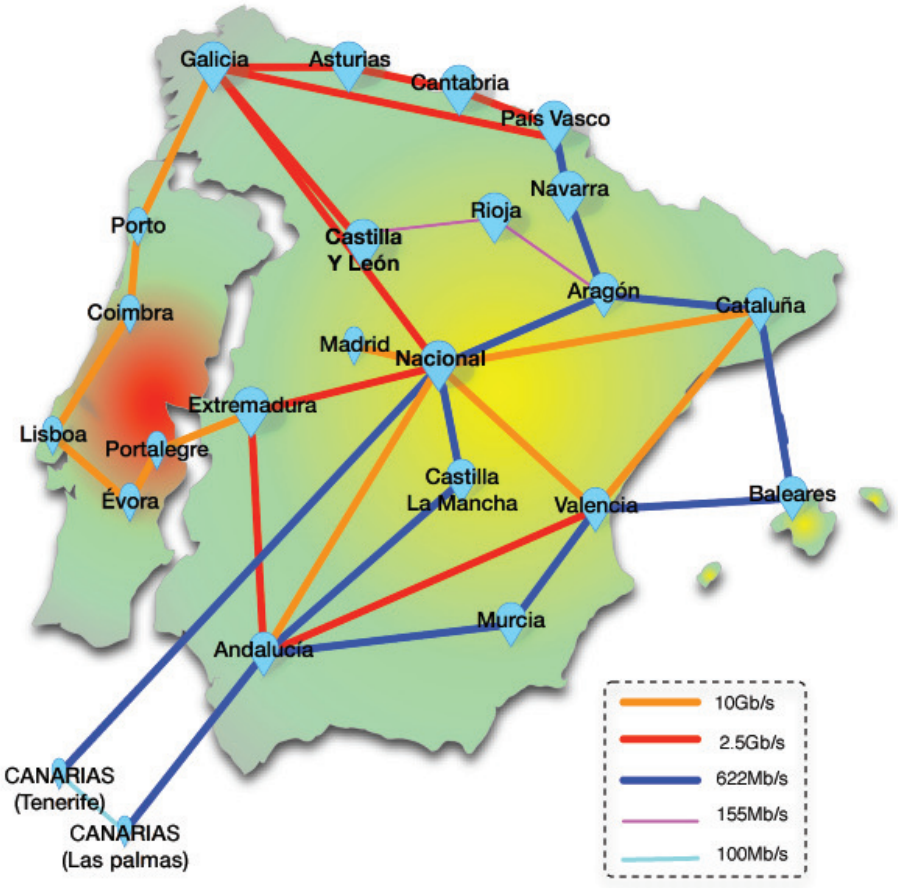

Figure 6: Communication Network: RedIris and RCTS maps

- MOABC: Population size = 101 (50 employed bees, 50 onlooker bees and 1 scout bee), mutation probability = 0.25.
- NSGA II: Population size = 100, crossover probability = 0.9, mutation probability = 0.1.

For all the algorithms the Maximum Time of Execution is 2 minutes.

7.3. Analysis

This section is divided in two main analysis. The first one studies the multi-objective properties for MOGSA and MOABC and also they are compared with the standard NSGA II. Moreover, MOABC, as a better algorithm

for our studies, is compared with two current meta-schedulers in the second analysis of this research. Due to the stochastic nature of multi-objective algorithms, each experiment performed in our study includes 30 independent executions, showing the corresponding statistical data in each case.

7.3.1. Multi-Objective comparison

In this analysis, we compare three multi-objective algorithms, two novel swarm algorithms MOGSA and MOABC from different fields (physical and biological) and the standard and well-known NSGA II, that is a very used algorithm in multi-objective optimization. This study uses two grid environments. The first part of this section evaluates the algorithms on the WWG and EU DataGrid combination testbed and the second part uses the IBERGRID testbed as a grid environment.

First, we compare the three algorithms using the hypervolume metrics. Tables 3, 4 and 5 show the average results and also the standard deviation. We observe that the reliability of the two swarm algorithms, MOABC and MOGSA, is greater than the standard NSGA II. MOABC and MOGSA have lower standard deviation in all the cases. However, the reliability between MOABC and MOGSA is similar. Moreover, the average hypervolume of MOABC is around a 57 percent being better than the 55 percent from MOGSA, and also MOGSA and MOABC are better than the 45 percent from NSGA II algorithm. This means that the solutions from MOABC and MOGSA are better than the solutions achieved by NSGA II.

To reinforce and visualize this research the Pareto fronts for each workflow are shown in Figure 7. All the algorithms follow the same behaviour with all the workflows. The resulting Pareto fronts of MOABC always dominate the solutions obtained by MOGSA and NSGA II. And the solutions found by MOGSA always dominate the results obtained by NSGA II. Moreover, a numerical comparison of these three algorithms regard to set coverage metrics is given in Table 6. Each cell gives the fraction of non-dominated solutions evolved by algorithm B, which are covered by the non-dominated points achieved by algorithm A [25]. Again, MOABC covers almost all the results obtained by MOGSA and NSGA II, and MOGSA dominates the results of NSGA II.

The same multi-objective study is carried out for the IBERGRID testbed. The hypervolume metrics is calculated per algorithm and workflow. Tables 7, 8 and 9 show the average of hypervolumes and the standard deviations. Hypervolume metrics demonstrates that MOABC obtains better set of solu-

tions than MOGSA an NSGA II, and also the set of solutions obtained by MOGSA is better than the NSGA II set of solutions. Moreover, MOABC and MOGSA have better reliability than NSGA II with a minimum standard deviation. These results are similar to the results obtained in the other study with the WWG and EU DataGrid testbed except the increase of the standard deviation from the NSGA II algorithm using the IBERGRID environment.

Furthermore, graphs of the resulting Pareto fronts are displayed in Figure 8 and also the direct comparison with the set coverage metrics is calculated in Table 10. These tests prove that MOABC usually covers all the solutions from the other algorithms as well as MOGSA covers the solutions from the NSGA II. This second test environment demonstrates a similar behaviour than the first one, highlighting MOABC as a better meta-scheduler than the other two. Moreover, it also proves that swarm algorithms have better behaviour than NSGA II in dealing with the job scheduling problem in grid environments.

7.3.2. Real schedulers comparison

This second kind of analysis compares the best multi-objective algorithm (MOABC), studied in the previous analysis, respect to two current and popular meta-schedulers - WMS and DBC - with all the workflows and the two topologies. Also, each experiment performed in our study includes 30 independent executions per algorithm and workflow. First, the WWG and EU DataGrid testbed is studied.

On the one hand, the WMS has been deployed with two requirements options. WMS needs requirements option to sort the resources by a rank of preference. The first option (Option 1) considers the resource response time. This is calculated sending a job per each available resource. The second option (Option 2) takes into account the free CPUs of every resource to calculate the rank of preferences. In these tests, the meta-scheduler is located in CERN. Table 11 shows the execution time varying the deadline for the experiments. Results demonstrate that the WMS in the two options needs more time to achieve the execution of all the jobs. Furthermore, when the deadline is more restrictive WMS is not able to complete the execution of the workflow while MOABC always offers a good solution to fulfill the deadline of an experiment.

On the other hand, the DBC is implemented and compared with the MOABC in Table 12. Results show that in spite of DBC tries to keep the

deadline of an experiment, when this deadline is more restrictive it is not able to complete the execution of the workflows. However, MOABC optimizes the solutions and always completes the execution of the workflows.

Using the IBERGRID topology, results show a similar behaviour regarding to the previous topology studied. The meta-scheduler is located in Ciemat-TIC in all these tests. Both WMS options are not able to execute all the jobs that compose the workflows when the deadline is more restrictive (Table 13). However, MOABC has not problems in carrying out the experiments. Remember that the jobs are interdependent, and therefore, the execution of all the jobs could be a must. Also, DBC has the same behaviour for this new topology obtaining worse results than the MOABC when the deadline is more restrictive (Table 14).

8. Conclusions and Future Work

This paper studies and compares three meta-scheduler algorithms based on multi-objective approaches. Two algorithms are inspired from the biology and physics fields - MOABC and MOGSA - that work according to swarm behaviour. Also a popular multi-objective genetic algorithm - NSGA II - is compared with the mentioned algorithms to evaluate the goodness of them. MOABC highlights because of its set coverage and hypervolume, being superior in all the cases than the other multi-objective algorithms. Because of that, MOABC is also compared with current schedulers as WMS and DBC to prove the relevance of this study. One more time, MOABC obtains better results than the two options of WMS and DBC with different deadlines of the workflows, which follow a DAG model with dependent jobs.

An extension of our proposed algorithm to support dynamic environments will be the next step of this research in order to offer fault tolerance when a resource falls down. A penalization of those bees or agents that allocate their jobs in invalid resources will be applied in order to discard them from the population for the following iterations.

As future work, we will apply other multi-objective metaheuristics to the job scheduling on grid environments, in order to make comparisons between the approaches presented in this work and the new ones. In particular, we will implement, adjust and study the MOEA/D [26], since it is one of the most prominent multi-objective optimizer used by the evolutionary computing community.

References

[1] D. Karaboga, B. Basturk, A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm, J. of Global Optimization 39 (2007) 459–471.

[2] D. Karaboga, B. Basturk, Artificial bee colony (ABC) optimization algorithm for solving constrained optimization problems, in: IFSA '07: Proceedings of the 12th international Fuzzy Systems Association world congress on Foundations of Fuzzy Logic and Soft Computing, Springer-Verlag, Berlin, Heidelberg, 2007, pp. 789–798.

[3] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, Gsa: A gravitational search algorithm, Information Sciences 179 (2009) 2232 – 2248.

[4] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast elitist multiobjective genetic algorithm: NSGA-II, IEEE Transactions on Evolutionary Computation 6 (2000) 182–197.

[5] R. Buyya, M. Murshed, Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing, Concurrency and Computation: Practice and Experience 14 (2002) 1175–1220.

[6] R. Buyya, M. Murshed, D. Abramson, A deadline and budget constrained cost-time optimisation algorithm for scheduling task farming applications on global grids, in: Int. Conf. on Parallel and Distributed Processing Techniques and Applications, Las Vegas, Nevada, USA, pp. 2183–2189.

[7] M. Wieczorek, A. Hoheisel, R. Prodan, Towards a general model of the multi-criteria workflow scheduling on the grid, Future Gener. Comput. Syst. 25 (2009) 237–256.

[8] D. Abramson, R. Buyya, J. Giddy, A computational economy for grid computing and its implementation in the nimrod-g resource broker, Future Generation Computer Systems 18 (2002) 1061–1074.

[9] F. Xhafa, A. Abraham, Computational models and heuristic methods for grid scheduling problems, Future Generation Computer Systems 26 (2010) 608 – 621.

[10] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, Q. Zhang, Multiobjective evolutionary algorithms: A survey of the state of the art, Swarm and Evolutionary Computation 1 (2011) 32 – 49.

[11] D. Kundu, K. Suresh, S. Ghosh, S. Das, B. Panigrahi, S. Das, Multi-objective optimization with artificial weed colonies, Information Sciences 181 (2011) 2441 – 2454.

[12] P. Chakraborty, S. Das, G. G. Roy, A. Abraham, On convergence of the multi-objective particle swarm optimizer, Information Sciences 181 (2011) 1411–1425.

[13] G. Ye, R. Rao, M. Li, A multiobjective resources scheduling approach based on genetic algorithms in grid environment, Grid and Cooperative Computing Workshops, International Conference on (2006) 504–509.

[14] J. Yu, M. Kirley, R. Buyya, Multi-objective planning for workflow execution on grids, in: GRID '07: Proceedings of the 8th IEEE/ACM International Conference on Grid Computing, IEEE Computer Society, Washington, DC, USA, 2007, pp. 10–17.

[15] B. Zeng, J. Wei, W. Wang, P. Wang, Cooperative grid jobs scheduling with multi-objective genetic algorithm, in: Parallel and Distributed Processing and Applications, volume 4742 of *Lecture Notes in Computer Science*, Springer Berlin / Heidelber, 2007, pp. 545–555.

[16] A. K. M. K. A. Talukder, M. Kirley, R. Buyya, Multiobjective differential evolution for workflow execution on grids, in: MGC '07: Proceedings of the 5th international workshop on Middleware for grid computing, ACM, New York, NY, USA, 2007, pp. 1–6.

[17] F. Pop, Optimization of resource control for transitions in complex systems, Mathematical Problems in Engineering 2012 (2012) 12.

[18] V. Khare, X. Yao, K. Deb, Evolutionary Multi-Criterion Optimization, volume 2632, Springer, Berlin / Heidelberg, 2003.

[19] A. K. M. K. A. Talukder, M. Kirley, R. Buyya, Multiobjective differential evolution for scheduling workflow applications on global grids, Concurr. Comput. : Pract. Exper. 21 (2009) 1742–1756.

[20] L. S. Pitsoulis, M. G. C. Resende, Greedy randomized adaptive search procedures, in: Handbook of Applied Optimization, Oxford University Press, 2001, pp. 168–183.

[21] T. Tsuchiya, T. Osada, T. Kikuno, Genetics-based multiprocessor scheduling using task duplication, Microprocessors and Microsystems 22 (1998) 197 – 207.

[22] M.-Y. Wu, D. D. Gajski, Hypertool: A programming aid for message-passing systems, IEEE Transactions on Parallel and Distributed Systems 1 (1990) 330–343.

[23] V. Boudet, Heterogenous task scheduling: a survey, Research Report RR-6895, INRIA, 2001.

[24] A. Sulistio, G. Poduval, R. Buyya, C. Tham, On incorporating differentiated levels of network service into gridsim, Future Gener. Comput. Syst. 23 (2007) 606–615.

[25] E. Zitzler, L. Thiele, Multiobjective optimization using evolutionary algorithms - a comparative case study, in: Proceedings of the 5th International Conference on Parallel Problem Solving from Nature, Springer-Verlag, London, UK, 1998, pp. 292–304.

[26] Q. Zhang, H. Li, MOEA/D: A multi-objective evolutionary algorithm based on decomposition, IEEE Transactions on Evolutionary Computation 2007 (2007).

9. Vitae

9.1. María Arsuaga-Ríos

Figure 9: María Arsuaga-Ríos

María Arsuaga Ríos is Computer Engineer from the University of Murcia (Spain). She got another two MSc. The first one was related to Grid Computing and e-Engineering at Cranfield University, (UK), in 2008. And the second MSc was about Information Technologies and Advanced Telematics at University of Murcia (Spain) in 2009. After her studies, she has been working on projects related to Data Mining, Ontology Systems, Bioinspired Algorithms and Optimization. She is doing her Ph.D. with the University of Extremadura (Spain). Her Ph.D. consists in researching different multi-objective strategies based on bioinspired algorithms to optimize the job scheduling problem in Grid environments. At the same time, she have taught artificial intelligence applied to Grid Computing in a MSc at the University of Extremadura. Currently, she is also enrolled in the project FESA that is being developed at CERN (Switzerland).

9.2. Miguel A. Vega-Rodríguez

Figure 10: Miguel A. Vega-Rodríguez

Miguel A. Vega-Rodríguez is a professor of Computer Architecture in the Dept. of Computer and Communications Technologies, University of Extremadura, Spain. He received a PhD degree in Computer Science from the University of Extremadura. Dr. Vega-Rodríguez has authored or co-authored more than 420 publications including journal papers, book chapters and peer-reviewed conference proceedings. In addition, he is editor and reviewer of several international JCR journals. Dr. Vega-Rodríguez's main research interests are parallel and distributed computing, reconfigurable computing, and evolutionary computing.

9.3. Francisco Prieto-Castrillo

Figure 11: Francisco Prieto Castrillo

Francisco Prieto Castrillo studied theoretical physics at Universidad Autónoma de Madrid. He obtained his Ph.D. in physics of complex systems and nonlinear dynamics. He has been involved for six years in the study of complex systems and their applications through different disciplines ranging from earthquake engineering to grid computing. At the present time his research is focussed on the optimization of distributed computing networks through complex systems based techniques.

Table 3: MOGSA Hypervolume properties per each workflow. Testbed WWG and EU DataGrid.

Workflows	Average (%)	Standard Deviation	Reference Point (Time (s), Cost (G$))
Gaussian	55.06	0.28	(1000, 10000)
Gauss-Jordan	55.53	0.22	(1200, 22000)
LU	54.48	0.47	(1200, 22000)
Find-Max	54.23	0.14	(2000, 10000)
FFT	55.94	0.32	(1200,22000)
Stencil	57.39	0.31	(1000, 15000)

Table 4: MOABC Hypervolume properties per each workflow. Testbed WWG and EU DataGrid.

Workflows	Average (%)	Standard Deviation	Reference Point (Time (s), Cost (G$))
Gaussian	57.44	0.14	(1000, 10000)
Gauss-Jordan	57.29	0.50	(1200, 22000)
LU	57.74	0.38	(1200, 22000)
Find-Max	56.57	0.12	(2000, 10000)
FFT	57.41	0.43	(1200, 22000)
Stencil	59.61	0.87	(1000, 15000)

Table 5: NSGA II Hypervolume properties per each workflow. Testbed WWG and EU DataGrid.

Workflows	Average (%)	Standard Deviation	Reference Point (Time (s), Cost (G$))
Gaussian	45.89	1.08	(1000, 10000)
Gauss-Jordan	47.13	0.66	(1200, 22000)
LU	48.02	0.70	(1200, 22000)
Find-Max	36.23	1.51	(2000, 10000)
FFT	48.92	0.62	(1200, 22000)
Stencil	46.92	1.12	(1000, 15000)

Table 6: Set Coverage comparison of MOABC, MOGSA and NSGA II per each workflow. Testbed WWG and EU DataGrid.

Coverage A $\geq$ B (%)								
Algorithm		Workflows						Average
A	B	Gaussian	Gauss-Jordan	LU	Find-Max	FFT	Stencil	
MOABC	MOGSA	92.30	94.44	94.44	95.93	95.93	90	93.84
MOGSA	MOABC	0	0	0	0	0	0	0
MOABC	NSGA II	100	100	100	100	100	100	100
NSGA II	MOABC	0	0	0	0	0	0	0
MOGSA	NSGA II	100	100	100	100	100	100	100
NSGAII	MOGSA	0	0	0	0	0	0	0

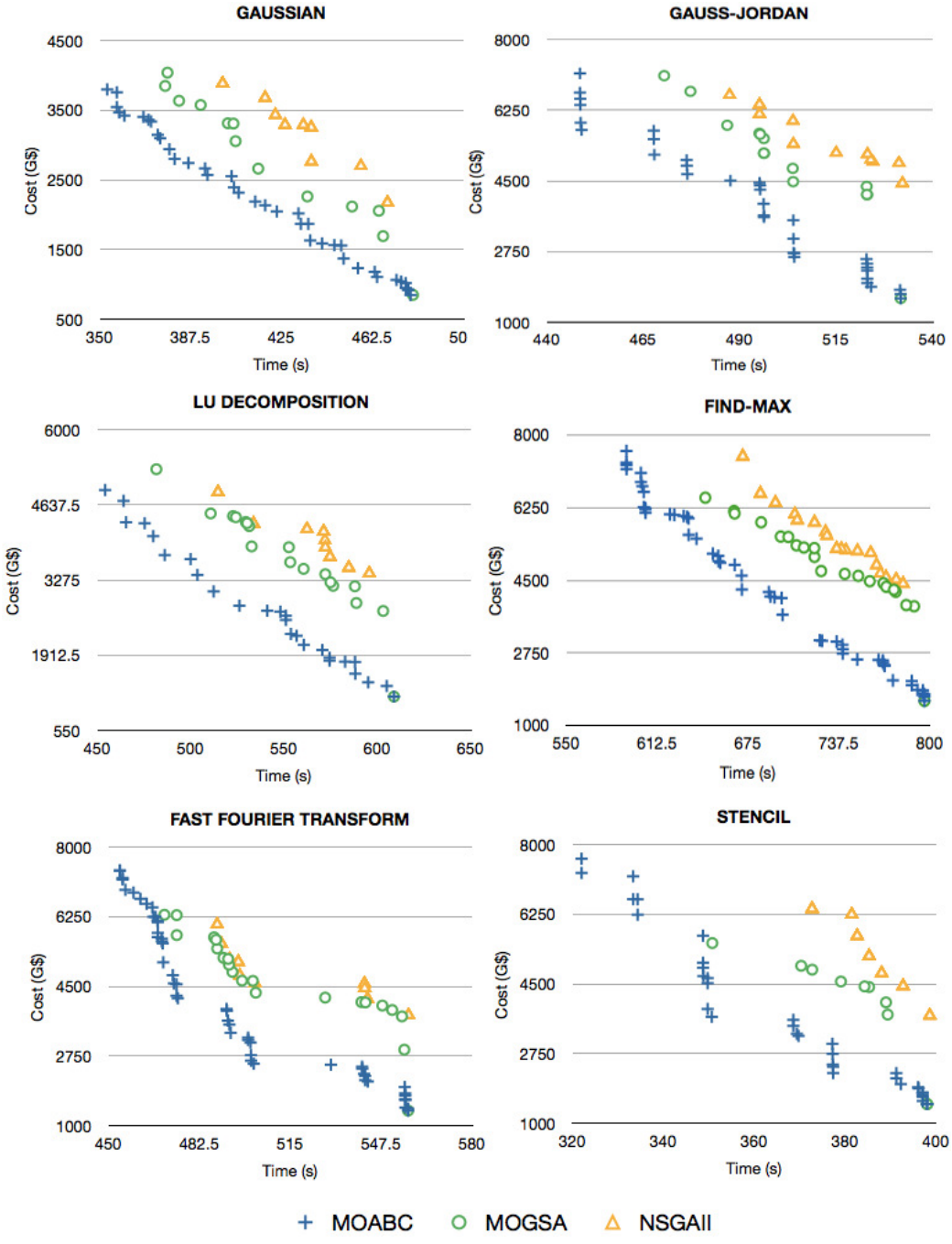

Figure 7: Pareto fronts per workflow and algorithm. Testbed WWG and EU DataGrid.

Table 7: MOGSA Hypervolume properties per each workflow. Testbed IBERGRID.

Workflows	Average (%)	Standard Deviation	Reference Point (Time (s), Cost (¢G$))
Gaussian	56.39	0.50	(65000, 6500000)
Gauss-Jordan	53.03	0.56	(80000, 8000000)
LU	52.98	0.57	(80000, 8000000)
Find-Max	52.91	0.49	(105000, 10500000)
FFT	53.72	0.47	(80000, 8000000)
Stencil	53.62	0.92	(60000, 6000000)

Table 8: MOABC Hypervolume properties per each workflow. Testbed IBERGRID.

Workflows	Average (%)	Standard Deviation	Reference Point (Time (s), Cost (¢G$))
Gaussian	60.69	0.35	(65000, 6500000)
Gauss-Jordan	58.16	1.02	(80000, 8000000)
LU	58.54	0.47	(80000, 8000000)
Find-Max	57.76	0.48	(105000, 10500000)
FFT	58.26	0.47	(80000, 8000000)
Stencil	60.23	0.67	(60000, 6000000)

Table 9: NSGA II Hypervolume properties per each workflow. Testbed IBERGRID.

Workflows	Average (%)	Standard Deviation	Reference Point (Time (s), Cost (¢G$))
Gaussian	44.80	2.40	(65000, 6500000)
Gauss-Jordan	34.39	3.20	(80000, 8000000)
LU	40.27	1.89	(80000, 8000000)
Find-Max	38.42	2.35	(105000, 10500000)
FFT	38.04	2.89	(80000, 8000000)
Stencil	33.88	4.18	(60000, 6000000)

Table 10: Set Coverage comparison of MOABC, MOGSA and NSGA II per each workflow. Testbed IBERGRID.

Coverage A $\geq$ B (%)								
Algorithm		Workflows						Average
A	B	Gaussian	Gauss-Jordan	LU	Find-Max	FFT	Stencil	
MOABC	MOGSA	93.75	87.5	75	81.8	88.88	88.88	85.97
MOGSA	MOABC	0	0	0	3.57	0	0	0.59
MOABC	NSGA II	100	100	100	100	100	100	100
NSGA II	MOABC	0	0	0	0	0	0	0
MOGSA	NSGA II	100	100	100	85.71	100	87.5	95.53
NSGAII	MOGSA	0	0	0	9.09	0	11.11	3.36

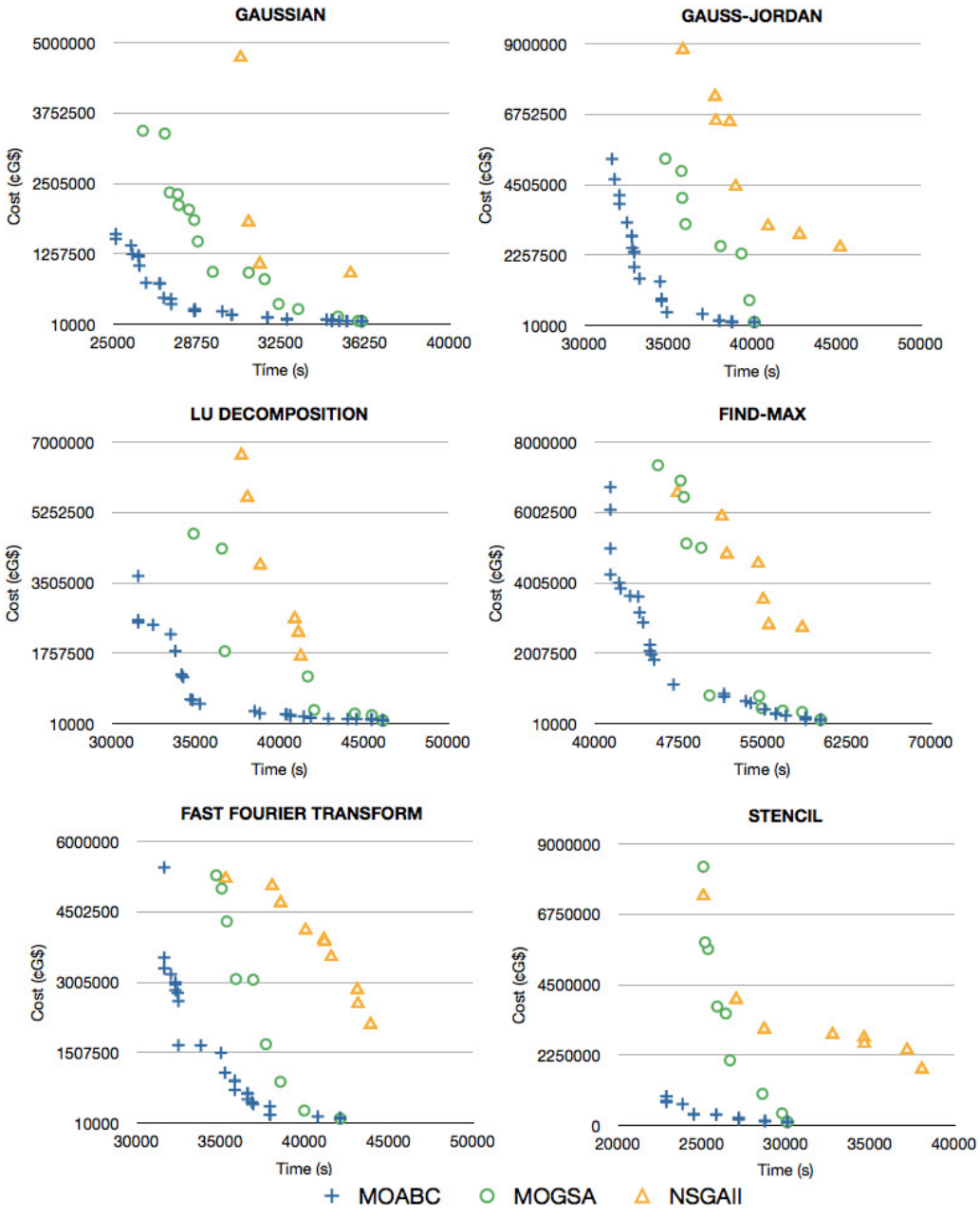

Figure 8: Pareto fronts per workflow and algorithm. Testbed IBERGRID.

Table 11: WMS vs MOABC: Successful executed jobs regard to deadline variation. Testbed WWG and EU DataGrid.

Workflows	Constraint	WMS (Option 1)		WMS (Option 2)		MOABC	
	Deadline	Time	Jobs	Time	Jobs	Time	Jobs
Gaussian	500	482.68	12	442.46	12	478.11	12
	450	455.11	10	442.46	12	449.53	12
	400	401.01	7	368.15	9	394.32	12
Gauss-Jordan	550	534.70	15	492.18	15	531.71	15
	500	428.57	14	492.18	15	496.24	15
	450	428.57	14	394.55	14	448.71	15
LU	600	585.78	13	561.46	14	594.96	14
	550	504.12	10	537.14	13	547.91	14
	500	424.56	9	463.76	10	499.81	14
Find-Max	750	721.42	15	736.29	18	749.87	18
	700	667.32	12	661.98	15	698.48	18
	650	561.18	11	612.12	12	639.30	18
FFT	550	507.13	11	515.58	15	542.62	15
	500	426.51	7	465.76	11	499.82	15
	450	426.51	7	391.51	7	443.91	15
Stencil	390	373.48	10	368.15	12	377.18	12
	360	319.42	8	343.69	10	350.59	12
	330	319.42	8	293.91	8	321.94	12

Table 12: DBC vs MOABC: Successful executed jobs regard to deadline variation. Testbed WWG and EU DataGrid.

Workflows	Constraint	DBC		MOABC	
	Deadline	Time	Jobs	Time	Jobs
Gaussian	500	480.82	12	478.11	12
	450	450.58	10	449.53	12
	400	400.77	9	394.32	12
Gauss-Jordan	550	533.41	15	531.71	15
	500	500.08	14	496.24	15
	450	450.08	14	448.71	15
LU	600	596.66	14	594.96	14
	550	550.00	12	547.91	14
	500	500.28	10	499.81	14
Find-Max	750	750.68	15	749.87	18
	700	700.01	12	698.48	18
	650	650.386	12	639.30	18
FFT	550	544.96	15	542.62	15
	500	500.25	11	499.82	15
	450	450.04	7	443.91	15
Stencil	390	378.98	12	377.18	12
	360	360.60	10	350.59	12
	330	330.70	8	321.94	12

Table 13: WMS vs MOABC: Successful executed jobs regard to deadline variation.Testbed IBERGRID.

Workflows	Constraint	WMS (Option 1)		WMS (Option 2)		MOABC	
	Deadline	Time	Jobs	Time	Jobs	Time	Jobs
Gaussian	35000	32641.82	6	28788.02	12	34751.61	12
	30000	22598.13	5	28788.02	12	29862.43	12
	25000	22598.13	5	23989.00	9	24818.09	12
Gauss-Jordan	40000	33262.57	12	31984.93	15	38759.33	15
	37500	33262.57	12	31984.93	15	37007.08	15
	35000	33262.57	12	31984.93	15	34894.49	15
LU	46000	40174.88	9	36784.05	14	45442.98	14
	38500	32641.82	6	36784.05	14	38488.01	14
	32000	22598.13	5	30385.92	10	31560.51	14
Find-Max	61000	58208.17	11	47977.90	18	60140.90	18
	51500	49892.24	7	47977.98	18	47014.72	18
	42000	33261.53	6	39979.79	12	41376.77	18
FFT	43500	35092.94	15	33584.06	15	42098.00	15
	37750	35092.94	15	33584.06	15	36925.65	15
	32000	31750.25	11	30384.92	11	32030.00	15
Stencil	29500	27619.46	6	23987.93	12	28735.14	12
	26250	20086.46	4	23987.93	12	25894.05	12
	23000	20086.46	4	22388.92	10	22868.49	12

Table 14: DBC vs MOABC: Successful executed jobs regard to deadline variation. Testbed IBERGRID.

Workflows	Constraint	DBC		MOABC	
	Deadline	Time	Jobs	Time	Jobs
Gaussian	35000	35000.68	10	34751.61	12
	30000	30000.92	9	29862.43	12
	25000	25000.27	6	24818.09	12
Gauss-Jordan	40000	38764.00	15	38759.33	15
	37500	37410.48	15	37007.08	15
	35000	35000.23	14	34894.49	15
LU	46000	45445.66	14	45442.98	14
	38500	38500.38	10	38488.01	14
	32000	32000.05	9	31560.51	14
Find-Max	61000	60145.21	18	60140.90	18
	51500	51500.85	12	47014.72	18
	42000	42000.41	11	41376.77	18
FFT	43500	42102.02	15	42098.00	15
	37750	37750.71	11	36925.65	15
	32000	32000.23	7	32030.00	15
Stencil	29500	29072.86	12	28735.14	12
	26250	26250.86	8	25894.05	12
	23000	23000.50	8	22868.49	12