MAGS – An Approach Using Multi-Objective Evolutionary Algorithms for Grid Task Scheduling

Miguel Camelo, Yezid Donoso, Harold Castro
Systems and Computing Engineering Department
Universidad de los Andes, Bogotá, Colombia
ydonoso@uniandes.edu.co

***Abstract*—** Grid task scheduling problem has been a research focus in grid computing for the past years. Some Deterministic, Heuristics or Metaheuristic scheduling approaches have been proposed to solve this NP-complete problem. However, these algorithms do not take the Multi-Objective nature of Grid Computing performance into account. In this paper we present a Multi-Objective approach using Evolutionary Algorithm (MOEA) to efficiently solve such kind of scheduling problems. Our proposal is based on NSGA-II MOEA algorithm combined with a set of Heuristics in different evolutionary operators which allow a fast convergence to optimal (or near-optimal) solutions. The results obtained by our proposed algorithm were compared and evaluated against Mono-Objective and Multi-Objective algorithms used for Grid task scheduling. The main contributions of this paper are the proposed mathematical model, the optimization model and the algorithm to solve it. Additionally we show the effectiveness and robustness of the proposed algorithm.

***Keywords*—** Evolutionary Algorithms, Grid, Task Scheduling, Heuristics, Metaheuristics, Meta-Scheduler, Multi-Objective Optimization, NP-Complete, NSGA-II, Pareto Front.

I. Introduction

GRID Computing technologies, specifically the ones used in High Throughput Computing (HTC) [1] systems, allow a large amount of processing over large periods of time. The key for these infrastructures to offer a high performance is the management and effective exploitation of all available computing resources [2]. Traditionally, tasks scheduling in computing environments has been addressed using mono-objective optimization [3] [4] [5]. Although this approach obtained good results on the performance of a single aspect (e.g. execution time and queue delays minimization), the Multi-Objective (MO) nature of Grid Computing [6] makes it an ineffective approach and necessary to use MO Optimization methods that can capture all the characteristics and requirements of different stakeholders of the Grid infrastructure, obtaining a set of trade-off solutions, all good in some aspect.

Our proposal uses the paradigm of Multi-Objective Evolutionary Algorithms (MOEA). These algorithms combined with a set of evolutionary operators that take advantage of the characteristics of the general operations of Grid Computing, solve efficiently and effectively the task scheduling problem in Grid computing environments, which is a NP-Hard problem [7]. On the other hand, evaluations about our algorithm performance showed that the set of non-dominated solutions found are closed to the Pareto-optimal front and uniformly spaced. Also the algorithm convergence is fast. The true Pareto Front was found using the classical method of ε-constrains MO Optimization and the Branch and Bound deterministic algorithm [7] [8].

The rest of the paper is organized as follows. Section 2 talks about related work and section 3 introduces the task scheduling problem in the Multi-Objective context and presents a general mathematical model of Grid Computing. The section 4 proposes an optimization model for Grid scheduling in HTC with 2 objectives. Section 5 proposes the algorithm based on NSGA-II to solve it. Finally, the set of performance tests where the proposed algorithm is compared against a mono-objective and multi-objective algorithms is presented in Section 6. Section 7 has the conclusions and future work.

II. Related Work

Task scheduling in Grid environments with a single objective has been one of the most extensively studied fields in recent years. Grid use in academic and scientific environment is usually modeled by independent and batch task scheduling techniques. Some examples of such techniques are genetic algorithms [9], data mining [10] or swarm intelligence [11]. Other works have studied scheduling technologies using

economic/market-based models [12]. The main disadvantage of these techniques is the optimization of one goal regardless of the Multi-Purpose Grid nature which leads to improve the performance of one characteristic and causing the deterioration of the others characteristics.

At the level of Grid task scheduling with multiple objectives using MO Optimization, very few papers are presented. In [6] the quality of the different found solutions were compared against the Ant Colony Optimization (ACO), the Particle Swarm Optimization (PSO), the Simulated Annealing (SA) and the Genetic Algorithms (GA) Metaheuristic. The results show that PSO and GA are highly efficient and effective in the task scheduling problem. Later these Metaheuristics were compare (combined with the classic MO Optimization *Weighted Sum* method) against a MOEA algorithm (it was not specified), optimizing the *makespan* and *flowtime* [7] [8] objective functions. Compared to other Metaheuristics, the MOEA algorithm presented better quality solutions, showing that this type of algorithms always converges to the optimal value on each execution. Unlike other Metaheuristics, where on each execution it could provide the optimal value or not, causing deviations between the solutions of different executions. Additionally, authors exposed the virtues of the MOEA to give a set of non-dominated solutions to extend the capabilities of the best solution choosing that fits the specific problem.

In [13], the authors propose an algorithm called Multi-Objective Resource Scheduling Approach - MORSA, which is a combination between NPGA and NSGA Algorithms. They combine the sorting algorithm of non-dominated solutions with the process of Niche Sharing to ensure diversity. An important feature is the incorporation of precedence relationship between tasks, something that was not considered on previous proposals. It uses the execution cost on resources and completion time (flow time) as objective functions. This proposal has the disadvantage of using first-generation MOEA algorithms, although they are better than classical methods of MO optimization, they are less efficient and effective than second-generation MOEA [14].

Finally, another interesting proposal is presented in [15] where the NSGA-II (a Second Generation MOEA) is used as base algorithm. They are aimed at balancing the load between autonomous sites (called Virtual Organizations or VOs [16]) while minimizing the response time of their tasks. Although this solution has the advantages and benefits of using the NSGA-II as base algorithm, authors do not incorporate any knowledge into the algorithm (using fast Heuristics or Metaheuristics), avoiding better results. This assertion is demonstrated in the results of our proposal.

III. Grid Computing as Multi-Objective Optimization Problem

The need to formulate the Grid resource management as a MO problem results from the characteristics of the Grid environment itself. This environment incorporates or knows a set of requirements of different stakeholders groups. Each group has its own point of view and policies that should be considered when evaluating different schedules of tasks from the point of view of different criteria or objectives.

Grid Characteristics can be captured in a formal mathematical model, which allows designing an optimization algorithm for the task scheduling problem. In general, the Grid is composed of one or more Virtual Organizations (VO), which is a set of individuals and/or domains that are governed by a common set of policies for sharing resources. The users objectives/preferences, resource providers and VO administrators (called stakeholders in the literature) are often inconsistent or conflicting with each other. A user is an entity that uses available resources to execute their work computer or any operation that consumes resources for a certain period of time. On the other hand, a resource provider is an entity that manages computational units (simple computers or local management resources systems) inside a single administrative domain. Users and resource providers can be organized dynamically between VOs, each one with different policies and preferences. Finally, a VO manager is the responsible entity for the inter-domain policies maintenance and control to exchange resources and security standards.

Grid Computing requires the use of specialized middleware to hide complexity integration of distributed resources within a domain or a VO. This middleware consists of services such as data management, data collection and monitoring, resource management, task scheduling, authorization, accounting, among others. But the service responsible for ensuring high efficiency in resource management is the service of Resource Management and Task Scheduling. This service is responsible for identifying requirements, matching resources to tasks (for an efficient implementation), submit to the selected resources the tasks executions and monitor them until they finish.

The problem considered in this work consists of a finite set of $|U|$ users $U = \{u_1, u_2, \ldots\ldots, u_{|U|}\}$ that need to send a $|T|$ quantity of task $T = \{t_{u1}, t_{u2}, \ldots\ldots, t_{u|T|}\}$ to resources that are provided by $|PR|$ different resources providers of a finite set $PR = \{pr_1, pr_2, \ldots\ldots, pr_{|PR|}\}$. Each resources provider has associated a set of $|R|$ resources of a finite set $R = \{r_1, r_2, \ldots\ldots, r_{|R|}\}$.

Each resource is described by a set of $|A|$ attributes (or resources features) $A = \{a_1, \ldots., a_{|A|}\}$. Each task contains $|HC|$ hard constraints of a finite set, $HC = \{hc_1, hc_2, \ldots\ldots, hc_{|HC|}\}$. A hard constraint hc_n, can be defined as a relation $\circledast$ between a resource's attribute $(a_{k,j,m}) \in A_{k,j}$ (i.e Operating System, Processor Architecture, CPU speed at least 2GHz) and a task requirement concerning its attribute $(a_{i,m}^{req})$, for each task i and its requirements A_i^{req}, where $i \in T$, $j \in R$, $m \in A$, $n \in HC$ and $k \in PR$.

The relationship is satisfied if $a_{k,j,m}$ matches with $a_{i,m}^{req}$, in other words if the attribute $a_{k,j,m}$ is greater, equal and/or lower than a required value by the attribute $a_{i,m}^{req}$, according to the case. For example, if the attribute $a_{i,m}^{req}$ indicates that it has to be greater than a z value, the relationship $\circledast$ is satisfied if $a_{k,j,m}$ is greater than the z value. Be s (a scheduling) the

assignments set ($\rightarrow$) of resources to execute all tasks $t_{u,i} \rightarrow r_{k,j}$, it is said that the assignment is feasible if $\forall hc \in HC$ meets the relationship $a_{i,m}^{req} \circledast a_{k,j,m}$. Additionally it can be set UR^{req} as the required resources quantity for a task (i.e. a 1GB Disk space or 4 CPUs).

It is important to verify that in the filter stage (compared with the General Architecture for Scheduling in Grid [17]) exists a strong restriction stating that the task of a user must have access rights to perform an operation (i.e. a task execution) over a resource from a resource provider. The permissions relationship of a user to access and use a resource (identified by $\circledcirc$) can be defined as: if $t_{u,i} \rightarrow r_{k,j} \Longrightarrow t_{u,i} \circledcirc r_{k,j}$, meaning assigning a resource to a task is valid if and only if the task owner has access and rights to use the allocated resource.

Given the definitions, notations and considerations that describe tasks scheduling problem in Grid environments, ***the Multi-Objective problem formal definition is:***

$$Minimization\ (or\ maximization)\ of$$

$$\boldsymbol{F}(s) = \left(f_1(s), f_2(s), \dots, f_{|fo|}(s)\right) \quad (1)$$

$$\boldsymbol{subject\ to}$$

$$a_{i,m}^{ref} \circledast a_{k,j,m}\ \forall A_{k,j \in a_{k,j,m}} \quad (2)$$

$$\forall_{k,j}\left(\sum_{i:(t_i \rightarrow tpr_{k,j}) \in s} ur_{i,p}^{req}\right) \leq ur_{k,j,p} \quad (3)$$

$$\forall t_{u,i} \rightarrow r_{k,j} \Longrightarrow t_{u,i} \circledcirc r_{k,j} \quad (4)$$

$$\boldsymbol{Where}$$

$$a_{k,j,m} \in A_{k,j},\ \ a_{i,m}^{ref} \in A_k,\ m = 1, \dots, |A| \quad (5)$$

$$\forall_{(t_{u,i} \rightarrow r_{k,j}) \in s} t_i \in ET(r_{k,j}) \quad (6)$$

$$s = \{t_1 \rightarrow r_{k,j}, t_2 \rightarrow r_{k,j}, \dots\dots\dots\dots, t_{|T|} \rightarrow r_{k,j}\},\ s \in \boldsymbol{S} \quad (7)$$

$$i \in \{1, \dots, T\},\ j \in \{1, \dots, R\},\ m \in \{1, \dots, A\},\ n \in \{1, \dots, RD\} \text{ and } k \in \{1, \dots, PR\} \quad (8)$$

Equation (2) indicates that all the attributes requirements of a task have to be satisfied under hard constraints. Equation (3) ensures that available resources meet the resource requirements for a task and equation (4) is the access rights constrain of a user to resources to execute their tasks. Equations (5) and (6) show the sets and attributes used in the mathematical model and equation (7) indicates a feasible allocation (s) of tasks to resources, which should be part of the set (*S*) of all feasible solutions that satisfy the problem constraints and meet the definition of the Pareto Front.

IV. Optimization Math Model for High Throughput Computing

Since there are multiple types of Grid, we must define an optimization model for a specific instance of Grid. The mathematical model's features of the MO scheduler that support MO task scheduling are defined based on the HPC Job Scheduling: Base Case and Common Cases of the information paper OGF GFD-I.100 [18], widely referred to as the High Throughput Computing (HTC) core. It is important to recognize that the tasks scheduler in batch tasks is only a part of the broad spectrum that represents management services in Grid Computing. This model can be extended to more objectives, other types of tasks and adding / removing constrains.

The characteristics assumed in the optimization model for task scheduling are presented below:

- A job is composed of a task and does not have a priority.
- The Task Scheduling will be on batch job.
- Each job can only be assigned to a single resource and this is executed exclusively.
- Resources and jobs are heterogeneous and are part of a finite set.
- Resources are discrete, however the attributes described above may have different domains.
- Pre-emption is not allowed.
- The time of application provisioning network delays, or any other time different from the processing time of a job is not considered. This is due to HTC Grid is only for computer processing.
- The resource to be scheduling is the amount of Millions of Instructions Per Second (MIPS) of the processor and the amount of RAM available in Giga Bytes.
- Two Objectives to optimize.

Additionally, we define a Grid task scheduling problem instance like a pair *(n,m)* where *n* and *m* indicate the total number of tasks and resources in the scheduling problem respectively. Given these characteristics and constraints the table 1 presents the set of variables and parameters that allow the creation of a mathematical optimization program for the Grid Scheduling Problem in HTC environments on Grid.

Variable	Description
x_{ijk}	Binary decision variable (0,1) that indicates whether the i task is executed as the k^{th} in the j^{th} processor
Parameter	**Description**
C_i	Time Completion of i task
p_i	Amount of processing requested by the task i in MIPS
s_j	Amount of processing offered by the j resource in MIPS
p_{ijk}	Total execution time of the i task which is executed as the k^{th} in the j^{th} processor (p_i/s_j).
M_i	Amount of memory requested by the i task in GBytes
N_j	Amount of memory offered by the j resource in GBytes

Table 1. Instances of the Problem

A. Objective Function

According to [18], the relevant objective function at Grid can be classified according to the stakeholders involved in the Grid (users, resource providers and administrators of the VO). For the model, we selected two of these functions to observe their behavior in the performance tests of the Multi-Objective Algorithm: load balancing (makespan minimization) and task

completion time minimization (flowtime minimization).

1) Load Balancing (minimize makespan)

To do load balancing in Grid Scheduling is necessary to assure that the finish time of the last job is the same on each resource. Minimizing completion time of the system's last task, also called makespan (C_{max}), leads to optimal load balancing. The total completion time of last task in a resource is described as follows:

$$C_j = \sum_{i=1}^{n}\sum_{k=1}^{n} k * p_{ijk} * x_{ijk} \quad j = 1,2,\dots,m \quad (9)$$

If $C_{max} = \max\{C_1, \dots, C_{|m|}\}$, then the objective function to load balancing is:

$$f(x)_1 = \text{minimizar } C_{max} \quad (10)$$

2) Minimize the total completion time of all task (minimize flow-time)

A user on the grid wants the execution of their tasks to be quick. Reducing the completion time of tasks generally involves a greater use of more powerful machines and shorter time in processor queues but this behavior leads to load imbalance between resources. It can be formally defined as:

$$\sum_{j=1}^{m} C_i = \sum_{i=1}^{n}\sum_{i=1}^{n} k * p_{ijk} * x_{ijk} \quad (11)$$

Evidently the reduction of this value is desired by the Grid users. Hence it is possible to define the second objective function as follows:

$$f_2(x) = \text{minimizar} \sum_{j=1}^{m} C_j \quad (12)$$

From the complexity point of view, it can be demonstrated that in the resulting combinatorial problem of Task Scheduling, the number of possible solutions that can be found given the characteristics of this problem is exponential $O(n^m)$ and in the worst case scenario a deterministic algorithm must check n^m solutions.

B. Model Constrains

Finally, given the set of objective functions, the set of constraints in the mathematical programming model to solve the multi-objective problem are:

$$\sum_{j=1}^{m}\sum_{k=1}^{n} x_{ijk} = 1 \ \forall i \in T \quad (13)$$

$$\sum_{i=1}^{n} x_{ijk} \leq 1 \ \forall j \in R, k \in T \quad (14)$$

$$C_{max} - \sum_{i=1}^{n}\sum_{k=1}^{n} p_{ijk} * x_{ijk} \geq 0 \ \forall j \in R \quad (15)$$

$$x_{ijk} \leq \frac{s_j}{p_i} \quad (16)$$

$$x_{ijk} \leq \frac{N_j}{M_i} \quad (17)$$

$$x_{ijk} \in \{0,1\} \quad (18)$$

Where R={1,2,...,m} is the set of available resources in the system and T={1,2,...,n} is the set of tasks in the batch.

The constraint (12) ensures that task *i* is assigned to one and only one position. The constraint (14) ensures that each position (j,k) has at most one task assigned to it. Since the C_{max} basically is a decision variable and not an element of the resource vector of optimization model, the constraint (15) ensures that the amount of processing on each machine is less than C_{max}. Constraints (16) and (17) are hard constraints on the model which ensure that the requirements on the amount of memory and processor request for a certain task are met. The constraint (18) forces the decision variable to be 0 or 1, as was described in Table 1.

V. MAGS: A NSGA-II BASED ALGORITHMS FOR GRID TASK SCHEDULING

In year 2000, Kalyanmoy Deb *et. al.* in [19] presented the Non-dominated Sorting Genetic Algorithm II (NSGA-II), which is a review of the NSGA algorithm of MOEA's first generation. NSGA-II allows complexity reduction, incorporates an elitism operator and eliminates the parameters on the diversity operator, allowing for greater transparency in the algorithm.

A. Evolutionary Algorithm Operators Configuration

The process of convergence of the solutions set to the Pareto Front is highly linked to the use of evolutionary operators. Now, the configuration parameters and operators designed for the NSGA-II applied to the Task Scheduling Problem are described. The algorithm 1 shows NSGA-II pseudo code applied in task scheduling and Figure 1 show the algorithm's operation.

Algorithm 1: NSGA-II Pseudo code for Task Scheduling.

```
Input: A maximum number of generations max_gen
Output: A population evolved
begin
    t = 0;
    Initialize population P_t with size N- Random,G.A,Min Min;
    Evaluate population according to the objective functions;
    while t < max_gen do
        Apply Binary tournament selection, crossover and mutation to
        P_t, to generate Q_t;
        R_t = P_t ∪ Q_t;
        Assign a hierarchy based on Pareto dominance to R_t;
        Assign crowding distance to R_t;
        Select N individuals of R_t, according to the crowding
        comparison operator to generate P_{t+1};
        t = t + 1;
    end
end
```

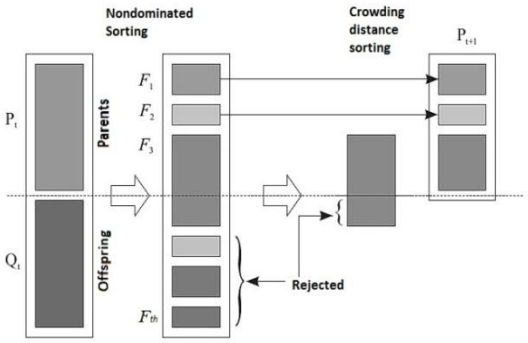

Fig. 1 NSGA-II Operation

1) Chromosome

The chromosome encoding is done indirectly. Given a chromosome of length *n*, the position of each gene in the chromosome is the task number identifier and the allele value (value the gene can take) *m* is the resource to which the task is assigned. If n is the number of tasks to be scheduled in the batch and m the amount of resources available, an instance (m, n) with $n = 6$ and $m = 3$ indicates that the chromosome that represents a solution is encoded as a vector of six positions, where each allele can take the values 1, 2 and 3, since we have 3 resources.

2) Comparison and Selection

The crowding comparison operator is used to choose the solution set of the last front population trying to enter the next generation population. Binary tournament operator is used into the creation of child population.

3) Crossover

The crossover operation can extend the search space by generating new solutions from the existing ones. Evaluating the multipoint crossover strategies it was found that they are highly destructive, generating highly random solutions that can cause the algorithm to get stuck in a local optimum area, giving low convergence to the Pareto front values. The selected crossover strategy is the single point crossover [14], which showed a good performance to solve this problem.

4) Mutation

The mutation operation provides Metaheuristics with local search functions and solution diversity. The mutation process acts in three different ways:

a) ***Fast Load Balancing*** between the most loaded (greater makespan) and the least loaded (less makespan). This Heuristic selects the most loaded resource and the less loaded of a chromosome and exchanges two task.
b) ***Random migration of tasks between resources.*** Given a chromosome, two genes are randomly selected and the value of their alleles is exchanged.
c) ***Min-Min strategy.*** Given a random point in the chromosome, the Min-Min Heuristics generate the remaining values of the chromosome.
d) ***Incorporation of new random population.*** Incorporates new solutions in a random fashion to extend the search space which must meet the problem constrains.

The strategy of mutation has a low complexity and increases the capabilities of the NSGA-II to find the real Pareto Front. The procedures a), b) and c) apply to (90%) of the individuals to be mutated and the remaining (10%) are eliminated and replaced by the procedure d).

5) Initial Population

The initial population in our algorithm is mostly randomly generated meeting the problem constraints, but maintaining a low complexity in the algorithm. On the other hand, a few individuals are generated with fast Metaheuristics/Heuristics, enabling greater convergence and higher speed of the algorithm.

The proposed initial Population uses a size of 100 individuals, where 95 (95%) are randomly generated, and of the remaining 5 (5%), four are generated by genetic algorithms (G.A) where one of the individuals minimizes the makespan and the other minimizes the flowtime. Finally, the last individual is generated by the Min-Min Heuristic.

6) Completion Criteria

The termination criterion used is the number of maximum generations, which was set at 500. This value was experimentally chosen because any increase upon this value, offered no further improvements in the solutions.

VI. Evaluations and Results

A. Mono-Objective Scenario

This evaluation presents important result about optimal values found in single objective optimization. We use an algorithm (Branch and Bound), and Heuristic (Min-Min algorithm) and a Metaheuristic (Genetic Algorithms) to find optimal values and to compare against MAGS extreme solutions. The Min-Min Heuristic [3] [4] and G.A. Metaheuristics presented in [6] good results in Grid Task Scheduling Mono-Objective problems.

The Mono-Objective evaluation presented in this part takes three instances of the problem into account (Table 2): a small (3,13), a medium (5,100) and a large (50,300) search space. The relaxation used in the Branch and Bound is solved by the dual simplex method [20]. The standard deviation for G.A. Metaheuristics and the MOEA was calculated after 20 algorithm executions. The relative error is computed respect to the optimum value found in Branch and Bound, which is the exact method and serves as a theoretical reference. In the intractable instances for exactly algorithms the reference was the MAGS results.

Instance	Task (n)	Resources (m)	Space Solutions (Total Solutions)
Small	13	3	1594323
Medium	100	5	7,88861E+69
Large	300	50	4.9E+509

Table 2. Instances of the Problem

1) Algorithms Configurations

Below we present the Mono-Objective algorithm configurations.

- **B&B:** This optimization strategy use Simplex Dual Algorithm to find lower bounds and Pure Depth First Heuristic to find integer optimal solution.
- **Min-Min:** Its configuration is based on algorithm present in [3] and [4].
- **Genetic Algorithm:** Table 3 presents the algorithms parameter configuration.
- **MAGS:** Table 4 presents the algorithms parameter configuration.

2) Results for Mono-Objective Evaluation

The evaluations shown in table 5,6 and 7 and figures 2, 3 and 4 show the quality of the solutions that MOEA finds incorporating knowledge of the Task Scheduling problem into evolutionary operators. In the instance (3,13), the Min-Min Heuristic showed a poor performance for both objective functions, but particularly in the makespan, showing a relative error greater than 70%. Genetic algorithms show very good results with relative errors less than 2%. On the other hand, the MOEA obtains the expected optimal values. The above statement is supported by having a relative error of 0. Another important result of the MOEA was its standard deviation, which was zero. Indicating that in each execution of the algorithm, it always found the global optimum of the objective functions.

Options	Value
Initial Population	100
Initial Population Creation	Random with bound constrains
Generation Number	500
Crossover	Single-Point
Crossover Probability	80%
Mutation	Uniform
Probability Mutation	20%
Selection	Binary Tournament
Total Executions	20

Table 3. G.A. parameters in Mono-Objective evaluation

Options	Value
Initial Population	100
Initial Population Creation	Random meet the bound constrains , Min-Min Heuristic (2 Individual) and G.A. Metaheuristic (3 Individuals)
Generation Number	500
Crossover	Single-Point
Crossover Probability	80%
Mutation	Fast Balancing, Fast Migration, Min-Min, Random Population
Probability Mutation	20%
Selection	Binary Tournament
Elitist Operator	Crowding Distance
Total Executions	20

Table 4. MAGS parameters in Mono-Objective evaluation

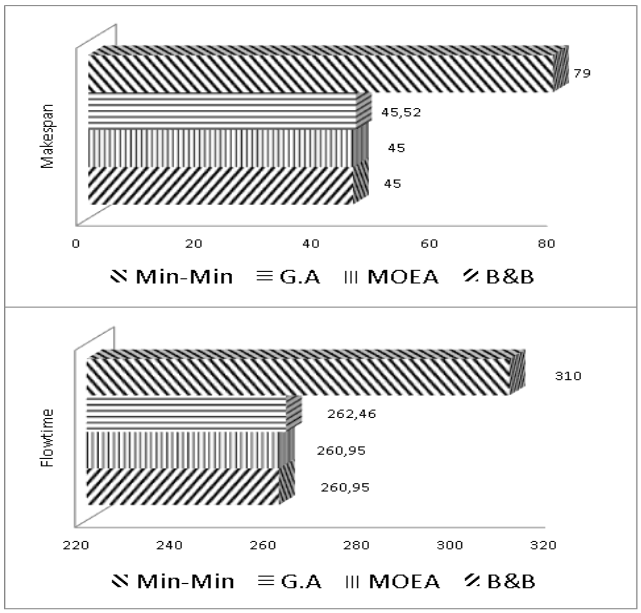

Fig. 2 Results of Instance (3,13)

Algorithm	Objective Function	Optimal Value	Standard Deviation	Relative Error (%)
B&B	**makespan**	45	------	------
	flowtime	260,95	------	------
G.A	**makespan**	45,52	0,28	1,155
	flowtime	262,46	0,69	0,578
Min-Min	**makespan**	79	------	75,55
	flowtime	310	------	18,79
MAGS	**makespan**	45	0	0
	flowtime	260,95	0	0

Table 5. Optimization results in the instance (3,13)

In the results of the instance (5,100), again the Min-Min algorithm presents low quality results, which are above 50% over the actual value of the makespan, showing its inefficiency to minimize this function. The opposite happens with the results obtained in the minimization of flowtime, which are acceptable since they are below 10% of the theoretical value. The G.A. continues to present difficulties in optimizing the makespan, with a relative error greater than 20% in contrast to the minimization of flowtime which is maintained below 1% compared to the theoretical optimum value obtained by the Branch and Bound. The G.A. shows high standard deviations especially in the makespan, indicating a low certainty as to find the optimum value in each run of the algorithm.

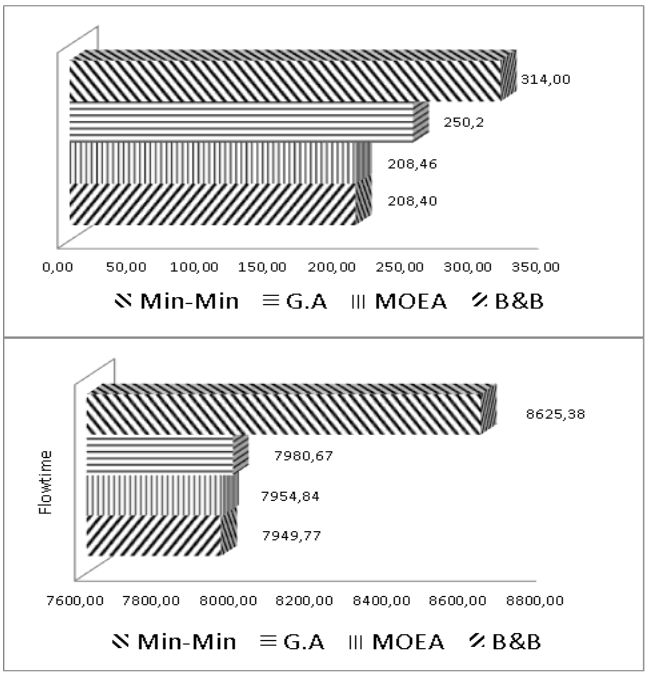

Fig. 3 Results of Instance (5,100)

In the instance (50-300), the complexity generated by the mathematical model and the computational complexity of the size of the search space, prevented the exact method from being executed. For this reason, the standard deviation for the Min-Min and G.A. was calculated with respect to the values found by the MOEA.

Algorithm	Objective Function	Optimal Value	Standard Deviation	Relative Error (%)
B&B	**makespan**	208,4	------	------
	flowtime	7949,77	------	------
G.A	**makespan**	250,2	13,71	20,05
	flowtime	7980,66	8,02	0,38
Min-Min	**makespan**	314	------	50,67
	flowtime	8625,37	------	8,498
MAGS	**makespan**	208,46	0,052	0,029
	flowtime	7954,83	1,15	0,06

Table 6. Optimization results in the instance (5,100)

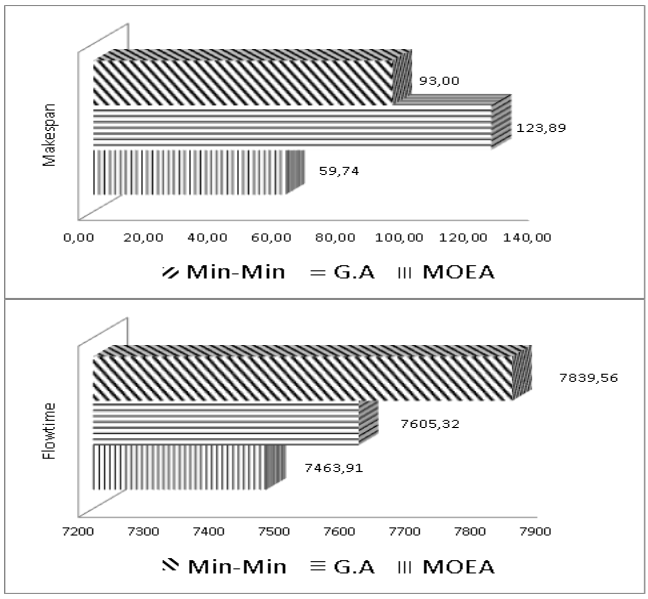

Fig. 4 Results of Instance (50,300)

Algorithm	Objective Function	Optimal Value	Standard Deviation	Relative Error (%)
B&B	**makespan**	Intractable	------	------
	flowtime	Intractable	------	------
G.A	**makespan**	123,89	14,85	107,39
	flowtime	7605,32	16,21	1,89
Min-Min	**makespan**	7839,55	------	55,68
	flowtime	93	------	5,03
MAGS	**makespan**	59,73	0,44	------
	flowtime	7463,90	4,1872	------

Table 7. Optimization results in the instance (50,300)

Continuing with the analysis of the instance (50,300), the G.A. had a terrible level of convergence in the makespan and its standard deviation showed that it happened in most of its algorithm executions. The Min-Min continues with its tendency to be close to 50% from the optimal makespan generated by the MOEA. Flowtime optimal values, found by the GA and Min-Min, are still good values (assuming that being below 5% is a good optimum value). It is important to acknowledge that a value less than 5% in the solution can be classified as good or very good, given that the magnitude of flowtime values that are close to 7900 seconds, and an error below 5% means a deviation less than 300 seconds from the optimum. The standard deviation of the GA rises, again showing problems of convergence towards the optimum in each algorithm execution.

B. Multi-Objective Scenario

One of the best ways to measure the quality of a Multi-Objective algorithm is using metrics to evaluate it against theoretical solutions of the models. In this work, we built a real Pareto Front evaluating the mathematical problem with the method of ε-constrains and using the deterministic/exact algorithm of Branch and Bound.

The obtain results allow us first to use the metric of

generational distance - *GD* (presented in [14]) to evaluate the Pareto front convergence found with our proposal against the real Pareto front found by the exact algorithm. On the other hand, it is necessary to evaluate the ability to generate well-distributed solutions through the real Pareto, so the metric spacing - *(S)* (presented in [14]) is use. Here are the formulas that relate them respectively.

$$\mathbf{DG} = \frac{\left(\sum_{i=1}^{|Q|} d_i^p\right)^{1/p}}{|Q|} \quad (19) \qquad \boldsymbol{S} = \sqrt{\frac{1}{|\boldsymbol{Q}|}\sum_{i=1}^{|\boldsymbol{Q}|}(\boldsymbol{d_i} - \bar{\boldsymbol{d}})^2} \quad (20)$$

The algorithm was run 20 times and on each run the GD and S metrics were obtained. Subsequently their average and standard deviation are obtained. The ε-constrains method configuration parameters are presented in table 8.

Options	Value
Integer Problem Relaxation Solution	Simplex-Dual
Search Heuristic	Pure Depth First
Quality Integer solution for Acceptance	0.05%
Max Time Exploration for Solution	3600 seconds
ε values calculated	30

Table 8. ε-constrains method configuration parameters.

1) Pareto Front, Spacing and GD Results

In this evaluation, three instances of the problem were used to calculate the real Pareto Front and the one generated by our proposal. Being one instance a pair like (m, n), where *m* is the number of machines and *n* the number of tasks. Then the instances used for evaluation were: small instance (3,13), which generates a search space of 1,594,323 solutions; medium instance (5,100), which generates a space of $7.88x10^{69}$ solutions and a big instance (50,300) with a search space of $4.9x10^{509}$ solutions. The obtained Pareto Fronts are presented in the figures 5, 6 and 7 and spacing and generational distance values in table 9.

The instance's (3,13) results show that all the points of the real Pareto were found by the MOEA, thus when computing its spacing and the generational distance, the values are below the unity because of the precision on the decimal values of the exact algorithm. These results confirm the convergence of the algorithm towards the real Pareto front. Furthermore, both the generational distance and the spacing held a low standard deviation, which shows that the algorithm converges always in every execution.

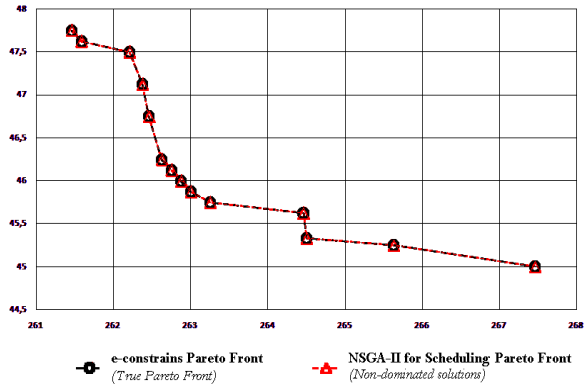

Fig. 5 Pareto Front in the instance (3,13)

In the (5,100) instance, the algorithm's efficiency was demonstrated again, even in bigger search spaces. The calculated metrics show the high quality and diversity of the Pareto's front non-dominated solutions found by the proposed algorithm. Although the spacing and the GD both increased, they are still low in scale of the objective functions.

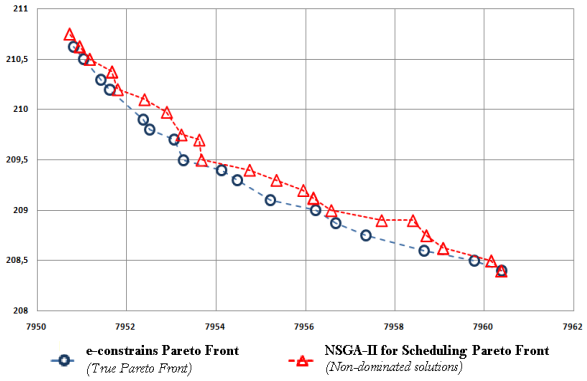

Fig. 6 Pareto Front in the instance (5,100)

Finally, in the 50-300 instance (large-scale combinatory problem), the program used to calculate the optimum values of the real Pareto did not work. The space and computational complexity of the mathematical programming model turned the deterministic algorithm of the classic method useless. The calculated spacing and its deviation showed that through each of the algorithm's execution, the Pareto's distribution almost always stayed constant. This indicates that the generated fronts on each execution were similar one to another, meaning convergence to the same optimum values.

Instance	Metric	Average	Standard deviation
(3,13)	GD	0,091	0,038
	Spacing	1.42	0.45
(5,100)	GD	1,29	0,75
	Spacing	1,44	1,18
(50,300)	GD	No calculated because ε-constrains method don't work (the memory RAM was exceeded)	
	Spacing	1,787	1,3217

Table 9. Generational Distance and Spacing metrics results

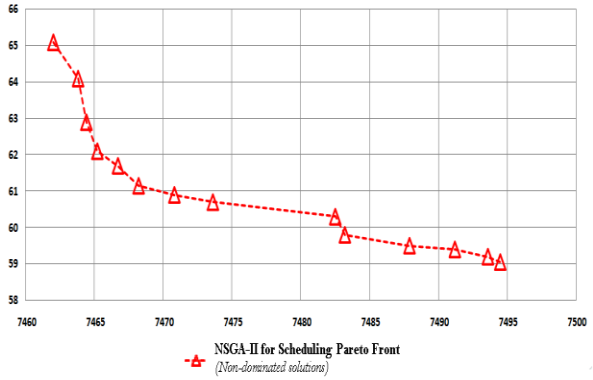

Fig. 7 Pareto Front in the instance (50,300)

2) Complexity Analysis

The computational complexity of NSGA-II algorithm is polynomial and equal to $O(MN^2)$ for each G generation, where M is the number of objective functions and N the individuals in the population, meaning its implementation complexity is (GMN^2) and this is independent of the number of resources and task. On the other hand, Branch and Bound, has a complexity equal to fully evaluate the search space of combinatorial problem and it is concluded that this algorithm is in the worst case scenario of exponential complexity $O(m^n)$, where m is the number of resources and n is the number of tasks. In a practical way, the execution times of our proposal and of the traditional method were measured. The table 10 shows the practical results of the execution time in both algorithms.

Instance (n,m)	MAGS - NSGA-II for Grid Scheduling (seconds)	ε-constrains using Branch and Bound (seconds)
3-13	10.31	>>1
10-50	32,36	62,5
5-100	37.88	147,8
20-200	49,27	<<3600
50-300	94,28	Intractable

Table 10. Execution Time of NSGA-II and Branch and Bound

This table shows the inefficiency of the deterministic algorithms in NP-Complete problems. In a small problem instance, the difference, in terms of execution time elapsed, between algorithms is low. However, in large problem instance the deterministic algorithm have very high execution therefore the optimization results should be obtained as an integer gap value respect to optimization problem relaxed solution. In the last instance (50,300), the deterministic algorithm does not finalize because its spatial complexity is beyond the capabilities of the machine used for assessments (Dual Core Xeon Processor and 4 GB of RAM).

VII. Conclusions and Future Work

The proposed MOEA showed the best results, demonstrating that the use of elitism operator carries the best individuals to the next generation by allowing the genetic material of these individuals to be used to create new members of the population through evolutionary operators, expecting that these new individuals are fitter than their predecessors. Additionally, the mutation operator designed in this proposal (which contains knowledge of the problem) coupled with the choice of algorithm parameters and the other operators through pre-evaluations enabled the construction of an algorithm that allows convergence to the real Pareto Front efficiently. It is also observed that even though each instance increased the complexity of the search space, convergence was held constant and the relative error remained low.

The Mono-Objective evaluations we showed that proposed MOEA is the excellent convergence to the optimum on every execution demonstrated by a low standard deviation. It is noteworthy that solutions far from the real optimal values, can generate a front of local non-dominated solutions (local optimum), avoiding the convergence to the global optimum. Additionally we showed that in large-scale task scheduling (a NP-Complete Problem) deterministic algorithms are highly inefficient and is necessary to use Metaheuristics as the NSGA-II, whose polinomial complexity allows us to find good solutions in a reasonable time.

On the other side, Multi-Objective evaluation in the medium-large scale size Grid scheduling problem (5-100 y 50-300) showed the difficulties suffered by the traditional deterministic-algorithm method when trying to find the exact Pareto front (because of its computational and spatial complexity) were noted. In the 5-100 instance, the computational complexity caused the Branch and Bound algorithm's execution to be very slow to find the ε values used to generate the real Pareto front using the MO classic method but the space complexity was reduced by means of using efficient algorithms like the dual-simplex, the Pure Depth First search Heuristic and the use of sparse matrixes, which permit the manipulation of millions of variables to solve medium-scale mathematical programming problems. Nonetheless although these solutions are acceptable in medium-scale problems, they are either limited or insufficient in large-scale problems. This was seen in the 50-300 instance where the exact algorithm failed its execution because of a need for more memory (above 4GB of RAM) when loading the restriction matrix. It is important to note that the proposed algorithm finds a set of optimal solutions (close to the Optimal Pareto Front). The charts presented and supported by the GD and spacing metrics, showed that the use of Metaheuristics is highly recommended to tackle Multi-Objective task scheduling problems which are of combinatory nature and generate a large scale and/or non-convex search space.

It is worthwhile to remember that HTC's task scheduling Problem includes scenarios of hundreds of thousands of resources and task that must be schedule quickly, where deterministic algorithms solutions cannot be used and therefore, it necessary to use alternatives such as the one suggested here, which showed to be efficient and effective when tackling the problem. As a future work, it is proposed to compare the algorithm with other MOEA on large-scale instances. Furthermore, it is also suggested to begin evaluating other evolutionary operators and incorporating new Heuristics on the mutation operator to improve a faster the algorithm's convergence to the real Pareto.

REFERENCES

[1] Condor - High Throughput Computing (HTC). [Online]. http://www.cs.wisc.edu/condor/htc.html

[2] Zhou Lei and Zhifeng, Allen, Gabrielle Yun, "Grid Resource Allocation," in *Grid Computing: Infraestructure, Service, and Applications*, Lizhe Wang, Wei Jie, and Jinjun Chen, Eds. Boca Raton: CRC Press, 2009, ch. 7, pp. 1172-188.

[3] Tracy D. Braun et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," in *Journal of Parallel and Distributed Computing*.: Academic Press, Inc, 2001, vol. 61, pp. 810 - 837.

[4] Hesam Izakian, Ajith Abraham, and Václav Snasel, "Comparison of Heuristics for Scheduling Independent Tasks on Heterogeneous Distributed Enviroments," vol. 1, pp. 8-12, 2009.

[5] Tracy D. Braun et al., "A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing systems," in *Eighth Heterogeneous Computing Workshop, 1999. (HCW '99) Proceedings*.: IEEE Computer Society, 1999, pp. 15-29.

[6] Fatos Xhafa and Ajith Abraham, *Metaheuristics for Scheduling in Distributed Computing Environments*.: Springer-Verlag Berlin Heidelberg, 2008, pp. 1-38, 247-272.

[7] Miguel L. Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Fifth Edition ed.: Springer, 2008.

[8] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt, and J. Weglarz, "Handbook on Scheduling: From Theory to Applications (International Handbooks on Information Systems)," in *International Handbook on Information Systems*, Primera Edición ed.: Springer, 2007, pp. 137-190, 271-311.

[9] A. Sanchez et al., *A dynamic-balanced scheduler for genetic algorithms for grid computing*, 1st ed.: WSEAS Transactions on Computers , 2009, vol. 8.

[10] Liu Meiqun, Gao Kun, and Wan Zhong, *A novel architecture for data mining grid scheduler*, 1st ed.: WSEAS Transactions on Systems, 2008, vol. 7.

[11] Da-Zhen Wang, Jun-Shan Zhan, Fang Wan, and Lei Zhu, *A Dynamic Task Scheduling Algorithm in Grid Environment*, 7th ed.: WSEAS Transaction on Computer, 2006, vol. 7.

[12] Massimiliano Caramia and Stefano Giordani, *Resource allocation in grid computing: an economic model*, 1st ed.: WSEAS Transactions on Computer Research, 2008, vol. 3, pp. 19-27.

[13] Guangchang Ye, Ruonan Rao, and Minglu Li, *A Multiobjective Resources Scheduling Approach Based on Genetic Algorithms in Grid Environment*. Hunan, China: Fifth International Conference on Grid and Cooperative Computing Workshops, 2006.

[14] Kalyanmoy Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. New York: John Wiley & Sons, Ltd., 2001.

[15] Christian Grimme, Joachim Lepping, and Alexander Papaspyrou, "Discovering Performance Bounds for Grid Scheduling by using Evolutionary Multiobjective Optimization," in *Proceedings of the 10th annual conference on Genetic and evolutionary computation*. Atlanta, GA, USA: ACM, 2008, pp. 1491-1498.

[16] Ian T. Foster, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," in *Proceedings of the 7th International Euro-Par Conference Manchester on Parallel Processing*.: Springer-Verlag, 2001, vol. 2150/2001, pp. 1-4.

[17] Open Grid Forum - Group Scheduling Working. (2001) Ten Actions When SuperScheduling. [Online]. http://www.ogf.org/documents/GFD.4.pdf

[18] OGSA HPC Profile WG. (2006) HPC Job Scheduling: Base Case and Common Cases. [Online]. http://www.ogf.org/documents/GFD.100.pdf

[19] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, "A fast elitist non-dominated sorting genetic algorithm for multi-objective optimisation: NSGA-II," *PPSN VI: Proceedings of the 6th International Conference on Parallel Problem Solving from Nature*, pp. 849–858, 2000.

[20] Mokhtars S. Bazaraa and John J. Jarvis, *Linear Programming and Network Flows*.: John Wiley & Sons, Inc., 1977, pp. 279-286.