Multi-objective Robust Static Mapping of Independent Tasks on Grids

Bernabé Dorronsoro, Pascal Bouvry, J. Alberto Cañero,
Anthony A. Maciejewski, *Fellow, IEEE*, and Howard Jay Siegel, *Fellow, IEEE*

***Abstract*— We study the problem of efficiently allocating incoming independent tasks onto the resources of a Grid system. Typically, it is assumed that the estimated time to compute each task on every machine is known. We are making the same assumption in this work, but we allow the existence of inaccuracies in these values. Our schedule will be robust versus such inaccuracies, ensuring that even when the estimated time to compute all the tasks is increased by a given percentage, the makespan of the schedule (i.e., the time when the last machine finishes its tasks) will not grow behind that percentage. We propose a new multi-objective definition of the problem, optimizing at the same time the makespan of the schedule and its robustness. Four well-known multi-objective evolutionary algorithms are used to find competitive results to the new problem. Finally, a new population initialization method for scheduling problems is proposed, leading to more efficient and accurate algorithms.**

I. Introduction

Task scheduling on heterogeneous parallel computing environments (such as *Grids* [1]) has been a very active research field in computer science, especially during the last few years. The reason is that more and more complex problems arise everyday, and in many cases they can only be tackled through parallel computing techniques. Due to this demand, both Grids and high performance computing centers are becoming more and more popular. Consequently, there is a clear need for efficiently scheduling the tasks to compute in these parallel systems. Typically, there are several goals that are required to be optimized by a scheduling algorithm, such as the *makespan* (finishing time of the last task), the QoS of the system (e.g., its response time), energy efficiency issues (such as the energy used by the resources to complete the tasks), the user benefit, and the resource utilization.

There are several different aspects of real Grids that should be considered in the design of a scheduling algorithm, such as inter-dependencies among tasks, heterogeneity of tasks and resources, and communication times. Additionally, a realistic scheduler should be robust enough to cope with the possible uncertainties that might occur during the execution of the scheduled tasks. Examples of uncertainties are inaccuracies in the estimated time to compute the tasks, delays in the communications, and even failures in the resources that could make it impossible to complete the assigned tasks.

Bernabé Dorronsoro, Pascal Bouvry, and J. Alberto Cañero work at the Faculty of Sciences and Communications, University of Luxembourg, and A. A. Maciejewski, ECE Dept., and H. J. Siegel, ECE Dept. and CS Dept., Colorado State University, CO, USA (email: {bernabe.dorronsoro, jacanero, pascal.bouvry}@uni.lu, {aam, hj}@colostate.edu).

Many different heuristics have been proposed in the literature for the problem of mapping tasks on computational Grids (e.g., [2]), however, due to its NP-complete nature [3], [4], *metaheuristic* algorithms [5], [6] stand out as highly appropriate tools for tackling this problem [7], [8], especially when its size is large.

In our study, we focus on the problem of allocating static independent tasks on heterogeneous computing environments. Therefore, we consider that a batch of tasks is received by the scheduler with some frequency, and it will be in charge of finding an appropriate mapping of tasks to the available set of machines for efficient execution. We assume in this work that the estimated run time needed to perform every task on the different machines is known *a priori* and this information is used by the scheduler (this assumption is usually made in the literature, e.g., [2], [9], [10]). However, because this is a strong assumption, we are relaxing it by allowing some percentage error (or *tolerance*) in this estimation. This way, we are defining a much more realistic problem. Our objective is to find a robust solution ensuring that the makespan will not grow beyond the maximum tolerance percentage.

We design in this paper a new multi-objective definition of our robust scheduling problem. Consequently, we use a metric that defines how robust the solution is against the allowed uncertainty, originally proposed in [11]. This robustness function will be optimized together with the makespan. For tackling this two objectives problem, we use and analyze several well-known multi-objective evolutionary algorithms (population based metaheuristics) [12], [13], [14].

The main contribution of this work is the mathematical formulation of a new multi-objective problem for the robust static resource allocation of independent tasks on Grids. To the best of our knowledge this is the first multi-objective definition for this problem in the literature. Additionally, the behavior of several state-of-the-art multi-objective evolutionary algorithms (MOEAs) has been compared on this new complex problem. Finally, as a third contribution, we propose a new method for initializing the population of the MOEAs, which creates a preliminary population of very diverse and accurate solutions (in comparison to random ones). This new method is compared against another initialization method that is well accepted in the literature.

The structure of this paper is as follows. Section II summarizes the main existing related work on robust and/or multi-objective resource allocation problems. Then, we provide some background on multi-objective optimization in

978-1-4244-8126-2/10/$26.00

Section III. After that, we describe in Section IV both the classical problem of static mapping of independent tasks and the new multi-objective definition we propose. We present the studied algorithms in Section V, and then we detail our experiments and analyze their results in Section VI. Finally, we conclude this paper in Section VII.

II. Related Work

As a first approach to robust scheduling, several works appeared in the literature proposing scheduling algorithms to find solutions that are somehow *flexible* to possible uncertainties due to breakdowns or some other kind of failures that may occur [15], [16], [17], [18], [19], [20], [21]. In this context, a *flexible* solution means that it is expected to be affected by these uncertainties (e.g., by some delay) to a lesser extent than a regular scheduler. However, these algorithms cannot offer any guarantee on the correct behavior of the scheduled solution after these uncertainties occur. In this case, a good robustness value means that the solution is highly flexible to the considered uncertainty in the sense that the variation in the makespan will be *smaller* than in the case of regular schedulers.

Other works focus on the optimization of the problem considering the worst case for the uncertain values. Thus obtaining the maximum possible robustness for the solutions [22], [23], but a low quality makespan.

Finally, Ali et al. proposed in [11] a general mathematical formulation of a robustness metric that can be applied to a variety of parallel and distributed systems. The authors apply this metric to two example systems, one of them being the static allocation of independent tasks to heterogeneous resources that we are considering in this paper. When adopting this robustness metric, it is guaranteed that if the collective difference of the actual task execution times versus the estimated times is within a certain calculated range, then the given makespan requirement will be met. This metric is used in [24] and [25], and we employ the metric in this paper in Section IV.

Regarding the existing multi-objective approaches for scheduling problems, there are a few works in the literature that consider a single fitness function defined as the weighted sum of the objectives to optimize, for instance makespan and flowtime [8], [26], [27]. Additionally, there exist several works using multi-objective optimization algorithms for the problem of mapping tasks on Grids, either considering dependencies between tasks [28], [29], deadlines [30], or static allocation of independent tasks [31], [32], [33], as the problem considered in this paper. However, these papers are considering objectives such as the resource utilization, the completion time of the resources, or the total execution time, but none is considering the robustness of the system as we do in this work. Perhaps, the only exception is [33], which is considering some kind of robustness by optimizing, together with the resource utilization, the resources reliability by assigning some static reliability values to every resource (i.e., a value meaning how reliable is the resource) in the problem definition.

III. Multi-objective Optimization Fundamentals

To make this paper self-contained, we present some basic notions of multi-objective optimization in this section [13], [14]. Specifically, the concepts of *multi-objective problem* (MOP), *dominance*, *Pareto optimal set*, and *Pareto front* are addressed. We are assuming here, without loss of generality, the minimization of all the objectives.

A general multi-objective optimization problem (MOP) is to find vectors $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]$ that are optimizing the vector of functions $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]$. Each $f_i(\vec{x})$ is a single-objective optimization problem, and it is one of the objectives to optimize in our MOP. The different objectives must be in conflict with the other ones, meaning that an increase in the quality of one of them will lead to worsen the values of (some of) the other ones. If the objectives were not in conflict, then we could reformulate the problem as a single-objective one.

In multi-objective optimization, it is not trivial to decide whether one solution is better than another one or not, because it could be better for several objectives, but worse for some other ones. Therefore, we say that a solution *dominates* another one if it is better for every objective. Two solutions are said to be *non-dominated* if neither dominates the other.

The goal of multi-objective optimization is to find the optimal set of non-dominated solutions to the problem, called the *Pareto optimal set*. Finally, the projection of the Pareto optimal set in the objectives domain is called the *Pareto optimal front* (i.e., the $\vec{f}(\vec{x})$ values for every $\vec{x}$ in the Pareto optimal set). Because the Pareto optimal front might contain a large number of solutions, a good multi-objective algorithm must look for a Pareto front with a limited number of solutions, and it should be as close as possible to the optimal front. Additionally, these solutions should be uniformly spread along the Pareto front; otherwise, they would not be very useful to the decision maker.

IV. Multi-objective Robust Mapping on Grids

We present in this section the new multi-objective problem of robust static mapping of independent tasks on Grids (which we call RSMP, standing for Robust Static Mapping Problem). Consider a set of n independent tasks $T = \{t_1, t_2, \ldots, t_n\}$ that must be scheduled onto the set of k heterogeneous machines $M = \{m_1, m_2, \ldots, m_k\}$. We assume that an estimate of the time to compute task t_i in every machine m_j, $ETC_{i,j}$, is known. Then, the RSMP problem is to allocate all of the tasks on the different machines in a way that the *makespan* of the schedule is minimized, while at the same time its *robustness* is maximized. These two objectives are defined bellow.

- Makespan is defined as the maximum completion time of all the resources used in the schedule. The completion time of a machine m_j in schedule S is defined as:

$$F_j(C) = ready_j + \sum_{t \in S(j)} C_{t,j} \ , \qquad (1)$$

where $ready_j$ is the time when machine m_j will be available (we consider $ready_j = 0$ for every machine

in this work), $S(j)$ is the set of tasks assigned to m_j, and C is a matrix with the actual times to compute the tasks in every machine (C = ETC when we are not considering errors in the estimation of the duration of tasks). Our makespan function is defined as:

$$f_M(\vec{x}) = \{\max\{F_j(C)\} \ , \tag{2}$$

where $\vec{x}$ represents an allocation.

- The robustness metric we are using in this work was originally presented in [11]. It is defined as the minimum of the robustness radii of every machine used in the schedule. The robustness radius for a given machine m_j is the smallest collective increase, based on the euclidean distance, in the error that would cost the finishing time of that machine to be τ times than original, and we calculate it with Eq. (3) taken from [11] —where M^{orig} is the makespan of the schedule with the estimated ETC values and $F_j(C^{orig})$ is the estimated time to compute the assigned tasks to machine m_j. Our robustness metric is the worst machine in terms of that value, as defined in Eq. (4). Specifically, by using this robustness metric, we ensure that if the collective difference of the actual task execution times versus the estimated times is within a certain calculated range, then the given makespan requirement will be met.

$$r_{\vec{x}}(F_j, C) = \frac{\tau \cdot M^{orig} - F_j(C^{orig})}{\sqrt{\text{number of applications allocated to } m_j}} \tag{3}$$

$$f_R(\vec{x}) = \{\min\{r_{\vec{x}}(F_j, C)\} \ . \tag{4}$$

Finally, our multi-objective RSMP problem is defined as finding schedules such that the makespan f_M is minimized and the robustness f_R is maximized —see Eq. (5). Notice that the two objectives to optimize are in conflict, because improving the makespan of a solution may lead to a decrease in its robustness, and vice-versa. Therefore our problem must be solved with multi-objective optimization techniques.

$$RSMP(\vec{x}) = \begin{cases} \text{minimize } f_M(\vec{x}) \\ \text{maximize } f_R(\vec{x}) \end{cases} \ . \tag{5}$$

V. The Algorithms

We have selected from the literature four multi-objective algorithms with different features to solve the new proposed problem. These algorithms are IBEA, MOCell, MOEA/D, and NSGA-II. They belong to the state of the art in multi-objective optimization, and therefore they will provide competitive results for our new problem. We would like to mention at this point that even when NSGA-II is not very competitive for problems with more than two objectives, it is still highly competitive (and probably the most referenced) for two-objectives problems, as is the case considered in this paper.

The *Indicator Based Evolutionary Algorithm*, IBEA [34], is characterized by the way in which the fitness values are computed. In this algorithm, a binary performance metric (also called *indicator*) must be defined by the designer according to his/her preferences, and solutions will be evaluated in terms of their adequacy in terms of this metric. One important feature of IBEA is that because of the use of such indicator, there is no need of including any diversification mechanism during the optimization process.

The *Multi-Objective Cellular* genetic algorithm, MOCell [35], is characterized by the use of an external archive, in which the non-dominated solutions found during the search are stored, and the use of a decentralized population. Specifically, individuals are arranged in a two dimensional toroidal grid, and only those individuals that are close to each other in the mesh are allowed to interact. The main advantage of using this population structure, called *cellular*, is that individuals are isolated by distance, and therefore good solutions will spread slowly through the population mesh, consequently keeping the diversity of solutions for longer [36]. MOCell takes advantage of the good solutions stored in the archive of non-dominated solutions by choosing one of these solutions as one of the parents during the breeding loop.

MOEA/D [37] tackles the MOP by optimizing a number of single-objective optimization problems (called subproblems) that are composed by aggregations of the different functions composing the MOP (the authors propose different aggregation techniques; from them, we use the Tchebycheff approach). Therefore, the different aggregated functions are approximating towards different regions of the Pareto front. A neighborhood relation is established among subproblems, and they are optimized by using information mainly from its neighboring subproblems.

The Non-dominated Sorting Genetic Algorithm, NSGA-II [38], is, perhaps, the most referenced algorithm in the multi-objective literature. It is a GA with a panmictic population. At each generation, an auxiliary population (with the same size as the original one) is generated by iteratively applying the genetic operators, then, both the current and the auxiliary populations are merged into one single new population of the same size for the next generation. The *Ranking* (an ordering of solutions from better to worse in terms of how many other solutions they dominate) and *Crowding* (assigning higher fitness to those solutions that are more isolated in the current Pareto front) processes are used to select the solutions for the next generation population.

VI. Experiments

We summarize and analyze in this section the experiments that were carried out for this work. Section VI-A presents the two problem classes we have considered. The configuration of the algorithms is then described in Section VI-B, and the metrics used to evaluate them are defined in Section VI-C. Finally, Section VI-D provides our main results, an analysis of them, and a comparison of the behavior of the different studied algorithms.

TABLE I

PARAMETERS OF THE STUDIED ALGORITHMS.

Population size	100 solutions MOCell: 10×10 population with C9 neighborhood
Population initialization	One solution with Min-min and the rest random Each solution: 25% random genes & 75% Min-min
Selection	Binary tournament
Recombination	two-points crossover, $p\_comb = 0.9$
Mutation	rebalance mutation, $p\_mut = 1.0/ChromosomeLength$
Local search	LMCTS, $p\_ls = 1.0$
Archive size	100 solutions
Stopping criterion	500,000 fitness function evaluations

A. Problem Classes

We have considered in this work two different problem classes, namely HTRH, with *High Task and Resources Heterogeneities*, and LTRH, having *Low Task and Resources Heterogeneities*. A high task heterogeneity represents highly different computation times for the different tasks. In contrast, the resource heterogeneity refers not only to different computational power, but also to different architectures, meaning that if a machine m_i is faster than m_j for a given task t_p, it does not mean that it will be faster for all tasks; it could happen that m_j is faster than m_i for computing some other tasks.

For each of the two studied problem classes, we generated 100 different instances, i.e., 100 different ETC matrices. The instances were generated following the coefficient-of-variation based ETC matrix generation method presented in [39]. All the considered instances are composed of 512 independent tasks that must be scheduled on a cluster of 16 machines. This instance size is well accepted in the literature [2], [8], [26], [27], [40], and it is out of the scope of this paper to tackle large instances that would possibly require the parallelization of the studied multi-objective algorithms, which would make the extensive studies presented in this work much more difficult.

B. Algorithms Configuration

We have set the same parameter values for all the algorithms. Their configuration is presented in Table I. We used the implementation of all the algorithms provided in the jMetal [41] framework.

Solutions are represented as an array $S = [S_1, S_2, \ldots, S_n]$, where n is the number of tasks, and every S_i is the identifier of the machine to which task i is assigned.

We are using populations of 100 solutions. In the case of MOCell, due to its structured population, we must decide the population shape and the neighborhood. We took a square population and a C9 neighborhood (i.e., the eight surrounding individuals plus the considered one), as suggested in the original MOCell paper.

Two different methods for initializing the population have been analyzed in this work. One of them, commonly used in the literature [2], [8], [27], [40], is to insert one *seed* (e.g., a solution generated with Min-min heuristic [4]), into a population of completely random solutions. In this paper, we call this initialization method *Seed*, and the algorithms using it are named NSGA-II, IBEA, MOEA/D, and MOCell. The other method studied for generating the initial population is

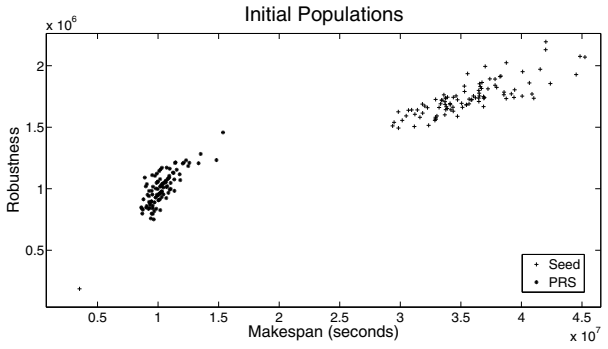

Fig. 1. Example of two initial populations generated with Seed and PRS for the HTRH problem.

presented in this paper for the first time, and we call it the *Partially Random Solution* (PRS) generation method. We use this method to initialize all the individuals in the population. According to this method, individuals are generated in the following way: a given percentage of the tasks (randomly selected) are assigned to the available machines in a random way; then, the other unassigned tasks are allocated according to the Min-min heuristic to complete the solution. After some preliminary experiments, we set the percentage of randomly allocated tasks to 25%; consequently, 75% of the solution is generated using the Min-min heuristic. The algorithms implementing this new population initialization method are called NSGA-II_PRS, IBEA_PRS, MOEAD_PRS, and MOCell_PRS in this paper. Notice that when setting the percentage of randomly assigned tasks to 0%, then the PRS method generates the same solution as Min-min.

We plot in Fig. 1 the solutions of two populations (100 solutions) generated with both the PRS and Seed methods. As we can see, Seed generates one single solution with low makespan and robustness values (in the bottom left corner), and 99 random solutions with high robustness values, but high makespan too. Therefore, Seed is generating high fitness solutions for one of the two objectives (either robustness or makespan), but poor values for the other objective. In contrast, the PRS method is generating solutions that are, as it can be seen in the picture, between these two extremes, i.e., solutions producingn a more appropriate tradeoff between robustness and makespan.

In all the algorithms, the binary tournament operator is used for selection, the two-points crossover for recombination (applied with probability $p\_comb = 0.9$), and the rebalance operator [8] for mutation (with probability $p\_mut = 1.0/ChromosomeLength$). Both the selection and recombination operators are well known in the literature. The rebalance mutation operates by randomly choosing one task from the 25% of the highest completion time machines, and assigning it to one of the 25% of the lowest completion time machines.

Every new generated solution undergoes a local search procedure with the goal of improving its makespan value. This local search is the *Local Minimum Completion Time Swap* (LMCTS), which consists of swapping all pairs of tasks involving a reduction in the completion time of the

affected machines (which involves "n choose 2" steps). For those algorithms using an external archive for storing the non-dominated solution, its size was set to the population size. The algorithms iterate until 500,000 fitness function evaluations of chromosomes are accomplished.

Finally, there are some additional parameters that are specific to some of the compared algorithms. Specifically, we use the hypervolume metric as quality indicator for IBEA algorithm, and the Tchebycheff approach [37] is used in MOEA/D to generate the scalar optimization problems to optimize by the algorithm.

C. Evaluation of the Algorithms

Several metrics, well known in the multi-objective literature, have been used to compare the performance of the algorithms:

- *Inverted Generational Distance (IGD)* [42]: Measures the proximity of the Pareto front to the optimal one. It is defined in Eq. (6), where d_i is the distance from point i in the Pareto front to the closest one in the optimal Pareto front, and n is the number of solutions in the Pareto front. In the case when the evaluated Pareto front consists of only solutions from the optimal Pareto front, this metric is equal to zero.

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{n} \quad . \tag{6}$$

- *SPREAD*: Estimates how well the solutions are distributed along the Pareto front. It was originally defined in [38] for two-objective problems, and later generalized for more than two objectives in [43]. This work uses the latter version – defined in Eq. (7) – which is more generic, even when the considered problem is a two-objectives one. The value for this metric is zero for an ideal distribution of the solutions in the Pareto front.

$$\Delta = \frac{\sum_{i=1}^{m}(d(e_i,S)) + \sum_{X\in S}\left|d(X,S) - \overline{d}\right|}{\sum_{i=1}^{m} d(e_i,S) + |S|\cdot\overline{d}} \quad . \tag{7}$$

  where S is a set of solutions, S^* is the set of Pareto optimal solutions, $(e_1,\ldots,e_m)$ are m extreme solutions in S^*, m is the number of objectives, and

$$d(X,S) = \min_{Y\in S, Y\neq X} ||f(X) - f(Y)||^2 \tag{8}$$

$$\overline{d} = \frac{1}{|S^*|}\sum_{X\in S^*} d(X,S) \quad , \tag{9}$$

  where f is the fitness function to optimize.

- *Hypervolume (HV)* [44]: Calculates the volume (in the objective space) covered by the solutions in the evaluated Pareto front Q. Mathematically, for each solution $i \in Q$, a hypercube v_i is constructed with a reference point W and the solution i as the diagonal corners of the hypercube. The reference point can simply be found by constructing a vector of worst objective function values. Thereafter, a union of all hypercubes is found and its hypervolume is calculated, as shown in Eq. (10). This metric takes its maximum value when all the solutions in the evaluated Pareto front belong to the optimal one.

$$HV = \text{volume}\left(\bigcup_{i=1}^{|Q|} v_i\right) \quad . \tag{10}$$

Therefore, we are analyzing distinct aspects of the Pareto fronts reported by the algorithms: IGD is measuring the convergence (i.e., distance to the optimal Pareto front), SPREAD is a measure of the distribution of the solutions in the Pareto front, and HV is considering these two issues (convergence and distribution) at the same time. In order to avoid any kind of bias in the metrics due to the different order of magnitude of the objectives to optimize, it is common to apply the metrics after a normalization of the objective function values with respect to the values in the optimal Pareto front. Because the optimal Pareto front is not known for the considered problems, we generate a reference Pareto front by merging all the Pareto fronts found by the four algorithms in the 100 runs into one for every problem instance studied. (See Fig. 2 for an example.) This reference Pareto front will be used for the normalization.

D. Results

We summarize in this section our results. All the experiments presented here were run on Intel Xeon 2.0 GHz processors with 4 MB of RAM under linux Debian 4.1 OS. The java version used is 1.5. Figures 3 and 4 present the results of every algorithm for the problems with low and high tasks and resources heterogeneities, respectively. Specifically, the boxplots are created with all the results obtained with every algorithm after 100 independent runs for every one of the 100 problem instances (the results of the different problem instances are comparable, since the ETC matrices were generated using gamma distributions with the same parameters). Therefore, every box is a sample of 10,000 results (100 problem instances, being solved 100 times each). In total, 160,000 independent runs of the algorithms (10,000 runs per algorithm $\times$ 8 algorithms $\times$ 2 problems) were performed for the results summarized in this section.

In the displayed boxplots, the bottom and top of the boxes represent the lower and upper quartiles of the data distribution, respectively, while the line between them is the median. The whiskers are the lowest datum still within 1.5 IQR of the lower quartile, and the highest datum still within 1.5 IQR of the upper quartile. The crosses are data not included between the whiskers. Finally, the notches in the boxes display the variability of the median between samples. If the notches of two boxes are not overlapped, then it means that there are statistical significant differences in the data with 95% confidence.

As it was mentioned in the previous section, the different Pareto fronts computed by the algorithms were normalized before applying the metrics in order to avoid some bias due to the different order of magnitudes in the two objectives to optimize. In this work, we do not know the optimal Pareto

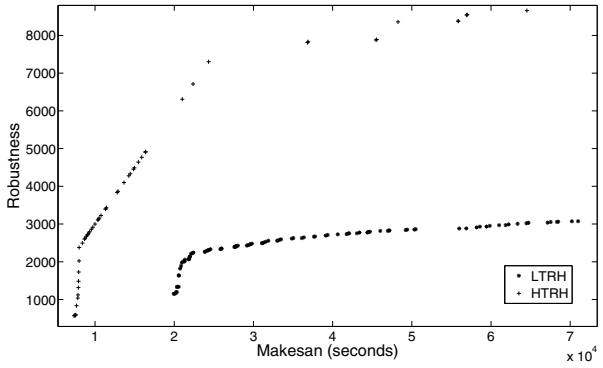

Fig. 2. Example reference Pareto fronts for instance number 1 of HTRH and LTRH problems (maximizing robustness and minimizing makespan), obtained after merging the Pareto fronts obtained by all the compared algorithms in the 100 independent runs.

front, so we therefore generated a reference Pareto front for every problem instance by merging all the Pareto fronts obtained by the different algorithms in the 100 runs. As an example, we plot in Fig. 2 the reference Pareto fronts we obtained for instance number 1 (out of the 100 ones we generated for this work) for the two studied problems. It can be seen in this figure that, in effect, the makespan values are one order of magnitude higher than the robustness, and it could influence the results of the metrics if these values were not normalized.

It stands out in Fig. 2 how fast the robustness of the solutions in the Pareto front is growing with the makespan for the high heterogeneities case with respect to LTRH, i.e., the algorithms can find solutions with better robustness values for the HTRH problem than for LTRH for the same makespan values. However, even when the algorithms find solutions with a higher robustness level for HRTH, they have difficulties finding them, as is demonstrated by the large gaps and few solutions in the upper right part of the front.

The studied algorithms are compared in figures 3 and 4. As we can see, most of the differences on the results reported by the algorithms are significant with 95% confidence level (the notches in the boxes do not overlap). Among all the algorithms, MOCell (initialized Seed) clearly stands out as the best performing algorithm for both the HTRH and LTRH problems and for the three studied metrics (with statistically significant differences in all cases). The only exception is the result for SPREAD in the HTRH problem, since even when MOCell finds more accurate Pareto fronts than the compared algorithms (as demonstrated by the results for IGD in Fig. 4), the solutions are not as well distributed as in the case of the others. We believe that the good performance of MOCell with respect to IBEA, MOEA/D and NSGA-II is due to its decentralized population: it is able to keep a high level of diversity in the solutions of the population, even in the presence of the seed (a high quality makespan solution), while the other compared algorithms are influenced by this seed in an important way, consequently loosing the population diversity too rapidly.

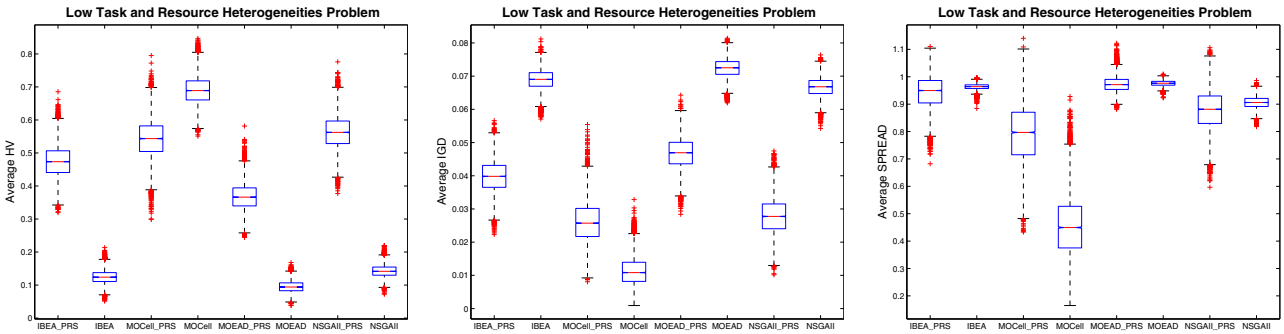

Fig. 3. Results for high heterogeneities problem (averaged on 100 runs for each of the 100 different instances).

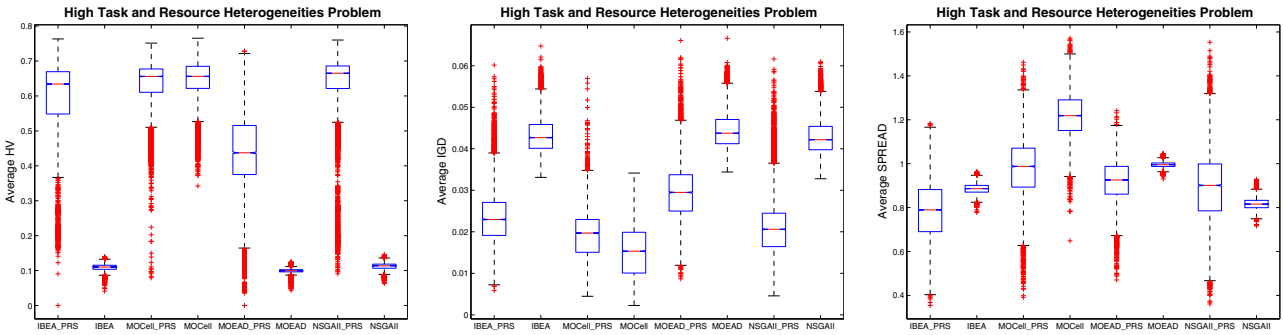

Fig. 4. Results for high heterogeneities problem (averaged on 100 runs for each of the 100 different instances).

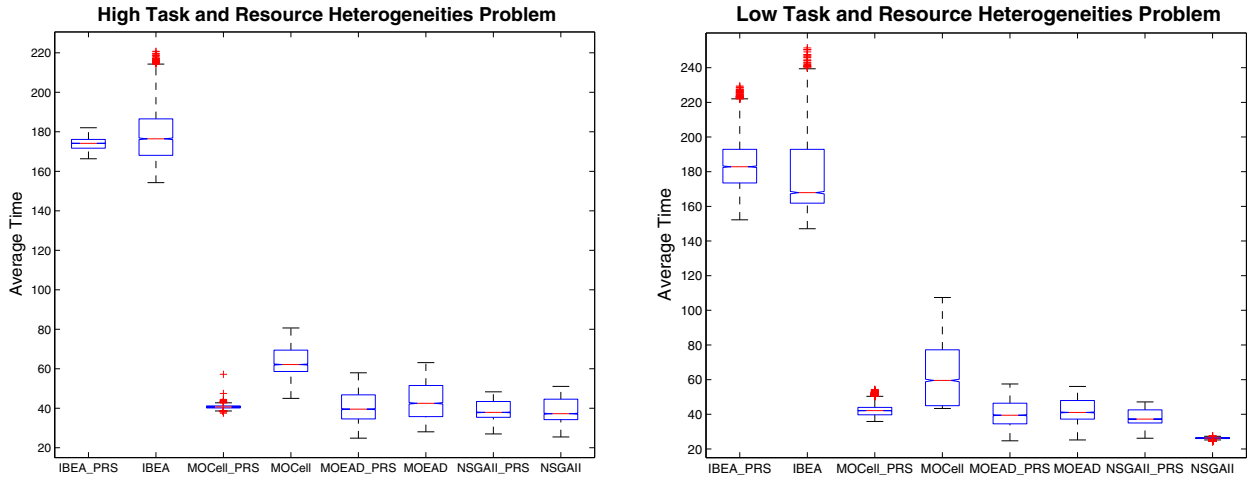

Fig. 5. Computational time (in seconds) for all the studied algorithms (averaged on 100 runs for each of the 100 different instances).

Regarding the population initialization method used, we can see that, in general, the algorithms with the new PRS method outperform the equivalent algorithms with the other compared method (Seed) in most cases. The only exceptions are MOCell, which outperforms MOCell_PRS, as well as all the other compared algorithms in most cases (as we just commented), and NSGA-II for SPREAD and HTRH problem.

We show in Fig. 5 the computational times, in seconds, required by all the algorithms to solve the HTRH and LTRH problems. It stands out that the algorithms using PRS for initializing the population are generally faster (with 95% confidence) than the equivalent ones using Seed. Notice that the termination condition is the same for all the algorithms (to reach 500,000 evaluations), and even when PRS is applied to generate all the individuals in the population (Seed generates only one solution with Min-min and all the other ones randomly), the algorithms running PRS are generally faster than the ones using Seed. The slowest studied algorithms are both IBEA and IBEA_PRS, more than three times slower than all the others for the two considered problems.

VII. Conclusions and Future Work

We have proposed in this paper a new multi-objective definition for the problem of robust static mapping of independent tasks on Grids. We have considered two different cases for this problem: those with high and low task and resource heterogeneities (called HTRS and LTRH, respectively). In order to provide competitive results for the new problems, they were solved with four well-known multi-objective evolutionary algorithms (MOEAs).

The different MOEAs were compared for the two studied problem classes in terms of three metrics; those measuring the convergence of the Pareto front (the inverted generational distance method), the dispersion of their solutions (spread metric), and both of them at the same time (hypervolume). MOCell was shown to be the best performing of the considered algorithms for the two studied problem classes.

Additionally, a new method for initializing the population of the MOEAs for scheduling problems, called PRS, was proposed. This method is to generate all the solutions in the population by randomly assigning part of the tasks to some of the available machines, and completing the solutions by applying the Min-min heuristic to the remaining unscheduled tasks. As a result, most of the MOEAs using this method obtained better solutions in shorter times.

As future work, we are considering the design of new variation operators (i.e., recombination, mutation, or local search) that would focus not only on the makespan improvement, but on the robustness as well. Additionally, we plan to consider larger problem instances (with both higher number of machines and tasks) in future work. For that, we are planning on using parallel algorithms and decomposing the problem. Finally, we think it could be interesting to perform an in depth study on the different possible parameterizations for the new PRS proposed population initialization method.

Acknowledgments

This research was supported in part by Luxembourg FNR GreenIT project (C09/IS/05), by the NSF under grants CNS-0615170 and CNS-0905399, and by the Colorado State University George T. Abell Endowment.

References

[1] I. Foster and C. Kesselman, *The Grid - Blueprint for a New Computing Infrastructure*. Morgan Kaufmann Publishers, 1998.

[2] T. D. Braun, H. J. Siegel, N. Beck, L. L. Bölöni, M. Maheswaran, A. I. Reuther, J. P. Robertson, M. D. Theys, B. Yao, D. Hensgen, and R. F. Freud, "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *Journal of Parallel and Distributed Computing*, vol. 61, no. 6, pp. 810–837, 2001.

[3] D. Fernandez-Baca, "Allocating modules to processors in a distributed system," *IEEE Transactions on Software Engineering*, vol. 15, no. 11, pp. 1427–1436, 1989.

[4] O. H. Ibarra and C. E. Kim, "Heuristic algorithms for scheduling independent tasks on nonidentical processors," *Jornal of the ACM*, vol. 24, no. 2, pp. 280–289, 1977.

[5] F. W. Glover and G. A. Kochenberger, *Handbook of Metaheuristics*, ser. International Series in Operations Research Management Science, F. W. Glover and G. A. Kochenberger, Eds. Kluwer Academic Publishers, 2003.

[6] H. Mühlenbein and H. M. Voigt, *Metaheuristics: Theory and Applications*. Kluwer Academic Publishers, 1996, ch. Gene pool recombination in genetic algorithms, pp. 53–62.

[7] A. Abraham, R. Buyya, and B. Nath, "Nature's heuristics for scheduling jobs on computational grids," in *The 8th IEEE International Conference on Advanced Computing and Communications (ADCOM)*. India: IEEE Press, 2000, pp. 45–52.

[8] F. Xhafa, E. Alba, B. Dorronsoro, and B. Duran, "Efficient batch job scheduling in grids using cellular memetic algorithms," *Journal of Mathematical Modelling and Algorithms*, vol. 7, no. 2, pp. 217–236, 2008.

[9] A. Ghafoor and J. Yang, "Distributed heterogeneous supercomputing management system," *IEEE Comput.*, vol. 26, no. 6, pp. 78–86, 1993.

[10] M. Kafil and I. Ahmad, "Optimal task assignment in heterogeneous distributed computing systems," *IEEE Concurrency*, vol. 6, no. 3, pp. 42–51, 1998.

[11] S. Ali, A. A. Maciejewski, H. J. Siegel, and J.-K. Kim, "Measuring the robustness of a resource allocation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 51, no. 7, pp. 630–641, 2004.

[12] T. Bäck, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. New York: Oxford University Press, 1996.

[13] C. A. C. Coello, G. B. Lamont, and D. A. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, 2007, second edition.

[14] K. Deb, *Multi-Objective Optimization using Evolutionary Algorithms*. Wiley, 2001.

[15] J. C. Bean, J. R. Birge, J. Mittenthal, and C. E. Noon, "Matchup schedules with multiple resources, release dates and disruptions," *Operations Research*, vol. 39, pp. 470–483, 1991.

[16] E. Hart, P. Ross, and J. Nelson, "Producing robust schedules via an artificial immune system," in *Proceedings of the IEEE World Congress on Computational Intelligence*, 1998, pp. 464–469.

[17] V. J. Leon, S. D. Wu, and R. H. Storer, "A game-theoretic control approach for job shops in the presence of disruptions," *International Journal of Production Research*, vol. 32, no. 6, pp. 1451–1476, 1994.

[18] ——, "Robustness measures and robust scheduling for job shops," *IIE Transactions*, vol. 26, no. 5, pp. 32–43, 1994.

[19] M. T. Jensen, "Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures," *Applied Soft Computing*, vol. 1, pp. 35–52, 2001.

[20] ——, "Generating robust and flexible job shop schedules using genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 275–288, 2003.

[21] S. D. Wu, E. Byeon, and R. H. Storer, "A graph-theoretic decomposition of the job shop scheduling problem to achieve scheduling robustness," *Operations Research*, vol. 47, no. 1, pp. 113–124, 1999.

[22] J. W. Herrmann, "A genetic algorithm for minimax optimization problems," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 2, 1999, pp. 1099–1103.

[23] P. Kouvelis and G. Yu, *Robust Discrete Optimization and Its Applications*. Norwell, MA: Kluwer Academic Publishers, 1997.

[24] P. Sugavanam, H. J. Siegel, A. A. Maciejewski, M. Oltikar, A. Mehta, R. Pichel, A. Horiuchi, V. Shestak, M. Al-Otaibi, Y. Krishnamurthy, S. Ali, J. Zhang, M. Aydin, P. Lee, K. Guru, M. Raskey, and A. Pippin, "Robust static allocation of resources for independent tasks under makespan and dollar cost constraints," *Journal of Parallel and Distributed Computing*, vol. 67, no. 4, pp. 400–416, 2007.

[25] A. M. Mehta, J. Smith, H. J. Siegel, A. A. Maciejewski, A. Jayaseelan, and B. Ye, "Dynamic resource allocation heuristics that manage tradeoff between makespan and robustness," *Journal of Supercomputing, Special Issue on Grid Technology*, vol. 42, no. 1, pp. 33–58, 2007.

[26] J. Carretero, F. Xhafa, and A. Abraham, "Genetic algorithm based schedulers for grid computing systems," *International Journal of Innovative Computing, Information and Control*, vol. 3, no. 5, pp. 1053–1071, 2007.

[27] F. Xhafa, J. Carretero, B. Dorronsoro, and E. Alba, "A tabu search algorithm for scheduling independent jobs in computational grids," *Computing and Informatics Journal, special issue on Intelligent Computational Methods and Models*, vol. 28, pp. 1001–1014, 2009.

[28] J. Yu, M. Kirley, and R. Buyya, "Multi-objective planning for workflow execution on grids," in *IEEE/ACM International Conference on Grid Computing*, 2007, pp. 10–17.

[29] G. Ye, R. Rao, and M. Li, "A multiobjective resources scheduling approach based on genetic algorithms in grid environment," in *Proceedings of the Fifth International Conference on Grid and Cooperative Computing Workshops (GCCW)*. IEEE Press, 2006, pp. 504–509.

[30] C. Grosan, A. Abraham, and B. Helvik, "Multiobjective evolutionary algorithms for scheduling jobs on computational grids," in *IADIS International Conference, Applied Computing*, 2007, pp. 459–463.

[31] I. De Falco, A. Della Cioppa, U. Scafuri, and E. Tarantino, "Multiobjective differential evolution for mapping in a grid environment," in *High Performance Computing and Communications (HPCC) Conference*, vol. 4782. Springer-Verlag, 2007, pp. 322–333.

[32] ——, "A multiobjective evolutionary approach for multisite mapping on grids," in *Parallel Processing and Applied Mathematics*, vol. 4967. Springer-Verlag, 2008, pp. 991–1000.

[33] I. De Falco, A. Della Cioppa, D. Maisto, U. Scafuri, and E. Tarantino, *A Multiobjective Extremal Optimization Algorithm for Efficient Mapping in Grids*, ser. Advances in Soft Computing. Springer-Verlag, 2009, vol. 58, pp. 367–377.

[34] E. Zitzler and S. Künzli, "Indicator-based selection in multiobjective search," in *Parallel Problem Solving from Nature (PPSN VIII)*, ser. Lecture Notes in Computer Science, vol. 3242. Springer, 2004, pp. 832–842.

[35] A. J. Nebro, J. J. Durillo, F. Luna, B. Dorronsoro, and E. Alba, "MOCell: A cellular genetic algorithm for multiobjective optimization," *International Journal of Intelligent Systems*, vol. 24, pp. 726–746, 2009.

[36] E. Alba and B. Dorronsoro, *Cellular Genetic Algorithms*, ser. Operations Research/Compuer Science Interfaces. Springer-Verlag Heidelberg, 2008.

[37] Q. Zhang and H. Li, "MOEA/D: A multi-objective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[38] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[39] S. Ali, H. J. Siegel, M. Maheswaran, D. Hensgen, and S. Ali, "Representing task and machine heterogeneities for heterogeneous computing systems," *Journal of Science and Engineering, Special 50 th Anniversary Issue*, vol. 3, no. 3, pp. 195–207, 2000.

[40] G. Ritchie and J. Levine, "A hybrid ant algorithm for scheduling independent jobs in heterogeneous computing environments," in *23rd Workshop of the UK Planning and Scheduling Special Interest Group (PLANSIG 2004)*, 2004.

[41] jMetal, a framework for multi-objective optimization. [Online]. Available: http://jmetal.sourceforge.net/

[42] D. A. V. Veldhuzen, "Multiobjective evolutionary algorithms: Classifications, analyses, and new innovations," Ph.D. dissertation, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, 1999.

[43] A. J. Nebro, F. Luna, E. Alba, B. Dorronsoro, J. J. Durillo, and A. Beham, "AbYSS: Adapting scatter search to multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 4, pp. 439–457, 2008.

[44] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.