BI-OBJECTIVE SCHEDULING IN HETEROGENEOUS GRID COMPUTING SYSTEMS USING A PARALLEL MICRO EVOLUTIONARY ALGORITHM

Santiago Iturriaga
Centro de Cálculo, Facultad de Ingeniería, Universidad de la República, Uruguay
Herrera y Reissig 565, Montevideo, Uruguay
siturria@fing.edu.uy

Sergio Nesmachnow
Centro de Cálculo, Facultad de Ingeniería, Universidad de la República, Uruguay
Herrera y Reissig 565, Montevideo, Uruguay
sergion@fing.edu.uy

ABSTRACT

This work presents the application of a micro evolutionary algorithm to solve a bi-objective scheduling problem in heterogeneous grid computing environments. A new formulation of the bi-objective scheduling problem is introduced, by considering the *makespan* and *weighted response ratio* metrics, which account for the total execution time and the waiting time respectively. The problem is NP-hard, and a parallel implementation of the proposed micro evolutionary algorithm is presented, in order to find accurate results in short execution times. The experimental analysis performed on both standard low-sized and large problem instances shows that efficient schedules are computed by the proposed method.

KEYWORDS. Scheduling, heterogeneous computing, evolutionary algorithms.

Main area: MH - Metaheuristics

1. Introduction

In the last decade, distributed computing platforms have been increasingly used to solve complex problems with large computing demands. Nowadays, the paradigm of grid computing is employed to work over a large distributed, heterogeneous, and loosely-coupled aggregation of processing elements that allow gathering the computing power required for complex applications (Foster, 1996). When using this kind of heterogeneous computing platforms, a capital problem consists in finding a tasks-to-machine assignment in order to fulfill specific requirements, usually related with the total time required to execute a bunch of tasks and/or with the quality of service provided by the computing infrastructure. The Heterogeneous Computing Scheduling Problem (HCSP) has emerged as an important optimization problem in the last decade, due to its ability to model the correct planning of distributed computing grid environments (Eshaghian 1996).

The HCSP is NP-hard (Garey and Johnson, 1979), even in its classical formulation that proposes to minimize a unique objective function. So, classical methods such as dynamic programming, linear programming, etc. are only useful to solve low-dimension problem instances. When tackling large scheduling scenarios, heuristics and metaheuristic techniques are promising methods to solve the HCSP, since they are able to compute accurate sub-optimal schedules in reasonable times. Among a large set of modern metaheuristic techniques, evolutionary algorithms (EAs) emerged as accurate and efficient methods to solve scheduling problems, showing the high level of problem-solving accuracy they get in many other areas of application (Back et al. 1997).

EAs have been applied to the single-objective HCSP that proposes to minimize the *makespan* metric in the last fifteen years (Wang 1997, Braun 2001, Zomaya 2001, Xhafa 2008). However, those HCSP variants that propose the simultaneous optimization of several scheduling efficiency metrics have been seldom tackled. In addition, problem instances that model realistic grid environments have rarely been faced, mainly due to the complexity of dealing with the underlying high-dimension optimization problem. Few works have studied parallel EAs in order to determine their ability to use the computing power of large clusters to improve the search efficacy and efficiency. Thus, there is still room to contribute in these lines of research by studying highly efficient parallel EA implementations, able to deal with large-size multiobjective HCSP instances by using the computational power of parallel and distributed environments.

In this line of work, this article presents a parallel micro EA applied to the bi-objective HCSP that proposes to minimize the *makespan* and *weighted response ratio* metrics. The main contributions of the work are to introduce a new formulation of the bi-objective scheduling problem in heterogeneous grid computing systems; and to develop an accurate parallel micro EA to solve it. Efficient numerical results are reported for the experiments performed on realistic problem instances. The analysis shows that the proposed EA is able to achieve high problem solving efficacy, outperforming the results obtained with traditional deterministic heuristics, while also exhibiting a good scalability behavior when solving high dimension problem instances.

The manuscript is structured as follows. Next section presents the HCSP, its mathematical formulation, and the deterministic heuristics used in the results comparison. Section 3 introduces evolutionary computing techniques and describes the new parallel micro EA proposed in this work. The implementation details of the parallel micro EA applied to the bi-objective HSCP are provided in Section 4. Section 5 presents and discusses the experimental analysis and results. The conclusions and possible lines for future work are formulated in Section 6.

2. Scheduling in heterogeneous computing and grid environments

This section introduces the bi-objective HCSP and its mathematical formulation, and describes the deterministic scheduling heuristics using in the comparative experimental analysis.

2.1 Bi-objective HCSP in grid computing environments

Most scheduling problems mainly concern about time, trying to minimize the time spent to execute a set of submitted tasks. The bi-objective version of the HCSP introduced in this work proposes to minimize two relevant metrics: the *makespan* and the *weighted response ratio*.

The makespan is defined as the time spent from the moment when the first task begins execution to the moment when the last task is completed (Leung 2004). It is the most usual metric to minimize in scheduling problems, since it evaluates the computing resources utilization.

Many other performance metrics have been considered to optimize in scheduling problems. They are usually related to the economic cost of executing an application, or to the quality of service (QoS), which is especially pertinent in grid infrastructures. The *response time* of each task is an indicator of the QoS of the system and it is an important metric from the user point-of-view, since it reflects the response time of a computational system for a set of submitted tasks. The *response ratio* metric evaluates the sum of the response times of each submitted task, and it is inversely dependent on each task estimated completion time, thus favoring short tasks over long tasks (Stallings 2001). A common approach in modern computational grid systems is to group the tasks in different classes, according to their importance or priority (Buyya 2002, Dong 2006). To model this scenario, the bi-objective HCSP tackled in this work uses the *weighted response ratio* (*wrr*) metric: each task is assigned a positive weight p_i which represents the relative priority of the task in the system. The *wrr* metric is then defined as the total response ratio of each task multiplied by its priority weight in the system (p_i), following a weighted approach that has been previously applied to the response time metric (Monte 2002).

2.2 Mathematical formulation

The following formulation presents the mathematical model for the MR-HCSP:

- Let there be an heterogeneous grid computing system composed of a set of processors or machines M = $\{m_1, m_2, ..., m_L\}$ (dimension L), and a collection of tasks T = $\{t_1, t_2, ..., t_N\}$ (dimension N) to be executed on the system,
- let there be an *execution time function* $ET : T \times M \rightarrow \mathrm{R}^+$, where $ET(t_i,m_j)$ is the time required to execute the task t_i in the machine m_j,
- let there be a *priority function* $P : T \rightarrow \mathrm{N}^+$, where $P(t_i)$ is the priority of the task ti in the system,
- let $F(t_i)$ be the finishing time of task t_i in the system;
- the goal of the bi-objective HCSP is to find an assignment of tasks to machines (a function $f : T^N \rightarrow M^L$) which simultaneously minimizes the *makespan*, defined in Equation 1, and the *weighted response ratio*, defined in Equation 2.

$$\max_{m_j \in M} \sum_{\substack{t_i \in T: \\ f(t_i)=m_j}} ET(t_i, m_j) \tag{1}$$

$$\sum_{\substack{t_i \in T: \\ f(t_i)=m_j}} P(t_i) \times \frac{F(t_i)}{ET(t_i, m_j)} \tag{2}$$

Both objectives are in conflict, as minimizing the makespan metric implies to execute the tasks in the minimum possible time, and the weighted response ratio inversely depends on the expected execution time.

In the previous HCSP formulation all tasks can be independently executed. This kind of applications frequently appears in many lines of scientific research, such as in Single-Program Multiple-Data applications used in multimedia processing, data mining, parallel domain decomposition of numerical models for physical phenomena, etc. Another scenario with independent tasks is when different users submit their (obviously independent) tasks to execute in grid and volunteer-based computing infrastructures -such as TeraGrid, WLCG, BOINC, Xgrid, etc.-, where non-dependent applications using domain decomposition are very often submitted for execution. Thus, the relevance of the bi-objective HCSP faced in this work is justified due to its importance in realistic distributed heterogeneous grid computing systems.

In this work, the bi-objective optimization problem is solved by applying an aggregate function approach that uses a weighted sum of makespan and weighted response ratio. Although the aggregation function approach is often outperformed by Pareto-based methods when solving multiobjective optimization problems, it is a common approach in the literature mainly due to two main advantages: it is computationally efficient, so it is recommended when the times available for search is short, and it is suitable for optimization problems with a convex Pareto front (Coello 2006). The aggregation function approach has been previously applied to other HCSP variants by Xhafa et al. (2007, 2008). In the bi-objective HCSP tackled in this work, since the makespan and weighted response ratio objectives are in different units, they have to be normalized considering a reference solution before the aggregation.

2.3 Traditional heuristics for scheduling in heterogeneous computing systems

Many deterministic heuristics have been proposed for scheduling in heterogeneous computing and grid systems (Kwok 1999). Most of them are based on simple iterative procedures that assign tasks to machines in a given order, trying to fulfill a specific criterion. Three deterministic heuristics have been used to provide a baseline for comparing the results achieved with the micro EA proposed in this work:

Minimum Completion Time (MCT) considers the set of tasks sorted in an arbitrary order. Then, it assigns each task to the machine with the minimum execution time for that task.

Sufferage identifies the task that if it is not assigned to a certain host, will *suffer* the most. The *sufferage value* is computed as the difference between the best MCT of the task and its second-best MCT, and this method gives precedence to those tasks with high sufferage value.

Min-Min greedily picks the task that can be completed the soonest. The method starts with a set U of all unmapped tasks, calculates the MCT for each task in U for each machine, and assigns the task with the minimum overall MCT to the machine that executes it faster. The mapped task is removed from U, and the process is repeated until all tasks are mapped. Min-Min improves upon the MCT heuristic, since it considers all the unmapped tasks sorted by MCT, and the availability status of the machines is updated by the least possible amount of time for every assignment. Min-Min is one of the most effective deterministic scheduler for a wide range of heterogeneous computing scenarios.

3. Evolutionary computation

This section presents evolutionary computation techniques, the CHC algorithm, and parallel EAs. After that, the new proposal of a parallel micro-CHC algorithm is introduced.

3.1 Evolutionary algorithms

EAs are randomized (i.e. non-deterministic) methods that emulate the evolutionary process of species in nature, in order to solve optimization, search, and other related problems (Back et al. 1997). In the last twenty-five years, EAs have been successfully applied for solving optimization problems underlying many real applications of high complexity.

An EA is an iterative method (each iteration is called a generation) that applies stochastic operators on a pool of individuals (the population) in order to improve their *fitness*, a measure related to the objective function of the optimization problem. Every individual in the population is the encoded version of a tentative solution of the problem. The population is initialized by either using a random method or a specific heuristic for the problem. The *evaluation* associates a fitness value to every individual, indicating its suitability to the problem. The probabilistic application of evolutionary operators like the *recombination* of parts of two individuals and *mutation* in their contents are guided by a selection-of-the-best technique to tentative solutions of higher quality.

The stopping criterion usually considers a fixed number of generations or execution time, a quality threshold on the best fitness value, or the detection of a stagnation situation. Specific policies are used in the *selection* of individuals to recombine and to determine which new individuals *replace* the old ones in each new generation. The EA returns the best solution ever found in the iterative process, taking into account the fitness function values.

3.2 The CHC algorithm

"Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation" (CHC) (Eshelman 1991) is a variant of EA that uses an elitist selection strategy that tends to perpetuate the best individuals in the population. CHC uses a special mating: only those parents which differ from each other by some number of bits are allowed to reproduce. The initial threshold for allowing mating is often set to 1/4 of the chromosome length. If no offspring is inserted into the new population, this threshold is reduced by 1. The recombination operator in CHC is Half Uniform Crossover (HUX), which randomly swaps exactly half of the bits that differ between the two parent strings. CHC does not apply mutation; diversity is provided by applying a re-initialization procedure, using the best individual found so far as a template for partially creating a new population after convergence is detected.

The pseudo-code of CHC presented in Algorithm 2 shows those features that make it different from classical EAs: the elitist selection strategy, the use of the HUX recombination operator, the absence of mutation, which is substituted by a re-initialization operator, and the mating restriction policy, that does not allow to recombine a pair of "too similar" individuals.

```
initialize (P(0))
generation ← 0;  distance ← 1/4 * chromosomeLength
while (not stopcriteria) do
        evaluate(P(generation))
        parents ← elitist_selection(P(generation))
        if (distance(parents) ≤ distance) then
                offspring ← HUX (parents)
                evaluate(offspring)
                newpop ← replace(offspring, P(generation))
        end
        if (newpop = P(generation)) then
                distance – –
        end
        generation ++
        P(generation) ← newpop
        if (distance = 0) then
                reinitialization (P(0))
                distance ← 1/4 * chromosomeLength
        end
end
return best solution ever found
```

Algorithm 2: Schema of the CHC algorithm

3.3 Parallel evolutionary algorithms

Parallel implementations became popular in the last decade as an effort to improve the efficiency of EAs. By splitting the population into several computing elements, parallel EAs allow reaching high quality results in a reasonable execution time even for hard-to-solve optimization problems (Alba 2005). The parallel EA proposed in this work is categorized within the *distributed subpopulations* model according the classification by Alba and Tomassini (2002): the original population is divided into several subpopulations, separated geographically from each other. Each subpopulation runs a sequential EA, so individuals are able to interact only with other individuals in the subpopulation. An additional *migration* operator is defined: within a given frequency some selected individuals are exchanged among subpopulations, introducing a new source of diversity in the EA.

```
initialize(P(0))
 generation ← 0
while (not stopcriteria) do
        evaluate(P(generation))
        parents ← selection(P(generation))
        offspring ← variation operators(parents)
        newpop ← replace(offspring, P(generation))
        generation ++
        P(generation) ← newpop
        if (sendmigrants)
                migrants ← selection for migration(P(generation)
                sendmigration(migrants)
        end
        if (recvmigrants)
            inmigrants ← recvmigration()
            P(generation) ← insert(immigrants, P(generation))
        end
end
return best solution ever found
```

Algorithm 3: Schema of a parallel evolutionary algorithm.

Algorithm 3 shows the generic schema for a distributed subpopulation parallel EA. It follows the generic schema of an EA, but including the new migration operator. Two conditions control the migration procedure: *sendmigrants* determines when the exchange of individuals takes place, and *recvmigrants* establishes whether a foreign set of individuals has to be received or not. *Migrants* denotes the set of individuals to exchange with some other subpopulation, selected according to a given policy. The schema explicitly distinguishes between *selection for reproduction* and *selection for migration*; they both return a selected group of individuals to perform the operation, but following potentially different policies. The *sendmigration* and *recvmigration* operators carry out the exchange of individuals among subpopulations according to a connectivity graph defined over them, most usually a unidirectional ring.

3.4 The parallel micro-CHC evolutionary algorithm

By splitting the global population into several subpopulations, PEAs significantly increase the computational efficiency of the search due to the limited interactions and the reduced population size within each subpopulation. However, EAs quickly lose diversity in the solutions when using small populations: the search suffers a premature convergence, and the quality of solutions stagnates. In CHC, the mating restriction policy and the reinitialization operator are usually not powerful enough to provide the required diversity to avoid premature convergence when using very small populations (i.e. less than 10 individuals). Many alternatives have been proposed in the related literature to overcome the loss of diversity on EAs. In the quest for designing a fast and accurate version of the CHC algorithm for solving scheduling problems, a parallel micro-CHC algorithm has been developed in our research group.

The proposed parallel micro-CHC algorithm combines a distributed subpopulation parallel model of the original CHC structure (using HUX and mating restriction) with two key concepts from the micro-genetic algorithm by Coello and Pulido (2001): an external population of elite solutions stored during the search, and an accelerated reinitialization using a specific randomized version of a well-known local search method to provide diversity within each subpopulation. A micro-population of eight individuals is used in each subpopulation of the parallel micro-CHC algorithm. The size of the external population is three individuals, and a simple remove-of-the-worst strategy is used each time a new individual is inserted in the elite set.

4. A parallel micro-CHC applied to the bi-objective HCSP

This section introduces the software library used to implement the parallel micro-CHC algorithm, and it also presents the implementation details to solve the HCSP.

4.1 The MALLBA library

MALLBA (Alba et al., 2006) is a library of algorithms for optimization that can deal with parallelism in a user-friendly and at the same time, efficient manner. The library implements several EAs as generic templates in *software skeletons* to be instantiated by the user with the specific features when solving a given problem. These templates incorporate the knowledge related to the resolution method, its interactions with the problem, and the parallel considerations. Skeletons are implemented by a set of *required* and *provided* classes implemented in C++ that represent an abstraction of the entities participating in the resolution method:

- The *provided classes* implement internal aspects of the skeleton in a problem-independent way. The most important provided classes are *Solver* (the algorithm) and *SetUpParams* (for setting the parameters of the implemented algorithms).
- The *required classes* specify information related to the problem. The *Problem* and *Solution* required classes encapsulate the problem-dependent entities needed by the resolution method. Depending on the skeleton, other classes may be required.

The MALLBA library is publicly available to download at the University of Málaga website http://neo.lcc.uma.es/mallba/easy-mallba.

The implementation of parallel micro-CHC is based on the CHC skeleton provided by MALLBA. Additional code was incorporated into the CHC skeleton to define and manage the elite population, to implement the special reinitialization and local search operators, and to include other features related to the bi-objective HCSP resolution. The details about the problem encoding, the evolutionary operators, and specific features are provided in the next subsections.

4.2 Problem encoding

A machine-based encoding was used to represent candidate solutions for the bi-objective HSCP in parallel micro-CHC. The machine-based encoding is a bidimensional structure that stores for each machine the list of tasks assigned to it, ordered regarding their execution precedence. This encoding simplifies the calculation of the fitness function for solutions after applying the variation operators. Figure 1 presents an example of the machine-oriented encoding.

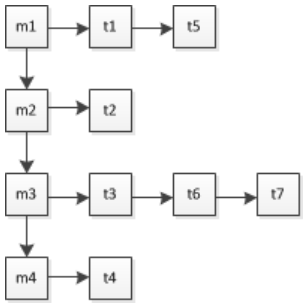

Figure 1: Machine oriented encoding.

4.3 Initialization

Several methods have been proposed to initialize the population when applying EAs to the single objective HCSP (Braun, 2001; Xhafa, 2008). Traditional deterministic scheduling heuristics have been used to start the evolutionary search from a set of useful suboptimal schedules. In the parallel micro-CHC algorithm applied to the bi-objective HCSP, one individual in the whole population is initialized using Min-Min and other using Sufferage, and one individual in each subpopulation is initialized using MCT. As for the rest of the population, 60% is initialized using a randomized version of MCT, and the remaining 40% is initialized at random.

4.3 Recombination

The parallel micro-CHC algorithm uses HUX to recombine characteristics of two solutions. In the HUX implementation for the bi-objective HCSP, a group of tasks to be recombined is chosen with uniform probability (0.5). When a chosen task is assigned to different machines in the selected parents, the machine to place that task in each offspring is chosen to optimize either the *makespan* or the *wrr* objectives with uniform probability (0.5).

4.4 Reinitialization

The reinitialization operator performs small perturbations in a given schedule, aimed at providing diversity to the population, in order to avoid the search from getting stuck in local optima. When a stagnation situation is detected two actions are performed: the local search operator is applied to the best solution in the population to further improve it, and after that, simple moves and swaps of tasks are randomly applied to the rest of the population to restart the search process from a hopefully unexplored location in the solution space.

4.5 Local search

Many approaches have been proposed to provide diversity and improve the efficacy of the search in EAs to solve the single objective HCSP. Most of the previous works concluded that local search methods are needed to find accurate schedules in short times. The works of Xhafa et al. (2007; 2008) explored several local search operators for solving low-dimension HCSP instances, but many of the proposals become ineffective when the problem instances grow.

In order to improve the population diversity, the parallel micro-CHC algorithm incorporates a randomized version of Problem Aware Local Search (PALS) (Alba and Luque 2007). Algorithm 4 presents the pseudo-code of the randomized PALS operator applied in the parallel micro-CHC algorithm for the bi-objective HCSP.

```
M ← Select a list of #MAX_MACH machines
end_search ← false
while ((count(M) > 0) and not end_search) do
        m = pop(M)
        trials ← 0
        while (trials < MAX_TRIALS) and (not end_search)) do
                Δ_BEST ← 0
                for (t_M = start_M to TOP_M) do
                        {Iterate on tasks of machine m}
                        for (t_T = start_T to TOP_T)
                                {Iterate on tasks of other machines}
                                Δ_CURRENT ← SwapFitnessImprovement (t_M,t_T)
                                if ( Δ_CURRENT > Δ_BEST ) then
                                        best_swap ← (t_M,t_T)
                                        Δ_BEST ← Δ_CURRENT
                                end
                        end
                end
                trials++
                if ( Δ_BEST > 0 ) then
                        s ← DoSwap(best_swap)
                        end_search ← true
                end
        end
end
```

Algorithm 4: Schema of the randomized PALS for the bi-objective HCSP.

Working on a given schedule *s*, the randomized PALS for the bi-objective HCSP selects a collection of machines *M* to perform the search. With high probability the search focuses on the machines with the largest fitness contribution, but it also introduces a chance of improving the fitness for other machines. Then, for each machine, the outer cycle iterates on TOP_M tasks assigned to machine *m* (randomly starting in task $start_M$), while the inner cycle iterates on TOP_T tasks assigned to other machines (randomly starting in task $start_T$). For each pair (t_M, t_T), the fitness improvement when swapping tasks t_M and t_T is computed, storing the best improvement found for the whole schedule on the $TOP_M \times TOP_T$ swaps evaluated. After the double cycle ends, the best move found is applied only if it improves the current solution fitness. For each machine, the process is applied until finding a schedule which improves the original fitness or after performing MAX_TRIALS attempts. If a better schedule is not found for the current machine, then the swap-search is performed on the next machine in M.

5. Experimental analysis

This section describes the computational platform used to develop and evaluate the proposed parallel micro-CHC algorithm and introduces the set of HCSP instances used to evaluate the efficacy of the proposed method. After that, the parameter settings experiments are presented. Last, the experimental results when solving realistic MR-HCSP instances are analyzed, by presenting a comparison with the results obtained using deterministic techniques.

5.1 Development and execution platform

The proposed parallel micro-CHC algorithm was implemented in C++, using the MALLBA library with the version 1.2.7p1 of MPICH, a well-known implementation of MPI (Gropp 1994). The experimental analysis was performed using a cluster with four Dell PowerEdge servers (QuadCore Xeon E5430 at 2.66 GHz, 8 GB RAM), CentOS Linux 5.2 operating system and Gigabit Ethernet LAN (cluster website: http://www.fing.edu.uy/cluster).

5.2 Test instances

A specific set of 60 HCSP instances was used to evaluate the proposed parallel micro-CHC algorithm: 12 standard instances from Braun et al. (2001) with dimension (tasks×machines) 512×16, and 48 large dimension instances (24 with dimension 1024×32 and 24 with dimension 2048×64). These large dimension instances were generated following the methodology by Ali et al. (2000) and using the two parametrization values proposed by Ali et al. (2000) and Braun et al. (2001). The new instances model realistic scheduling scenarios on small and medium-sized heterogeneous computing and grid infrastructures.

5.3 Parameter settings

A stopping criterion fixed at 90 s. of execution time is used for the parallel micro-CHC algorithm, following several works proposing efficient parallel EAs applied to the single-objective HCSP by Xhafa (2007; 2008) and the previous work (Nesmachnow et al. 2011). This is an efficient time limit for scheduling in realistic distributed HC and grid infrastructures such as volunteer-computing platforms, distributed databases, etc., where large tasks -with execution times in the order of minutes, hours and even days- are submitted to execution.

Instead of fixing an arbitrary set of parameter values for the parallel micro-CHC, a configuration analysis was performed to determine the best values for the crossover (p_C) and reinitialization (p_R) probabilities, the number of subpopulations (#I) and their size (#pop). The parameter setting experiments were performed over a subset of six HCSP instances with dimension 512×16. The candidate values for the studied parameters were: p_C: 0.8, 0.9, 1.0; p_R: 0.7, 0.9, 1.0; #I: 4, 8, 16; #pop: 10, 20, 40. The parameter configuration that obtained the best results in the experiments was: $p_C = 1.0$, $p_R = 0.9$, #I =16, #pop =10.

5.2 Results and discussion

This subsection reports the experimental results when solving the MR-HCSP. The results for the problem instances by Braun et al. are presented separately, since these benchmark instances have been very often solved by the research community. In addition, since a large set of high dimension MR-HCSP instances were solved (48), a graphic summary and the average improvements over the Min-Min results for these instances are reported.

Table 1 presents the results computed by parallel micro-CHC for the standard problem instances by Braun et al. The best, average, and standard deviation on the two metric results obtained in the 30 independent executions performed for each problem instance are reported. The best improvements achieved by parallel micro-CHC over the best deterministic heuristic results (computed by Min-Min) are also shown.

Instance	Min-Min		EA (avg. ± std. dev)		EA (best)		Improvement	
	Makespan	WRR	makespan	*wrr*	makespan	*wrr*	makespan	*wrr*
u_c_hihi.0	8460680.0	46084.6	7899879.3±33641.3	38151.0±578.9	7869300.0	38516.8	7.0 %	16.5 %
u_c_hilo.0	161805.0	36170.9	157442.5±640.9	28027.1±660.9	157081.0	28158.9	2.9 %	22.2 %
u_c_lohi.0	275837.0	48269.5	259246.8±1672.3	39137.8±809.8	258165.0	39084.0	6.4 %	19.0 %
u_c_lolo.0	5441.4	36057.9	5352.5±21.1	27352.8±784.1	5328.8	27156.8	2.1 %	24.7 %
u_i_hihi.0	3513920.0	17862.1	3087658.6±23247.9	16639.7±533.8	3046180.0	16502.5	13.3 %	7.6 %
u_i_hilo.0	80755.7	23502.6	76100.3±669.1	20710.6±113.0	75691.3	20664.5	6.3 %	12.1 %
u_i_lohi.0	120518.0	17630.7	107372.7±586.7	16318.4±101.8	106358.0	16286.5	11.8 %	7.6 %
u_i_lolo.0	2785.6	24238.9	2624.0±16.0	21588.9±114.3	2608.8	21744.2	6.4 %	10.3 %
u_s_hihi.0	5160340.0	25884.0	4473874.7±43251.8	21739.5±365.2	4441760.0	21554.3	13.9 %	16.7 %
u_s_hilo.0	104375.0	27566.5	99679.6±521.9	23536.8±222.0	99639.8	23475.9	4.5 %	14.8 %
u_s_lohi.0	140284.0	26006.6	130829.4±1060.0	21817.1±410.4	129560.0	21791.8	7.7 %	16.2 %
u_s_lolo.0	3806.8	27608.1	3601.1±18.6	23347.4±159.6	3578.5	23341.5	6.0 %	15.5 %

Table 1: Bi-objective HCSP results for the instances from Braun et al.

The results in Table 1 indicate that accurate schedules are computed when using the parallel micro-CHC algorithm. When compared with the deterministic Min-Min results, significant improvements on the makespan and *wrr* metrics are obtained (up to **13.9%** in the makespan, and up to **22.2%** in the *wrr*). The proposed EA showed a robust behavior, indicated by the very small standard deviation values in both metrics (below 1.6%) obtained in the 30 executions performed.

Figure 1 reports the best improvements obtained by the parallel micro-CHC over the Min-Min results when solving the 48 large dimension MR-HCSP instances (24 with dimension 1024×32 and 24 instances with dimension 2048×64).

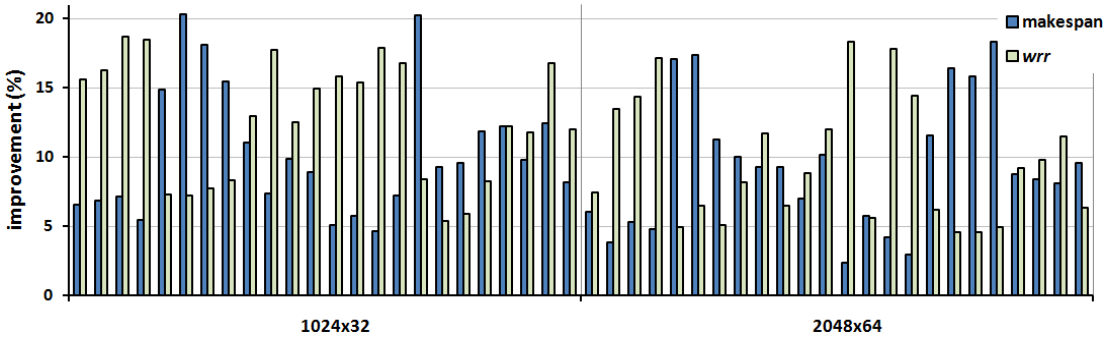

Figure 2: Improvements over Min-Min for instances with dimension 1024×32 and 2048×64.

The improvements reported in Figure 2 demonstrate that the proposed parallel micro-CHC algorithm has a good scalability behavior when solving large dimension MR-HCSP instances. Improvements up to **20.36%** in the makespan values and up to **18.75%** in the *wrr* values (with respect to the Min-Min solution) were obtained.

Table 2 summarizes the average improvements over the Min-Min results (averaged for the 24 MR-HCSP instances for each dimension), obtained when solving the MR-HCSP for the three problem dimensions studied.

dimension	avg. makespan improvement	avg. *wrr* improvement
512×16	7.35 %	15.26 %
1024×32	10.35 %	12.70 %
2048×64	9.33 %	9.56 %

Table 2: Overall average improvements for the bi-objective HCSP for all dimensions.

The results in Table 2 indicate that acceptable accurate are computed by the proposed EA, even when solving high-dimension HCSP scenarios. The average improvements over the Min-Min solution were up to 10.35% in the makespan values and up to 15.26 in the *wrr* values. The improvements in the *wrr* metric slightly decreased when solving the largest problem instances, suggesting that there is still room to further improve the proposed evolutionary search.

6. Conclusions and future work

This work has presented a parallel micro-CHC evolutionary algorithm to solve a bi-objective version of the scheduling problem in heterogeneous grid computing environments. The problem formulation considers two metrics that account for the resource utilization time (makespan) and the user point-of-view (the new *weighted response ratio* metric).

The proposed evolutionary algorithm combines the original structure of CHC with a distributed subpopulation parallel model using micro populations and other specific features inspired from multiobjective optimization. A specific local search operator was designed to compute accurate schedules in reduced execution times.

The experimental analysis performed on both standard and new problem instances showed that the parallel micro-CHC evolutionary algorithm is able to obtain accurate results when using a fixed time stopping criterion of 60 s. Improvements up to **10.35**% in the makespan metric and up to **15.26**% in the weighted response ratio metric were obtained when comparing with the best results computed using well-known deterministic scheduling algorithms. In addition, the parallel micro-CHC evolutionary algorithm showed a good scalability behavior when solving the new large-sized problem instances.

Two main lines have been identified for future work: to further improve the results, and to tackle even larger problem scenarios. Regarding the first line, new operators should be devised in order to compute more accurate values of the makespan and weighted response ratio metrics by performing a more intelligent neighborhood search in the randomized PALS operator. On the other hand, advanced parallel computing techniques (such as multithreading implementations of the evolutionary operators) should be applied to the current implementation of the parallel micro-CHC EA in order to improve the computational efficiency, allowing to face even larger instances of the scheduling problem that model nowadays heterogeneous and grid computing systems.

References

Alba, E. (2005), *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.

Alba, E., Almeida, F., Blesa, M., Cotta, C., Diaz, M., Dorta, I., Gabarró, J., González, J., León, C., Moreno, L., Petit, J., Roda, J., Rojas A., and Xhafa, F. (2006), MALLBA: A library of skeletons for combinatorial optimisation. *Parallel Computing*, 32(5-6):415–440.

Alba, E., and Luque, G. (2007), A new local search algorithm for the DNA fragment assembly problem. In *Proc. of 7th European Conference on Evolutionary Computation in Combinatorial Optimization*, volume 4446 of Lecture Notes in Computer Science, 1–12. Springer.

Alba, E., and Tomassini, M. (2002), Parallelism and evolutionary algorithms. *IEEE Trans. Evol. Comput*., 6(5):443–462.

Ali, S., Siegel, H., Maheswaran, M., Ali, S. and Hensgen, D. (2000), Task execution time modeling for heterogeneous computing systems. In *Proc. of the 9th Heterogeneous Computing Workshop*, 185, Washington, DC, USA. IEEE Computer Society.

Back, T. Fogel, D., and Michalewicz, Z. (Eds.). (1997), *Handbook of Evolutionary Computation*, IOP Publ. Ltd., Bristol, UK.

Braun, T., Siegel, H., Beck, N., Boloni, L., Maheswaran, M., Reuther, A., Robertson, J., Theys, M., Yao, B., Hensgen, D. and Freund, R. (2001). A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. *J. Par. Distrib. Comput*.,61(6):810–837.

Buyya, R. (2002). Economic-based Distributed Resource Management and Scheduling for Grid Computing. Ph.D. Thesis, Monash University, Melbourne, Australia.

Coello, C., Lamont, G., and Veldhuizen, D. (2006). Evolutionary Algorithms for Solving Multi-Objective Problems. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Coello, C., and Pulido, G. (2001). A micro-genetic algorithm for multiobjective optimization. In *Proc. of the 1st Int. Conf. on Evolutionary Multi-Criterion Optimization*, 126–140, London, UK.

Dong, F., Luo, J., Gao, L., and Ge, L. (2006). A Grid Task Scheduling Algorithm Based on QoS Priority Grouping. *In Proceedings of the Fifth International Conference on Grid and Cooperative Computing (GCC '06)*. IEEE Computer Society, Washington, DC, USA, 58-61.

Eshaghian, M. (1996), Heterogeneous Computing. Artech House, 1996.

Eshelman, L. (1991), The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination. In *Foundations of Genetics Algorithms*, 265–283. Morgan Kaufmann.

Foster, I. and Kesselman, C. (1998). The Grid: Blueprint for a Future Computing Infrastructure. Morgan Kaufmann.

Garey, M. and Johnson, D. (1979), Computers and intractability. Freeman.

Gropp, W., Lusk, E., Doss, N., and Skjellum, A. (1996), A high performance, portable implementation of the MPI message passing interface standard. *Parallel Computing*, 22(6):789–828.

Kwok, Y. and Ahmad, I. (1999), Static scheduling algorithms for allocating directed task graphs to multiprocessors. *ACM Computing Surveys*, 31(4):406–471.

Leung, J., Kelly, L., and Anderson, J. (2004), Handbook of Scheduling: Algorithms, Models, and Performance Analysis. CRC Press.

Monte, J. and Pattipati, K. (2002), Scheduling Parallelizable Tasks to Minimize Make-Span and Weighted Response Time. *IEEE Trans. on Syst., Man, and Cybern.*, Part A, 32(3):335–345.

Nesmachnow, S., Cancela, H., and Alba, E. (2011), Heterogeneous computing scheduling with evolutionary algorithms. *Soft Computing*, 15(4):685–698.

Stallings, W. (2001), Operating Systems: Internals and Design Principles. Prentice Hall.

Wang, L., Siegel, H., Roychowdhury, V., and Maciejewski, A. (1997), Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach. *J. Par. Distrib. Comput.*, 47(1):8–22.

Xhafa, F., Carretero, J., and Abraham, A. (2007), Genetic algorithm based schedulers for grid computing systems. *Int. Journal of Innovative Computing Information and Control*, 3(5):1–19.

Xhafa, F., Alba, E., and Dorronsoro, B. (2008), Efficient batch job scheduling in grids using cellular memetic algorithms. *Journal of Mathematical Modelling and Algorithms*, 7(2):217–236.

Zomaya, A. and Teh, Y. (2001), Observations on using genetic algorithms for dynamic load-balancing. IEEE Trans. Parallel Distrib. Syst., 12(9):899–911.