Multiobjective scheduling on distributed heterogeneous computing and grid environments using a parallel micro-CHC evolutionary algorithm

Sergio Nesmachnow
Centro de Cálculo, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
Email: sergion@fing.edu.uy

Santiago Iturriaga
Centro de Cálculo, Facultad de Ingeniería
Universidad de la República
Montevideo, Uruguay
Email: siturria@fing.edu.uy

***Abstract*—This work presents the application of a parallel micro-CHC evolutionary algorithm to the scheduling problem in heterogeneous computing environments, to minimize the makespan and weighted response ratio objectives. The studied problem is NP-hard, and significant effort has been made to develop efficient methods to compute accurate schedules in reduced execution times. Efficient numerical results are reported in the experimental analysis performed on both well-known and new large problem instances that model medium-sized grid environments. The parallel micro-CHC achieves a high problem solving efficacy and shows a good scalability behavior when facing high dimension instances.**

***Keywords*-parallel evolutionary algorithms; scheduling; heterogeneous computing; grid;**

I. Introduction

Nowadays, distributed computing platforms usually include a large number of heterogeneous devices that provides the computing power needed to solve complex problems arising in many areas of application. The expression *grid computing* defines the set of distributed computing techniques that work over a large loosely-coupled virtual supercomputer, formed by many heterogeneous computing devices widespread around the globe. This infrastructure has made it feasible to provide pervasive and cost-effective access to a collection of distributed computing resources for solving problems that demand large computing power [1].

A key problem when using distributed heterogeneous computing (HC) consists in finding a scheduling for a set of tasks to be executed. The goal is to assign the computing resources by satisfying some efficiency criteria, usually related to the total execution time, the resource utilization, or the quality of service (QoS). Scheduling problems have been widely studied in operational research [2], [3], and the *heterogeneous computing scheduling problem* (HCSP) became important due to the popularization of distributed computing and the growing use of heterogeneous clusters [4], [5].

Traditional scheduling problems are NP-hard [6], thus classic exact methods are only useful for solving instances of reduced size. Heuristics and metaheuristics methods are then used to compute efficient schedules in reasonable times for large problem instances.

Evolutionary algorithms (EAs) have emerged as flexible and robust metaheuristics for scheduling, achieving the high level of problem solving efficacy also shown in many other application areas [7]. EAs usually require larger execution times than ad-hoc scheduling heuristics, but they can find better solutions. So, EAs are competitive schedulers for HC systems where large tasks (with execution times of minutes, hours, or days) are submitted for execution. In order to further improve the efficiency of EAs, parallel implementations have been used to enhance and speed up the search, allowing to reach high quality results in reasonable execution times even for hard-to-solve optimization problems [8].

EAs have been applied to the single-objective HCSP [9]–[13], but the simultaneous optimization of several objectives has been seldom tackled. Realistic HCSP instances in grid environments have rarely been faced, mainly due to the complexity of dealing with the underlying high-dimension optimization problem. In addition, few works have studied parallel methods and their ability to use the computing power of large clusters to improve the search. Thus, there is still room to contribute in these lines of research by studying highly efficient parallel EAs, able to deal with large-size HCSP instances by using the computational power of parallel and distributed environments.

In this line of work, the main contribution of this article is to solve a new HCSP variant that proposes the simultaneous optimization of the makespan and weighted response ratio objectives, using a parallel micro-CHC (pμ-CHC) algorithm. Efficient numerical results are reported in the experimental analysis performed on both well-known and large problem instances. The analysis shows that pμ-CHC is able to achieve high problem solving efficacy, and also to exhibit a good scalability when solving high dimension problem instances.

The manuscript is structured as follows. Section II presents the problem formulation. Section III introduces EAs and describes the pμ-CHC algorithm. Section IV describes the implementation details of pμ-CHC applied to the HSCP. The discussion of the experimental analysis and results are presented in Section V, while the conclusions and possible lines for future work are formulated in Section VI.

II. Scheduling in parallel HC systems

A HC system is composed of many computers (*machines*), and a set of tasks to be executed on the system. A task is the atomic unit of workload, so it cannot be divided into smaller chunks, nor interrupted after it is assigned to a machine. The execution times of any task vary from one machine to another, so there will be competition among tasks for using those machines able to execute them in the shortest time.

The most usual objective to minimize in scheduling is the *makespan*, defined as the time spent from the moment when the first task begins to the moment when the last task is completed. However, other efficiency objectives have been considered in scheduling problems [3]. When considering several objectives, usually in conflict with each other, scheduling is a complex multiobjective problem.

The response time of each task is an important objective from the user point-of-view. The *response ratio* [14] evaluates the sum of the response time, but it inversely depends on the task estimated completion time, thus favoring short tasks over long tasks. Minimizing the response ratio of each task is important for the QoS of the system, but from an economic cost perspective not every task contributes the same to the QoS. A common approach in modern computational grid environments is to group task in different classes according to their importance, or priority [15]. To model these two previous features, in this work we introduce the *weighted response ratio* (*wrr*) objective: each task is assigned a positive weight p_i which represents the relative priority of the task in the system. The *wrr* is then defined as the total response ratio of each task multiplied by its weight in the system, p_i. This kind of weighted approach has been previously applied to the response time objective [16].

The following is the mathematical formulation for the HCSP variant studied in this work (MR-HCSP):

- given an HC system composed of a set of machines $M = \{m_1, m_2, \ldots, m_L\}$ and a collection of tasks $T = \{t_1, t_2, \ldots, t_N\}$ to be executed on the system,
- let there be an *execution time function* $ET : T \times M \rightarrow \mathbf{R}^+$, 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* $P : T \rightarrow \mathbf{N}^+$, where $P(t_i)$ is the priority of the task t_i in the system,
- let $F(t_i)$ be the finishing time of task t_i in the system;
- the goal of the MR-HCSP is to find an assignment of tasks to machines (a function $f : T^N \rightarrow M^L$) which simultaneously minimizes the *makespan* (Eq. 1), and the *weighted response ratio* (Eq. 2).

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

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

To deal with both objectives, this work combines the makespan and *wrr* in a linear aggregation function. Although the aggregate function approach is often outperformed by Pareto-based methods, it is a common approach in the literature, with three main advantages [17]: i) it is suitable for optimization problems with a convex Pareto front, ii) it is computationally efficient (recommended when the times available for search is short), and iii) it is well-suited for the decomposition approach used in pμ-CHC. In the MR-HCSP, since the makespan and *wrr* objectives are in different units, they have to be normalized before the aggregation.

Several deterministic heuristics have been proposed for HC scheduling [18]. Three of them have been used in this work to seed the population in the pμ-CHC algorithm:

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 ET 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.

III. Evolutionary algorithms

EAs are non-deterministic methods that emulate the evolution of species in nature, which have been successfully applied for solving optimization problems underlying many complex real-life applications in the last twenty years [7].

An EA is an iterative technique (each iteration is called a *generation*) that applies stochastic operators on a population of individuals which encode tentative solutions of the problem in order to improve their *fitness*, a measure related to the objective function. The initial population is generated at random or by using a specific heuristic for the problem. An evaluation function associates a fitness value to every individual, indicating its suitability to the problem. Iteratively, the probabilistic application of *variation operators* like the *recombination* of two individuals or random changes (*mutations*) in their contents are guided by a selection-of-the-best technique to tentative solutions of higher quality.

The stopping criterion usually involves 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 for the *selection* of individuals to recombine and to determine which new individuals *replace* the older ones in each new generation. The EA returns the best solution found, regarding the fitness function values.

A. The CHC algorithm

The CHC acronym stands for "Cross generational elitist selection, Heterogeneous recombination, and Cataclysmic mutation" [19]. CHC (Algorithm 1) is a specialization of a traditional 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.

Algorithm 1 Schema of the CHC algorithm.

```
1:  initialize(P(0))
2:  generation ← 0
3:  ux_threshold ← 1/4 * chromosomeLength
4:  while not stopcriteria do
5:     parents ← selection(P(generation))
6:     if hamming_distance(parents) ≥ ux_threshold then
7:        offspring ← HUX(parents)
8:        evaluate(offspring)
9:        newpop ← replace(offspring, P(generation))
10:    end if
11:    if newpop == P(generation) then
12:       distance − −
13:    end if
14:    generation ++
15:    P(generation) ← newpop
16:    if ux_threshold == 0 then
17:       re-initialization(P(generation))
18:       ux_threshold ← 1/4 * chromosomeLength
19:    end if
20: end while
21: return best solution ever found
```

B. 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 processing elements, parallel evolutionary algorithms (PEAs) allow reaching high quality results in a reasonable execution time even for hard-to-solve optimization problems [8]. The PEAs proposed in this work are categorized within the *distributed subpopulations* model according the classification by Alba and Tomassini [20]: the original population is divided into several subpopulations (*demes*), separated geographically from each other. Each deme runs a sequential EA, so individuals are able to interact only with other individuals in the deme. An additional *migration* operator is defined: occasionally some selected individuals are exchanged among demes, introducing a new source of diversity in the EA.

C. The parallel micro-CHC evolutionary algorithm

By splitting the global population, PEAs allow achieving high computational efficiency due to the limited interaction and the reduced population size within each deme. However, EAs quickly lose diversity in the solutions when using small populations, and the search stagnates, due to a premature convergence. The mating restriction and the reinitialization operator used in CHC are usually not powerful enough to provide the required diversity to avoid premature convergence in the parallel model when using very small populations. The pμ-CHC algorithm was conceived as a fast and accurate version of CHC for solving optimization problems.

pμ-CHC is based on theoretical concepts from Goldberg [21], some ideas about micro-genetic algorithm (μ-GA) by Krishnakumar [22], and the implementation of μ-GA for multiobjective optimization by Coello and Pulido [23].,

pμ-CHC combines a distributed subpopulation parallel model of the original CHC (using HUX and mating restriction) with two key concepts from μ-GA: an external population of elite solutions stored during the search, and an accelerated reinitialization using a randomized version of a well-known local search (LS) method to provide diversity within each deme. A micro-population (i.e., 8 to 12 individuals) is used in each deme of pμ-CHC. The size of the external population is three individuals, and a simple remove-of-the-worst strategy is used when a new individual is inserted in the elite set. Fig. 1 presents a graphic schema of the distributed subpopulations model used in pμ-CHC.

In order to deal with the multiobjective MR-HCSP, a decomposition approach similar to the one used in the MOEA/D algorithm for multiobjective optimization [24] is employed in pμ-CHC. Each subpopulation solves a specific subproblem, each one using a different pair of weights for the makespan and wrr objectives. This approach allows to sample the Pareto front of the problem, despite that the method does not use a Pareto-based fitness assignment.

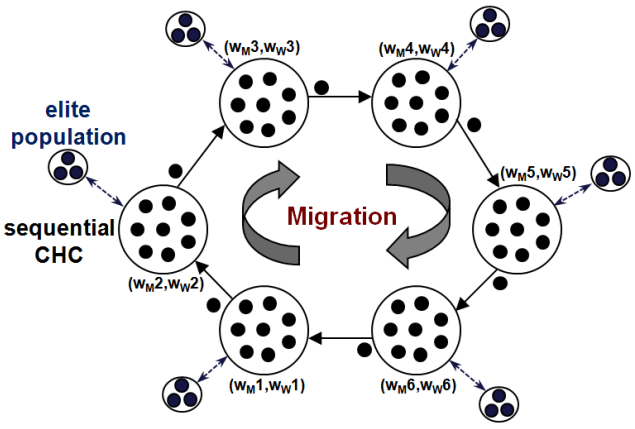


Figure 1. Schema of the decomposition approach for multiobjective optimization in the pμ-CHC algorithm.

IV. A MICRO-CHC ALGORITHM FOR THE MR-HCSP

This section presents the implementation details of the pμ-CHC algorithm to solve the MR-HCSP.

A. The MALLBA library

MALLBA [25] is a library of optimization algorithms that deals with parallelism in a user-friendly and efficient manner. MALLBA implements EAs and other metaheuristics as generic templates in *software skeletons*, which incorporate the knowledge related to the resolution method, its interactions with the problem, and the parallelism. Skeletons are implemented by *required* and *provided* C++ classes that abstracts the entities 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 algorithms' parameters).
- The *required classes* specify information related to the problem. Each skeleton includes the *Problem* and *Solution* required classes, that encapsulate the problem-dependent entities needed by the resolution method.

pμ-CHC was implemented using the CHC skeleton in MALLBA. Additional code was included to define and manage the elite population, and to implement the variation operators and other features for the MR-HCSP resolution.

B. Problem encoding

A machine-based encoding (see an example in Figure 2) was used to represent MR-HCSP solutions. A bidimensional dynamic structure is used to store the list of tasks assigned to each machine in the system, ordered by the execution precedence. By using the machine-oriented encoding pμ-CHC significantly simplifies the fitness calculation for MR-HCSP solutions after the variation operators are applied.

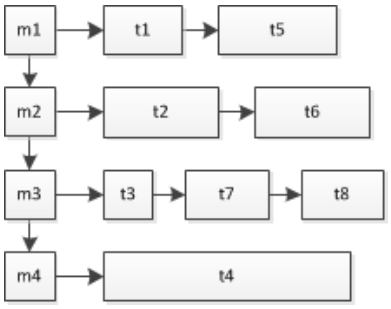


Figure 2. Machine-oriented encoding.

C. Initialization

Several methods have been proposed to generate the initial population when applying EAs to the single-objective HCSP [10], [12], [13], [26]. A usual option is to use ad-hoc scheduling heuristics to start the evolutionary search from a set of useful suboptimal schedules.

In the pμ-CHC applied to the MR-HCSP, exactly one individual in the whole population is initialized with Min-Min and other one with Sufferage, and one individual in each subpopulation is initialized with MCT. As for the rest of the population, 60% is initialized using a randomized MCT and the remaining 40% is initialized at random.

D. Exploitation: recombination

The pμ-CHC uses HUX to recombine characteristics of two solutions. In the MR-HCSP HUX implementation, a group of tasks to be recombined are chosen with uniform probability (0.5). When a chosen task is assigned to different machines in its 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).

E. Exploration: 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 different actions are performed: the LS operator is applied to the best solution in the population to further improve it and after that, simple tasks move-and-swap are randomly applied to the rest of the population to restart the search process from a hopefully unexplored location in the solution space.

F. Local search: randomized PALS

Most of the related works concluded that LS methods are needed within any EA to find accurate HCSP solutions in short times [12], [13], [26]. In order to improve the population diversity, pμ-CHC incorporates a randomized version of Problem Aware Local Search (PALS) [27].

Algorithm 2 presents the pseudo-code of the randomized PALS for the MR-HCSP. Working on a given schedule s, the method selects a collection of machines M to perform the search. With high probability the search focuses on improving the assignment for the machines contributing the most to the global solution fitness, but it also introduces a chance of improving the assignment for other machines too. 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.

Algorithm 2 Randomized PALS for the MR-HSCP.

```
1:  M ← Select a list of #MAX_MACH machines
2:  end_search ← false
3:  while count(M) > 0 and not end_search do
4:      m = pop(M)
5:      trials ← 0
6:      while trials < MAX_TRIALS and not end_search do
7:          Δ_best ← 0
8:          for t_M = start_m to TOP_M do
9:              {Iterate on tasks of machine m}
10:             for t_T = start_t to TOP_T do
11:                 {Iterate on tasks of other machines}
12:                 Δ_current ← SwapImprovFitness(t_M, t_T)
13:                 if Δ_current > Δ_best then
14:                     best_swap ← (t_M, t_T)
15:                     Δ_best ← Δ_current
16:                 end if
17:             end for
18:         end for
19:         trials ← trials + 1
20:         if Δ_best > 0 then
21:             s ← DoSwap(best_swap)
22:             end_search ← true
23:         end if
24:     end while
25: end while
```

G. Migration

The migration operator in pμ-CHC considers the subpopulations connected in a unidirectional ring topology. It uses an elitist selection for migration policy that exchanges the best two individuals between demes (the received individuals substitutes the worst ones in the destination deme) with a migration frequency of 100 generations.

V. Experimental analysis

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

A. HCSP instances

To evaluate the proposed pμ-CHC, a specific set of 84 MR-HCSP instances was randomly generated following the methodology by Ali et al. [28], and using two parametrization values [10], [28]. These instances model realistic medium-sized heterogeneous computing and grid infrastructures. Four problem dimensions were studied: (tasks$\times$machines) 512$\times$16, 1024$\times$32, 2048$\times$64, and 4096$\times$128.

B. Development and execution platform

pμ-CHC was implemented in C++, using MALLBA and MPICH 1.2.7p1. The experimental analysis was done in a cluster with four Dell PowerEdge (QuadCore Xeon E5430, 2.66 GHz, 8 GB RAM), CentOS Linux and Gigabit Ethernet.

C. Performance metrics

Several metrics for multiobjective optimization have been considered in this work to evaluate pμ-CHC [17].

The *efficacy* metrics evaluate the quality of results by analyzing the convergence towards the Pareto front:

- The number of non-dominated solutions found (*ND*).
- Inverted Generational Distance (IGD): the (normalized) sum of the distances between the non-dominated solutions found and a set of uniformly distributed points v in the Pareto front P^* (Eq. 3).

$$IGD = \frac{1}{|P^*|} \sum_{v \in P^*} d(v, P) \tag{3}$$

The *diversity* metrics evaluate the distribution of the solutions, analyzing the ability of sampling the Pareto front:

- Spread: evaluates the dispersion of non-dominated solutions in the calculated Pareto front, considering the extreme points of the true Pareto front (Eq. 4).

$$spread = \frac{\sum_{h=1}^{k} {d_h}^e + \sum_{i=1}^{ND} \left(\bar{d} - d_i\right)^2}{\sum_{h=1}^{k} {d_h}^e + ND \times \bar{d}} \tag{4}$$

- Relative hypervolume (RHV): evaluates the ratio of the volumes (in the objective functions space) covered by the computed Pareto front and by the real Pareto front.

In Eq. 4, d_i is the distance between the i-th solution in the calculated Pareto front and its nearest neighbor, $\bar{d}$ is the average of all d_i, and ${d_h}^e$ is the distance between the extreme of the h-th objective function in the true Pareto front and the closest point in the calculated Pareto front.

The true Pareto front, which is unknown for the MR-HCSP instances studied, was approximated by gathering the non-dominated solutions found in the 30 independent executions performed for each instance.

D. Parameter settings

The main objective of the research is to study the ability of pμ-CHC to *efficiently* solve the MR-HCSP. Thus, a stopping criterion fixed at 90 s. of execution time is used, following several works on parallel EAs applied to the single-objective HCSP [12], [13], [29]. This is an efficient time limit for scheduling in realistic distributed HC and grid infrastructures where large tasks -with execution times in the order of minutes, hours and even days- are submitted to execution.

A configuration analysis studied the best values for the crossover (p_C) and reinitialization (p_R) probabilities, the number of demes ($\#I$) and its size ($\#pop$) in pμ-CHC. The parameter setting analysis was performed over a subset of six MR-HCSP instances with dimension 512$\times$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, 15, 20.

The best makespan and *wrr* results were obtained with the parameter configuration p_C=1.0, p_R=0.9, $\#I$=16, $\#pop$=10.

Table I
EXPERIMENTAL RESULTS FOR THE 512×16 HCSP INSTANCES FROM BRAUN ET AL. [10]

instance	ND		IGD		Spread		RHV	
	pμ-CHC	**MOCHC**	**pμ-CHC**	**MOCHC**	**pμ-CHC**	**MOCHC**	**pμ-CHC**	**MOCHC**
u_c_hihi.0	13.60±1.63	**49.77±7.59**	**1.00±0.39**	1.65±0.30	1.22±0.13	**1.00±0.05**	**1.00±0.02**	0.87±0.02
u_c_hilo.0	14.80±1.10	**22.20±4.45**	**1.00±0.11**	6.99±0.44	1.24±0.12	**1.00±0.10**	**1.00±0.01**	0.44±0.01
u_c_lohi.0	14.43±1.38	**48.00±5.38**	**1.00±0.15**	2.29±0.30	1.12±0.14	**1.00±0.02**	**1.00±0.03**	0.76±0.02
u_c_lolo.0	14.70±1.02	**24.23±5.09**	**1.00±0.11**	5.40±0.28	**1.00±0.19**	1.36±0.04	**1.00±0.02**	0.55±0.01
u_i_hihi.0	4.97±1.47	**45.37±5.53**	2.82±0.26	**1.00±0.31**	1.11±0.02	**1.00±0.08**	**1.00±0.05**	0.93±0.04
u_i_hilo.0	7.83±1.32	**27.37±4.00**	**1.00±0.16**	6.53±0.31	1.26±0.16	**1.00±0.10**	**1.00±0.03**	0.58±0.01
u_i_lohi.0	4.67±1.09	**64.57±10.07**	1.07±0.15	**1.00±0.15**	1.14±0.04	**1.00±0.10**	**1.00±0.04**	0.84±0.04
u_i_lolo.0	8.30±1.53	**27.47±3.77**	**1.00±0.19**	6.77±0.36	1.07±0.27	1.00±0.06	**1.00±0.04**	0.63±0.02
u_s_hihi.0	8.80±1.79	**44.07±7.18**	**1.00±0.12**	1.94±0.36	1.26±0.17	**1.00±0.09**	**1.00±0.03**	0.84±0.03
u_s_hilo.0	10.80±1.52	**26.47±4.26**	**1.00±0.11**	6.01±0.51	1.25±0.13	**1.00±0.08**	**1.00±0.02**	0.42±0.02
u_s_lohi.0	9.60±1.59	**44.63±8.58**	**1.00±0.14**	2.59±0.51	1.22±0.16	**1.00±0.11**	**1.00±0.02**	0.74±0.05
u_s_lolo.0	11.40±1.75	**39.13±6.89**	**1.00±0.13**	8.15±0.46	1.00±0.18	1.05±0.04	**1.00±0.01**	0.42±0.01

Table II
SCALABILITY ANALYSIS: MOEAS FOR THE MULTIOBJECTIVE HCSP

dimension	ND		IGD		Spread		RHV	
	pμ-CHC	**MOCHC**	**pμ-CHC**	**MOCHC**	**pμ-CHC**	**MOCHC**	**pμ-CHC**	**MOCHC**
512×16	10.32±1.64	**53.15±8.54**	**1.00±0.15**	2.11±0.26	1.13±0.14	**1.00±0.07**	**1.00±0.02**	60.47±0.03
1024×32	9.97±1.79	**27.01±8.28**	**1.00±0.07**	1.21±0.03	**1.00±0.08**	1.01±0.02	**1.00±0.03**	81.48±0.08
2048×64	6.70±1.76	**23.65±8.05**	1.05±0.09	**1.00±0.02**	**1.00±0.08**	1.04±0.02	**1.00±0.04**	71.52±0.14
4096×128	5.67±1.80	**24.48±11.82**	1.16±0.13	**1.00±0.02**	**1.00±0.10**	1.08±0.01	**1.00±0.07**	70.59±0.14

E. Results and discussion

This subsection discusses the experimental results of pμ-CHC applied to the MR-HCSP. In order to have a baseline to perform a comparison, the MOCHC algorithm [30] was applied with the same parameter configuration than pμ-CHC to the same HCSP instances. MOCHC follows a sequential evolutionary approach, without applying the randomized PALS local search operator. The Min-Min (single-objective) results are also used as a baseline to analyze the pμ-CHC results. The experimental results for the 512×16 HCSP instances by Braun et al [10] are presented separatly, since this benchmark is frequently used by the research community.

Table I reports the results for the set of 512×16 HCSP instances. The average and standard deviation values for each metric in the 30 independent executions of each EA were computed, and the Kruskal-Wallis test was performed to analyze the statistical confidence of the results. The best results for each metric are marked in bold when the computed p-value is below 10^{-3}.

The results in Table I indicate that pμ-CHC computed better schedules than MOCHC for the 512×16 HCSP instances. MOCHC found a large number of non-dominated solutions, but they are always dominated by the solutions computed using pμ-CHC, as it is demonstrated by the (normalized) values of IGD (smaller values of IGD mean a better approximation to the Pareto front) and RHV (larger values of RHV mean a better coverage). In addition, the spread results (smaller values of the spread metric mean a better distribution), demonstrate that pμ-CHC correctly samples the Pareto front.

Table II summarizes the results obtained in the scalability analysis that solve the large HCSP instances. It reports the average and standard deviation values computed in the 30 independent executions of each EA performed for each instance dimension. The results show that pμ-CHC computed the best IGD values for two dimensions, the best spread values for three dimensions, and the best values of RHV for all the problem dimensions studied. The number non-dominated solutions decrease when solving the largest HCSP instances, but accurate schedules are found. The RHV values confirm both a good quality of solutions and a correct sampling of the Pareto front. The previous results indicate a good scalability behavior of the proposed pμ-CHC algorithm when solving large HCSP instances.

Table III reports the best improvements obtained by pμ-CHC over the best deterministic results for each objective (computed by Min-Min). Acceptable improvements -up to 14.0%- are obtained by pμ-CHC in the makespan objective, and significant improvements -up to 61.4%- are obtained in the *wrr*. This is an expected result, since Min-Min computes schedules with accurate makespan values [18], but it does not account for the *wrr* or other QoS-related objectives.

The best and average improvements obtained by pμ-CHC over the (single-objective) Min-Min results are summarized in Table IV for each problem dimension studied. The improvement values demonstrate that pμ-CHC is an accurate scheduler even for the largest problem dimension studied, outperforming the Min-Min results for up to **14.0%** (more than 6.0% in average) in the makespan objective and up to **72.1%** (more than 22% in average) in the *wrr* objective.

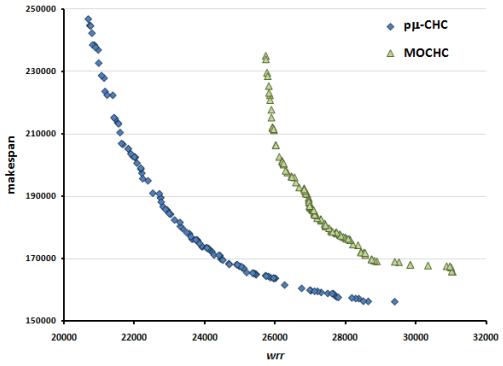
Table III
Pμ-CHC IMPROVEMENTS OVER MIN-MIN RESULTS (512×16).

instance	Min-Min		p-μCHC		improvement	
	makespan	wrr	makespan	wrr	makespan	wrr
u_c_hihi	8460680.0	46084.6	7916760.0	19027.2	6.4%	58.7%
u_c_hilo	161805.0	36170.9	156242.0	20687.6	3.4%	42.8%
u_c_lohi	275837.0	48269.5	255943.0	18645.5	7.2%	61.4%
u_c_lolo	5441.4	36057.9	5259.6	20244.4	3.3%	43.9%
u_i_hihi	3513920.0	17862.1	3042690.0	15415.1	13.4%	13.7%
u_i_hilo	80755.7	23502.6	74798.7	17301.2	7.4%	26.4%
u_i_lohi	120518.0	17630.7	105402.0	14985.9	12.5%	15.0%
u_i_lolo	2785.7	24238.9	2598.1	17709.5	6.7%	26.9%
u_s_hihi	5160340.0	25884.0	4435520.0	16936.4	14.0%	34.6%
u_s_hilo	104375.0	27566.5	98697.7	18056.4	5.4%	34.5%
u_s_lohi	140284.0	26006.6	127456.0	17071.3	9.1%	34.4%
u_s_lolo	3806.8	27608.1	3530.2	17519.0	7.3%	36.5%

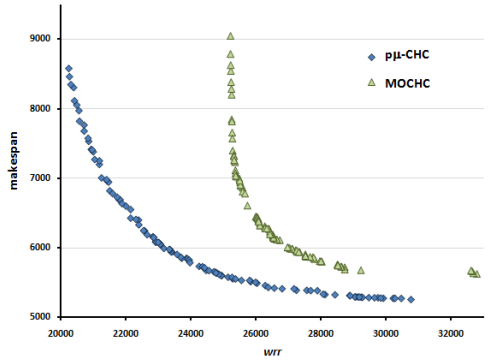
Table IV
Pμ-CHC IMPROVEMENTS OVER MIN-MIN RESULTS.

dimension	best improvement		avg. improvement	
	makespan	wrr	makespan	wrr
512×16	14.0%	61.4%	8.0%	35.7%
1024×32	19.2%	63.3%	9.5%	34.5%
2048×64	15.1%	62.9%	7.1%	22.8%
4096×128	24.6%	72.1%	5.9%	21.9%

Figure 3 presents two samples of the Pareto fronts computed by pμ-CHC and MOCHC for the problem instances u_c_lohi and u_c_lolo in the set from Braun et al.



(a) u_c_lohi (512×16)



(b) u_c_lolo (512×16)

Figure 3. Two samples of Pareto fronts found.

The Pareto fronts in Figure 3 exemplifiy the algorithmic behavior also shown for the other problem instances solved. The results demonstrate that significant improvements in the objective functions values and the diversity of the computed solutions are obtained when using pμ-CHC when compared with those found by the MOCHC algorithm.

VI. CONCLUSIONS AND FUTURE WORK

This article has presented the application of the parallel micro-CHC evolutionary algorithm to the scheduling problem in heterogeneous computing environments, to minimize the makespan and weighted response ratio objectives.

The proposed pμ-CHC was designed to solve standard benchmark instances of the problem, by using simple exploration operators that allow the method to scale in order to efficiently solve large-dimension HCSP instances. A specific local search operator based on PALS was included to compute accurate schedules. A decomposition approach similar to the one used in the MOEA/D algorithm was employed in pμ-CHC, where each subpopulation solves a specific subproblem, each one using a different pair of weights for the makespan and wrr objectives.

The comparative analysis demonstrated that pμ-CHC is an effective method for the simultaneous optimization of the makespan and wrr objectives. The multiobjective approach applied in pμ-CHC was capable to compute accurate solutions for standard HCSP instances, obtaining schedules with accurate trade-off values between the makespan and wrr objectives, and a correct sampling of the Pareto front for each problem. pμ-CHC computed significantly better solutions than a sequential MOCHC algorithm applied to the problem, and significant improvements were also obtained when comparing with the (single objective) makespan and *wrr* results computed by the Min-Min heuristic. The scalability analysis showed that pμ-CHC was able to compute accurate schedules for a test suite of large HCSP instances with up to 4096 tasks and 128 machines.

The results obtained in the experimental analysis indicate that pμ-CHC is a useful scheduler for distributed HC and grid systems when considering the makespan and wrr objectives to optimize.

The main lines for future work include to study specific techniques for improving the results and the efficiency of pμ-CHC, and to extend the proposed methods to solve other multiobjective HCSP variants. Regarding the first issue, specific exploration operators based on improving the proposed objectives can be incorporated in pμ-CHC in order to further improve the wrr results and the population diversity, allowing to compute more accurate scheduling when solving large HCSP instances. On the other hand, further work is also needed to study the efficacy and efficiency of the pμ-CHC algorithm for solving other multiobjective variants of the HCSP.

REFERENCES

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

[2] H. El-Rewini, T. Lewis, and H. Ali, *Task scheduling in parallel and distributed systems*. Prentice-Hall, Inc., 1994.

[3] J. Leung, L. Kelly, and J. Anderson, *Handbook of Scheduling: Algorithms, Models, and Performance Analysis*. CRC Press, 2004.

[4] R. Freund, V. Sunderam, A. Gottlieb, K. Hwang, and S. Sahni, "Special issue on heterogeneous processing," *J. Parallel Distrib. Comput.*, vol. 21, no. 3, 1994.

[5] M. Eshaghian, *Heterogeneous Computing*. Artech House, 1996.

[6] M. Garey and D. Johnson, *Computers and intractability*. Freeman, 1979.

[7] T. Bäck, D. Fogel, and Z. Michalewicz, Eds., *Handbook of evolutionary computation*. Oxford University Press, 1997.

[8] E. Alba, *Parallel Metaheuristics: A New Class of Algorithms*. Wiley-Interscience, 2005.

[9] L. Wang, H. Siegel, V. Roychowdhury, and A. Maciejewski, "Task matching and scheduling in heterogeneous computing environments using a genetic-algorithm-based approach," *J. Parallel Distrib. Comput.*, vol. 47, no. 1, pp. 8–22, 1997.

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

[11] A. Zomaya and Y. Teh, "Observations on using genetic algorithms for dynamic load-balancing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 12, no. 9, pp. 899–911, 2001.

[12] F. Xhafa, J. Carretero, and A. Abraham, "Genetic algorithm based schedulers for grid computing systems," *International Journal of Innovative Computing Information and Control*, vol. 3, no. 5, pp. 1–19, 2007.

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

[14] W. Stallings, *Operating Systems: Internals and Design Principles*. Prentice Hall, 2001.

[15] R. Buyya, "Economic-based distributed resource management and scheduling for grid computing," Ph.D. dissertation, Monash University, Melbourne, Australia, 2002.

[16] J. Monte and K. Pattipati, "Scheduling parallelizable tasks to minimize makespan and weighted response time," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. 32, no. 3, pp. 335–345, 2002.

[17] C. Coello, G. Lamont, and D. Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[18] Y. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Comput. Surv.*, vol. 31, no. 4, pp. 406–471, 1999.

[19] L. Eshelman, "The CHC adaptive search algorithm: how to have safe search when engaging in nontraditional genetic recombination," in *Foundations of Genetics Algorithms*. Morgan Kaufmann, 1991, pp. 265–283.

[20] E. Alba and M. Tomassini, "Parallelism and evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 443–462, 2002.

[21] D. Goldberg, "Sizing populations for serial and parallel genetic algorithms," in *Proc. of the 3rd Int. Conf. on Genetic Algorithms*, San Francisco, USA, 1989, pp. 70–79.

[22] K. Krishnakumar, "Micro genetic algorithms for stationary and nonstationary function optimization," in *Proc. of the SPIE Intelligent Control and Adaptive Systems Conference*, 1989.

[23] C. Coello and G. Pulido, "A micro-genetic algorithm for multiobjective optimization," in *Proc. of the 1^{st} Int. Conf. on Evolutionary Multi-Criterion Optimization*, London, UK, 2001, pp. 126–140.

[24] Q. Zhang and H. Li, "MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[25] E. Alba, F. Almeida, M. Blesa, C. Cotta, M. Diaz, I. Dorta, J. Gabarró, J. González, C. León, L. Moreno, J. Petit, J. Roda, A. Rojas, and F. Xhafa, "MALLBA: A library of skeletons for combinatorial optimisation," *Parallel Computing*, vol. 32, no. 5-6, pp. 415–440, 2006.

[26] F. Xhafa and B. Duran, "Parallel memetic algorithms for independent job scheduling in computational grids," in *Recent Advances in Evolutionary Computation for Combinatorial Optimization*. Springer, 2008, pp. 219–239.

[27] E. Alba and G. Luque, "A new local search algorithm for the DNA fragment assembly problem," in *Proc. of 7^{th} European Conference on Evolutionary Computation in Combinatorial Optimization*, 2007, pp. 1–12.

[28] S. Ali, H. Siegel, M. Maheswaran, S. Ali, and D. Hensgen, "Task execution time modeling for heterogeneous computing systems," in *Proc. of the 9^{th} Heterogeneous Computing Workshop*, Washington, DC, USA, 2000, p. 185.

[29] S. Nesmachnow, H. Cancela, and E. Alba, "Heterogeneous computing scheduling with evolutionary algorithms," *Soft Computing*, vol. 15, no. 4, pp. 685–698, 2011.

[30] A. Nebro, E. Alba, G. Molina, F. Chicano, F. Luna, and J. Durillo, "Optimal antenna placement using a new multi-objective CHC algorithm," in *Proc. of the 9^{th} annual conference on genetic and evolutionary computation*, New York, USA, 2007, pp. 876–883.