Comparative Analysis of Intra-Algorithm Parallel Multiobjective Evolutionary Algorithms: Taxonomy Implications on Bioinformatics Scenarios

Sergio Santander-Jiménez and Miguel A. Vega-Rodríguez

Abstract—Parallelism has become a recurrent tool to support computational intelligence and, particularly, evolutionary algorithms in the solution of very complex optimization problems, especially in the multiobjective case. However, the selection of parallel evolutionary designs often represents a difficult question due to the multiple variables that must be considered to attain an accurate exploitation of hardware resources, along with their influence in solution quality. This work looks into this issue by conducting a comparative performance analysis of intra-algorithm parallel multiobjective evolutionary algorithms running on shared-memory configurations. We consider different design trends including A) generational approaches based on measurements of solution quality plus diversity, B) generational approaches based on measurements of solution quality exclusively, and C) non-generational approaches. Following these trends, a total of six representative algorithms are applied to tackle a challenging bioinformatics problem as a case study, phylogenetic reconstruction. Experimentation on real-world scenarios point out the main advantages and weaknesses of each design, outlining guidelines for the selection of methods according to the characteristics of the employed hardware, evolutionary properties, and the parallelism exploitation capabilities of the evaluated approaches.

Index Terms—Comparative analysis, taxonomy, parallelism, multiobjective optimization, evolutionary computation, NP-hard problems.

1 Introduction

In the last years, the need to solve NP-hard optimization problems has played a prevailing role in research directions for a wide variety of scientific domains. Such problems involve the processing of a decision space S to find an optimal solution $s \in S$ according to an objective function $f : S \rightarrow \Re$. In this sense, many real-world problems are characterized by the presence of not a single but multiple objective functions $f_1, f_2, ..., f_n$ which belong to an objective space Z. Tackling these multiobjective optimization problems (MOPs) implies finding those solutions which represent the best possible trade-offs among the considered objectives, the Pareto-optimal set. Their NP-hard nature justifies the efforts in developing stochastic methods to address them, being specially significant the results obtained by using evolutionary algorithms (EAs) and, in the multiobjective case, multiobjective evolutionary algorithms (MOEAs) [1].

In spite of the use of evolutionary approaches, real-world problems are becoming increasingly difficult to solve in reasonable times due to the presence of different factors that increase their temporal complexity, including large search spaces, high dimensionality, and time-consuming objective functions. This issue is even more noticeable in MOPs, which are affected not only by the presence of computationally demanding operations (e.g. in the evaluation procedures) but also by the fact that these problems require finding a set of Pareto solutions instead of a single one as in traditional single-objective optimization. As a result, the use of serial approaches no longer satisfies the temporal requirements needed to address real-world optimization problems. Fortunately, the advances in hardware development have opened the door to the solution of such problems by taking advantage of the parallelism opportunities found in evolutionary algorithmic designs. The combination of parallelism and evolutionary computation indeed provides a number of benefits in both single and multiobjective scenarios, including reduced processing times, improved robustness, and boosted solution quality [2], [3].

Focusing on the multiobjective case, the computationally demanding features shown by MOPs motivate the interest in analyzing intra-algorithm parallelization approaches, which apply parallelism to minimize execution time and speed up the optimization process [4]. A key aspect is given by the fact that all the current hardware platforms are parallel. Therefore, when adopting MOEAs, the researcher must consider not only problem-related requirements but also other questions, such as the characteristics of the hardware and, especially, the intrinsic available parallelism shown by the evolutionary algorithm. In fact, a fundamental issue lies on examining the impact of MOEA algorithmic designs on the temporal gains observed in their parallel versions, in order to determine the parallel MOEA that best fits the preferences of the expert in terms of solution quality and

- *Sergio Santander-Jiménez is with the INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Lisboa 1000-029, Portugal (e-mail: sergio.jimenez@tecnico.ulisboa.pt).*
- *Miguel A. Vega-Rodríguez is with the Department of Computer and Communications Technologies, University of Extremadura, Escuela Politécnica, Campus Universitario s/n, Caceres 10003, Spain (e-mail: mavega@unex.es)*

execution time. Therefore, the development of effective, efficient parallel approaches involves multiple variables that are often complex to evaluate.

This work aims to study intra-algorithm parallel evolutionary designs for multiobjective optimization, undertaking a comparative analysis of parallel MOEAs on computationally demanding optimization scenarios. Rather than stating if one particular MOEA design is better or worse than others, the key idea lies on emphasizing the advantages and disadvantages of different parallel MOEA approaches attending to solution quality and parallelism potential, in order to support researchers in the choice of accurate optimization procedures according to the characteristics of the underlying hardware. To this end, we examine the multiobjective and parallel performance attained by different algorithms on a challenging bioinformatics problem, phylogenetic reconstruction [5]. We consider generational designs that integrate multiobjective mechanisms to promote 1) solution quality and diversity (represented by the Non-Dominated Sorting Genetic Algorithm II, NSGA-II [6], Strength Pareto Evolutionary Algorithm 2, SPEA2 [7], and Indicator-Based Evolutionary Algorithm, IBEA [8]) and 2) only solution quality (Multiobjective Firefly Algorithm, MOFA [9], and Multiobjective Evolutionary Algorithm Based on Decomposition, MOEA/D [10]). Along with them, non-generational designs (Multiobjective Artificial Bee Colony, MOABC [11]) are also considered. As a case study, this work will be focused on shared-memory multicore systems, since they represent one of the most widespread hardware trends in both commodity and high performance platforms.

This paper is organized as follows. The next section examines different attempts and proposals in the literature to characterize parallel EAs, outlining the scope of this study. Section 3 describes the different parallel MOEA designs considered in this study while Section 4 details the real-world problem selected for experimentation purposes. The evaluation and discussion of the described parallel MOEAs is carried out in Section 5. Finally, Section 6 includes concluding remarks and future work lines.

2 Related Work

Throughout the years, different authors have pointed out the close relationship between parallelism and evolutionary computation, reporting the use of parallel EAs and other bioinspired designs in telecommunications, engineering, industry, etc. We refer the reader to [2] and [3] for comprehensive surveys on this subject. This section summarizes the different attempts published in the literature to characterize parallel EAs and their multiobjective counterparts.

Among the initial works which tried to provide insight into parallel metaheuristics, we can highlight the classification of parallel genetic algorithms (GAs) by Cantú-Paz [12]. There, four categories were distinguished: global single-population master-slave GAs, multi-population coarse-grained GAs, single-population fine-grained GAs, and hierarchical GAs. The first class takes advantage of parallelism opportunities in the processing of individuals to parallelize the computation of time-consuming operations. The multi-population GA defines several subpopulations managed by different processors, each one running a complete GA with migration operators to exchange individuals periodically. In fine-grained GAs, the population is spatially partitioned in small subpopulations composed of one or few individuals, each one handled by a processor, using neighbourhoods for selection and mating. Finally, the hierarchical or hybrid approach combines the previous classes, typically defining a multi-population GA at the upper level and a master-slave, fine-grained, or another coarse-grained GA at the bottom.

Regarding the general EA case, Alba and Tomassini reported a taxonomy focused on the classification of EA models and parallel implementations [13]. Two main EA models were defined in this work. The first one is the panmictic EA, which handles a single panmictic population where evolutionary operators take place globally considering any individual. The second model, named as structured EA, uses structured populations in the shape of islands (distributed EAs) or diffusion grids (cellular EAs). Considering this distinction, four basic parallelization schemes were discussed: global parallelization under master-slave principles, coarse-grained, fine-grained, and hybrid approaches.

Another well-known classification of parallel metaheuristics was due to Talbi [4], who defined inter and intra-algorithm parallelization strategies under three hierarchical levels: algorithmic, iteration, and solution levels. At the algorithmic level, problem-independent inter-algorithm parallelization is applied to boost search capabilities. This level included the island model and the diffusion model. On the other hand, at the iteration and solution levels, intra-algorithm schemes are used to reduce execution time by exploiting parallelism opportunities related to the algorithmic design (iteration level, problem-independent approach) and task/data parallelism at the objective function calculations (solution level, problem-dependent approach). Combinations of the above mentioned approaches were also included in this classification.

Van Veldhuizen et al. addressed the adaptation of parallel EA paradigms to the MOEA case [14], identifying different parallelization strategies that were further discussed by Jaimes and Coello in [15]. Focusing on master-slave designs, three basic approaches were outlined: 1) distribution of the population members in such a way that each slave computes all the n objective functions for the assigned individuals; 2) distribution of sets of population members where partitions of slaves are responsible for the computation of $k<n$ different objective functions; and 3) decomposition of each objective function in a problem-dependent fashion. Regarding the island model, four main variants were defined: 1) homogeneous (identical MOEAs/parameters in each island); 2) heterogeneous (different MOEAs/parameters); 3) computation of different objective function subsets per island; and 4) computation of different regions of the genotype or phenotype domains per island. For the diffusion model, they pointed out several neighbourhood shapes, including square, rectangle, cube, and multi-dimensional approaches.

Talbi et al. also adapted their classification to the multiobjective context [16], providing hints to undertake problem-dependent parallelization when considering several independent solvers, fitness evaluation under decomposition, and fitness evaluation under multiple runs. More recently, Luna and Alba pointed out in [3] centralized and distributed approaches to manage non-dominated solutions

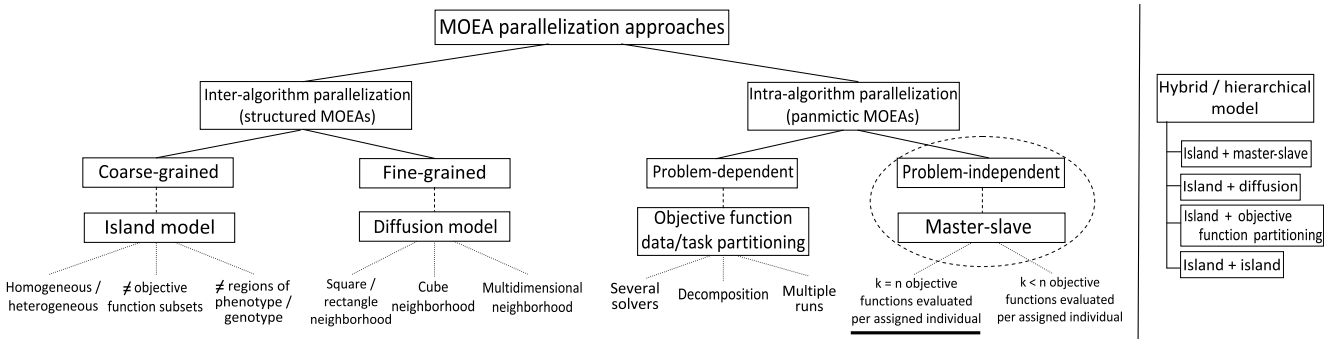


Fig. 1. General classification of parallel MOEA approaches and strategies, highlighting the scope of this work

in structured MOEAs. Coello suggested in [17] the idea of incorporating surrogate models (functional approximations) into parallel MOEAs as a way to deal with computationally expensive real-world problems. Finally, Talbi analyzed the opportunities provided by the emergence of novel hardware platforms and programming models, identifying the challenges to be addressed in the parallel MOEA research field on the way to achieve Exascale performance [18].

Figure 1 summarizes the different paradigms and strategies in these taxonomies for the multiobjective case. Our work is focused on examining the problem-independent intra-algorithm branch by considering different MOEA approaches, discussing the influence of their algorithmic designs on parallel performance. Other attempts in the literature to study intra-algorithm approaches were mostly aimed at evaluating multiobjective quality, like [19] which considered the specific case of distributed computing on grid architectures. We aim to go a step further by analyzing and discussing parallel MOEAs taking into account not only multiobjective quality, but also parallel performance and their relationship with MOEA features. Moreover, our study is undertaken considering the more general case of shared-memory machines, which are among the most commonly used hardware setups in metaheuristics research. Finally, we consider implementations whose worker tasks include, besides other time-consuming operations, the evaluation of n objective functions per assigned individual, as it represents the most usual approach and avoids the load imbalance issues associated to the $k < n$ alternative [14]. A summary of the main contributions of this work is given below:

- A comparative analysis of parallel MOEAs based on the problem-independent intra-algorithm paradigm, considering different design trends and six representative examples of parallel MOEAs for shared-memory systems (NSGA-II, SPEA2, IBEA, MOEA/D, MO-FA, and MOABC).
- An in-depth evaluation of these approaches under both parallel metrics (speedup and efficiency) and multiobjective indicators (hypervolume and spacing), conducting experimental assessment on a challenging real problem, phylogenetic reconstruction.
- The identification of key opportunities and flaws in each design trend, suggesting guidelines for the selection of parallel MOEA approaches by considering solution quality requirements and parallel computing capabilities in the underlying hardware setup.

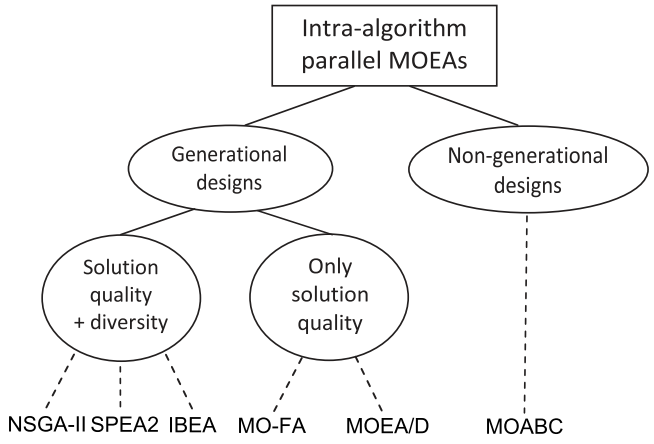


Fig. 2. Intra-algorithm parallel MOEAs considered in this study

3 Intra-algorithm Parallel MOEAs

This section describes the intra-algorithm parallel MOEAs studied in this work. We consider a shared-memory scenario involving N_C processing cores with the same memory address space, being this globally-shared memory the element where all the coordination and synchronization among execution threads take place. Our parallel implementations follow the OpenMP standard [20], which defines a fork/join model for performing parallel computations via worksharing and other compiler directives. Figure 2 shows the general overview of the considered parallel MOEA categories, distinguishing the different trends herein evaluated.

3.1 Generational Designs

In a generational design, the parallelization is undertaken by distributing independent time-consuming operations in the main sections that compose each generation of the MOEA. Only when all the parallel tasks belonging to a section have been processed, the algorithm can proceed with the next one. The term 'generational' refers to the fact that a new generation can only be started after completing all the tasks in the current one. Hence, a synchronous style of parallel programming is commonly adopted to implement these approaches. The evaluation procedures have traditionally been identified as the main source of parallelism to be exploited. However, real-world problems often involve additional computationally demanding operations whose

Algorithm 1 Generational Parallel NSGA-II/SPEA2/IBEA

```
1:  Initialize Data Structures (PopulationStructs, ParetoFront)
2:  #pragma omp parallel num_threads (num_threads)
3:  Initialize Population (PopulationStructs, popSize, num_threads)
4:  while ! stop criterion reached do
5:      #pragma omp single
6:          Fitness Assignment (PopulationStructs)
7:          Environmental Selection (PopulationStructs)
8:      #pragma omp for schedule (scheduleType)
9:      for i = 1 to popSize do
10:         Selection, Crossover, and Mutation (P'_i, PopulationStructs, crossover-
            Prob, mutationProb)
11:         Evaluate Generated Solution (P'_i)
12:     end for
13:     #pragma omp single
14:         Update Pareto Front (ParetoFront, PopulationStructs∪P')
15: end while
```

parallelization can lead to noticeable reductions of execution time e.g., the computation of evolutionary operators when complex data structures are used in the solution encoding. We follow this guideline in our implementations with the aim of reducing serial components.

Different generational MOEAs are distinguished according to the multiobjective mechanisms used to assess solutions. We examine here two strategies: approaches based on solution quality and diversity and approaches based on solution quality only.

3.1.1 Approaches Based on Measurements of Solution Quality and Diversity

These designs are characterized by the inclusion of mechanisms for assessing solutions in terms of convergence and diversity. In our analysis, these approaches are represented by three widely-used representative MOEAs: NSGA-II [6], SPEA2 [7], and IBEA [8]. The consideration of these algorithms within the generational trend follows the guidelines of their standard definitions, which exhibit intrinsically generational properties. In these algorithms, the optimization process is handled by using the traditional EA scheme, generating new solutions by using selection, crossover, and mutation operators. The main differences among them are related to the data structures used to manage the population and the implemented fitness assignment strategies.

While NSGA-II introduces fast non-dominated sorting and crowding distance ordering over a combined parent + offspring population to determine the next parent population, SPEA2 uses an archive of promising individuals for parent selection purposes that is updated through environmental selections considering strength values and nearest-neighbour density information. IBEA integrates the information retrieved by multiobjective indicators to assess the quality of the individuals in the population. Indicator-based fitness values are defined to identify the most promising solutions that must remain in the population and be used in the generation of offspring. Although a wide range of quality indicators can be considered in this framework, IBEA is commonly used under indicators that combine measurements of convergence and diversity, such as those based on hypervolume [21].

The parallel implementations of NSGA-II, SPEA2, and IBEA are based on the scheme shown in Algorithm 1. In these MOEAs, parallelizable and non-parallelizable sections can be distinguished at each generation. For example, the fitness assignment and environmental selection steps represent operations difficult to parallelize, due to the fact that they could potentially be affected by data dependencies issues and read/write hazards. These operations will be handled by a single thread, using for this purpose the *#pragma omp single* directive (lines 5-7 in Algorithm 1). The update of the Pareto front structure in the final steps of a generation is also included among these serial operations (lines 13-14). Regarding parallelizable sections, worksharing efforts in these algorithms take place at the offspring processing loop, since the generation and evaluation of new candidate solutions can be handled in parallel by different execution threads. Therefore, *#pragma omp for* can be used to distribute the iterations of this loop (lines 9-12). This parallel loop can be executed under static or dynamic scheduling policies (schedule clause), depending the choice on the characteristics of the tackled problem and their effect over the times required to compute evolutionary operators and objective functions for different candidate solutions.

3.1.2 Approaches Based on Measurements of Solution Quality Exclusively

The main feature in these designs lies on the use of lightweight fitness mechanisms that allow the algorithm to determine quickly the quality of the individuals, discarding other information for the sake of reducing fitness assignment times. We also include in this category approaches that reduce computational complexity by decomposing a MOP into scalar subproblems, which are optimized using solution quality information from their neighbouring subproblems. To illustrate them, we use a multiobjective adaptation of the Firefly Algorithm, MO-FA [9], [22], and the most representative decomposition-based MOEA, MOEA/D [10]. These solution quality-only methods, MO-FA and MOEA/D, show intrinsic generational properties in their original definitions, being suitable for inclusion in the generational trend.

MO-FA models the interactions of a firefly population, based on movements towards the position of partners showing brighter flashing light patterns. The multiobjective search engine in MO-FA proceeds by making pairwise comparisons of solution quality under the Pareto dominance relation [1]. By using Pareto dominance, the algorithm can compare solutions on the basis of their objective function scores. Let P_i and P_j be two individuals with solutions $P_i.s$ and $P_j.s$. If P_i is dominated by P_j ($P_j \succ P_i$), the algorithm carries out the generation of a new candidate solution by computing firstly a measurement δ_{ij} of the distance which separates $P_i.s$ from $P_j.s$: $\delta_{ij} = \sqrt{\sum_{k=1}^{d}(P_i.s_k - P_j.s_k)^2}$, where $P_i.s_k$ and $P_j.s_k$ refer to the k-th component (decision variable) of each solution. Once this value has been calculated, each component of $P_i.s$ is updated as follows:

$$P_i.s_k = P_i.s_k + \beta_0 e^{-\gamma\delta_{ij}^2}(P_j.s_k - P_i.s_k) + \alpha(rand[0,1] - \frac{1}{2}), \quad (1)$$

where β_0 is an attractiveness factor, γ an environmental absorption coefficient, α a randomization factor, and rand[0, 1] a random number from a uniform distribution in the interval [0, 1]. The second term in Eq. 1 denotes the degree $P_i.s$ will move towards $P_j.s$ while the third term introduces

Algorithm 2 Generational Parallel MO-FA

```
1:  Initialize Data Structures (P, ParetoFront)
2:  #pragma omp parallel num_threads (num_threads)
3:  Initialize Population (P, popSize, num_threads)
4:  while ! stop criterion reached do
5:      #pragma omp single
6:          idDominatedFfs, numDominatedFfs, idDominatingFfs ← 0
7:          for i = 1 to popSize do
8:              if ∃ P_j: P_j ≻ P_i then
9:                  idDominatedFfs[numDominatedFfs] ← i
10:                 numDominatedFfs ← numDominatedFfs + 1
11:                 idDominatingFfs[i] ← idDominatingFfs[i] ∪ j /*∀P_j: P_j≻P_i*/
12:             end if
13:         end for
14:         auxPop ← P
15:     #pragma omp for schedule (scheduleType)
16:     for i = 1 to numDominatedFfs do
17:         idDom ← idDominatedFfs[i]
18:         Move Firefly (P_idDom, auxPop, idDominatingFfs[idDom], β_0, γ, α)
19:         Evaluate Generated Solution (P_idDom)
20:     end for
21:     #pragma omp single
22:         Update Pareto Front (ParetoFront, P)
23: end while
```

Algorithm 3 Generational Parallel MOEA/D

```
1:  Initialize Data Structures (P, ParetoFront)
2:  #pragma omp parallel num_threads (num_threads)
3:  Initialize Population (P, λ, popSize, num_threads)
4:  while ! stop criterion reached do
5:      #pragma omp for schedule (scheduleType, T)
6:      for i = 1 to popSize do
7:          Selection, Crossover, and Mutation (y, P, T, crossoverProb, mutation-
            Prob)
8:          Evaluate Generated Solution (y)
9:          #pragma omp critical
10:             Update Reference Point (y, z*)
11:         for each P_j in the neighbourhood of P_i do
12:             if g_j(y|λ^j, z*) improves g_j(P_j|λ^j, z*) then
13:                 Update Neighbouring Subproblem (P_j, y)
14:             end if
15:         end for
16:     end for
17:     #pragma omp single
18:         Update Overlapped Neighbourhoods (P, T)
19:         Update Pareto Front (ParetoFront, P)
20: end while
```

variability into $P_i.s$ to promote the exploration of the search space. These movements are repeated for each P_j in the population that dominates P_i, proceeding next with the computation of objective functions for the resulting solution.

The idea behind the parallel implementation of MO-FA (Algorithm 2) consists of distributing firefly movements among execution threads. To this end, several issues must be addressed. Particularly, we have to minimize load imbalance in the processing of individuals, as movements are only applied over dominated individuals in the population and, for each one, a variant number of dominating individuals must be considered in the calculations. To attain balanced workload per thread, we pre-process the population to determine the number of dominated individuals (lines 5-13 in Algorithm 2), storing their identifiers as well as those from their dominating counterparts. In this way, we can remove the Pareto dominance if-condition from the movement loop, which is parallelized by using *#pragma omp for* (lines 15-20). This parallel loop benefits from the use of dynamic scheduling, so that we can counterbalance the effect of having a variant number of dominating individuals involved in each iteration. In addition, we introduce a backup population (line 14) to keep a stable copy of the population, thus avoiding read/write hazards when updating individuals.

On the other hand, MOEA/D tackles a MOP by decomposing it into multiple scalar optimization subproblems, in such a way that the i-th individual in the population represents the best solution found for the i-th subproblem. This decomposition is commonly undertaken by adopting the Tchebycheff approach [10], which defines the objective function of the i-th subproblem g_i as:

$$minimize\ g_i(x|\lambda^i, z^*) = max_{1\leq j\leq n}\left\{\lambda_j^i|f_j(x) - z_j^*|\right\}, \quad (2)$$

where x is the solution under evaluation, $f_j(x)$ the score at the j-th objective for x, λ^i the weight vector for the i-th subproblem, and z^* a reference point whose coordinates represent the best found objective scores.

Each subproblem in MOEA/D is optimized by considering information from their T closest neighbouring subproblems, generating new candidate solutions by applying evolutionary operators within the neighbourhood. Once a new solution y has been obtained, it is used to update z^* (in case it contains a better score at the j-th objective) and also the solutions in the neighbourhood (in case $g_i(y)$ is better than $g_i(x)$, being x the current best solution at the neighbouring subproblem i). These steps are repeated over each subproblem, updating accordingly the Pareto front structure throughout this evolutionary process.

The parallel implementation of MOEA/D is presented in Algorithm 3. In order to accomplish the parallel treatment of subproblems (lines 5-16 in Algorithm 3), it must be ensured that the processing of solutions inside each partition of neighbours is carried out by the same execution thread to respect data dependencies. For this purpose, a chunk size of T iterations is set in the scheduling clause of *#pragma omp for* (line 5), so that the iterations of the loop are distributed in accordance with the partitions defined by T. The reference point z^* represents a variable globally shared by all the threads and therefore its update must be performed under *#pragma omp critical* (lines 9-10) to avoid read/write hazards. Once the parallel processing of subproblems has finished for the current generation, a *#pragma omp single* section is introduced to carry out the update of overlapping neighbours between partitions and the management of the Pareto front (lines 17-19).

These parallel designs (Algorithm 2 and 3) distinguish themselves from the solution quality plus diversity approach (Algorithm 1) in the fact that the identified serial / parallel sections comprise different tasks with different time complexity. It is worth remarking that the serial / critical sections in MO-FA and MOEA/D do not involve mandatory environmental selections based on measurements of convergence and diversity over the whole population like in Algorithm 1, but only operations for supporting the parallel loop that imply less computational effort.

3.2 Non-generational Designs

Non-generational parallel MOEAs are based on the idea of allowing evolutionary mechanisms to proceed as soon as a new candidate solution has been generated. That is, the execution of the algorithm can go on without requiring all the parallel tasks which compose a generation to be completed. Therefore, the 'generation' idea is no longer considered, since the execution threads can carry out new parallel tasks

Algorithm 4 Non-generational MOABC: Master Thread

```
1:  while ! stop criterion reached do
2:      for i = 0 to num_threads-2 do
3:          while BeeFIFO[i] has elements do
4:              Lock (FIFOSem[i])
5:              InconsPop.P_beeId ← Pop (BeeFIFO[i])
6:              Unlock (FIFOSem[i])
7:          end while
8:      end for
9:      if solutions have been retrieved then
10:         Fast Non Dominated Sort / Crowding (InconsPop.P, popSize)
11:         Compute Selection Array (InconsPop.probVector, InconsPop.P, popSize/2)
12:         Lock (PopSem[i]) /* ∀i : i = 0 to num_threads-2 */
13:         Interchange Population Structures Pointers (ConsPop, InconsPop)
14:         Unlock (PopSem[i]) /* ∀i : i = 0 to num_threads-2 */
15:         ParetoFront ← Update Pareto Front (ParetoFront, ConsPop.P)
16:     end if
17: end while
18: Send Termination Notification (i) /* ∀i : i = 0 to num_threads-2 */
```

Algorithm 5 Non-generational MOABC: Worker Threads

```
1:  while ! Termination Notification received do
2:      Lock (PopSem[threadID])
3:      if threadID < num_threads/2 then
4:          Get Employed (employedData, ConsPop.P, indices)
5:      else
6:          Get Onlooker (onlookerData, ConsPop.P, ConsPop.probVector, indices)
7:      end if
8:      Unlock (PopSem[threadID])
9:      if threadID < num_threads/2 then
10:         Perform Employed Tasks (beeSolution, employedData, mutationRate)
11:     else
12:         Perform Onlooker Tasks (beeSolution, onlookerData, mutationRate)
13:     end if
14:     if beeSolution.trial_counter > limit then
15:         Perform Scout Bee Tasks (beeSolution, mutationRate)
16:     end if
17:     Lock (FIFOSem[threadID])
18:     Push (BeeFIFO[threadID], beeSolution)
19:     Unlock (FIFOSem[threadID])
20: end while
```

without waiting for others. As a result, these MOEAs are often implemented by adopting an asynchronous style of parallel programming [23].

These designs are represented in this work by MOABC [11], [24]. This algorithm addresses MOPs by modelling the way honey bees locate and exploit food sources. For this purpose, three main search mechanisms are defined. The first one is the employed bees exploitation, which is applied over the first half of the population. Given an individual P_i associated to a solution $P_i.s$, a new candidate solution is generated under the employed bee formulation as follows:

$$P_i'.s_k = P_i.s_k + \phi(P_i.s_k - P_j.s_k), \tag{3}$$

where k refers to the k-th component of the solution, P_j a randomly chosen individual, and ϕ a random number from a uniform distribution generated in the interval [-1, 1]. By using a greedy approach, the new candidate solution will replace the one originally contained in P_i iff $P_i' \succ P_i$.

The second mechanism models the behaviour of onlooker bees, operating over the second half of the population to exploit the best employed solutions. To this end, Pareto ranking and crowding distances are calculated along with an array of selection probabilities. The onlooker exploitation is conducted by mutating the selected solution to generate a new one, which will be retained in the memory of the processed individual if it is non-dominated with regard to the selected solution. The last mechanism, the scout bees search, addresses local optima issues by operating over individuals that have not been successfully improved in a 'limit' number of trials. Those solutions that verify this condition are replaced by randomly generated ones, which can be optimized by using other procedures to make them able to compete with the remaining solutions.

As the execution of scout searches is variable and depends on the state of the optimization process, this step would introduce load imbalance and waiting times under a generational perspective. In fact, the general model of the algorithm is strongly built upon non-generational principles (e.g., employed bees operate independently from other types of bees and their tasks should be performed without waits, scout bees must take action immediately upon local optima detection to avoid additional unsuccessful attempts, etc.). This issue explains why MOABC represents a suitable example of non-generational approaches, in comparison to other algorithms that show a more generational-oriented design in their standard definitions.

The parallel implementation of MOABC organizes execution threads under master-worker roles after being initialized (using *#pragma omp parallel*). While the master keeps updated the structures that give support to the optimization process, the workers generate and evaluate new candidate solutions in such a way that a worker can start a new task immediately after finishing the previous one. Two population structures are used: 1) a consistent structure to maintain the current state of the population available to the workers and 2) an inconsistent structure, exclusively handled by the master, to manage solutions pending integration into the population. Interactions between the master and the workers require the use of queues to allow results communications and semaphores to manage conflicting master-worker concurrent accesses to shared structures.

Master operations (Algorithm 4) proceed by checking the queues for the arrival of new solutions, storing them in the inconsistent population upon detection (lines 2-8 in Algorithm 4). The inconsistent structure is then processed by applying a fast non-dominated sort and crowding distance ordering, along with determining new selection probabilities (lines 10-11). In this way, the new solutions are integrated into the population and the status of the search is updated in the inconsistent structure. To make it available to the workers, the master interchanges the consistent and inconsistent structures pointers (lines 12-14), updating afterwards the Pareto front (line 15).

Since the population in MOABC is structured in two halves, the worker thread identifier determines the indices of the individuals to be processed and also which kind of approach (employed or onlooker formulation) is applied to generate new candidate solutions. Worker tasks (Algorithm 5) begin by reading the corresponding individuals from the consistent structure (lines 2-8 in Algorithm 5), defining a critical section to guarantee that worker readings are not performed while the master is modifying the consistent structure pointer. Afterwards, the worker generates a new solution under employed or onlooker principles and evaluates it (lines 9-13), conducting also a scout bee search if needed (lines 14-16). The result is then introduced into the queue (lines 17-19) under mutual exclusion with the

readings performed at the master side. After finishing this task, the worker proceeds with a new one, repeating these steps until receiving a termination notification.

4 Problem Formulation

Since the parallel MOEAs research field lacks a well-established framework of benchmark functions for evaluation purposes [2] and the computational complexity of classical -serial- MOEA benchmarks often does not meet the computing requirements needed to accurately assess parallel performance [16], the consideration of real-world MOPs represents a fitting scenario for the comparative analysis of parallel MOEAs. For testing purposes, we have selected a real-world problem from the bioinformatics field: phylogenetic reconstruction [5]. This problem is aimed at processing a biological sequence alignment of size NxM (where N is the number of sequences and M the sequence length) to infer the evolutionary relationships of the organisms characterized in the input data. Evolutionary events are modelled by using a phylogenetic tree $T = (V, E)$, where V is the node set and E defines ancestor-descendant relationships. Phylogenetic reconstruction can be tackled as a MOP which involves the optimization of different phylogenetic functions, such as parsimony $P(T)$ (to be minimized) and likelihood $L(T)$ (to be maximized):

$$P(T) = \sum_{i=1}^{M} \sum_{(u,v)\in E} C(u_i, v_i). \tag{4}$$

$$L(T) = \prod_{i=1}^{M} \sum_{x,y\in\Lambda} \pi_x \left[P_{xy}(t_{ru}) L_p\left(u_i = y\right)\right] \times [P_{xy}\left(t_{rv}\right) L_p\left(v_i = y\right)]. \tag{5}$$

In Equation 4, $(u, v) \in E$ refers to the branch between two nodes $u, v \in V$, u_i and v_i represent the state values (according to an alphabet Λ, e.g. the nucleotide / amino acid state alphabet) at the i-th site in the sequences of u and v, and C measures the presence (1) or lack (0) of a state change. As for Equation 5, π_x represents the stationary probability of a state $x \in \Lambda$, $P_{xy}(t)$ the mutation probability from x to another state y within a time t (branch length value), $r \in V$ the root node with descendants u, v, and $L_p(u_i = y)$, $L_p(v_i = y)$ the partial likelihoods of observing y at the i-th site for u and v. Likelihood calculations are supported by the use of probabilistic models of sequence evolution known as evolutionary models [25].

Phylogenetic reconstruction shows a number of challenging features [26] that justify its suitability for testing purposes. Firstly, this NP-hard problem is characterized by huge search spaces, which grow exponentially with the number of input sequences N according to the expression $(2N-5)!!$. A second complexity factor lies on the evaluation procedures, since their temporal costs generally increase linearly with the length of the input sequences M. In this sense, the likelihood function is remarkably time-consuming due to the high number of floating-point operations involved in its computation. Another issue is given by the fact that different solutions can give rise to different processing and evaluation times, in accordance with their topological features. As a result, this problem involves high execution times and load imbalance issues that put to test the effectiveness and efficiency of parallel MOEAs.

TABLE 1
Phylogenetic datasets

Dataset	Num. of sequences	Sequence length	Data Description
rbcL_55	55	1314	rbcL gene nucleotide data [31]
Fungi_88	88	3329	Thermophilic fungi protein data [32]
mtDNA_186	186	16608	Human mitochondrial DNA data [33]
RDPII_218	218	4182	Prokaryotic RNA data [34]
ZILLA_500	500	759	rbcL gene nucleotide data [35]

TABLE 2
Input parameter settings - parametric study

Parameter	Values				
Population size	24	48	72	96	128
Crossover probability	50%	60%	70%	80%	90%
Mutation probability	5%	10%	15%	20%	25%
Archive size (SPEA2)	10	25	50	75	100
Fitness scaling factor κ (IBEA)	0.05	0.07	0.10	0.25	0.50
I_{HD} reference point (IBEA)	(1.10, 1.10)	(1.25, 1.25)	(1.50, 1.50)	(1.75, 1.75)	(2, 2)
Attractiveness factor β_0 (MO-FA)	0.10	0.25	0.50	0.75	1
Absorption coefficient γ (MO-FA)	0.10	0.25	0.50	0.75	1
Randomization factor α (MO-FA)	0.05	0.10	0.15	0.20	0.25
Neighbourhood size T (MOEA/D)	Population size / N_C				
Limit trial number (MOABC)	5	15	25	35	45

TABLE 3
Input parameter settings - final values

Stop criterion	10,000 evaluations		
Nucleotide sequence model	General Time Reversible (GTR+Γ) [5]		
Amino acid sequence model	Le and Gascuel model (LG08+Γ) [36]		
SPEA2		MO-FA	
Population size	96	Population size	128
Crossover probability	70%	Attractiveness factor β_0	1
Mutation probability	5%	Absorption coefficient γ	0.50
Archive size	75	Randomization factor α	0.05
NSGA-II		MOABC	
Population size	96	Population size	96
Crossover probability	70%	Mutation probability	5%
Mutation probability	5%	Limit trial number	25
IBEA		MOEA/D	
Population size	96	Population size	96
Crossover probability	70%	Crossover probability	70%
Mutation probability	5%	Mutation probability	5%
Fitness scaling factor κ	0.05	Neighbourhood size T	96 / N_C
I_{HD} reference point	(2, 2)		

TABLE 4
Data for metrics calculations: serial execution time (in seconds)

Dataset	NSGA-II	SPEA2	IBEA
rbcL_55	5294.76	5355.62	5372.39
Fungi_88	54202.84	54602.34	54982.11
mtDNA_186	47135.49	47313.75	48083.64
RDPII_218	50452.11	51045.57	51334.49
ZILLA_500	69702.12	70377.89	71307.04
Dataset	MO-FA	MOEA/D	MOABC
rbcL_55	5486.04	5247.58	5610.64
Fungi_88	56424.79	53930.55	52370.39
mtDNA_186	47700.12	47116.05	45379.13
RDPII_218	54507.41	49104.02	48218.18
ZILLA_500	77365.74	68802.96	65459.84

TABLE 5
Data for metrics calculations: hypervolume reference points

	Ideal point		Nadir point	
Dataset	Parsimony	Likelihood	Parsimony	Likelihood
rbcL_55	4774	-21569.69	5279	-23551.42
Fungi_88	33155	-147630.78	35974	-160398.35
mtDNA_186	2376	-39272.20	2656	-43923.99
RDPII_218	40658	-132739.90	45841	-147224.59
ZILLA_500	15893	-79798.03	17588	-87876.39

TABLE 6
Parallel performance results: speedups and efficiencies on the 48-core AMD Opteron system

	Speedup					Efficiency(%)				
Algorithm	8 cores	16 cores	24 cores	32 cores	48 cores	8 cores	16 cores	24 cores	32 cores	48 cores
rbcL_55										
NSGA-II	6.831±0.087	12.279±0.172	16.772±0.178	20.178±0.284	26.435±0.444	85.393	76.745	69.884	63.057	55.073
SPEA2	6.873±0.067	12.157±0.168	16.553±0.159	19.865±0.286	26.075±0.405	85.912	75.978	68.970	62.077	54.323
IBEA	6.950±0.104	12.321±0.137	16.834±0.232	20.208±0.233	26.556±0.497	86.869	77.005	70.142	63.149	55.326
MO-FA	7.206±0.079	13.217±0.099	17.574±0.238	21.332±0.254	28.471±0.416	90.079	82.607	73.226	66.661	59.315
MOEA/D	7.073±0.054	12.949±0.140	18.055±0.275	21.824±0.313	29.074±0.436	88.408	80.936	75.229	68.200	60.571
MOABC	**7.235±0.072**	**14.464±0.167**	**21.639±0.171**	**28.823±0.250**	**40.694±0.322**	**90.432**	**90.398**	**90.161**	**90.072**	**84.779**
Fungi_88										
NSGA-II	7.406±0.040	13.849±0.127	19.533±0.206	23.338±0.372	29.333±0.515	92.576	86.553	81.386	72.931	61.110
SPEA2	7.403±0.015	13.712±0.130	19.327±0.245	22.796±0.380	28.634±0.510	92.540	85.703	80.529	71.237	59.654
IBEA	7.405±0.036	13.869±0.075	19.555±0.181	23.458±0.458	29.665±0.483	92.559	86.679	81.477	73.307	61.802
MO-FA	**7.490±0.024**	13.934±0.138	19.764±0.244	24.029±0.310	30.765±0.533	**93.620**	87.086	82.348	75.092	64.093
MOEA/D	7.484±0.054	13.956±0.140	19.788±0.276	24.406±0.313	31.159±0.436	93.552	87.228	82.448	76.268	64.914
MOABC	7.424±0.041	**14.810±0.122**	**22.037±0.233**	**29.219±0.312**	**42.808±0.470**	92.794	**92.562**	**91.820**	**91.309**	**89.184**
mtDNA_186										
NSGA-II	7.208±0.148	12.405±0.264	17.084±0.359	20.858±0.405	27.018±0.518	90.096	77.533	71.184	65.181	56.288
SPEA2	7.170±0.143	12.346±0.260	16.542±0.347	19.950±0.426	26.188±0.508	89.621	77.164	68.924	62.344	54.558
IBEA	7.210±0.144	12.897±0.271	17.562±0.278	21.173±0.392	27.695±0.386	90.122	80.604	73.173	66.165	57.698
MO-FA	**7.453±0.144**	13.402±0.251	17.880±0.409	22.077±0.421	29.620±0.527	**93.161**	83.763	74.500	68.990	61.709
MOEA/D	7.305±0.062	13.114±0.132	18.059±0.205	21.953±0.389	29.248±0.345	91.311	81.963	75.247	68.602	60.934
MOABC	7.243±0.058	**14.455±0.167**	**21.636±0.153**	**28.835±0.202**	**40.850±0.316**	90.537	**90.341**	**90.151**	**90.109**	**85.104**
RDPII_218										
NSGA-II	7.277±0.158	13.177±0.252	17.425±0.373	20.943±0.429	27.358±0.591	90.960	82.355	72.604	65.447	56.996
SPEA2	7.153±0.123	13.093±0.241	17.490±0.374	20.348±0.424	26.495±0.571	89.418	81.832	72.875	63.586	55.197
IBEA	7.297±0.135	13.369±0.261	18.007±0.393	21.211±0.574	27.617±0.759	91.207	83.558	75.031	66.283	57.535
MO-FA	**7.509±0.121**	13.576±0.212	18.062±0.416	22.376±0.462	30.134±0.626	**93.866**	84.849	75.260	69.925	62.779
MOEA/D	7.309±0.171	13.465±0.226	18.247±0.417	22.673±0.587	30.217±0.860	91.366	84.153	76.027	70.854	62.951
MOABC	7.260±0.084	**14.501±0.178**	**21.627±0.209**	**28.812±0.255**	**42.864±0.362**	90.745	**90.632**	**90.111**	**90.037**	**89.301**
ZILLA_500										
NSGA-II	7.650±0.089	14.260±0.177	20.254±0.226	25.459±0.383	35.325±0.421	95.622	89.127	84.391	79.561	73.593
SPEA2	7.642±0.155	14.289±0.192	20.222±0.271	24.597±0.373	34.400±0.479	95.526	89.306	84.257	76.864	71.668
IBEA	7.684±0.181	14.572±0.103	20.726±0.199	25.637±0.311	35.854±0.505	96.054	91.075	86.358	80.115	74.696
MO-FA	**7.764±0.087**	**15.113±0.253**	21.589±0.359	27.121±0.362	38.536±0.548	**97.050**	**94.455**	89.952	84.754	80.284
MOEA/D	7.702±0.095	15.066±0.210	21.738±0.125	26.845±0.379	38.240±0.727	96.277	94.164	90.574	83.890	79.668
MOABC	7.532±0.096	15.052±0.162	**22.568±0.243**	**30.060±0.356**	**44.761±0.424**	94.156	94.072	**94.034**	**93.937**	**93.252**
Mean Results										
NSGA-II	7.274	13.194	18.214	22.155	29.094	90.930	82.463	75.890	69.235	60.612
SPEA2	7.248	13.119	18.027	21.511	28.358	90.603	81.996	75.112	67.223	59.080
IBEA	7.309	13.406	18.537	22.337	29.478	91.364	83.784	77.236	69.804	61.412
MO-FA	**7.484**	13.848	18.974	23.387	31.505	**93.555**	86.553	79.058	73.084	65.636
MOEA/D	7.375	13.710	19.177	23.540	31.588	92.183	85.689	79.905	73.563	65.808
MOABC	7.339	**14.656**	**21.901**	**29.150**	**42.395**	91.735	**91.603**	**91.256**	**91.093**	**88.324**

To tackle this problem, solutions are encoded by means of NxN-sized floating-point matrices containing evolutionary distances between organisms [27]. The reconstruction of the phylogenies associated to the matrices processed by the MOEAs is carried out by using the BIONJ method [5]. In addition, topological and branch length optimizations [28] are applied to complement the tree-building method. These implementation decisions have been adopted in accordance with results from previous research [29], [30], in such a way that we are herein considering the best found solution encoding and reconstruction methods for addressing the multiobjective formulation of the problem.

Regarding evolutionary operators, NSGA-II, SPEA2, and IBEA include binary tournament selection, uniform crossover based on row interchanges [27], and a mutation operator that modifies matrix entries according to a gamma-distributed factor [31]. MOEA/D also uses the mentioned crossover and mutation operators, being the selection handled by choosing two random subproblems in the neighbourhood of the currently processed one [10]. MO-FA adapts the movement operator (Equation 1) to the matrix case, applying it over the entries $m[u, v]$ for each pair of species u, v. This idea is also used in the employed formula in MOABC (Equation 3), including gamma-based mutation in the onlooker calculations and the initialization and optimization of new solutions in the scout bee searches. Due to the characteristics of the problem, our implementations apply dynamic policies in the scheduling of parallel loops.

5 Experimentation and Evaluation

This section undertakes the experimental assessment of the parallel MOEAs under analysis. In a first step, we will report the parallel results obtained by the MOEAs over different system / problem sizes, along with evaluating the generated Pareto fronts under different multiobjective quality metrics. Afterwards, each design trend will be discussed by identifying advantages and disadvantages. Finally, the implications of the attained results will be pointed out. In previous works, we gave account of successful comparisons with other bioinformatics tools and methods from the literature when using multiobjective algorithms in the tackled problem [22], [24]. Therefore, such comparisons are out of the scope of this paper, which is focused on the comparative evaluation and discussion of parallel MOEA trends.

Experiments have been conducted on a shared-memory multiprocessor system composed of four twelve-core AMD Opteron 'Magny-Cours' 6174 processors (a total of 48 cores) at 2.2 GHz with 12MB L3 cache and 64GB DDR3 RAM, running Ubuntu 14.04 LTS as operating system and using GCC 5.2.1 (-O3 optimization flag) to compile the tested software. Five real phylogenetic datasets (given by Table 1) were considered to test the methods under evaluation.

For statistical validation purposes, results samples were examined according to the following statistical methodology [37] (with a confidence level of 95%). The Kolmogorov-Smirnov test was firstly used to find out if the samples

TABLE 7
Multiobjective performance results - hypervolume and spacing

Dataset	NSGA-II	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
			Hypervolume (%)			
rbcL_55	71.237±0.155	71.227±0.153	71.432±0.052	71.473±0.079	71.418±0.048	**71.732±0.018**
Fungi_88	77.797±0.021	77.790±0.047	77.809±0.015	77.814±0.044	77.806±0.028	**77.872±0.019**
mtDNA_186	69.718±0.112	69.631±0.158	69.830±0.084	70.004±0.008	69.981±0.006	**70.021±0.014**
RDPII_218	73.625±0.054	73.727±0.066	74.302±0.062	**74.725±0.076**	74.631±0.118	74.641±0.050
ZILLA_500	71.800±0.034	71.799±0.019	72.337±0.042	72.959±0.023	72.671±0.026	**73.016±0.012**
Mean	72.835	72.834	73.142	73.395	73.301	**73.456**
			Spacing (%)			
rbcL_55	0.080	**0.060**	0.094	0.128	0.102	0.087
Fungi_88	0.078	**0.059**	0.077	0.123	0.083	0.067
mtDNA_186	0.101	0.099	**0.078**	0.103	0.107	0.079
RDPII_218	0.025	0.043	0.021	0.033	0.029	**0.017**
ZILLA_500	0.111	0.127	0.108	0.051	0.091	**0.031**
Mean	0.079	0.077	0.076	0.087	0.082	**0.056**

TABLE 8
Statistical testing results - hypervolume I_H (✓=significant differences, ×=non-significant)

rbcL_55

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	×	✓	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			×	×	✓
MO-FA				✓	✓
MOEA/D					✓

Fungi_88

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	×	✓	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			×	×	✓
MO-FA				×	✓
MOEA/D					✓

mtDNA_186

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	✓	✓	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			✓	✓	✓
MO-FA				✓	✓
MOEA/D					✓

RDPII_218

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	✓	✓	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			✓	✓	✓
MO-FA				✓	✓
MOEA/D					×

ZILLA_500

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	×	✓	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			✓	✓	✓
MO-FA				✓	✓
MOEA/D					✓

TABLE 9
Statistical testing results - speedup for 48 cores SU (✓=significant differences, ×=non-significant)

rbcL_55

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	✓	×	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			✓	✓	✓
MO-FA				✓	✓
MOEA/D					✓

Fungi_88

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	✓	×	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			✓	✓	✓
MO-FA				✓	✓
MOEA/D					✓

mtDNA_186

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	✓	✓	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			✓	✓	✓
MO-FA				✓	✓
MOEA/D					✓

RDPII_218

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	✓	×	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			✓	✓	✓
MO-FA				×	✓
MOEA/D					✓

ZILLA_500

Method	SPEA2	IBEA	MO-FA	MOEA/D	MOABC
NSGA-II	✓	✓	✓	✓	✓
SPEA2		✓	✓	✓	✓
IBEA			✓	✓	✓
MO-FA				×	✓
MOEA/D					✓

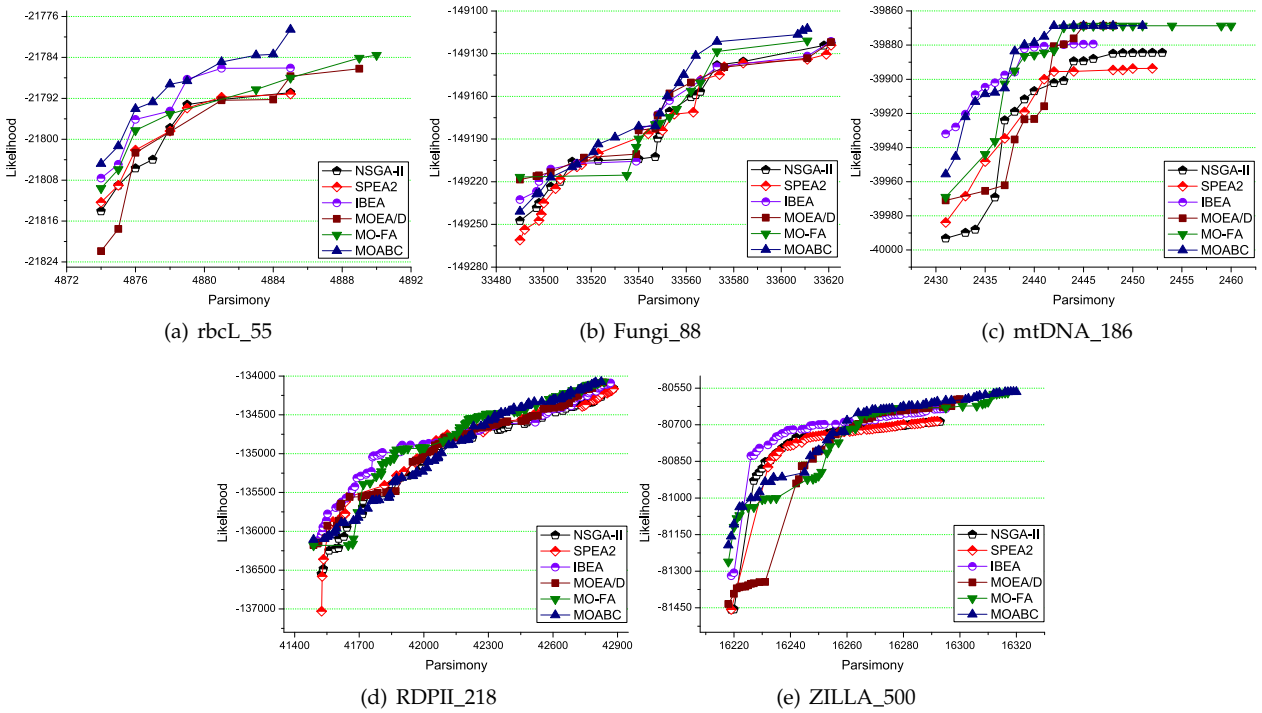
under comparison followed a Gaussian distribution. If so, the Levene test was conducted to study homoscedasticity, applying accordingly the ANOVA test if homogeneity in variances was detected. In the remaining cases (non-Gaussian distributions / no homogeneity in variances), we applied the Wilcoxon-Mann-Whitney test.

Regarding MOEAs configuration, we conducted parametric studies in which a range of uniformly distributed values for each input parameter were examined in order to find the best input parameter settings for each MOEA. Those parameters that showed a more noticeable influence in the multiobjective quality (hypervolume) of the generated Pareto fronts were configured firstly, proceeding subsequently with the remaining ones. Table 2 details the parameter values tested in these parametric studies, while Table 3 provides the best settings found in the comparisons. The neighbourhood size parameter in MOEA/D was set to a value dependent of the number of cores used N_C in order to have useful workload for all the execution threads. The evolutionary models GTR+Γ and LG08+Γ for phylogenetic analysis were adopted in our experiments according to the outputs of the software jModelTest and ProtTest [5], [38].

5.1 Performance Metrics and Results

We have applied different metrics to examine the parallel and multiobjective results achieved by each parallel MOEA. Parallel performance has been assessed under speedup and efficiency [20], two metrics which measure the improvement in execution time observed with regard to the serial version of the application and the average utilization of processing units. As for multiobjective metrics, we have employed 1) hypervolume [21] to calculate the area of the objective space which is weakly-dominated by at least one point in the outcome of a MOEA; and 2) spacing [39] to measure the uniformity of the Pareto front distribution in terms of distances between neighbouring points. For each analyzed dataset, Table 4 provides the execution times reported by each MOEA serial counterpart, while Table 5 defines the ideal and nadir points used in hypervolume calculations. Considering these points, hypervolume computations were performed over normalized objective scores in the scale [0,1] to avoid the influence of different ranges in objective values.

To test parallel performance, we have performed 11 independent runs per parallel MOEA and dataset over system configurations involving 8, 16, 24, 32, and 48 cores. Table 6 provides the observed median speedups and interquartile ranges (columns 2-6) along with the corresponding efficiencies (columns 7-11) for each algorithm. In general terms, the non-generational MOABC reports the best overall scalability, being able to speed up the computations up to 44.76x when using all the 48 processing cores. In fact, the efficiency values for this design denote a more stable behaviour on the solution of different problem sizes, ranging from 84.78% (rbcL_55) to 93.25% (ZILLA_500) in comparison to the efficiencies attained by the remaining algorithms: 60.57-79.67% (MOEA/D), 59.32-80.28% (MO-FA), 55.33-74.70% (IBEA), 55.07-73.59% (NSGA-II), and 54.32-71.67% (SPEA2). Among the generational approaches, MO-FA and MOEA/D achieve the most satisfying results, even surpassing MOABC when lower numbers of cores are involved in the computations.



(a) rbcL_55 (b) Fungi_88 (c) mtDNA_186

(d) RDPII_218 (e) ZILLA_500

Fig. 3. Multiobjective performance results - Pareto fronts taken from the median hypervolume executions

Regarding multiobjective performance, we have analyzed the outcomes from 31 independent runs per dataset and parallel MOEA (31 samples that were randomly taken from the same experimentation carried out for the parallel performance study). Table 7 provides the observed median hypervolume results (upper side), along with the spacing values (bottom side) for the median-hypervolume fronts represented in Figure 3. In the case of MOEA/D, the results in these tables measure the median behaviour observed for the different configurations of neighbourhood sizes considered in the parallel performance experiments.

The hypervolume scores point out the non-generational MOABC as the algorithm that reports the most satisfying results under this metric (a mean value of 73.5%, obtaining the highest score in rbcL_55, Fungi_88, mtDNA_186, and ZILLA_500), followed by the generational MO-FA (73.4%, achieving the best score in RDPII_218) and MOEA/D (73.3%). Competitive hypervolume results are also obtained by NSGA-II, SPEA2, and IBEA, reporting the latter significant performance in datasets with a reduced number of input sequences (rbcL_55 and Fungi_88). Nevertheless, NSGA-II, SPEA2, and IBEA take a more noticeable role in the spacing comparison, since they give account of satisfying distributions of solutions in four datasets that result into mean spacing values below 0.08. MOABC also verifies significant performance from the diversity perspective, in accordance with the spacing values reported for RDPII_218, and ZILLA_500 (mean value of 0.06).

The results of the statistical testing of multiobjective and parallel results are shown in Tables 8 and 9. Table 8 gives account of statistically significant differences in hypervolume in most of the comparison scenarios. Non-significant differences were found between NSGA-II and SPEA2 in rbcL_55 (P-value = 0.4), Fungi_88 (0.8), and ZILLA_500 (0.3), IBEA and MO-FA / MOEA/D in rbcL_55 (0.1 / 0.3) and Fungi_88 (0.1 / 0.1), MOEA/D and MO-FA in Fungi_88 (0.1), and finally MOEA/D and MOABC in RDPII_218 (0.3). Regarding speedups, the statistical testing in Table 9 confirms that statistically significant differences were found in almost all the cases where the entire hardware infrastructure was used for acceleration purposes, being non-significant differences only verified between members of the same parallel MOEA category (NSGA-II - IBEA in rbcL_55 and RDPII_218, and MOEA/D - MO-FA in RDPII_218 and ZILLA_500).

5.2 Other Hardware Platforms

In order to further evaluate the parallel MOEA trends under study, we now examine the parallel results obtained in two alternative hardware infrastructures: 1) a medium-sized multiprocessor system composed of two eight-core Intel Xeon E5-2630v3 processors (a total of 16 processing cores) at 2.4 GHz with 20MB L3 cache and 80GB DDR3 RAM; and 2) a low-sized commodity system comprising an Intel i7-2600 CPU (4 processing cores) at 3.4 GHz with 8MB L3 cache and 8GB DDR3 RAM. Table 10 includes the serial times used as reference for the calculation of speedups and efficiencies in these two hardware setups.

The achieved median speedups and efficiencies (from 11 independent runs per experiment) are reported in Table 11 (Xeon setup, using system configurations of 8 and 16 cores) and Table 12 (i7 setup, using 4 cores). The multiprocessor scenario (16 cores) in Table 11 shows that the non-generational MOABC is able to achieve a mean speedup value of 14.22x, thus representing the best trend in overall terms with a mean efficiency of 88.84%. However, this scenario also gives account of the fact that the generational approaches based on solution quality (MO-FA

TABLE 10
Serial execution time (in seconds) for the Xeon and i7 platforms

	NSGA-II		SPEA2		IBEA	
Dataset	Xeon	i7	Xeon	i7	Xeon	i7
rbcL_55	2728.15	2501.44	2784.75	2520.71	2793.45	2528.84
Fungi_88	29854.38	29387.12	30225.22	29782.92	30417.72	30037.95
mtDNA_186	24344.49	23455.70	24468.90	23796.92	24569.57	23791.62
RDPII_218	25850.61	24254.79	25916.94	24477.56	26360.49	24614.88
ZILLA_500	35537.28	33214.71	35625.95	33305.39	35775.17	33379.55
	MO-FA		MOEA/D		MOABC	
Dataset	Xeon	i7	Xeon	i7	Xeon	i7
rbcL_55	2798.48	2538.53	2697.85	2470.54	2810.04	2586.45
Fungi_88	31442.41	30652.01	29732.70	28652.08	29737.44	28244.67
mtDNA_186	24624.69	24236.39	24089.69	23124.18	23349.16	22203.46
RDPII_218	26522.65	24580.83	25497.85	24100.44	25045.55	23689.02
ZILLA_500	36078.48	33780.20	35209.83	33035.28	35127.26	32793.54

TABLE 11
Speedups and efficiencies on the 16-core Intel Xeon system

	Speedup		Efficiency(%)	
Algorithm	8 cores	16 cores	8 cores	16 cores
rbcL_55				
NSGA-II	7.293±0.068	12.219±0.233	91.164	76.370
SPEA2	7.201±0.095	12.182±0.235	90.009	76.140
IBEA	7.352±0.087	12.250±0.092	91.897	76.560
MO-FA	**7.424±0.057**	12.907±0.072	**92.803**	80.669
MOEA/D	7.364±0.093	12.628±0.164	92.056	78.926
MOABC	7.305±0.062	**13.753±0.067**	91.316	**85.955**
Fungi_88				
NSGA-II	7.337±0.073	13.762±0.077	91.711	86.015
SPEA2	7.216±0.012	13.541±0.148	90.196	84.633
IBEA	7.470±0.050	13.974±0.183	93.370	87.337
MO-FA	7.523±0.059	14.131±0.073	94.039	88.319
MOEA/D	**7.626±0.010**	**14.310±0.024**	**95.322**	**89.441**
MOABC	7.481±0.057	14.237±0.061	93.513	88.980
mtDNA_186				
NSGA-II	7.331±0.119	12.471±0.201	91.633	77.942
SPEA2	7.235±0.131	12.461±0.272	90.436	77.880
IBEA	7.434±0.132	12.514±0.202	92.927	78.214
MO-FA	**7.579±0.146**	12.950±0.225	**94.733**	80.939
MOEA/D	7.541±0.130	12.875±0.166	94.269	80.466
MOABC	7.529±0.019	**14.273±0.050**	94.110	**89.203**
RDPII_218				
NSGA-II	7.323±0.154	13.114±0.158	91.535	81.961
SPEA2	7.279±0.101	13.021±0.338	90.990	81.381
IBEA	7.397±0.135	13.242±0.365	92.456	82.763
MO-FA	**7.538±0.197**	13.844±0.384	**94.219**	86.523
MOEA/D	7.478±0.090	13.758±0.457	93.478	85.985
MOABC	7.395±0.106	**14.382±0.200**	92.442	**89.890**
ZILLA_500				
NSGA-II	7.403±0.074	13.955±0.281	92.532	87.220
SPEA2	7.410±0.020	13.850±0.191	92.621	86.564
IBEA	7.399±0.121	14.184±0.282	92.491	88.649
MO-FA	7.637±0.106	**14.527±0.319**	95.460	**90.794**
MOEA/D	**7.646±0.038**	14.494±0.183	**95.580**	90.585
MOABC	7.353±0.124	14.429±0.309	91.910	90.181
Mean Results				
NSGA-II	7.337	13.104	91.715	81.902
SPEA2	7.268	13.011	90.850	81.320
IBEA	7.410	13.233	92.628	82.705
MO-FA	**7.540**	13.672	**94.251**	85.449
MOEA/D	7.531	13.613	94.141	85.081
MOABC	7.413	**14.215**	92.658	**88.842**

and MOEA/D) attain the best parallel results in the two most time-consuming datasets, Fungi_88 (MOEA/D) and ZILLA_500 (MO-FA). In fact, these algorithms represent the best trend in the single-processor scenario (8 cores), reporting mean efficiencies of 94.25% and 94.14% in comparison to the 92.66% from MOABC. The generational NSGA-II, SPEA2, and IBEA also achieve satisfying performance in the 8-core configuration, especially for the case of ZILLA_500 where these algorithms improve the results from MOABC.

Regarding Table 12, the 4-core scenario from the i7 platform highlights MO-FA and MOEA/D as the parallel MOEAs that accomplish the best exploitation of commodity resources (showing efficiencies in the range 93.46% - 98.72% and 93.10% - 98.66%, respectively). The generational methods based on solution quality and diversity are also

TABLE 12
Speedups and efficiencies on the 4-core Intel i7 system

	Speedup	Efficiency(%)	Speedup	Efficiency(%)
Algorithm	4 cores	4 cores	4 cores	4 cores
	rbcL_55		Fungi_88	
NSGA-II	3.677±0.052	91.920	3.860±0.041	96.495
SPEA2	3.556±0.044	88.907	3.853±0.021	96.317
IBEA	3.698±0.018	92.458	3.877±0.020	96.913
MO-FA	**3.739±0.020**	**93.464**	3.882±0.017	97.055
MOEA/D	3.724±0.043	93.099	**3.893±0.042**	**97.323**
MOABC	3.402±0.045	85.051	3.558±0.030	88.958
	mtDNA_186		RDPII_218	
NSGA-II	3.729±0.019	93.225	3.825±0.083	95.632
SPEA2	3.712±0.017	92.811	3.821±0.066	95.535
IBEA	3.771±0.055	94.279	3.831±0.060	95.774
MO-FA	**3.797±0.039**	**94.930**	**3.879±0.017**	**96.971**
MOEA/D	3.796±0.012	94.909	3.855±0.026	96.386
MOABC	3.556±0.013	88.911	3.551±0.016	88.773
	ZILLA_500		Mean Results	
NSGA-II	3.863±0.026	96.568	3.791	94.768
SPEA2	3.862±0.032	96.562	3.761	94.026
IBEA	3.895±0.015	97.380	3.814	95.361
MO-FA	**3.949±0.038**	**98.724**	**3.849**	**96.229**
MOEA/D	3.946±0.032	98.657	3.843	96.075
MOABC	3.560±0.033	89.012	3.526	88.141

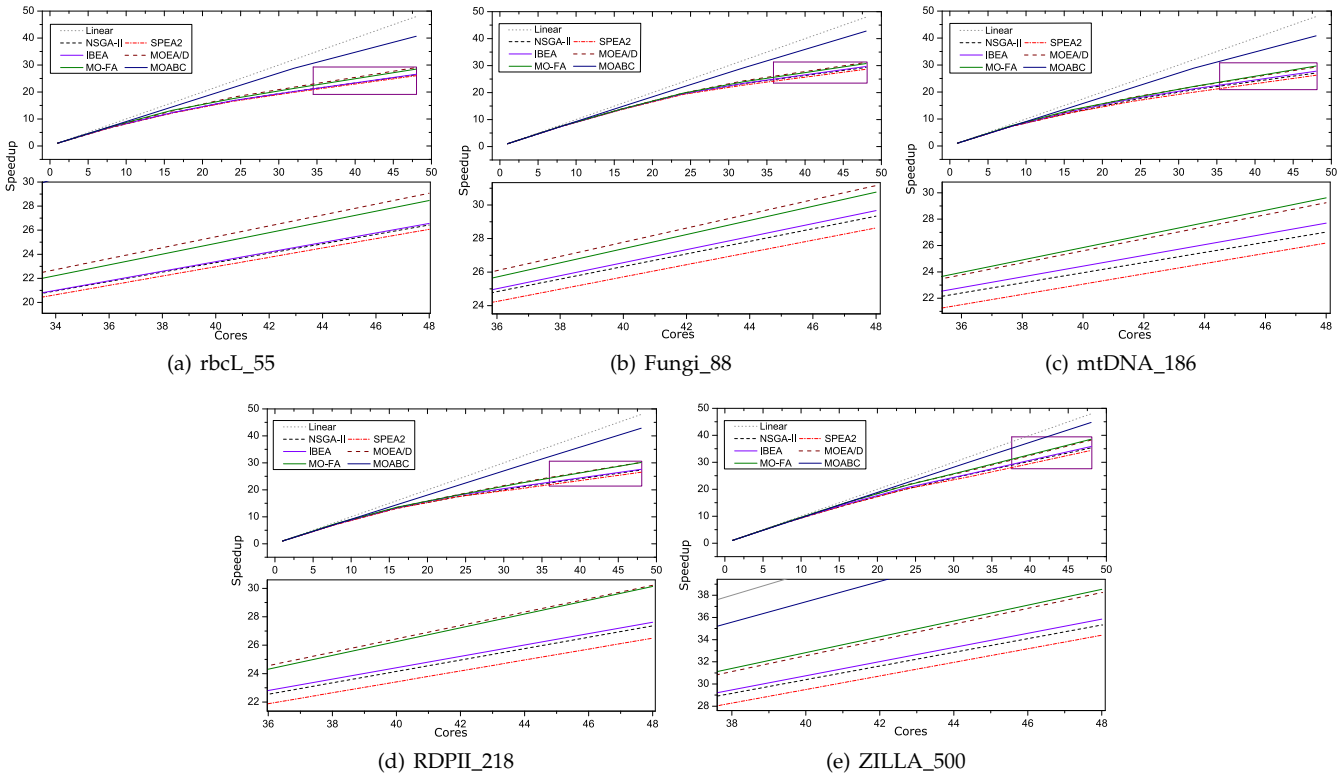
able to achieve mean efficiencies over 94%, representing the second best approach for the considered low-sized setup. On the other hand, the non-generational MOABC reports efficiencies in the range 85.05% - 89.01%, thus not reaching the parallel results of the generational categories.

5.3 Discussion

By plotting the speedups obtained by each parallel MOEA, we can distinguish three main trends in accordance with the different categories considered in this study. Figure 4 provides a representation of the parallel scalability attained by each parallel MOEA on the 48-core AMD system. On the basis of this representation, we now discuss the relationship between the characteristics of each parallel design and the observed parallel and multiobjective results, identifying key advantages and weaknesses for each category.

5.3.1 Non-generational Designs

Attending to parallel scalability, the leading trend in our comparative analysis is given by the non-generational approach represented by MOABC. The observed parallel results give account of how the idea of allowing execution threads to carry out parallel tasks in a continuous way (without the synchronizations introduced by the generational scheme) leads to a significant exploitation of shared-memory resources (with efficiencies over 84% in all the evaluation scenarios for the AMD system). Particularly, the solution of the two most time-consuming datasets (Fungi_88 and ZILLA_500) can be undertaken with 48 cores in only 20 and 24 minutes, respectively, representing significant time reductions in comparison to the 14.5 and 18.2 hours required by the serial implementation. On measuring the impact of overhead sources in ZILLA_500, we observed that these sources contribute only a 2% of the overall execution time for 48 cores. On the other hand, the overhead for the generational designs has an impact, in average terms, of 17% over the observed execution time. Hence, the performance penalty introduced by overhead times is almost 9 times lower in the case of the non-generational approach. This explains the improved parallel results reported when the whole shared-memory system was employed.



(a) rbcL_55 (b) Fungi_88 (c) mtDNA_186

(d) RDPII_218 (e) ZILLA_500

Fig. 4. Evolution of median speedups for each parallel MOEA, in comparison with the theoretical linear speedup (AMD Opteron system)

From a multiobjective perspective, MOABC is able to report significant performance in the solution of the tackled problem. Particularly, the attained hypervolume scores point out the accuracy of this approach to obtain high-quality Pareto solutions. The different multiobjective search mechanisms integrated in this parallel metaheuristic allow it to undertake the optimization process by following different strategies. These include the intra (employed bees) and inter-level (employed-onlooker interactions) collaborative exploitation of promising solutions identified through Pareto rankings, along with the scout bee exploration tasks to address local optima issues. In addition, the spacing indicator suggests that MOABC leads to a good distribution of solutions in almost all the datasets, thanks to the use of diversity mechanisms which complement the convergence ones in the fitness assignment procedure.

The main weakness of this design is shown in scenarios involving low numbers of processing cores. For the 8-core case of the AMD setup, the efficiencies from MOABC are lower than the ones reported by MO-FA - MOEA/D (in Fungi_88, mtDNA_186, RDPII_218, and ZILLA_500), NSGA-II - IBEA (RDPII_218 and ZILLA_500), and SPEA2 (ZILLA_500). Such behaviour is also observed in the 8-core scenario from the Xeon system, becoming the issue even more noticeable in the 4-core i7 system (where MOABC reports the lowest efficiencies in the comparison). This is due to the fact that the non-generational approach requires the use of one core to manage master tasks exclusively, thus not participating in the generation and evaluation of new candidate solutions. The fact of using N_C-1 cores to handle parallel tasks leads to a worsening in performance with regard to the generational approaches, which can effectively use N_C cores in the parallel loops. This issue affects the ability of the non-generational approach to exploit multicore resources on lower-sized systems, despite the improved scalability shown for higher system configurations.

5.3.2 *Generational Designs Based on Measurements of Solution Quality Exclusively*

The second most significant trend in scalability identified in Figure 4 is given by the generational approaches MO-FA and MOEA/D, whose differences with NSGA-II, SPEA2, and IBEA become more noticeable as we increase the number of cores. The improvement observed in parallel scalability with regard to the other generational methods is motivated by the lower single sections fraction required and also by the significant efficiency observed at the processing of workshared loops. For 48 cores, the average workshared efficiency in MO-FA and MOEA/D shows values of 66.3% and 66.9%, respectively, outperforming the ones obtained by NSGA-II, SPEA2, and IBEA (61.3%, 59.9%, and 62.1%).

These factors along with the ability to use the total number of available cores in parallel computations explain the improved speedups and efficiencies reported for the 8-core configurations not only over NSGA-II, SPEA2, and IBEA, but also over MOABC. In fact, MO-FA and MOEA/D are still able to improve MOABC in the case of 16 cores for ZILLA_500 (AMD and Xeon setups) and Fungi_88 (Xeon). Moreover, these parallel MOEAs report the best parallel performance for the 4-core i7 setup, confirming their significance in scenarios involving low/medium-sized systems and computationally demanding problem instances.

Attending to multiobjective results, the search strategies implemented in MO-FA lead to the attainment of significant

hypervolume scores, reporting Pareto sets of good multiobjective quality according to the convergence property. However, since this design prioritizes lighter fitness calculations over the introduction of diversity information, the fronts tend to show poor uniformity in their distributions, as suggested by the spacing indicator. Figure 3 reveals that, in general terms, the algorithm is biased towards the processing of solutions surrounding the extreme points and the middle of the Pareto front (as it can be observed more clearly in Fungi_88 and ZILLA_500).

Although relevant hypervolume results are also attained, the multiobjective performance of the parallel MOEA/D shows a strong dependence on the neighbourhood size imposed by the availability of hardware resources. When the neighbourhoods are reduced to define effective workloads for all the processing cores, a general degradation of the generated results can be observed. For the case of RDPII_218 and ZILLA_500, the hypervolumes obtained in the 48-core scenario show a statistically significant worsening (from 74.63 and 72.67 to 74.02 and 72.14, respectively). An impact is also observed in the spacing scores, which become similar to the ones reported by MO-FA. In this sense, it is worth mentioning that the decomposition mechanism in the serial version of MOEA/D is claimed to attain good spreads of solutions [10]. However, the serial MOEA/D is governed by strict data dependencies (i.e. the solution of a particular subproblem depends on the solution of the previous ones for the whole population) that have to be re-organized for parallelization purposes.

Consequently, the improved parallel performance shown by these generational approaches comes at the expense of an impact in some multiobjective properties. These potential issues in the generated outcomes represent the main disadvantage of the algorithms considered in this design trend.

5.3.3 *Generational Designs Based on Measurements of Solution Quality and Diversity*

From a parallel scalability perspective, the generational NSGA-II, SPEA2, and IBEA represent the last trend in this comparative analysis. In these designs, the non-parallelizable fraction of the application is governed not only by the introduction of synchronization points due to the generational approach, but also by the more prominent role played by serial sections in the calculations when both quality and diversity mechanisms are considered and employed to process the whole population. According to Amdalh's law [20], increased non-parallelizable fractions have a higher impact as a limiting factor in the achievable parallel scalability, leading this category to the less significant parallel results in the 48-core setup. Consequently, these parallel MOEAs attain more satisfying performance in medium and low-sized infrastructures, in accordance with the results achieved in the Xeon and i7 setups.

Regarding multiobjective performance, the observed hypervolumes show that these evolutionary designs do not reach, in overall terms, the mean quality attained by the other approaches under analysis. In spite of that, the spacing values suggest a satisfying distribution of solutions in the Pareto fronts obtained by NSGA-II, SPEA2, and IBEA. In fact, if we discard the most parsimonious solutions obtained in ZILLA_500, the spacing scores become 0.029 (NSGA-II), 0.032 (SPEA2), and 0.030 (IBEA), thus outperforming MO-FA and MOEA/D in this dataset along with matching MOABC. As a result, this class of generational MOEAs (involving solution plus diversity mechanisms) becomes more relevant in optimization scenarios where a more uniformly-distributed range of solutions is desirable, despite the loss of parallelism potential with regard to other designs.

5.4 Implications

The previous experimental assessment gave account of the main opportunities and flaws of each design in this comparative analysis. When choosing among these intra-algorithm parallel MOEA approaches, we must take into account not only the preferences of the expert in terms of multiobjective performance but also the characteristics of the underlying hardware setup where the MOEAs are run.

The first scenario we consider here is the one where the parallel MOEA runs on a commodity platform. These widely-used systems are characterized by the presence of a single or a low number of multicore processors (comprising a total number of around 4-8 cores). According to our results, the generational approaches represent a good choice to attain an accurate exploitation of parallel resources in these systems. If priority is put on the achievement of high parallel efficiencies, we can rely on the use of designs like MO-FA and MOEA/D, which showed the additional advantage of obtaining a significant exploitation of resources also in medium-sized infrastructures and good multiobjective results from a hypervolume perspective. On the other hand, NSGA-II, SPEA2, and IBEA may be selected instead in order to accomplish the fulfillment of the diversity property. Finally, the non-generational approach MOABC can lead to performance degradation especially in low-sized infrastructures, as shown in the case of the 4-core i7 setup. More specifically, the solution of time-demanding MOPs on commodity hardware setups is affected by the master-core issue, since we are dedicating one entire processing core to supporting operations instead of collaborating on the parallelization of highly complex calculations.

The second scenario is given by the execution of parallel MOEAs on high performance computing systems. These hardware configurations comprise a high number of processing units whose accurate exploitation depends on the scalability capabilities of the parallel design. In this scenario, the non-generational approach arises as the most suitable choice to take full advantage of the huge parallel processing potential in these systems (although generational approaches like MO-FA and MOEA/D can also lead to significant parallel results over complex problem instances, as shown in ZILLA_500). One important feature to be considered is given by the fact that, in this kind of designs, the theoretical scalability is limited by the population size. In the multiobjective case, we are often dealing with complex optimization processes involving a number of individuals which tends to be higher than the number of cores in current shared-memory multicore multiprocessor systems. As a result, the theoretical limit does not show any relevant influence in that context. On the other hand, this limit can play a key role in the exploitation of large-scale systems

with hundreds or thousands cores. In order to deal with the limitation imposed by the population size, we can apply either simple strategies (such as using job-level parallelism to carry out multiple independent runs of the parallel MOEA) or more complex techniques which introduce changes in the algorithmic design. For example, several threads could be processing a copy of the same individual in the population but applying different operators, in such a way that different candidate solutions are generated from common starting points. However, the use of such techniques requires further research, as they modify the behaviour of the search engine and, consequently, the boundary between parallelism and a potential impact in solution quality must be established.

Summing up, non-generational intra-algorithm parallel designs represent a robust choice when the complexity of the problem requires the use of advanced hardware setups. On their behalf, the generational approaches still find their room of application when the access to high performance systems is not possible. In this case, the selection of a particular generational MOEA must be carried out according to the expected effects of the algorithmic design over the desired parallel and multiobjective performance.

6 Conclusions

The present work has focused on the analysis of parallel metaheuristic designs for multiobjective optimization. As the time complexity of real-world MOPs keeps growing with the adoption of more realistic assumptions, thorough studies are required to determine the most accurate strategies to speed up the computations and achieve high-quality outcomes. For this purpose, we have undertaken a comparative analysis of intra-algorithm parallel MOEAs with the aim of identifying the influence of their algorithmic features and parallelism properties on the attained performance. The parallel MOEA designs herein considered include generational approaches integrating solution quality plus diversity mechanisms (NSGA-II, SPEA2, IBEA), solution quality mechanisms exclusively (MO-FA, MOEA/D), and non-generational approaches (MOABC).

We have adapted these parallel designs to tackle the challenging phylogenetic reconstruction problem, performing our experimentation on three parallel systems comprising 4, 16, and 48 processing cores, respectively. Different metrics to measure parallel performance (speedup, efficiency) and multiobjective quality (hypervolume, spacing) have given account of the main advantages and weaknesses of each parallel MOEA under evaluation. In terms of parallel scalability, we have identified the non-generational parallel designs as the most suitable approach for exploiting setups with high numbers of cores, followed by the generational parallel designs focused on solution quality, whose results become more relevant on low and medium-sized system configurations. On the basis of these results and the quality of the generated Pareto fronts, we have discussed key guidelines for the selection of parallel MOEAs on commodity and high performance computing platforms.

Our future work lines are aimed at providing further insight into the performance of parallel MOEAs on other hardware platforms beyond the shared-memory scenario. We will conduct the comparative evaluation of intra-algorithm parallel designs on heterogeneous systems, distributed environments, and hybrid shared-distributed memory setups, in order to verify if the conclusions of the current study can be extended to platforms with different parallel processing capabilities. For example, for the case of shared-nothing architectures and distributed systems in general, it is needed an accurate, in-depth assessment of the relationship between the algorithmic design and the multiple factors that govern parallel performance in distributed contexts. Such factors include communication patterns, message sizes and network latencies, processes management and topological organization, load balancing mechanisms, synchronization patterns and coordination among processes, replication of data structures, and fault tolerance requirements [40]. As for heterogeneous systems, we will discuss the suitability of adapting intra-algorithm parallel MOEAs to co-processors architectures such as GPUs, taking into account the impact of stochastic elements and population sizes in the achievable exploitation of parallel resources [41], [42]. We will also study hybrid approaches in which co-processors are applied to support CPU-based intra-algorithm schemes by accelerating operations with large data parallelism (e.g., internal hypervolume calculations [43]) and problem-dependant ones (acceleration of the inner loops of the objective functions). We will finally focus on examining in detail the application of non-generational approaches on large-scale systems.

Acknowledgments

This work was partially funded by the AEI (State Research Agency, Spain) and the ERDF (European Regional Development Fund, EU), under the contract TIN2016-76259-P (PROTEIN project). Thanks to the Junta de Extremadura for the GR15011 grant provided to the group TIC015. Sergio Santander-Jiménez is supported by the Post-Doctoral Fellowship from FCT (Fundação para a Ciência e a Tecnologia, Portugal) under Grant SFRH/BPD/119220/2016.

References

[1] K. Deb, "Multi-Objective Evolutionary Algorithms," in *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 995–1015.

[2] E. Alba, G. Luque, and S. Nesmachnow, "Parallel metaheuristics: recent advances and new trends," *International Transactions in Operational Research*, vol. 20, no. 1, pp. 1–48, 2013.

[3] F. Luna and E. Alba, "Parallel Multiobjective Evolutionary Algorithms," in *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 1017–1031.

[4] E. G. Talbi, "Parallel Evolutionary Combinatorial Optimization," in *Springer Handbook of Computational Intelligence*. Springer, 2015, pp. 1107–1125.

[5] P. Lemey, M. Salemi, and A.-M. Vandamme, *The Phylogenetic Handbook: a Practical Approach to Phylogenetic Analysis and Hypothesis Testing*. Cambridge: Cambridge Univ. Press, 2009.

[6] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A Fast and Elitist Multi–Objective Genetic Algorithm: NSGA-II," *IEEE Trans. Evol. Comput.*, vol. 6, no. 2, pp. 182–197, 2002.

[7] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm," in *Proc. of EUROGEN'02*, 2002, pp. 95–100.

[8] E. Zitzler and S. Künzli, "Indicator-Based Selection in Multiobjective Search," in *Parallel Problem Solving From Nature VIII*, ser. LNCS, vol. 3242. Springer Verlag, 2004, pp. 832–842.

[9] X. S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.

[10] 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.
[11] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: artificial bee colony (ABC) algorithm and applications," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 21–57, 2014.
[12] E. Cantú-Paz, "A Survey of Parallel Genetic Algorithms," *Calculateurs Parallèles, Réseaux et Systèmes Répartis*, vol. 10, no. 2, pp. 141–171, 1998.
[13] E. Alba and M. Tomassini, "Parallelism and Evolutionary Algorithms," *IEEE Trans. Evol. Comput.*, vol. 6, no. 5, pp. 443–462, 2002.
[14] D. V. Veldhuizen, J. Zydallis, and G. Lamont, "Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms," *IEEE Trans. Evol. Comput.*, vol. 7, no. 2, pp. 144–173, 2003.
[15] A. L. Jaimes and C. Coello, "Applications of Parallel Platforms and Models in Evolutionary Multi-Objective Optimization," in *Biologically-Inspired Optimisation Methods, Studies in Computational Intelligence*. Springer Verlag, 2009, vol. 210, pp. 23–49.
[16] E. G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, and C. Coello, "Parallel Approaches for Multiobjective Optimization," in *Multiobjective Optimization*, LNCS. Springer, 2008, vol. 5252, pp. 349–372.
[17] C. Coello, "Multi-objective Evolutionary Algorithms in Real-World Applications: Some Recent Results and Current Challenges," in *Advances in Evolutionary and Deterministic Methods for Design, Optimization and Control in Engineering and Sciences*. Springer, 2015, pp. 3–18.
[18] E. G. Talbi, "Parallel Multi-objective Evolutionary Algorithms," in *Proc. of the Seventh International Conference on Bioinspired Optimization Methods and their Applications, BIOMA 2016*, 2016, pp. 21–48.
[19] J. J. Durillo, A. J. Nebro, F. Luna, and E. Alba, "A Study of Master-Slave Approaches to Parallelize NSGA-II," in *11th International Workshop on Nature Inspired Distributed Computing (NIDISC 2008)*. IEEE Computer Society, 2008, pp. 1–8.
[20] B. Chapman, G. Jost, and R. van der Pas, *Using OpenMP: Portable Shared Memory Parallel Programming*. Cambridge, MA, USA: The MIT Press, 2007.
[21] N. Beume, C. M. Fonseca, M. López-Ibáñez, L. Paquete, and J. Vahrenhold, "On the Complexity of Computing the Hypervolume Indicator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 5, pp. 1075–1082, 2009.
[22] S. Santander-Jiménez and M. A. Vega-Rodríguez, "Parallel Multiobjective Metaheuristics for Inferring Phylogenies on Multicore Clusters," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1678–1692, 2015.
[23] S. Santander-Jiménez and M. A. Vega-Rodríguez, "Asynchronous Non-Generational Model to Parallelize Metaheuristics: A Bioinformatics Case Study," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 7, pp. 1825–1838, 2017.
[24] S. Santander-Jiménez and M. A. Vega-Rodríguez, "On the Design of Shared Memory Approaches to Parallelize a Multiobjective Bee-Inspired Proposal for Phylogenetic Reconstruction," *Information Sciences*, vol. 324, pp. 163–185, 2015.
[25] M. Steel, *Phylogeny: Discrete and Random Processes in Evolution*. Philadelphia: Society for Industrial & Applied Mathematics, 2016.
[26] M. Ott, J. Zola, S. Aluru, A. D. Johnson, D. Janies, and A. Stamatakis, "Large-scale phylogenetic analysis on current HPC architectures," *Sci. Programming*, vol. 16, no. 2-3, pp. 255–270, 2008.
[27] L. Poladian, "A GA for maximum likelihood phylogenetic inference using neighbour-joining as a genotype to phenotype mapping," in *GECCO 2005*, 2005, pp. 415–422.
[28] A. Goëffon, J. Richer, and J. Hao, "Progressive Tree Neighborhood Applied to the Maximum Parsimony Problem," *IEEE/ACM Trans. Comput. Biol. Bioinform.*, vol. 5, no. 1, pp. 136–145, 2008.
[29] S. Santander-Jiménez and M. A. Vega-Rodríguez, "Performance Analysis of Multiobjective Artificial Bee Colony Implementations for Phylogenetic Reconstruction," in *Proc. of NaBIC 2014*. IEEE, 2014, pp. 35–40.
[30] S. Santander-Jiménez and M. A. Vega-Rodríguez, "A Comparative Study on Distance Methods Applied to a Multiobjective Firefly Algorithm for Phylogenetic Inference," in *GECCO 2013 Companion*, 2013, pp. 1587–1594.
[31] P. O. Lewis, "A Genetic Algorithm for Maximum-Likelihood Phylogeny Inference Using Nucleotide Sequence Data," *Molecular Biology and Evolution*, vol. 15, no. 3, pp. 277–283, 1998.
[32] I. Morgenstern *et al.*, "A molecular phylogeny of thermophilic fungi," *Fungal Biology*, vol. 116, no. 4, pp. 489–502, 2012.
[33] M. Ingman and U. Gyllensten, "mtDB: Human Mitochondrial Genome Database, a resource for population genetics and medical sciences," *Nucleic Acids Research*, vol. 34 (D749-D751), 2006.
[34] J. R. Cole *et al.*, "The Ribosomal Database Project (RDP-II): sequences and tools for high-throughput rRNA analysis," *Nucleic Acids Research*, vol. 33 (Database issue D294-D296), 2005.
[35] M. W. Chase *et al.*, "Phylogenetics of seed plants: An analysis of nucleotide sequences from the plastid gene rbcl," *Annals of the Missouri Botanical Garden*, vol. 80, no. 3, pp. 528–580, 1993.
[36] S. Lee and O. Gascuel, "An improved general amino acid replacement matrix," *Mol. Biol. Evol.*, vol. 25, no. 7, pp. 1307–1320, 2008.
[37] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures. 5th edition*. NY, USA: Chapman & Hall/CRC, 2011.
[38] D. Darriba, G. L. Taboada, R. Doallo, and D. Posada, "jModelTest 2: more models, new heuristics and parallel computing," *Nature Methods*, vol. 9, no. 8, pp. 772–772, 2012.
[39] C. Coello, C. Dhaenens, and L. Jourdan, *Advances in Multi-Objective Nature Inspired Computing*. Berlin / Heidelberg: Springer, 2010.
[40] A. D. Kshemkalyani and M. Singhal, *Distributed Computing: Principles, Algorithms, and Systems*. New York, NY, USA: Cambridge University Press, 2008.
[41] W. Zhu, A. Yaseen, and Y. Li, "DEMCMC-GPU: An Efficient Multi-Objective Optimization Method with GPU Acceleration on the Fermi Architecture," *New Generation Computing*, vol. 29, no. 2, pp. 163–184, 2011.
[42] M. L. Wong and G. Cui, "Data Mining Using Parallel Multi-objective Evolutionary Algorithms on Graphics Processing Units," in *Massively Parallel Evolutionary Computation on GPGPUs*. Springer, 2013, pp. 287–307.
[43] E. M. López, L. M. Antonio, and C. Coello, "A GPU-Based Algorithm for a Faster Hypervolume Contribution Computation," in *EMO 2015: Evolutionary Multi-Criterion Optimization*, ser. LNCS, vol. 9019. Springer Verlag, 2015, pp. 80–94.

Sergio Santander-Jiménez received the Ph.D. degree in Computer Engineering from the University of Extremadura, Spain, in 2016. He is currently a Post-doctoral Fellow and a Senior Researcher at the R&D Instituto de Engenharia de Sistemas e Computadores (INESC-ID), Instituto Superior Técnico (IST), Universidade de Lisboa (UL), Portugal. He has co-organized several international workshops on high-performance computing, computational intelligence, computational biology and bioinformatics, reviewing articles on these topics for multiple international JCR-indexed journals. His main research interests include evolutionary and bioinspired computing, multi-objective optimization, parallel and distributed computing, and their applications to real-world biological problems.

Miguel A. Vega-Rodríguez received the Ph.D. degree in Computer Engineering from the University of Extremadura, Spain, in 2003. He is currently an Associate Professor (accredited as Full Professor) of computer architecture in the Department of Computer and Communications Technologies, University of Extremadura. He has authored or co-authored more than 630 publications including journal papers (more than 120 JCR-indexed journal papers), book chapters, and peer-reviewed conference proceedings, for which he got several awards - such as Best Paper Awards in ISDA'11, IBERGRID'11, ICEC'09, and IEA-AIE'08. He has contributed to the organization of several international conferences and workshops as general chair or co-chair. In addition, he has edited over 10 special issues of international JCR-indexed journals. His main research interests include parallel and distributed computing, evolutionary computation, bioinformatics, and reconfigurable and embedded computing.