Parallel Multi-Objective Evolutionary Algorithms: A Comprehensive Survey

Jesús Guillermo Falcón-Cardona[a,*], Raquel Hernández Gómez[a], Carlos A. Coello Coello[a,b], Ma. Guadalupe Castillo Tapia[c]

[a] *CINVESTAV-IPN, Department of Computer Science, Av. IPN No. 2508, Col. San Pedro Zacatenco, Mexico City, Mexico, 07300*
[b] *Basque Center for Applied Mathematics (BCAM) & Ikerbasque, Spain.*
[c] *UAM Azcapotzalco, Departamento de Administración, Av. San Pablo No. 180, Mexico City, Mexico, 02200*

Abstract

Multi-Objective Evolutionary Algorithms (MOEAs) are powerful search techniques that have been extensively used to solve difficult problems in a wide variety of disciplines. However, they can be very demanding in terms of computational resources. Parallel implementations of MOEAs (pMOEAs) provide considerable gains regarding performance and scalability and, therefore, their relevance in tackling computationally expensive applications. This paper presents a survey of pMOEAs, describing a refined taxonomy, an up-to-date review of methods and the key contributions to the field. Furthermore, some of the open questions that require further research are also briefly discussed.

Keywords: Multi-objective optimization, evolutionary algorithms, parallel computing

2010 MSC: 00-01, 99-00

[⋆] The first and second authors acknowledge support from CONACyT to pursue graduate studies in computer science at CINVESTAV-IPN. Carlos A. Coello Coello acknowledges support from the CONACyT project no. 1920, from a 2018 SEP-Cinvestav grant (application no. 4), and the Basque Government through the BERC 2018-2021 program by the Spanish Ministry of Science.

[*] Corresponding author

Email addresses: `jfalcon@tec.mx` (Jesús Guillermo Falcón-Cardona), `rhernandez@computacion.cs.cinvestav.mx` (Raquel Hernández Gómez), `ccoello@cs.cinvestav.mx` (Carlos A. Coello Coello), `mgct@azc.uam.mx` (Ma. Guadalupe Castillo Tapia)

1. Introduction

Many real-world applications can be formulated as multi-objective optimization problems (MOPs), which involve the simultaneous optimization of several, often conflicting, objective functions. The solution to an MOP consists in finding, not one, but rather a set of incomparable solutions, which cannot be improved in one objective without deteriorating at least another one. A very popular and effective way to cope with MOPs is by means of multi-objective evolutionary algorithms (MOEAs) [1] because they can find nearly optimal solutions in a single run without imposing any particular assumptions such as continuity or differentiability. Instead, these methods are based on randomized search strategies that mimic Darwin's principle of natural selection. However, and despite their evident advantages, MOEAs require a significant number of objective function evaluations, becoming unaffordable for certain applications that demand an intensive use of computational resources.

The computational cost of an MOEA at each iteration is affected by two factors: 1) the computational complexity of evaluating the objective functions, and 2) the scalability of the input parameters. In the first case, a vast majority of MOPs cannot be expressed in an algebraic form. Thus, their evaluation is conducted by time-consuming simulators. Although some attempts have been made to reduce the execution time of MOEAs by exploiting knowledge acquired during the search [2, 3], the quality of the final solutions often worsens. Examples of the second case include applications that deal with high dimensional spaces, i.e., MOPs having four or more objectives, or having hundreds, even thousands of decision variables. These kind of MOPs, also known as many-objective optimization problems (MaOPs) [4] and large-scale multi-objective optimization problems [5], respectively, considerably increase the running time of MOEAs [6, 7, 8]. Some other applications need a large population size for improving accuracy [9] or covering more regions of the search space [10]. Though most MOEAs run in expected polynomial time regarding their population size, the storage limit also becomes an issue.

Parallel MOEAs (pMOEAs) are an attractive option to address the above mentioned issues, where the basic idea is to sub-divide the operation of the MOEA into several tasks. Each of these tasks is solved simultaneously on a different processing unit or *deme*[1] and, once all of them have been completed, the results are combined to provide a solution to the MOP [11]. Moreover, processing units can be on the same machine, or distributed in a collection of machines interconnected by a network [12]. The wide acceptance of pMOEAs is mainly because they can produce substantial gains in terms of execution time and, in some cases, they can also improve the accuracy of the results with respect to their serial counterparts [13]. Throughtout the years, several pMOEAs have been proposed (see [14] and [15] for a condensed list of approaches), aiming to advance the state-of-the-art or to tackle a specific problem. However, a constant factor has been the use of the four main types of parallel models: master-slave, island, diffusion, and hybrid models [16, 13] (see Figure 1). The master-slave model is one the of the simplest ways to parallelize MOEAs. Its backbone idea is the parallelization of the operations of an MOEA that do not require information about all the individuals. The island model takes inspiration from nature where populations evolve simultaneously in isolation. Hence, several independent MOEAs could be executed in different processors. The diffusion model, also known as fine-grained parallelization, spatially distributes a single population among several processing units, i.e., employing a neighborhood structure. This paradigm is well-suited for massively parallel computers. Finally, hybrid models come from the combination of the three previous models. These four parallel models have promoted the design and implementation of most of the currently available pMOEAs.

To date, several tutorials and surveys on pMOEAs have been published, aiming to provide the basics on the matter and describe the most recent approaches [17, 14, 18, 1, 19, 16, 15, 13]. In 2003, Van Veldhuizen *et al.* [17] published a

---

[1] A Deme is a subgroup of individuals of the population (or even one individual), which explores the search space, using a search engine.

journal paper which not only gives useful guidelines for the design and implementation of pMOEAs, but also makes observations about parallel architectures, benchmarks, performance metrics, and estimations of their computational time. Furthermore, the authors introduced innovative concepts for the migration and replacement schemes, as well as a generic pseudocode for different parallel models, using for this purpose a specialized notation. However, no taxonomy was proposed and they only provided the review of four approaches. An extended version of this work, published in a book chapter [1, p.443] in 2007, included a comprehensive literature review of more than 40 implementations of pMOEAs, dating back to 1992. In 2005, Nebro *et al.* [14] published a book chapter where they proposed one of the first taxonomies of pMOEAs and a wide review of approaches, considering features such as programming language, connection topology, parallel model, and application domain. In spite of providing a review of more than 40 pMOEAs (from 1993 to 2004), no specific discussion on the advantages and drawbacks of the approaches is provided. Later on, Luna *et al* [18] and Luna and Alba [15], in 2006 and 2015, respectively, extended the work of Nebro *et al.* [14] by refining their taxonomy and aggregating the characteristics of up to 80 pMOEAs from 1993 to 2011. Nevertheless, the review of approaches is exclusively focused on reporting their main features (according to the taxonomies) without a discussion of properties and performances. On the other hand, López Jaimes and Coello [16] presented a review of modern parallel platforms and the way in which they can be exploited to implement pMOEAs. Additionally, an explanation of the four parallel models and a review of five approaches from 2000 to 2007 is included. Last but not least, Talbi *et al.* [19] published a general overview of pMOEAs, providing a taxonomy for parallel meta-heuristics and exact methods. The authors discussed parallel and distributed architectures and estimated the computational time for the master-slave model. The literature review provided in this case includes more than 30 pMOEAs published between 1995 and 2007, covering their design and implementation. Finally, in 2019, Talbi [13] extended this work by refining the proposed taxonomy and by analyzing the implementation issues of pMOEAs.

However, a review of recent approaches was not incorporated.

Due to the critical impact of pMOEAs and the continued development of more sophisticated proposals, this paper aims to provide a comprehensive review of approaches. To this aim, we propose a refinement of Talbi's taxonomy [13], considering different characteristics of pMOEAs. Hence, we aim to propose here a unified taxonomy. Based on the taxonomy, we analyze in detail the advantages and drawbacks of several pMOEAs published from 2002 until 2020. Hence, this is first survey that reviews pMOEAs published from 2011 until 2020. We include approaches in the review that introduce new ideas on the design of pMOEAs that push forward the state-of-the-art. Additionally, we briefly describe some pMOEAs that have been applied to solve real-world problems. Finally, some future research trends in the area are outlined.

The remainder of this paper is organized as follows. Section 2 introduces the basic concepts that will be used throughout the rest of the paper as well as the parallel models that have been utilized for the design of pMOEAs. Section 3 is devoted to present our proposed taxonomy as well as a comprehensive review of state-of-the-art pMOEAs. Section 5 describes some real-world applications where pMOEAs have been used. Section 6 proposes some possible future research trends. Finally, our main conclusions are provided in Section 7.

2. Background

This section is devoted to provide some basic definitions to make the paper self-contained. First, we define the general multi-objective optimization problem. Then, we outline the basics of multi-objective evolutionary algorithms (MOEAs). Finally, we describe the general parallelization models of MOEAs, i.e., master-slave, island, diffusion, and hybrid models.

2.1. Multi-Objective Optimization

According to Coello *et al.* [1], the formal definition of an MOP is as follows:

$$
\begin{aligned}
\text{Minimize}\quad & f(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})) && (1)\\
\text{subject to}\quad & g_j(\vec{x}) \geq 0, \qquad j = 1, 2, \ldots, p, && (2)\\
& h_k(\vec{x}) = 0, \qquad k = 1, 2, \ldots, q, && (3)\\
& x_i^L \leq x_i \leq x_i^U,\ i = 1, 2, \ldots, n, && (4)
\end{aligned}
$$

where $\vec{x} = (x_1, x_2, \ldots, x_n)^T \in \mathcal{X}$ is a *decision vector*, $\mathcal{X} \subseteq \mathbb{R}^n$ is the *decision space*, $f(\vec{x}) \in \mathcal{Y}$ is an objective vector of m (≥ 2), normally conflicting, *objective functions* $f_i : \mathcal{X} \to \mathbb{R}, i = 1, \ldots, m$, and $\mathcal{Y} \subseteq \mathbb{R}^m$ is the *objective space*. The constraints (2)-(4) determine the set of *feasible solutions*.

When tackling an MOP, an order relation is necessary to compare solutions. The Pareto dominance relation, which is a binary relation that imposes a strict partial order in $\mathbb{R}^m$, has been commonly employed. Given two vectors $\vec{x}, \vec{y} \in \mathbb{R}^n$, it is said that $\vec{x}$ *Pareto dominates* $\vec{y}$ (denoted by $\vec{x} \prec \vec{y}$), if $f_i(\vec{x}) \leq f_i(\vec{y})$ for all $i = 1, 2, \ldots, m$ and $\exists j \in \{1, 2, \ldots, m\}$ such that $f_j(\vec{x}) < f_j(\vec{y})$. In multi-objective optimization, the common optimality criterion is defined in terms of the Pareto dominance relation as follows: a vector of decision variables $\vec{x}^* \in \mathcal{X}$ is *Pareto optimal* if there does not exist another $\vec{x} \in \mathcal{X}$ such that $\vec{x} \prec \vec{x}^*$. Due to the conflicting nature of the objective functions, the solution to an MOP is the set of all Pareto optimal solutions, known as the *Pareto set*. The image of the Pareto set is the *Pareto front* (PF) whose elements represent the best possible trade-offs among the objectives.

2.2. Multi-Objective Evolutionary Algorithms

In the last 20 years, MOEAs have become a popular choice for tackling complex MOPs [1, 20, 21, 22]. MOEAs are stochastic black-box metaheuristics based on the principles of Darwin's natural evolution that can approximate the Pareto front in one execution due to their population-based nature. The main goals of MOEAs are to produce solutions as close as possible to the Pareto front, having good diversity and covering the whole Pareto front. MOEAs are set-based methods that at each iteration t maintain a population $P_t = \left\{\vec{x}^1, \vec{x}^2, \ldots, \vec{x}^\mu\right\}$

that has to be evolved, using a set of evolutionary operators: mating or parent selection (π), variation (ν), and survival selection (σ). In the following, let A^N represent a subset of size N of the set A and let $(A, \mathcal{M})$ be a multiset of the set A, where $\mathcal{M} : A \rightarrow \mathbb{N}$.

1. $\pi : \mathcal{X}^{\mu} \rightarrow (\mathcal{X}^{\mu}, \mathcal{M})$. This operator chooses from $\mathcal{X}^{\mu}$, μ solutions that will shape the mating pool, i.e., the parent solutions employed to create new ones via the variation operators. Since a solution can be chosen zero or more times, the mating pool is represented as the multiset $(\mathcal{X}^{\mu}, \mathcal{M})$. In the literature, there are several mating selection mechanisms, but the common strategies are random sampling and binary tournament [1].
2. $\nu : (\mathcal{X}^{\mu}, \mathcal{M}) \rightarrow \mathcal{X}^{\lambda}$. Variation operators produce λ new individuals from the mating pool. Variation operators aim to explore and exploit the decision variable space. A wide range of variation operators have been proposed to handle different encodings of the decision vector. Regarding real-numbers encoding, the most common choices are: simulated binary crossover (SBX) and polynomial-based mutation (PBX) [23].
3. Survival selection σ determines the solutions that will shape the next generation. This selection function has two variants: (1) $\sigma_{(\mu,\lambda)} : \mathcal{X}^{\lambda} \rightarrow \mathcal{X}^{\mu}$ in which the best μ ($\mu \leq \lambda$) solutions from $\mathcal{X}^{\lambda}$ replace the parent population, and (2) $\sigma_{(\mu+\lambda)} : (\mathcal{X}^{\mu} \cup \mathcal{X}^{\lambda}) \rightarrow \mathcal{X}^{\mu}$ where the best μ individuals from the union of parents and offspring solutions are chosen. Regarding the latter selection scheme, the simplest case is the steady-state selection, which refers to $\sigma_{\mu+1}$. However, $\sigma_{(\mu+\lambda)}$ is preferred to incorporate elitism which is an important strategy of MOEAs to guide the population towards the true Pareto front of an MOP [1].

Based on its evolutionary operators, an MOEA can be defined by the iterative rule: $P_{t+1} = \sigma\left\{P_t \cup \nu[\pi(P_t)]\right\}$. Additionally, some MOEAs use a secondary population (also known as archive) $\mathcal{A}$ that keeps the non-dominated solutions found so far as a way to introduce elitism. At the end, the final population or the archive is returned as the *approximation set*. The general framework of an MOEA is shown in Algorithm 1. First, the main population $P_{t=0}$ and the archive $\mathcal{A}$ are initialized in lines 1 and 2, respectively. Lines 4 to 11 show the main loop of an MOEA. At each iteration, mating selection is performed to select μ parent solutions from either P_t or $\mathcal{A}$. Variation operators (e.g., crossover and mutation) further explore this set M of parent solutions to create the set $\mathcal{O}$ of λ offspring solutions. Then, the archive $\mathcal{A}$ is updated using the non-dominated offspring solutions in line 7. The next step involves performing the survival selection strategy, using either a (μ, λ) or a $(\mu + \lambda)$ selection rule, to shape the next generation P_{t+1}. Finally, an MOEA returns P_t or $\mathcal{A}$, depending on its specific design. However, both of them contain an approximation to the Pareto optimal front. Table 1 presents a summary of the most representative MOEAs, briefly describing which operators π, ν, and σ they use as well as some additional observations.

ALGORITHM 1: MOEA's general framework

```
1  Generate initial population $P_0$ of size $\mu$;
2  Initialize $\mathcal{A}$ with the non-dominated solutions from $P_0$ ;
3  $t \leftarrow 0$;
4  while stopping criterion is not fulfilled do
5      $M \leftarrow$ Select $\mu$ parents from $P_t$ or $\mathcal{A}$;
6      $\mathcal{O} \leftarrow$Generate a set of $\lambda$ offspring solutions based on $M$, using
         variation operators;
7      Update archive $\mathcal{A}$ using $\mathcal{O}$;
8      $S \leftarrow$ Select survival solutions using a $\sigma_{(\mu,\lambda)}$ or $\sigma_{(\mu+\lambda)}$ strategy;
9      $P_{t+1} \leftarrow S$;
10     $t \leftarrow t + 1$;
11 end
12 return $P_t$ or $\mathcal{A}$
```

The MOEAs presented in Table 1 (and, in general, all the MOEAs designed so far) have a remarkable aspect in common: they were designed to tackle com-

Table 1: Features of the most representative MOEAs

Year	Optimizer/ Reference	Full Name	Parent Selection (π)	Variation (ν)	Survival Selection (σ)	Observations
1994	NSGA [24]	Nondominated Sorting Genetic Algorithm	Stochastic remainder selection. Dummy fitness based on non-dominated fronts and sharing in decision variable space.	Binary operators.	Non-elitist approach: (μ, μ).	Computational complexity of $O(\lvert P\rvert^3 m)$. Performance sensitive to σ_{share} parameter.
1998	SPEA [25]	Strength Pareto Evolutionary Algorithm	Tournaments (including external solutions). Fitness based on Pareto dominance.	Single-point crossover and binary mutation.	Elitist scheme with selection (μ, μ).	External archive pruned to a limit size using a clustering method (average linkage).
1999	PAES [26]	Pareto Archived Evolution Strategy	Not applicable, except for the variant $(\mu + \lambda)$, whith $\mu > 1$ or $\lambda > 1$. In this case the selection is performed via tournaments.	Random mutation (similar to a hill-climber).	Elitist $(1 + 1)$, comparisons are made using Pareto dominance and a degree of crowding, which is estimated from an external archive.	External archive truncated to a fixed size by an adaptive grid, which recursively splits the objective space in $O(2^{lm})$ subdivisions, where l is a user defined parameter.
2000	NSGA-II [20]	Nondominated Sorting Genetic Algorithm II	Tournaments based on a preference relation ($\prec_n$), which considers non-dominated fronts and crowding distance.	Simulated binary crossover (SBX) and Polynomial-based mutation.	Elitist scheme $(\mu + \mu)$, based on $\prec_n$.	Computational complexity of $O(\lvert P\rvert^2 m)$. Constraint handling by modifying Pareto dominance.
2001	SPEA2 [27]	Strength Pareto Evolutionary Algorithm 2	Tournaments. Fitness based on Pareto dominance and density information (kNN method).	Depending on the problem, same as SPEA or NSGA-II.	Elitist scheme $(\mu + \lambda)$, truncation considers distances among individuals.	Boundary points are preserved. Computational complexity of $O(\lvert P\rvert^2(\log \lvert P\rvert + m))$.
2004	IBEA [28, 29]	Indicator-Based Evolutionary Algorithm	Binary tournaments. Fitness is given by an exponential function, which calculates the performance indicator of each distinct pair of individuals.	Depending on the problem, same as SPEA or NSGA-II.	Elitist scheme $(\mu + \mu)$, which iteratively removes the worst individual, updating fitness values.	Framework of binary indicators, compliant with Pareto dominance. Required scaling factor $\kappa > 0$, dependent on the problem and the indicator used.
2005	SMS-EMOA [30]	S-metric Selection Evolutionary Multi-Objective Optimization	Random sampling.	Simulated binary crossover (SBX) and Polynomial-based mutation.	Elitist $(\mu + 1)$. The solution with the worst hypervolume contribution of the last non-dominated front is removed.	Expensive computational complexity of $O(\lvert P\rvert^{m-1})$. Hypervolume is maximized through generations.
2007	MO-CMA-ES [31]	Multi-Objective Covariance Matrix Adaptation Evolution Strategy	In the scheme $\lambda(\geq 1)$ every individual in the population is mutated.	Sampling from a multi-variate normal distribution with the parent as the mean vector and an adaptive covariance matrix.	Elitist scheme $\mu(1 + \lambda)$. Population is ranked according to the non-dominated fronts and the contributing hypervolume (or crowding distance).	Invariant against rotation, scaling and translation of the search space. Parameter adaptation of the mutation distribution per individual. Computationally expensive for problems with many variables or objectives.
2007	MOEA/D [32]	Multi-Objective Evolutionary Algorithm based on Decomposition	Random sampling from the neighborhood T_i of each i-th individual, associated with a scalar optimization problem.	Only one offspring is created. Same as SPEA2, including differential evolution.	Elitist scheme $(\mu + \mu)$. A child i replaces solutions from T_i if its corresponding scalarizing function is equivalent or better.	Framework of scalarizing functions with a complexity of $O(\lvert P\rvert \lvert T\rvert m)$. Required set of weight vectors and neighborhood size ($\lvert T\rvert \ll \lvert P\rvert$).
2011	HypE [33]	Hypervolume Estimation Algorithm for Multi-Objective Optimization	Binary tournaments using the hypervolume contributions of the population.	Depending on the problem, SBX and Polynomial-based mutation.	Elitist scheme $(\mu + \mu)$. Solutions with the worst hypervolume contributions of the last non-dominated front are discarded.	If $m \leq 3$ the exact hypervolume is calculated, otherwise it is estimated using Monte Carlo simulation, where a large number of sampling points is required.
2014	NSGA-III [34, 35]	Nondominated Sorting Genetic Algorithm III	Binary tournaments for constraint MOPs, otherwise random sampling.	SBX and Polynomial-based mutation	Elitist $(\mu + \mu)$, crowding distance is replaced by a niching strategy, which requires a set of well spread reference points.	Computational complexity of $O(\lvert P\rvert^2 m)$. Algorithm designed for solving MOPs up to 15 objectives.

plex MOPs. Unlike mathematical programming techniques proposed by the Multi-Criterion Decision Making community, MOEAs are flexible techniques that can be easily applied on a plethora of nonlinear MOPs without additional requirements and assuming less strict mathematical properties [1]. However, an important drawback of MOEAs is that they normally require a considerably large number of objective function evaluations to obtain good results. Moreover, many of the real-world MOPs are computationally expensive since they depend on time-consuming methods for computing the objective functions and constraints. Hence, the application of sequential MOEAs may be limited. In this context, parallelism appears as an alternative to design MOEAs such that they can solve complex MOPs in a reasonable amount of time [17]. In contrast to sequential MOEAs, parallel MOEAs are attractive for the following reasons (among others) [16]: (1) pMOEAs can improve the search ability of sequential MOEAs, (2) due to this improved search ability, the possibility to get stuck in suboptimal solutions is reduced, (3) pMOEAs allow the use of larger population sizes, (4) they tend to enhance the properties of the Pareto front approximations, and (5) pMOEAs are more suitable to solve large-scale MOPs and MaOPs. In the next section, we present the main parallel models that can be used to define pMOEAs.

2.3. Parallel Models of MOEAs

Parallel paradigms can be utilized to decompose some problem (task and/or data) in order to decrease the execution time. These paradigms may also allow for exploring more of the solution space, potentially finding *better* solutions in the same amount of time as when using a serial implementation. A pMOEA seeks to find as good or better MOP solutions in less time than its serial MOEA counterpart and/or searching more of the solution space in the same amount of execution time (i.e., the goals are to increase efficiency and effectiveness). However, the use of parallel methodologies generally implies the utilization of additional computational resources. pMOEAs have followed four main parallelization models that have been inherited from single-objective metaheuristics

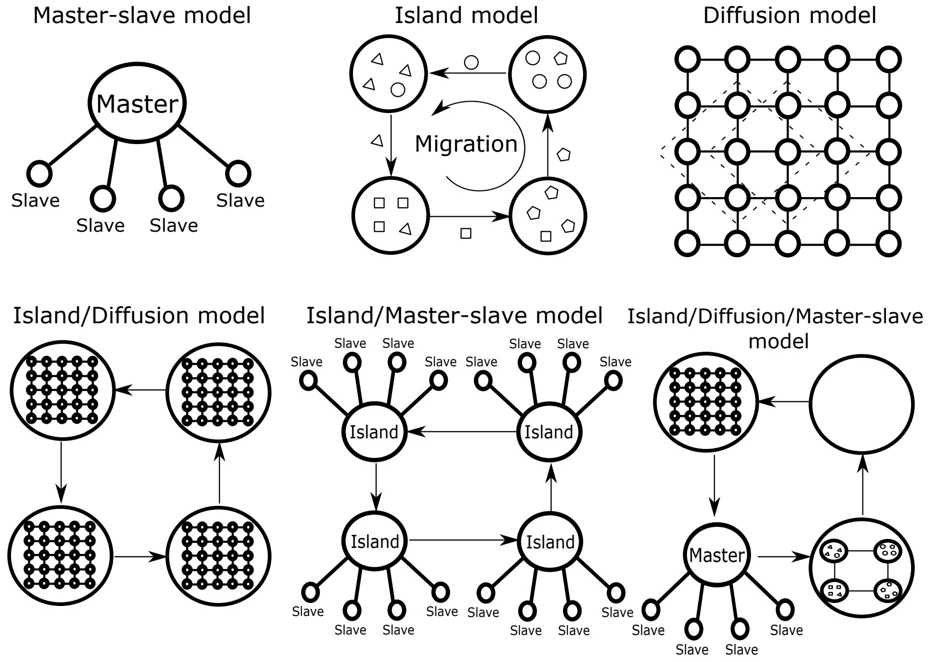

Figure 1: Classical parallel models: master-slave, island model, diffusion, and different hybrid models.

[16]: (1) master-slave or global parallelization, (2) the island model, also known as coarse-grained parallelization, (3) the diffusion model that is also denoted as fine-grained parallelization, and (4) hybrid models which are based on combinations of the three previous models. These parallel models are illustrated in Figure 1. In the following, these four parallel models will be broadly described.

Master-slave. The master-slave model is suitable to parallelize the operations of an MOEA that do not require information about all individuals [15]. Thus, a single population is managed by a *master* and the evaluation of the objective functions and the application of the variation operators[2] are done simultaneously by the *slaves*. The master-slave model does not alter the search behavior of the

---

[2] Normally, the parallelization of these operators is omitted, since they may introduce a high communication overhead when parents are sent to slaves. A better practice is to perform tournament selection among the subset of individuals.

underlying sequential MOEAs, but instead, it makes them faster, especially when computationally expensive objective functions are tackled [16, 15].

The master-slave model, also known as *global parallelization*, can be implemented either on shared or on distributed memory computers. In the former case, the population is stored in a shared memory and each slave processor reads the individuals assigned to it and writes the results back. On the latter, a master processor stores the population and is responsible for sending the individuals to the slave processors and collecting the results. The number of individuals assigned to any slave may be constant, but in some cases it may be necessary to balance the computational load among the available processors. Furthermore, it is possible to disseminate objective function evaluations across the slaves, instead of population members, and even more, it is possible to let some slaves specialize in the evaluation of a specific objective function [36, 37, 38, 39]. It is worth noting that if the evaluations have a very low computational cost, the time spent in the communications could be easily higher than the time required for performing any computation. This is specially true in distributed memory machines where the overhead produced by each message might be considerable.

The master-slave model can be implemented following a synchronous or asynchronous strategy [13]. In a synchronous master-slave pMOEA, the master waits to receive the results of all the population before proceeding to the next generation. Here, the search behavior is conceptually identical to a serial MOEA, with its execution time being the only difference. One disadvantage of this approach in distributed memory environments is that synchronization may cause a waste of resources, since slave processors sit idle while the master is doing its work and this is accentuated when the machines are not homogeneous. On the other hand, in an asynchronous master-slave pMOEA, the master does not stop to wait for any slow slave and thus, the search space exploration is different from a MOEA being executed on a serial processor. In the case of shared memory architectures, the master must lock individuals until they are ready to be incorporated in the population.

Island model. This model is inspired by the natural phenomenon of populations evolving in relative isolation, such as might happen within some oceanic chain of islands [1, 16]. Sometimes this approach has been called *distributed model* as it is usually implemented on distributed memory computers. Furthermore, it has also been called *coarse-grained model* or *cooperative parallelism* [13], since islands contain a large number of individuals. In technical terms, the island model divides the overall population into a number of independent *subpopulations* that live in *islands* (processors) and evolve through the execution of a serial MOEA. Islands are connected in a physical or logical topology, such as a star, line, tree or ring (just to name a few). Occasionally, neighboring islands interact with each other, exchanging some individuals as shown in Figure 1. This operation is known as *migration* and it introduces extra parameters: the number of solutions to migrate, how often migration occurs, and the migration policy that determines which individuals migrate and which are replaced in the receiving subpopulation [1]. These parameters along with the topology play an important role in the performance of an island pMOEA, because they determine how fast (or how slow) a good solution spreads to other subpopulations. Additionally, an island pMOEA can manage the Pareto front approximation either in a centralized way (i.e., by a master island that maintains the global nondominated solutions) or locally (i.e., managed by each sub-MOEA during the computation).

Sometimes, island pMOEAs with a star topology and high migration rates are mistaken with master-slave algorithms. The difference lies in that islands evolve subpopulations during a certain number of generations (called *epochs*), applying all the steps of an MOEA and after this period, they migrate individuals. If the communication is synchronous, islands have to wait until they receive their corresponding individuals from the central island, and in the same way, the central island will be idle while peripheral islands are working. When the central island performs steps in addition to those performed by the other islands (like keeping an external archive of non-dominated solutions), this pMOEA is considered heterogeneous. On the other hand, master-slave pMOEAs keep one population that is distributed among processors at each generation, with the purpose of being reproduced and/or evaluated. The remaining steps of the algorithm are performed by the master. Finally, as can be noticed, communication rates are higher in the master-slave model than in the island model.

Island pMOEAs are very popular because they are a simple extension of serial MOEAs and few changes are needed to implement a migration operator [16]. Furthermore, their behavior differs from their serial counterpart, allowing the formation of niches, which can improve the search. Algorithms based on this model can considerably reduce the execution time and its use is recommended for clusters of computers with very limited communication among them. They are also suitable for problems with large search spaces where a large population size is required. A disadvantage of island pMOEAs is that their diversity may be poor, since some parts of the Pareto front could be missed by its islands.

Diffusion. Unlike island pMOEAs where the population is divided into subpopulations of a considerable size, the diffusion model or *cellular MOEA* typically considers subpopulations composed of one individual [16, 15]. Hence, this has led some people to use the term *fine-grained parallelism* for this sort of approach. A neighborhood structure is imposed on the processors that hold the subpopulations such that each subpopulation can only interact with its nearest neighbors. The neighborhood structure is shown in Figure 1 with a dashed line, covering the involved processors. Variation operators are applied only within these (possibly) overlapping neighborhoods. The neighborhood geometry could be a square, a rectangle, a cube, or any other shape depending upon the number of dimensions associated with the diffusion algorithm's topological design. Each geometry reflects some associated number and arrangement of neighbors within a multi-dimensional grid. As good solutions arise in different areas of the local topology, the aim is that they spread or diffuse slowly throughout the entire population due to the overlapping or dynamically changing neighborhoods that it adopts. According to Luna and Alba [15], this model is well suited for massively parallel computers. However, it can be used sequentially on a regular computer or using graphics processing units (GPUs). In this model, there is no migration per se and communication costs may then be very high within a neighborhood.

Hybrid models. Few researchers have attempted to combine different methods to parallelize MOEAs, producing the so-called hybrid or hierarchical pMOEAs [16, 15]. Some of these hybrid algorithms add an additional degree of complexity to the already complicated scene of parallel approaches, but other hybrids manage to keep the same complexity as one of their components. When two or more methods for parallelizing MOEAs are combined, they form a hierarchy. At the upper level most hybrid pMOEAs implement a coarse-grained parallel model and at the lower level they can adopt a coarse-grained model or a fine-grained strategy. For instance, considering an island model in the upper level, it could be combined with a master-slave, island model or diffusion model in the lower level as depicted in Figure 1.

3. Parallel Multi-Objective Evolutionary Algorithms

This section is devoted to introduce our refined taxonomy. Then, we present a review of pMOEAs which have been proposed from 2002 to 2020, analyzing their main design properties, advantages and drawbacks. The selection of pMOEAs is based on their contributions to push forward the state-of-the-art.

3.1. Our Proposed Taxonomy

In 2019, Talbi [13] presented a taxonomy that classifies pMOEAs in three levels of parallelization: (1) algorithmic-level, (2) iteration-level, and (3) solution-level. The algorithmic-level encompasses the parallel execution of either independent or cooperative MOEAs, i.e., pMOEAs following the master-slave model or the island model, respectively. In this level of parallelization, Talbi indicates that pMOEAs could handle the current Pareto front in a centralized or distributed way, and the objective and decision spaces could be managed following a global or partitioned approach. Unlike the algorithmic-level where multiple

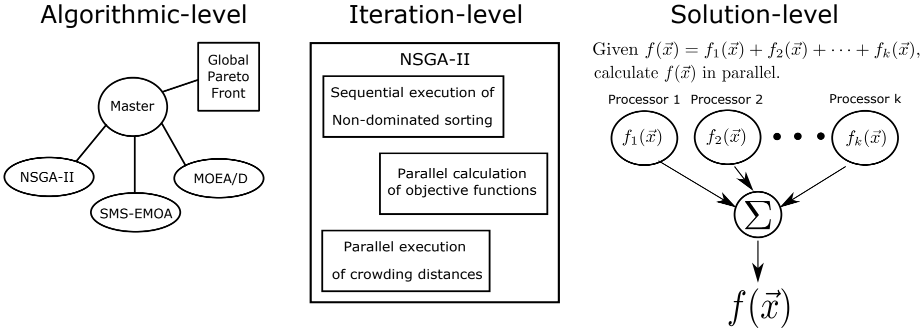
Algorithmic-level
Global Pareto Front
Master
NSGA-II
SMS-EMOA
MOEA/D
Iteration-level
NSGA-II
Sequential execution of Non-dominated sorting
Parallel calculation of objective functions
Parallel execution of crowding distances
Solution-level
Given $f(\vec{x}) = f_1(\vec{x}) + f_2(\vec{x}) + \cdots + f_k(\vec{x})$, calculate $f(\vec{x})$ in parallel.
Processor 1
Processor 2
Processor k
$f_1(\vec{x})$
$f_2(\vec{x})$
$f_k(\vec{x})$
$\sum$
$f(\vec{x})$

Figure 2: Examples of the three levels of parallelization.

MOEAs are executed in parallel, the iteration-level focuses on parallelizing the main loop of a single MOEA. This level gives special attention to handle the population in parallel since a CPU-time consuming component of any MOEA is the evaluation of the objective functions. Hence, the author indicates that the population can be divided into a number of subsets according to different criteria (for instance, by objective functions). It is worth noting that these two levels of parallelization are not dependent on the target MOP which implies that only problem-independent components are parallelized. In contrast, the solution-level focuses on the parallelization of a single solution that involves the calculation of the objective functions and/or the constraints. Two types of strategies are considered: decomposition of data where the calculation of functions is done in parallel for each partition of the data; and, function decomposition, where a single function can be decomposed into many sub-functions. Figure 2 shows examples of the three levels of parallelization. The algorithmic-level depicts the parallel execution of NSGA-II, SMS-EMOA, and MOEA/D as slave nodes and a master node having the global Pareto front. Regarding the iteration-level, it is shown the parallel execution of some of its mechanisms, namely, the evaluation of objective functions and the calculation of the crowding distances. Finally, the iteration-level example illustrates a function decomposition calculated in parallel.

In Figure 3, we present our taxonomy which refines the one proposed by

Talbi [13]. The refinements were done based on a comprehensive review of pMOEAs. Our taxonomy keeps the three levels of parallelization mentioned before. However, the main difference is the addition of new properties (shown in boxes in the figure) for the three levels: we added the subcategory *parallelism of mechanisms* to the iteration-level, and we considered the *Predator-Prey* parallel model [40] that has not been broadly studied and we also added the *Consumer-Producer* parallel model [41] which has not been used in pMOEAs but we think it has a great potential. We should emphasize that this taxonomy does not aim to categorize a whole pMOEA, but instead, it focuses on the different levels of parallelization. It is possible that a pMOEA uses different parallel levels which will produce a multi-level scheme (also known as hybrid model). The algorithmic-level and iteration-level represent the main categories since they are problem-independent classes. We propose that the iteration-level is divided into two subclasses: *parallelism of population* in which the population is split and distributed among different processors and *parallelism of mechanisms* where MOEAs' mechanisms such as the selection schemes or the update of the external archives are to be parallelized. The property *parallel model* is the backbone of both the algorithmic-level and the iteration-level. This property encompasses the well-known master-slave, island model, and diffusion model but we also include the predator-prey and consumer-producer models. The parallel model is characterized by six additional properties:

1. *Pareto front*: it determines if the Pareto front approximation is managed in a centralized or distributed way during the evolutionary process.
2. *Decision space*: a pMOEA can explore the decision space in a global or in a partitioned way, where the latter has been especially employed when dealing with large-scale MOPs.
3. *Objective space*: as in the previous property, the parallel components of a pMOEA can explore the objective space in an unrestricted way or they can be focused on specific regions of the objective space.
4. *Nodes*: this category indicates if the processors execute homogeneous or

heterogeneous algorithmic components, for instance, MOEAs with the same mechanisms and parameters. It is not related to the hardware characteristics.

5. *Distribution*: it determines if the assigned processors of a pMOEA are static throughout the evolutionary process or if they dynamically change, depending, for instance, on load-balancing conditions.
6. *Communication*: it dictates how the different processors that execute the pMOEA communicate with each other. Typically, synchronous and asynchronous communication schemes have been implemented. However, a recent study [42] proposes to have semi-asynchronous communication or even an adaptive communication scheme switching between synchronous and asynchronous strategies, depending on the execution conditions.

These six categories characterize in a better way the parallel model of both the algorithmic-level and the iteration-level. However, the solution-level, which was directly taken from Talbi's taxonomy, is only related to the communication property. This is because this parallel level is just in charge of parallelizing the calculation of a single solution which requires a communication scheme to compute the whole solution.

In the following sections, we provide a comprehensive review of state-of-the-art pMOEAs that have been published from 2002 to 2020. The proposals are presented according to the parallelization levels that they use. In this light, Tables 2, 4, and 6 show the algorithmic-, iteration-, and multi-level pMOEAs, respectively. All these tables summarize the main characteristics, according to the taxonomy, of the approaches. On the other hand, Tables 3, 5, and 7 include information about their implementation, i.e., parallel architecture, communication library, programming language, and operating system.

3.2. Algorithmic-level Parallelization

In this section, we describe the algorithmic-level pMOEAs that have been published so far. Table 2 summarizes the proposals according to the properties

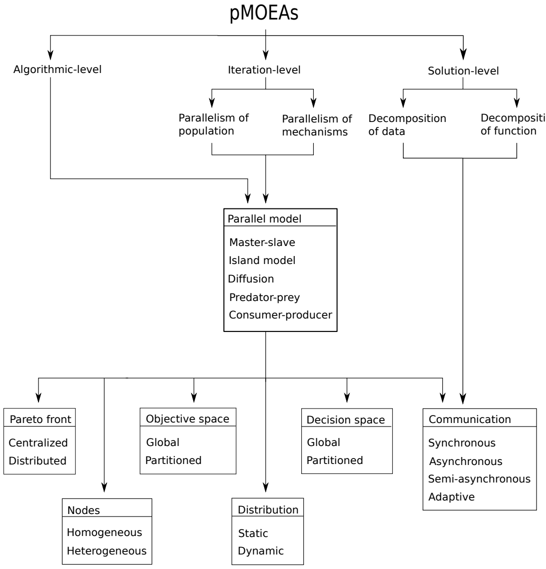

Figure 3: Proposed taxonomy of pMOEAs.

shown in our taxonomy in Figure 3 and Table 3 shows their implementation features.

In 2002, Kamiura *et al.* [43] introduced the Multi-Objective Genetic Algorithm with Distributed Environment Scheme (MOGADES) which is an island pMOEA with static homogeneous nodes that borrows some ideas from SPEA2 and NSGA-II. Here, the fitness is given by a weighted sum. Thus, to derive Pareto optimal solutions, each deme has a different search direction (weight vector) within the global decision space. Furthermore, two archives are maintained in each island and both participate during the offspring generation. One preserves solutions with the best fitness values and the other keeps the non-dominated individuals. When the latter exceeds its allowable size, fitness sharing is performed without requiring any extra parameter. After synchronous migration, weight vectors are updated taking into account the distance between neighboring islands (two islands are neighbors only if they have contiguous search directions). MOGADES was compared with SPEA2 and NSGA-II on two test problems, using bit flip mutation and two point crossover. Good re-

Table 2: Algorithmic-level proposals: main characteristics. The following terms are employed: master-slave model (MS), island model (IM), synchronous (Sync), asynchronous (Async), global (G), partitioned (P), centralized (C), distributed (D), homogeneous (Ho), heterogeneous (He), static (St) and dynamic (Dyn).

Name	**Baseline Algorithm**	**Parallel model**	**Communication**	**Decision Space**	**Objective Space**	**Pareto Front**	**Nodes**	**Distribution**	**Year**	**Reference**
MOGADES	NSGA-II & SPEA2	IM	Sync	G	P	D	Ho	St	2002	[43]
DCMOGA	MOGA & SOGA	IM	Sync	G	P	C	He	St	2002	[44]
AsyncMOGA	MOGA	IM	Async	G	G	D	Ho	St	2002	[45]
SIM	NSGA & SOGA	IM	Sync	G	P	C	He	St	2003	[46]
pPAES	PAES & MOGA	MS	Sync	P	G	C	He	Dyn	2004	[47]
Cone-separated NSGA-II	NSGA-II	IM	Sync	G	P	D	Ho	St	2004	[48]
kosMOEA	NSGA-II	IM	Sync	G	P	C	Ho	St	2005	[49]
kssMOEA	NSGA-II	IM	Sync	P	G	C	Ho	St	2005	[49]
MRMOGA	MOGA	IM	Sync	P	G	D	He	St	2005	[50]
MFED	SPEA	IM	Sync	G	G	D	Ho	St	2006	[51]
Parallel Hyper-heuristic	SPEA, SPEA2 & NSGA-II	IM	Async	G	G	C	He	St	2008	[52]
PNSGA	NSGA-II	IM	Sync	G	G	C	He	St	2009	[53]
SMPGA	PNSGA	IM	Sync	G	G	C	He	St	2010	[54]
AEMOA	NSGA-II & MO-CMA-ES	MS	Async	G	G	C	He	St	2011	[55]
EMaOEA	GrEA, SPEA2+SDE & NSGA-III	MS	Sync	G	G	D	He	St	2017	[56]
PasMoQAP	Memetic MOEA	IM	Async	G	G	D	Ho	St	2017	[57]
MOEA/D_{sp}^{pe}	MOEA/D	MS	Sync	G	G	C	Ho	St	2017	[58]
PEA	Own approach	MS	Sync	P	G	C	Ho	St	2018	[59]
PCPMOEA	Own approach	MS	Async	G	P	C	Ho	St	2019	[60]
S-PAMICRO	SMS-EMOA	IM	Async	G	G	D	Ho	St	2020	[61]
IMIA	Indicator-based MOEAs	IM	Sync	G	G	D	He	St	2021	[62]

Table 3: Algorithmic-level proposals: implementation characteristics.

Name	**Parallel architecture**	**Communication library**	**Programming language**	**Operating system**	**Reference**
MOGADES	Not defined	Not defined	Not defined	Not defined	[43]
DCMOGA	Not defined	Not defined	Not defined	Not defined	[44]
asyncMOGA	Multi-computer (PC-Cluster)	Not defined	Not defined	Not defined	[45]
SIM	Parallel virtual machines	MPI	Not defined	Not defined	[46]
pPAES	Multi-core (4 cores)	pThreads and MPI	C	Linux	[47]
Cone-separated NSGA-II	Multi-computer (PC-Cluster)	MPI	C	Linux	[48]
kosMOEA	Multi-core (1 to 6 cores)	Not defined	Not defined	Not defined	[49]
kssMOEA	Multi-core (1 to 6 cores)	Not defined	Not defined	Not defined	[49]
MRMOGA	Multi-computer (16-node PC-Cluster)	MPI	C	Linux Red Hat	[50]
MFED	Multi-computer (5-node PC-Cluster)	MPI	C++	Windows XP	[51]
Parallel Hyper-heuristic	Not defined	Not defined	Not defined	Not defined	[52]
PNSGA	Not defined	Not defined	Not defined	Not defined	[53]
SMPGA	Not defined	Not defined	Not defined	Not defined	[54]
AEMOA	Multi-computer (40- and 100-node PC-Cluster)	Not defined	Not defined	Not defined	[55]
EMaOEA	Not defined	Not defined	Not defined	Not defined	[56]
PasMoQAP	Multi-computer (32-node Grid)	ParadisEO framework	C++	Not defined	[57]
MOEA/D$_{sp}^{pe}$	Multi-computer (2-node PC-Cluster)	MapReduce and Spark	Java	Not defined	[58]
PEA	Not defined	Not defined	Not defined	Not defined	[59]
PCPMOEA	Multi-core	Java threads	Java	Not defined	[60]
S-PAMICRO	Multi-computer (10-node PC-Cluster)	MPI	C	Linux	[61]
IMIA	Multi-core (5 cores)	OpenMP	C	RedHat	[62]

sults were derived with widespread solutions. One drawback of this approach is that the assignment of weight vectors is not generalized for more than two objectives. Also, additional processing is required for normalizing the objective functions.

The Distributed Cooperation model of Multi-Objective Genetic Algorithm (DCMOGA) [44] is an example of the island model with a star topology. It has $m + 1$ nodes where the central node executes a MOEA (e.g., MOGA, SPEA2 or NSGA-II) and each of the remaining m nodes optimizes a designated objective function by means of any single-objective genetic algorithm (SOGA). This approach is considered heterogeneous with global decision space and partitioned objective space, since one deme behaves different from the others and m demes specialize their search in disjoint regions. The central nodes maintains the global nondominated solutions while the other nodes have archives of best solutions and the stopping criterion is given by a predefined number of function evaluations. At each epoch, extreme solutions[3] are synchronously interchanged between the central and the peripheral nodes. If the migrated solution is better in the i^{th} objective either in the central node or in the i^{th} peripheral node, the local population is decreased and more solutions are migrated. When there is no increase in quality in any of the peripheral nodes from one epoch to the next one, all of them replace the SOGA by an MOEA. DCMOGA was compared with respect to basic MOEAs on different test problems, using binary encoding, two-point crossover and bit flip mutation. Its authors concluded that DCMOGA produced better spread Pareto fronts. One shortcoming of this approach is that the workload is not well distributed after the migration process, because subpopulations may have different sizes. Also, processors may become idle when the computation time of the algorithms is not equivalent.

Horii et al. [45] presented an island pMOEA, where asynchronous migration is launched in a deme until solutions have converged. Furthermore, diversity is maintained introducing migrants from the subpopulation which has the most

---

[3] Individuals that optimize an specific objective function.

different individuals. The rationale behind this scheme relies on the assumption that dissimilar genotypes of good quality promote higher schemata during recombination. Therefore, two measures are introduced: one for convergence, which is based on the covariance matrix of individuals' objective functions and another for estimating discrepancy, which uses Euclidean distances of average genotypes between subpopulations. Each processor executes an independent MOGA [63] with global decision and objective spaces, employing real-valued encoding, BLX-α and tournament selection. The proposal is homogeneous and the Pareto front approximation is distributed all over the nodes. Authors compared the proposed approach with respect to its synchronous version in four bi-objective problems, concluding that the asynchronous version was better in a multi-frontal problem. Moreover, the algorithm does not require migration parameters, but instead, the criterion for convergence needs a threshold which must be established *a priori*. This threshold is problem-dependent, and it is also sensitive to the scaling of the objective functions.

In the Specialized Island Model (SIM) [46] every deme is responsible for optimizing a subset of the original objective functions with global decision space. Hence, it is possible to have islands optimizing both multiple objectives or a single objective function. All islands employ two selection techniques: roulette and tournament. Those islands specialized in single-objective optimization apply a sharing technique with elitism, while the remaining use the non-dominated sorting approach of NSGA. Synchronous Migration is conducted by binary tournaments, and the replacement policy is random. The island optimizing all the objective functions maintains the global non-dominated individuals. The authors tested seven scenarios of SIM, varying the topology, specialization and number of demes on an three-objective artificial problem. Their experimental results showed that the connected models always outperformed their isolated counterparts, ensuring a large number of non-dominated solutions. Some open issues were the optimum values of the migration parameters, the radius for sharing, and that some processors could be idle due to to the heterogeonous nature of SIM.

Coello and Reyes [47] present the parallelization of a coevolutionary algorithm that adopts the master-slave model, uses the Pareto ranking scheme of MOGA and the adaptive grid of PAES. In this approach, the search space is divided into regions (as many as the decision variables of the problem) and such regions are assigned to demes for independent exploration. At each generation, an analysis of each region is performed in order to determine if some intervals of the variables are discardable or divisible, thus deleting or creating more demes, respectively. In this way, demes *cooperate* for obtaining the Pareto front and *compete* for individuals, in such a way that the size of each subpopulation is adjusted based on their contribution to the non-dominated set. Two implementations were considered, one using shared memory (threads) and another one using distributed memory (MPI). They were compared with respect to the serial version of the same MOEA and with respect to PAES on two bi-objective problems. The results indicated that both parallel implementations produced important gains in execution time while maintaining the quality of the results with respect to the serial version. One disadvantage of this approach is that is not scalable neither in variable space nor in objective space.

The cone-separated NSGA-II [48], inspired by an island model, divides the objective space into several equal regions with global decision space, using hyperplanes passing through a reference point. Each region is assigned to one processor for exploration and the borders of the regions are treated as constraints. Thus, solutions outside the designated region are dominated by solutions within it. Individuals that violate constraints are candidates to be migrated into demes that are valid. Moreover, at each epoch, subpopulations synchronize in order to normalize objective functions, partition the search space and share extreme solutions to neighborhood demes. On problems with two objectives, this pMOEA showed to be more efficient than the standard island model with and without migration. However, on problems with three objectives, the distribution of individuals was poor, forming accumulations at the borders between regions. Two main shortcomings of this pMOEA are that processors may have idle time when Pareto fronts are not evenly distributed. Also, partitioning into cones becomes complicated as the number of objectives increases.

Streichert et al. [49] proposed an island model for parallelizing NSGA-II [20] with a partitioning technique based on K-Means clustering applied in both, variable and objective spaces, giving rise to the kssMOEA and kosMOEA, respectively, both with and without zone constraints.[4] At every epoch, solutions are gathered, partitioned and redistributed back to processors. All demes use real-valued representation, one-point crossover and self-adaptive mutation. The four variants were compared with the cone-separated NSGA-II [48] and the standard island NSGA-II with and without migration. Experimental evidence did not support that the two variants with zone constraints were able to perform well on the test problems. On the other hand, the approaches without zone constraints performed as well or better as the island model of NSGA-II with migration. Moreover, these two approaches can solve problems of high dimensionality or problems which have contiguous search spaces. One disadvantage of this approach is that the mutation probability is too high (1.0), which indicates that the exploration of the search space is not guided by selection, but by a random process.

Thre Multiple Resolution Multi-Objective Genetic Algorithm (MRMOGA) [50] is an island model pMOEA with heterogeneous nodes. Since Pareto optimal solutions are found in fewer iterations using low resolution representations than using higher resolution ones, MRMOGA divides the decision space into hierarchical levels with well-defined overlaps. Hence, each island has its own encoding with a different resolution. The low-resolution islands have the purpose of approaching quickly towards the Pareto front. Afterwards, high-quality individuals stored in local archives of these islands are synchronously migrated into high-resolution islands, replacing the worst solutions in terms of Pareto dominance, in order to exploit the regions nearby these solutions. The migration process is performed only if the ratio of replacements in the last k iterations in the external archive is less than a user-supplied threshold. MRMOGA was

---

[4] Demes are limited to their specific region based on a constrained dominance principle.

compared with respect to an island model-based NSGA-II with a star topology on the bi-objective Zitzler-Deb-Thiele (ZDT) [64] and Osyczka [1] test problems. The authors indicated that MRMOGA has more merit in problems with large-scale search spaces since the division of the decision space allows it to find nearly optimal solutions in regions that otherwise would be difficult to find. A drawback of this proposal is the criterion to execute the migration process which is only based on the counting of replacements and it requires a parameter that seems to be problem-dependent.

The Multi-Front Equitable Distribution (MFED) [51] parallelizes SPEA [25] on a distributed memory, using the island model with a star topology. Each deme is assigned to one processor and evolves independently a population with global decision and objective spaces and it has an external archive. At every epoch, the k first fronts of each deme are gathered by the central deme. Then, the first k global fronts are equally redistributed among demes adopting for this purpose a clustering technique. Each deme replaces its population and external archive. At the end, the non-dominated solutions are extracted from the archives. This approach ensures that non-dominated solutions are never lost and each deme will receive a diversified approximate Pareto front created from all the clusters. The disadvantage of this approach is that two new parameters are introduced: the number of clusters and the number k of fronts to be distributed. For the latter, the authors conclude that a small number is better in order to preserve convergence.

León et al. [52] presented a new island scheme based on the cooperation of a set of MOEAs and a hyper-heuristic, that grants more computational resources to those algorithms that show a more promising behavior. A coordinator node is in charge of maintaining the global solution and selecting the configurations[5] that are executed on the demes. The global solution set is obtained by merging the local results achieved by each of the demes and its size is limited using the

---

[5] A configuration consists of an MOEA plus the variation operators and the set of user-supplied parameters which define them (population size, mutation and crossover rates, etc.).

crowding distance operator [20]. Besides the global stopping criterion, a local stopping criterion is defined for the execution of the MOEAs on the demes. When the local stopping criterion is reached, the configuration is scored using a performance indicator. Then, the coordinator applies the hyper-heuristic, selecting the configuration that will continue executing on the idle deme. If the configuration has changed, the subpopulation is replaced by a random subset of the current global solution. The scheme was tested on the ZDT and Walking-Fish-Group (WFG) [65] test problems for instances with two objectives, using SPEA, SPEA2, NSGA-II, IBEA and four variation operators. In total, sixteen configurations were considered for four islands using an all-to-all topology, asynchronous communication and elitist migration. The proposal was among the best pMOEAs when compared with an algorithm that randomly changes the configuration on demes and the homogeneous islands of each of the sixteen configurations. One disadvantage of this approach is that is not scalable for problems with more than three objectives.

In 2009, Wang and Ju [53] proposed an island model version of NSGA-II denoted as PNSGA. PNSGA employs only two self-evolving islands: an elite population island (EP) that stores the global non-dominated individuals and the search population island (SP) whose purpose is to explore the search space. PNSGA implements a synchronous migration scheme where SP firstly sends to EP its current non-dominated individuals after crossover and mutation operations. Then, EP divides its population (which includes the recenlty immigrant solutions) into non-dominated individuals and dominated ones where the latter are migrated to SP. In consequence, the best individuals are further optimized in EP and the dominated individuals are sent to SP in order to increase the diversity which could improve the exploration of the decision space. PNSGA was mainly tested on the ZDT benchmark and it was compared with respect to NSGA-II and MOCLPSO [66]. Experimental results based on the Generational Distance (GD) indicator [67] showed that PNSGA outperforms NSGA-II and MOCLPSO. PSNSGA has three important drawbacks: the high overhead related to the migration process which is executed at every generation, the proposal inherits from NSGA-II its bad performance on MaOPs [4], and the diversity of solutions in EP (which is the final Pareto front approximation) could be very poor.

The Selective Migration Parallel Genetic Algorithm (SMPGA) [54] improves the migration strategy of PNSGA, aiming to improve the diversity of the final Pareto front approximation. Similarly to PNSGA, SMPGA is an island model pMOEA using the EP and SP islands. However, each island adopts different crossover strength according to their roles. Unlike PNSGA, in SMPGA not all the non-dominated individuals in SP are sent to EP but just the ones that meet a migration qualification. If a candidate solution in SP is mutually non-dominated with respect to all the solutions in EP, then, a grid-based strategy is performed to determine if the candidate solution would improve the diversity of EP. If so, the solution in EP in the most crowded grid-location is replaced. SMPGA outperformed both NSGA-II and PNSGA regarding GD and a diversity quality indicator proposed in [20]. Although SMPGA generates Pareto front approximations with better diversity than PNSGA, it preserves the other two drawbacks of the latter, i.e., the high overhead in the migration process and its bad performance on MaOPs.

Yagoubi et al. [55] suggest an asynchronous master-slave algorithm (AE-MOA) for optimizing the combustion of a diesel engine. In this approach, the authors independently parallelize MO-CMA-ES [31] and NSGA-II, in such a way that as soon as a slave becomes available, the master sends an offspring to be evaluated and, then, it is inserted back in the population on a first-come first-served basis. This scheme is recommended for costly problems, where the time required to evaluate individuals may be different due to heterogeneous hardware or numerical simulations. Nonetheless, a region of the Pareto front may be missed when this region requires a higher computational cost [68]. In such context, the authors modified the mating selection scheme, introducing a predefined probability P_s for choosing between the standard selection of the algorithm at hand, and a tournament selection solely based on the duration of the individuals' evaluation. An open issue is how to define an appropriate value of P_s. The pMOEAs obtained better results, in terms of both convergence and runtime, than their generational and synchronized versions on the ZDT and IHR test functions.

Currently, there is wide range of many-objective evolutionary algorithms (MaOEAs) [4] exhibiting specific convergence and diversity properties. Zhou *et al.* [56] proposed an ensemble of MaOEAs (denoted as EMaOEA) following the master-slave strategy. Each slave node executes an independent MaOEA. At each iteration, the offspring populations created by all the MaOEA are synchronously sent to the master node where they are merged and a copy of the whole set of offspring solutions is sent back to the slaves. Then, each slave executes the selection mechanism of the associated MaOEA to shape the population for the next iteration. In the end, the populations of all slaves are collected to form the final Pareto front approximation. EMaOEA was implemented using three MaOEAs: GrEA [69], SPEA2+SDE [70] and NSGA-III [34]. Its performance was compared with the three baseline MaOEAs using the Deb-Thiele-Laumanns-Zitzler (DTLZ) [71] and WFG benchmark problems with 4, 5, 6, 8, and 10 objective functions. SPEA2+SDE was the best algorithm for DTLZ problems regarding the Inverted Generational Distance (IGD) indicator [72] while EMaOEA outperforms all the algorithms on the WFG problems using the HV indicator. Since EMaOEA is a heterogeneous approach due to the use of multiple MaOEAs, it produces idle times in slaves since the execution times of them are not homogeneous. Furthermore, the communication with the master node at each iteration implies a high overhead.

The multi-objective quadratic assignment problem is a time-consuming combinatorial problem that was tackled by Sanhueza *et al.* using a parallel asynchronous memetic algorithm, denoted as PasMoQAP [57]. This proposal uses that island model to independently evolve several subpopulations using a memetic algorithm. The connection topology is represented by a complete graph where migration is done asynchronously and the solutions are selected using a tournament selection and a Pareto-based elitist replacement. Each island has an external archive to store the local non-dominated solutions. A local search procedure is launched for a given time to explore the nearby regions of the solutions in each archive. PasMoQAP was compared with an island-based version of NSGA-II using 22 problems with 2, 3 and 4 objective which were produced using an instance generator. Both pMOEAs were configured to use 5, 8, 11, 16, and 21 islands. PasMoQAP consistently outperformed, based on the hypervolume indicator (HV) [73], the island-based version of NSGA-II when using 11 islands. The experimental results showed that increasing the number of islands does not necessarily imply a higher quality of the results. In fact, this may degrade the performance of PasMoQAP.

MOEA/D [21] has a great potential to be parallelized because of the division of an MOP into many single-objective optimization problems (SOPs). In 2017, Ying *et al.* proposed to parallelize MOEA/D using the Spark technology [74]. The proposal, denoted as MOEA/D_{sp}^{pe}, generates a population in a master node that is completely copied to the N slaves, where N corresponds to the number of weight vectors used by MOEA/D. These copies are evolved during a given number of generations and, then, the best solution for each weight vector is identified. These best solutions are sent to the master node so that there are N solutions for each weight vector. From all these solutions, the best N ones are determined using the standard MOEA/D selection to shape the global population for the next batch of generations. According to the IGD indicator, MOEA/D_{sp}^{pe} performs similarly to MOEA/D using the three-objective DTLZ1-DTLZ4 problems. When dealing with time-consuming objective functions, MOEA/D_{sp}^{pe} would be a very bad option since it consumes a lot of function evaluations. Furthermore, as the master node is processing all the solutions coming from the slaves, these ones will be idle due to the use of a synchronous strategy.

The existing selection mechanisms of MOEAs need to collect and compare all the candidate solutions to balance both convergence and diversity. This implies a series of dependent subprocesses. To overcome this issue, Chen *et al.* introduced a new master-slave parallel evolutionary algorithm (PEA) [59]. The slaves are in charge of achieving convergence while diversity is emphasized in the external archive of the master node that stores the global non-dominated solutions.

PEA classifies the decision variables into two categories: convergence-related and diversity-related variables. Slaves evolve solutions using only convergence-related variables. At the end of each generation, slaves check if the variance of the fitnesses of all their solutions is less than a given threshold. If so, the solution having the best fitness value in each slave is sent to the master node, and a new population is generated, using the convergence- and diversity-related variables, on the basis of the global external archive. In the external archive, the dominated individuals are first removed and if the number of non-dominated individuals exceeds the allowable size, a steady-state selection based on dissimilarity values is performed, retaining the extreme solutions. PEA was compared with respect to several MOEAs (from which the most relevant are NSGA-II and MOEA/D) on the MaF test instances [75] with 3, 5, 8, and 10 objective functions. The experimental results, based on IGD, showed that PEA outperformed the selected MOEAs, producing well-diversified approximation sets. Additionally, PEA showed significant speed ups using 2, 4, 6, and 8 cores.

Kantour *et al.* proposed the Parallel Criterion-based Partitioning MOEA (PCPMOEA) [60] whose main idea is to launch multiple asynchronous MOEAs with different populations. Each slave processor independently executes an MOEA that aims to minimize the distance between the current Pareto front and the ideal point. Additionally, each MOEA targets a specific region of the objective space by using a two-stage selection mechanism. First, a Pareto-based selection is performed to keep the multi-objective character and, then, a criterion-based selection gives preponderance to a given objective function. On the other hand, the master entity periodically adjusts and redirects the search process in real time of all the slaves. In order to do so, the master performs a global selection among the elite individuals generated by the slaves, and, then, it partitions the individuals according to the objective functions in order to update the local archives of the slaves which are focused on a given objective function as well. In this way, the master process helps the slaves to keep them enclosed and dedicated to their allocated objective space. PCPMOEA was tested on the multi-objective knapsack problem (MOKP) with 250, 500, and 750 items and 2 and 3 objective functions. Its performance was compared with respect to that of NSGA-II, SPEA2, MOEA/D, and NSGA-III, using multiple quality indicators, emphasizing the use of HV and IGD. PCPMOEA showed competitive HV and IGD values but it couldn't outperform NSGA-III. A reason for this behavior is the asynchronous updating of the archives in the slaves, which could produce outdated results. Additionally, PCPMOEA loses diversity since each slave is mostly focused on a given objective function due to the use of criterion-based selection.

As the dimensionality of the objective space increases, the computation of the individual contributions to the hypervolume indicator becomes very expensive [76]. Hence, SMS-EMOA becomes unaffordable for MaOPs. Hernández and Coello [8] observed that regardless of the number of objectives, execution time of SMS-EMOA was almost negligible when using small populations (less than 12 invididuals). This observation promoted the design of the Parallel Micro Optimizer based on the $\mathcal{S}$ metric (S-PAMICRO) [61] which employs the island model to split the population into many micro-populations (of at most 12 individuals) that are independetly evolved by a serial SMS-EMOA with an external archive and global decision and objective spaces. Each island performs asyncronous uniform-random migration, following an unidirectional ring topology. The individuals to be migrated are randomly chosen from the archive and a copy of them are sent to the destination island where an elitist-ranking replacement is executed. If the number of solutions in each archive exceeds its allowable size, an image-analysis pruning technique [77] is executed to reduce the archive's size. The execution time of S-PAMICRO is significantly less than that of SMS-EMOA. For the DTLZ and WFG problems with 2, 3, 4, 6, 8, and 10 objectives, S-PAMICRO generates Pareto front approximations with high-quality HV values which are similar to those produced by the standard island version of SMS-EMOA.

Motivated by the critical issue of performance dependence of several MOEAs, Falcón-Cardona *et al.* [62] proposed the island-based multi-indicator algorithm (IMIA). IMIA employs an island model to simultaneously execute five steady-state indicator-based MOEAs (IB-MOEAs) based on the quality indicators HV, R2 [78], IGD$^+$ [79], ϵ^+ [80], and Δ_p [81]. The core idea of IMIA is to compensate for the weaknesses of a given IB-MOEA with the strengths of the others. Here, a weakness is referred to the inability of an IB-MOEA to produce well-diversified solutions on MOPs with irregular Pareto front shapes. IMIA uses an all-to-all connection topology, synchronously migrating solutions at a user-supplied number of function evaluations. Furthermore, every IB-MOEA has a Pareto front approximation in an external archive and a centralized Pareto front is maintained, gathering solutions from all the IB-MOEAs and using a Riesz s-energy-based [82] selection criterion. The experimental results showed that due to the cooperation of the multiple IB-MOEAs, IMIA presents a Pareto-front-shape invariant behavior which makes it suitable to solve MOPs with regular and irregular Pareto front geometries. Moreover, the study revelead patterns on the discovery of solutions. In other words, depending on the Pareto front shape, one IB-MOEA may consistenly produce more (and better) solutions than another one.

3.2.1. Discussion

The design of algorithmic-level approaches has significantly changed along the years. Currently, due to the advances in the design of MOEAs and the new technological tools, we have very effective algorithmic-level pMOEAs. From the approaches in Table 2, EMaOEA, S-PAMICRO, and IMIA present important performance results. A common design strategy of these three approaches is the execution of multiple MaOEA that have different search properties, resulting in the generation of Pareto front approximations with good convergence and diversity. Moreover, these approaches are, overall, the best ones when tackling many-objective optimization problems. Due to the employment of micro-populations, S-PAMICRO allows the use of multiple instances of SMS-EMOA to tackle MaOPs which has been commonly prohibitive when using a panmictic sequential SMS-EMOA. On the other hand, it is worth noting that IMIA is the only pMOEA (taking into account the three different levels of parallelization) that presents an invariance property to the Pareto front shapes. In other words, IMIA can produce Pareto front approximations with high diversity regardless of the geometrical shape of the Pareto front. IMIA has this invariance property due to the cooperation of multiple IB-MOEAs (with different search skills) on an island model, using micro-populations as S-PAMICRO. This aspect is remarkable since in recent years we have faced an overspecialization problem when designing MOEAs [83]. Other approaches have followed the direction of balancing convergence and diversity, for example, PEA, PNSGA and SMPGA. The difference between these approaches is that PEA divides the decision space by analyzing which decision variables impact the convergence property and which ones are reflected on the diversity in objective space. PNSGA and SMPGA implement two archives, one oriented to convergence and the other one to diversity, in a similar way to the Two_Arch2 algorithm [84]. If the aim is to tackle large-scale MOPs, MRMOGA is the only algorithmic-level pMOEA that has been proposed. MRMOGA defines different levels of precision of the decision variables, where low-resolution variables look at rapidly approaching the population towards the Pareto front while high-resolution variables aim to exploit the previously found solutions, refining the Pareto front approximation. Finally, AEMOEA adopts a first-come first-served scheme to assign work load to the nodes. This is specially useful when dealing with the calculation of heterogeneous objective functions, alleviating idle times in the processors.

Regarding the theoretical computational complexity of the approaches, it is difficult to analyze it due to the heterogenous design strategies, communication schemes, and paralel architectural aspects. Overall, IMIA is the only proposal where the theoretical runtime complexity is described. IMIA is governed by the computational cost of executing SMS-EMOA with a micro-population. Since S-PAMICRO also uses multiple instances of SMS-EMOA, we could conclude that S-PAMICRO and IMIA have similar runtime complexities. However, the former was designed to be executed in a massively parallel way, i.e., under a cluster of computers while IMIA uses a multi-core architecture. For the remaining algorithms, all of the them present distinct execution times (and speed ups).

MRMOGA presents sublinear (and linear) speed ups on certain cases although it was tested on two-objective problems. S-PAMICRO produces significant reductions in execution time in comparison with a panmictic sequential and an standard island-based SMS-EMOA. Regarding PEA, as the number of cores increases, it showed a consistent reduction in execution time. However, it remains unclear which is the maximum number of cores for which this behavior is kept.

3.3. Iteration-level Parallelization

This section is devoted to describe representative approaches of iteration-level pMOEAs. Table 4 presents the main characteristics of these proposals and Table 5 summarizes their implementation details. Then, we further explain the details of each pMOEA.

Grimme *et al.* proposed in [85, 40] the use of a Predator-Prey parallelization model of MOEAs. In this parallel model, the population is structured usually in a toroidal grid which is populated by both predator and prey individuals. In this model, prey individuals are nodes in the grid. Preys correspond to solutions to the MOP and predators may define selection criteria, variation operators, among others, such that they apply pressure to the preys to reach Pareto optimal solutions (for further information see [40]. Based on this parallel model, the authors proposed a parallel Predator-Prey MOEA (pPPMOEA) [85] to tackle a multi-objective job shop scheduling problem. In order to employ pPP-MOEA using a specific target objective function, it is necessary to define a set of variation operators, a neighborhood structure for determining the individuals that are exposed to selection and reproduction, as well as a walking function for the movement pattern on the spatial structure. Theoretically, predators are independent agents which implies an asynchronous communication model. However, the authors do not explain what kind of communication strategy they followed. Due to the recent use of the predator-prey parallel model, pPP-MOEA represents an academic approach to solve the considered problem and further research is needed to improve the results and to get a better understanding of the predator-prey model. The current results just showed that the approach is

Table 4: Iteration-level proposals: main characteristics. The following terms are employed: population (Pop), mechanisms (Mech), master-slave model (MS), island model (IM), predator-prey (PP), diffusion (Diff), synchronous (Sync), asynchronous (Async), global (G), partitioned (P), centralized (C), distributed (D), homogeneous (Ho), heterogeneous (He), static (St), dynamic (Dyn), and not defined (N/D).

Name	Baseline Algorithm	Parallelism of	Parallel model	Communication	Decision Space	Objective Space	Pareto Front	Nodes	Distribution	Year	Reference
pPP-MOEA	Own approach	Pop and Mech	PP	N/D	G	G	D	He	St	2008	[85]
GPU-MOEA	NSGA-II	Pop and Mech	MS	Sync	G	G	C	Ho	St	2009	[86]
pSMS-EMOA	SMS-EMOA	Pop	MS	Async	G	G	C	Ho	St	2010	[3]
pMOEA/D	MOEA/D	Pop	MS	Sync	G	P	C	Ho	St	2010	[87]
PMOPSO	MOPSO	Pop	MS	Sync	G	G	C	Ho	St	2012	[88]
PQEA	MOEA/D	Pop	MS	Sync	G	P	C	Ho	St	2012	[89]
AMS-DEMO	DEMO	Pop	MS	Async	G	G	C	Ho	St	2013	[90]
GPU-MOEA/D-ACO	MOEA/D-ACO	Pop	MS	Sync	G	P	C	Ho	St	2014	[91]
PaDe	MOEA/D	Pop	IM	Async	G	P	D	He	St	2014	[92]
MP-MOEA/D	MOEA/D	Pop	Diff	Sync or Async	G	P	D	Ho	St	2015	[93]
GPU-NSGA-II	NSGA-II	Mech	MS	Sync	G	G	C	Ho	St	2015	[94]
PMEA	MOEA/D, IBEA$_{\epsilon+}$ SPEA2+SDE	Mech	MS	Sync	G	G	C	He	St	2016	[95]
MOMA	SMS-EMOA	Mech	MS	Sync	G	G	C	Ho	St	2016	[96]
CCSMPSO	SMPSO	Pop	IM	Sync	P	G	D	Ho	St	2016	[97]
SCC-NSGA-II	NSGA-II	Pop	MS	Sync	P	G	C	Ho	Dyn	2017	[98]
SA^{2}EA	SAEA	Pop	MS	Sync Async Semi-Async Adaptive	G	G	C	He	St	2017	[42]
PDL-MOEA	MOEA-DLA	Pop	IM	Sync	G	G	C	Ho	St	2017	[99]
pCMOEA/D-DMA	CMOEA/D-DMA	Pop	Diff	Sync	G	P	C	Ho	St	2019	[100]

Table 5: Iteration-level proposals: implementation characteristics.

Name	**Parallel architecture**	**Communication library**	**Programming language**	**Operating system**	**Reference**
pPP-MOEA	Not defined	Not defined	Not defined	Not defined	[85]
GPU-MOEA	CPU-GPU	CUDA	C	Windows XP Professional	[86]
pSMS-EMOA	Multi-computer (12-node Grid)	HTCondor and MPI	C	Linux	[3]
pMOEA/D	Multi-core (1 to 32 threads)	jMetal	Java	Linux OpenSuse and Ubuntu	[87]
PMOPSO	Not defined	Not defined	Not defined	Not defined	[88]
PQEA	Not defined	Not defined	Not defined	Not defined	[89]
AMS-DEMO	Multi-computer (16-node PC-Cluster)	MPI	C	Linux Fedora	[90]
GPU-MOEA/D-ACO	CPU-GPU	CUDA	C	Linux Ubuntu	[91]
PaDe	Multi-core (8 cores)	PaGMO	C++	Not defined	[92]
MP-MOEA/D	Multi-computer (128-node PC-Cluster)	MPI	C++	Not defined	[93]
GPU-NSGA-II	CPU-GPU	CUDA	C	Linux Ubuntu	[94]
PMEA	Parallel virtual machine	MPI	C	Not defined	[95]
MOMA	CPU-GPU	CUDA	C	Linux Fedora	[96]
CCSMPSO	Multi-core	Not defined	Not defined	Not defined	[97]
SCC-NSGA-II	Multi-core	Matlab	M	Windows 7	[98]
SA^{2}EA	Not defined	Not defined	Not defined	Not defined	[42]
PDL-MOEA	Multi-computer (PC-Cluster)	MPI	C++	Linux Ubuntu	[99]
pCMOEA/D-DMA	Not defined	Not defined	Not defined	[100]	

able to find nearly optimal solutions, but its performance is not as good as that of other pMOEAs considered in this survey.

In 2009, Wong [86] punctualized that more than 99% of the execution time of MOEAs is concentrated on performing dominance-checking and non-dominated selection procedures. To overcome this issue, he proposed an implementation of a pMOEA based on the use of GPUs. This GPU-MOEA is an iteration-level pMOEA that uses both parallelism of population and parallelism of mechanisms adopting the master-slave model. The slaves are in charge of independently evaluating the objective function values, of performing part of the stochastic tournament to select parent solutions, and also of performing crossover and mutation. Additionally, the dominance-checking and non-dominated selection are parallelized. The master node is focused only on collecting the global non-dominated solutions. The ZDT and DTLZ benchmarks were employed to assess the performance of this proposal using CPU and GPU configurations. As expected, the GPU-based version outperformed the CPU approach in terms of execution time, achieving speed ups of up to 14.6x. This approach is particularly interesting since it can be considered as a fully-parallelized iteration-level pMOEA because all the mechanisms as well as the population are processed in parallel.

Klinkenberg et al. [3] parallelized a modified version of SMS-EMOA based on the master-slave model with the aim of handling bi-objective problems with costly evaluations. For this purpose, two strategies are introduced. One replaces the variation operators of NSGA-II by a new mutation that creates several offspring from an individual and self-adapts its step-sizes. The second one splits the population into several slave nodes that evaluate their assigned solutions using a surrogate model[6] and filters out the most promising for being evaluated. Additionally, slaves should share its meta-model with the peers. The authors

---

[6] A surrogate model approximates the function values of a new solution using the results of previously evaluated solutions. The cost of training and using this model is relatively lower than the exact evaluation.

implemented this approach using the MPI library on a grid of 12 PCs having Intel Pentium 4 processors, and tested it on the Lamé superspheres benchmark [101] and on a real-world problem. The authors showed that the parallelization in addition to the surrogate model had a big impact on performance, considerably reducing the number of evaluations and achieving almost linear speed ups.

Nebro and Durillo proposed a parallel version of MOEA/D (pMOEA/D) [87] based on the master-slave model, targeting multi-core processors. The main population of pMOEA/D is partitioned among the threads, but the neighborhoods overlap. In consequence, some elements belonging to a specific thread could be modified by other thread when the population is updated. pMOEA/D employs three synchronized regions: (1) when threads are executing the offspring generation process so that the parent solutions remain unchanged, (2) during the computation of the ideal point, and (3) when the whole population is being updated using the neighborhood structure. Overall, pMOEA/D is executing in parallel the generation and evaluation of offspring solutions, which is useful when tackling time-consuming objective functions. pMOEA/D was compared with a sequential MOEA/D using eight bi-objective problems and one with three objective functions, analyzing both final quality of the Pareto front approximations and speed up. Although pMOEA/D was configured to use 1, 2, 4, 8, 16, and 32 threads, it was only able to produce Pareto fronts similar in quality to MOEA/D when using up to 8 threads. The use of more than eight threads made pMOEA/D not only to produce poor Pareto front approximations but also increased the execution time, i.e., no speed up was achieved. These results were obtained using benchmark functions which are not time-consuming. The authors also showed the results for a modified problem where they added useless loops to imitate a time-consuming MOP. For this problem, pMOEA/D was able to produce speed ups using eight or more threads. This observation emphasizes that it is necessary to test pMOEAs using time-consuming problems such that it is possible to analyze their actual advantages in computationally expensive MOPs.

Liu *et al.* [88] proposed the parallel multi-objective particle swarm optimization algorithm (PMOPSO) as an alternative to solve a soil sampling optimization model. PMOPSO uses the master-slave model to divide the set of particles into subgroups. Slave nodes store the subgroups, calculate the fitness values of the particles and change their locations. It is worth noting that only particles belonging to the same group can communicate among themselves. The master node holds the information from the slave nodes, searches for the best visited locations of the particles and for the global best visited location of the swarm and synchronously shares this information with all the slaves. PMOPSO was applied on a real-world bi-objective soil sampling problem related to the Hengshan County, Shanxi Province, in China. A four-threaded configuration was implemented to execute the proposal and it was compared with respect to a sequential MOPSO. Regarding the quality of solutions, both algorithms behave similarly. However, the convergence rate of PMOPSO was relativily slower which implies that it would need a large number of function evaluations to converge.

The Parallel Quantum Evolutionary Algorithm (PQEA) [89] is a quantum MOEA/D based on the master-slave model. Each slave is associated with a set of SOPs (defined by the weighted sum scalarizing function) which are grouped according to their similarity, calculating the Euclidean distance between the weight vectors. A q-bit individual is assigned to each slave to simultaneously optimize the corresponding SOPs, taking advantage of the linear superposition of all possible states. At each iteration, the quantum individuals are observed to obtain their corresponding values and, thus, update the approximate optimal solutions per weight vector and neighborhood. PQEA was compared to MOEA/D on the MOKP, using 250, 500, and 750 items with 2 and 3 objective functions. Both the coverage of two sets indicator [64] and GD showed that PQEA was able to produce Pareto front approximations with higher quality than MOEA/D and it exhibits a better convergence rate. Evidently, due to the current developments on quantum computing, the implementation of PQEA and its further utilization is not completely possible and affordable. However, this is indeed an interesting path for future research on parallel MOEAs.

Aiming to solve time-intensive MOPs, Depolli *et al.* [90] proposed the Asynchronous Master-Slave Differential Evolution for Multi-Objective Optimization Algorithm (AMS-DEMO). At each iteration, the master process executes the variation operators to generate the offspring population which is then split into several subpopulations. Each slave node receives in a non-blocking way, the individuals to calculate their objective functions. In order to use asynchronous communication to its full extent, AMS-DEMO implements queues of solutions. Hence, each slave continuously evaluates solutions as long as the master process sends them, and it only briefly interrupts the chain of continuous evaluations by sending the last evaluated solution to the master. It is worth emphasizing that due to its asynchronous nature, AMS-DEMO could be efficiently executed on heterogenous computer architectures, computers with a varying background load, and a dynamic number of processors. For every created solution, AMS-DEMO needs to communicate twice with the slave. Hence, the use of AMS-DEMO in MOPs whose evaluation time is comparable to the communication time will not produce any computational gains. Consequently, the real application of AMS-DEMO is related to time-consuming MOPs.

MOEA/D-ACO [102] is a version of MOEA/D that replaces its genetic operators by the main mechanisms of an ant colony optimization algorithm. In [91], a MOEA/D-ACO exploiting the large-scale parallelization of the graphics processing units (GPUs) is introduced. This approach, denoted as GPU-MOEA/D-ACO, takes advantage of the parallel nature of real-life ants which act as independent agents. Since each thread in a GPU shares a limited memory, it is not suitable to map an ant per thread. Instead, the authors decided to map an ant (related to a specific subproblem) per block of threads. Hence, each ant is in charge of constructing a solution and updating the pheromone structures. Additionally, when all the ants have generated their new solutions, the master process is executed in the CPU to update the global archive of non-dominated solutions. GPU-MOEA/D-ACO and MOEA/D-ACO were compared using nine combinations of the multi-objective travelling salesman problem (MTSP) with 2, 3, and 4 objective functions. GPU-MOEA/D-ACO was able to produce similar HV values than MOEA/D-ACO but with speed ups of up to 8.5x. Similarly to AMS-DEMO, the real usefulness of GPU-MOEA/D-ACO is clear when dealing with time-consuming objective functions such that the synchronous communication overhead with the CPU is negligible. On the other hand, due to the physical characteristics of the limited GPU's memory and registers, it is infeasible to handle a large number of subproblems using this approach.

The Parallel Decomposition (PaDe) [92] approach consists of a parallel scheme for MOEA/D. PaDe uses the asynchronous generalized island model to efficiently exchange chromosomic material among the islands. Hence, a fixed migration topology is defined by the proximitity of the weight vectors. After decomposing an MOP into N subproblems, each one is assigned to an island. The evolution of the islands is executed by k threads, such that each one takes care of N/k islands. The topology implemented in PaDe follows the logical structure imposed by the neighborhoods created in MOEA/D. Hence, an island is connected to the T islands that compose its neighborhood. Each island has a population of size $T+1$ that is evolved using the self-adaptive differential evolution algorithm. Once an island evolves during a given number of iterations, it performs its asynchronous migration process where its worst T solutions are replaced using the best solutions in its neighbohood. A critical aspect of PaDe is the definition of the ideal point which is employed to normalize the objective values of the whole population. Currently, PaDe uses $\vec{z}^* = (0, \ldots, 0)$ as the ideal point which will only allow to normalize populations lying in the positive orthant. The authors mentioned that an updating mechanism of the ideal vector will break the island asynchronicity. The bi-objective ZDT problems and the DTLZ test instances with 3, 4, and 5 objective functions were employed to compare PaDe with MOEA/D (using the differential evolution variation operators). Unfortunately, the authors only showed a comparison of a sequential PaDe with MOEA/D in terms of solution qualtiy where, regarding the hypervolume indicator, MOEA/D outperformed PaDe. In contrast, as it was expected, PaDe had better CPU-times.

Derbel *et al.* [93] proposed the Message-Passing MOEA/D (MP-MOEA) that implements a fine-grained pMOEA using the diffusion model. MP-MOEA is based on μ processing units ($p_i, i = 1, \ldots, \mu$), where each p_i handles a single subproblem related to the weight vector $\vec{w}^i$. The neighborhood structure of each weight vector defines the connections among the Processing Units (PUs), thus, each PU communicates with its T closest neighbors. It is worth noting that each PU stores its best solution found so far as well as a copy of the best solutions of its neighbors. At each iteration, a PU first performs the mating selection, variation, and, if necessary, it replaces its best solution and the local copy of its neighbors' solutions. If the best solution was replaced during the last $t_{\max}$ iterations, the PU needs to distribute it among its neighbors so that all of them have fresh information. The authors proposed both synchronous and asynchronous distribution processes and, in order to reduce the communication latency, the parameter $t_{\max}$ was introduced. Regarding the synchronous process, the PU distributes its best solution and, then, it waits to receive the best solutions from its neighbors to update the local copies of solutions. Clearly, an advantage of this type of communication mechanism is that all the PUs will have updated information before executing the next batch of $t_{\max}$ iterations. However, idle times may be introduced when a PU waits for the other ones to send a message and, evidently, the communication overhead is high. On the other hand, when using asynchronous communication, the PU distributes its best solution and, then, it checks if there are pending messages containig best solutions from its neighbors. This approach reduces both the communication overhead and the idle times. However, the PUs could operate with outdated information. MP-MOEA/D was tested on bi-objective ρMNK-landscapes with decision spaces of size 128, 256, 512, 1024, and 2048 (i.e, large-scale problems). The C++ MPI library was employed to implement MP-MOEA/D. Synchronous MP-MOEA/D, Asynchronous MP-MOEA/D, and the sequential MOEA/D were compared using HV and the ϵ^+ indicator [80]. Synchronous MP-MOEA/D showed to be better than MOEA/D in 50% of the test instances but Asynchronous MP-MOEA/D is significantly faster, where the acceleration depends on the neighborhood size and the dimensionality of the problem. In consequence, there is a trade-off between quality of the final approximation sets and the speed up that can be achieved.

Unlike the above described iteration-level proposals that parallelize the population, Gupta and Tan [94] proposed an iteration-level NSGA-II whose main target is the parallelization of the non-dominated sorting algorithm [20]. When the non-dominated sorting algorithm processes a population of $N \in [10^2, 10^3]$ solutions with 2 to 5 objective functions, the overall computational time is small. However, when $N \in [10^4, 10^5]$ and the dimensionality of the objective space is greater or equal to 4, the targeted mechanism is time-consuming. In consequence, a GPU-based non-dominated sorting algorithm is proposed. We should emphasize that this approach does not reduce the number of Pareto dominance comparisons. Instead, it performs the comparisons in parallel, making NSGA-II computationally economical and fast. In addition to the GPU-based non-dominated sorting, Gupta and Tan also introduced a GPU-based crowding distance selection. Hence, this NSGA-II uses parallelized selection mechanisms. This GPU-NSGA-II was compared to the sequential NSGA-II on the ZDT and DTLZ problems, the latter ones using from 2 up to 10 objective functions. To emphasize the advantages of the approach, both algorithms used populations of 20480, 25600, and 30720 individuals. In all cases, GPU-NSGA-II showed speed ups from 4.5x up to 14.5x. Clearly, this approach is only useful when managing large population sizes which could be beneficial when dealing with MaOPs that need lots of individuals to approximate the Pareto front. However, when using the commonly employed populations of hundreds of solutions, GPU-NSGA-II could introduce a great overhead due to its use of synchronous communications.

The Parallel Multi-Strategy Evolutionary Algorithm (PMEA) [95] executes in parallel three selection mechanisms over the main population. The underlying idea of PMEA is to increase its exploration ability by using decomposition-based [21], indicator-based (using the ϵ^+ indicator [28]), and shift-based [70] selection mechanisms. To this aim, the master-slave model is incorporated where the master node controls the whole evolutionary process by performing mating selection and variation operators to create an offspring population. Then, a copy of the current population and the offspring population are distributed to the slaves where the corresponding selection mechanisms are executed. Finally, the survival solutions are sent back to the master using MPI. The communication in both sides is done synchronously. It should be noted that the mating selection aims to balance the utilization of the three subpopulations and, in contrast to other pMOEAs, PMEA evaluates the objective functions of all the individuals in the master node which would not be very efficient when considering time-consuming objective functions. PMEA was mainly compared with MOEAs using the adopted selection mechanisms, i.e., MOEA/D [21], IBEA$_{\epsilon+}$ [28], and SPEA2+SDE [70] adopting the DTLZ and WFG test problems with 3, 5, 8, 10, and 15 objective functions. Regarding the hypervolume indicator, PMEA outperformed the baseline MOEAs with respect to which it was compared, which implies that this strategy properly combines the strengths of the three selection schemes while compensating for their respective weaknesses.

Manoatl Lopez and Coello proposed a Multi-Objective Memetic Algorithm (MOMA) [96] that uses SMS-EMOA as the global optimizer and executes multiple local search processes based on the IGD$^+$ indicator [79]. The authors developed a version of SMS-EMOA that adopts a GPU-based computation of the hypervolume contributions [7] and which evolves the population until a certain percentage of the maximum number generations has been executed. Then, an IGD$^+$-based local search is launched from each point of the reference set that IGD$^+$ needs, using the GPUs. Each thread determines the N closest solutions to the reference vector using the modified Euclidean distance (d^+) of the IGD$^+$ indicator. Based on this neighborhood, the mating selection is applied to create an offspring individual using the differential evolution operators, where one of the three parents is selected using d^+ and the other two are randomly chosen. Each thread synchronously communicates its neighboorhod and offspring solutions such that the next global population is formed, selecting the closest solutions to the reference vectors. Furthermore, the reference set is created using the technique proposed in [103] such that the current set of solutions is adjusted to a convex, linear, or concave shape by applying a set-based version of

Newton's method. It is worth emphasizing that the execution of the local search mechanism is based on the use of fine-grained parallelism but instead of a diffusion parallel model, MOMA follows the master-slave paradigm. MOMA was compared with respect to SMS-EMOA, HypE, and a sequential MOMA using the DTLZ and WFG test problems with 2 and 3 objective functions. Since all the MOEAs being compared optimize the hypervolume indicator, this indicator was utilized to assess their performance. MOMA was able to produce Pareto front approximations with marginally inferior HV values than SMS-EMOA but exhibiting the best execution times. The GPU-based computation of the hypervolume contributions helped MOMA to reduce its execution time and the use of the IGD$^+$-based local search engines improved the solution's quality.

The Cooperative Coevolutionary MOEA (CCMOEA) [104] defines a framework that evolves one population per objective function simultaneously. Taking advantage of its properties, Atashpendar *et al.* [97] proposed to couple the Speed-Constrained Multi-Objective Particle Swarm Optimization Algorithm (SMPSO) into CCMOEA, giving rise to CCSMPSO. The proposal breaks down the main population into several subpopulations, each of which is in charge of evolving a subset of the decision variables, using SMPSO via the island model. At each iteration of CCSMPSO, the subpopulations are evolved in parallel during one generation using SMPSO. Afterwards, each island broadcasts its best local partial solutions to update the local archives of the islands. In other words, all subpopulations evaluate complete solutions through a cooperative exchange of individuals which implies a connection topology where each island is connected to the remaining ones. When the stopping criterion is met, CCSMPSO merges all the local archives to build the Pareto front approximation. CCSMPSO was compared with respect to SMPSO, and both NSGA-II and SPEA2 under the CCMOEA framework on the bi-objective ZDT and DTLZ test suites. Three evaluation criteria were adopted: (1) quality of the approximation sets, using HV and the ϵ^+ indicators, (2) computational time, and (3) convergence speed. SMPSO outperformed CCSMPSO in terms of both the HV and the ϵ^+ indicator. On the other hand, CCSMPSO obtained speed ups that ranged from

3.5x to 4.6x in comparison to SMPSO. Finally, the authors reported that CC-SMPSO was the fastest algorithm to reach HV-based convergence through the evolutionary process.

Dynamic multi-objective optimization problems (DMOPs) involve solving an MOP where objectives, constraints, and/or decision variables change over time. Xu *et al.* [98] proposed the Spearman rank Correlation-based Parallel Cooperative Coevolutionary algorithm (SCC-NSGA-II) to handle DMOPs with changing variables. SCC-NSGA-II dynamically groups the decision variables using Spearman rank correlation analysis. Then, a subpopulation is employed to evolve the decision variables in each group during one generation. To this purpose, SCC-NSGA-II implements a master-slave model where the evaluation of complete solutions is done in the master node. Hence, a synchronous communication scheme is implemented to share the partial solutions. If the number of variables changes, then the master node needs to reorganize the groups. In order to assess the performance of SCC-NSGA-II, the authors employed the ZDT1 and the three-objective DTLZ1 problems with changing-variables. Three NSGA-II versions were employed in the comparative study: the original version, one using random grouping and another one using uniform grouping. According to the different IGD variables averaged over T time scales, SCC-NSGA-II outperformed the three adopted algorithms. However, no further performance analysis was performed. Hence, there is a lack of results that allow to determine the superiority of SCC-NSGA-II. This also calls for the need of a test suite of dynamic MOPs for parallel MOEAs.

Conventional synchronous pMOEAs need to wait for evaluations of all solutions in a population which causes to waste much idle time. To overcome this issue, Harada and Takadama [42] proposed the Self-Adaptive Semi-Asynchronous Evolutionary Algorithm (SA^{2}EA) that continuously evolves solutions whenever one solution completes its evaluation. SA^{2}EA controls its degree of asynchrony by changing the number of waited solutions based on the variance of their evaluation time. If the variance of the evaluation time is not large, it is more efficient to evolve new solutions after waiting some or most evaluations to employ their information on the whole evolutionary process. In contrast, if the variance is large, an asynchronous approach is still a good choice to reduce idle times. Hence, the search ability of this proposal depends on the trade-off between idle times to wait for solution evaluations and search efficiency. SA^{2}EA is implemented under a master-slave model where the nodes communicate with the master in an adaptive way, i.e., sometimes the communication could be synchronous, asynchrnonous, or semi-asynchronous. Additionally, the master node keeps the global non-dominated solutions. The authors employed NSGA-II as the baseline algorithm to implement a complete synchronous NSGA-II (CSNSGA-II), a semi-asynchronous NSGA-II (SANSGA-II), and a self-adaptive semi-asynchronous NSGA-II (SA^{2}NSGA-II). All algorithms employed 100 slave nodes and the bi-objective ZDT and WFG test suites were adopted for comparison. As expected, SANSGA-II and SA^{2}NSGA-II were faster than CSNSGA-II due to the reduction of idle times. Regarding the quality of approximation sets based on HV, SA^{2}NSGA-II produced similar results to CSNSGA-II while the latter outperformed SANSGA-II. This is an insight that the exploitation of idle times and its search ability allowed SA^{2}NSGA-II not only to be faster than CSNSGA-II but also to produce results of similar quality.

The application of pMOEAs is not only restricted to conventional MOPs since they have also been applied to solve dynamic MOPs and robust optimization problems. Such is the case of the Parallel Double-Level MOEA (PDL-MOEA) [99] in which a single-objective uncertain optimization problem is translated into a bi-objective MOP by conserving the expectation and the variance as two objectives. Hence, PDL-MOEA returns a set of solutions representing different stabilities. The double-level design is the master-slave parallel model where the master level maintains the global non-dominated solutions and at the slave level the problem is decomposed into a set of subproblems that are solved in parallel. PDL-MOEA is based on the MOEA with double-level archives (MOEA-DLA) [105] in which at the global level, a swarm works at the MOP level while in the sub-problem level the MOP is decomposed and a particle is in charge of a subproblem. Due to this scheme, the master node in

PDL-MOEA focuses on a well-diversified set of non-dominated solutions and the slaves accelerate convergence towards the Pareto front by solving the related subproblems. PDL-MOEA was compared with respect to MOEA-DLA, MOEA/D, and NSGA-II on the ZDT test problems, on DTLZ1-DTLZ3 and WFG1-WFG3, and on seven robust optimization problems (ROPs). Regarding HV, PDL-MOEA performed better than MOEA-DLA and it outperformed MOEA/D and NSGA-II. On the other hand, for the ZDT, DTLZ, and WFG test problems PDL-MOEA achieved speed ups of up to 1.22x and for the ROPs the speed ups were of up to 8.68x with respect to MOEA-DLA. The reason for this behavior is that conventional benchmarks are not time-consuming. Therefore, the communication time is greater than the evaluation time of the objective functions which produces no significant speed ups. However, the ROPs are time consuming and, hence, the advantages of PDL-MOEA in this domain can be clearly observed since in those cases, it achieves significant speed ups.

The Constrained MOEA/D with Directed Mating and Archives of Infeasible Solutions (CMOEA/D-DMA) [106] was designed to tackle constrained MaOPs. For this purpose, it maintains the best feasible solution for each subproblem and a set of infeasible solutions. To generate new solutions, it employs directed mating which requires the feasible solution and some of the infeasible solutions. Hence, it does not employ neighborhood structures. In 2019, Miyakawa *et al.* [100] proposed a parallel CMOEA/D-DMA, denoted as pCMOEA/D-DMA, that launches a thread for each subproblem, i.e., a thread is associated with a single solution. The neighboring relations associated to each weight vector are employed to define the communication scheme, thus, an individual can only communicate with its nearest neighbors. In consequence, pCMOEA/D-DMA is a fine-grained pMOEA that uses the diffusion model. For each subproblem, the best feasible solution is stored and an archive of infeasible solutions is maintained so that directed mating can be performed. If there is enough information to execute the directed mating, an individual does not need to communicate with its neighbors which implies no communication overhead because the offspring generation and the solution update can be performed in parallel. Otherwise, the individual communicates with its neighbors to retrieve the necessary information. To examine the potential of pCMOEA/D-DMA, it was compared with CMOEA/D-DMA and MOEA/D on constrained bi-objective knapsack problems with 500 items, focusing only on the final quality of the approximation sets. Unfortunately, the authors just presented a reduced comparison that showed that the hypervolume values related to pCMOEA/D-DMA remain constant as the granularity of the parallelization increases. However, MOEA/D showed better HV values.

3.3.1. Discussion

Unlike algorithmic-level approaches that involve the parallelization of whole algorithms, iteration-level proposals look at parallelizing either the evaluation of individuals or complete mechanisms of an MOEA. In this level of parallelization, the framework of MOEA/D has attracted a lot of attention due to its inherent potential of parallelization, giving rise to pMOEA/D, PQEA, GPU-MOEA/D-ACO, PaDe, MP-MOEA/D, and PMEA. From these approaches, pMOEA/D, PQEA, and GPU-MOEA/D-ACO are similar according to our proposed taxonomy, being the partioning of the objective space and the use of a master-slave model the two main characteristics. In contrast, PaDe and MP-MOEA/D represent interesting approaches. PaDe is an asynchronous pMOEA based on the island model which allows the maintenance of a distributed Pareto front. As in the case of pMOEA/D, PQEA, and GPU-MOEAA/D-ACO, it uses a partitioned objective space. MP-MOEA/D is one of few approaches that employs the diffusion model and has the capacity to perform asynchronous and synchronous communication among the demes. Moreover, it was implemented using MPI which encourages its execution in massively parallel architectures. Regarding the communication strategy, SA^{2}EA is the only pMOEA (taking into account all the parallel levels) that has an adaptive communication scheme. The demes of SA^{2}EA can communicate in a synchronous, asynchronous, and semi-asynchronous fashion depending on the worklod conditions. As in the case of the parallel hyper-heuristic [52] in the previous section, SA^{2}EA can be taken as a cornerstone to the design of configurable pMOEAs. Another MOEA that has been constantly parallelized is the SMS-EMOA. Regarding iteration-level proposals of SMS-EMOA, pSMS-EMOA aims to tackle very expensive MOPs by dividing the evaluation of individuals in several nodes. However, pSMS-EMOA suffers from the high computational cost of computing the contributions to the hypervolume indicator which makes it prohibitive for MaOPs. In contrast, MOMA is an approach that parallelizes two mechanisms: the computation of the calculation of the hypervolume contributions and it launches several IGD$^+$-based local search mechanisms to improve the current solutions. Nevertheless, neither pSMS-EMOA nor MOMA can obtain the quality results of S-PAMICRO [61] at low execution times.

Regarding computational time, the authors of SA^{2}EA provide a remarkable study that analyzes the impact of both a synchronous and an asynchronous communication scheme. Asynchronicity effectively reduces the overall computational time although it comprises the quality of the Pareto front approximation. On the other hand, a synchronous communication implies slighty larger runtimes at the expense of better Pareto front approximations. Hence, a multi-objective optimization problem can be formulated based on the use of a synchronous or an asynchronous scheme. SMPSO is able to obtain speed ups of up to 4x on solving large-scale versions of ZDT and DTLZ problems, which makes it a promising approach when dealing with large-scale MOPs. Regarding MOMA, the authors presented two versions: a CPU-based MOMA and a GPU-MOMA. Both parallel versions operate faster than the sequential MOMA but the results showed that the use of GPU hardware considerably increases the speed ups. Overall, this effect is consistent in the various GPU-based approaches such as GPU-NSGA-II, GPU-MOEA/D-ACO, and GPU-MOEA. Regarding the MOEA/D approaches, MP-MOEA/D presents promising results due to the massive use of parallel resources (it was executed on a 128-node cluster). Moreover, MP-MOEA/D is able to solve large-scale MOPs of up to 2048 variables. Nevertheless, MP-MOEA/D was only tested on a bi-objective landscape problem. In contrast, PaDe presents almost-linear speed ups on the three-, four-, and five-objective DTLZ problems.

Hence, PaDe rises as the fastest MOEA/D-like algorithm.

3.4. Solution-level Parallelization

Unlike the algorithmic- and iteration-levels that are problem independent, the solution-level completely depends on the MOP being tackled. As initially proposed by Talbi [13], solution-level focuses on the parallel evaluation of a single solution, considering objective functions and constraints but they need to be computationally expensive. The current state-of-the-art benchmark problems do not include any time-consuming functions. At most, some researchers have added useless loops to the test problems to emulate a time-consuming functions [87]. Evidently, the real expensive MOPs can be found in real-world applications [1]. In consequence, solution-level must be considered as a fundamental part in our taxonomy (even though there are no current approaches that consider this level to the best of the authors' knowledge) since in real-world applications it is much more likely to find time-consuming MOPs that require the parallelization of their functions using either of the two proposed ways (i.e., decomposition of data or decomposition of functions). Additionally, it is very important to propose benchmark problems that emulate or consider computationally expensive functions.

3.5. Multi-level Parallelization

In this section, the multi-level pMOEAs (also known as hybrid approaches) are described. The main characteristics of these proposals are summarized in Table 6 while Table 7 shows the implementation details. It is worth noting in the table the existence of solution-level proposals.

The Distributed Cooperation model of Multi-Objective Genetic Algorithm with Environmental Scheme (DCMOGADES) [107] is a multi-level approach that uses both algorithmic- and iteration-levels. It combines DCMOGA [44] and MOGADES [43] where the former imposes the algorithmic-level structure and the latter is in charge of the iteration-level. In this light, $N + 1$ islands are initially created using a star topology (as employed in DCMOGA), where

Table 6: Multi-level proposals: main characteristics. The following terms are employed: master-slave model (MS), island model (IM), parallelism of population (PPop), synchronous (Sync), asynchronous (Async), global (G), partitioned (P), centralized (C), distributed (D), homogeneous (Ho), heterogeneous (He), static (St), dynamic (Dyn), and not applicable (N/A) when an approach does not use a parallel level.

Name	**Algorithmic-level**	**Iteration-level**	**Solution-level**	**Communication**	**Decision Space**	**Objective Space**	**Pareto Front**	**Nodes**	**Distribution**	**Year**	**Reference**
DCMOGADES	IM	PPop IM	N/A	Sync	G	P	C	Ho	Dyn	2002	[107]
Hierarchical approach in distributed MOEAS	MS	PPop IM	N/A	Sync Async	G or P	G or P	D	He	St	2008	[108]
Multi-level NSGA-II	N/A	PPop IM	Decomposition of functions	Sync	G	G	D	Ho	St	2015	[38]
CPU-GPU NSGA-II	N/A	PPop MS	Decomposition of data	Sync or Async	G	G	C	Ho	Dyn	2018	[39]

Table 7: Multi-level proposals: implementation characteristics.

Name	**Parallel architecture**	**Communication library**	**Programming language**	**Operating system**	**Reference**
DCMOGADES	Not defined	Not defined	Not defined	Not defined	[107]
Hierarchical approach in distributed MOEAS	Multi-computer (8-node PC-Cluster and 24-node Grid))	Java MPI (Cluster) gLite (for Grid)	Java (Cluster and Grid)	Linux Scientific Linux	[108]
Multi-level NSGA-II	Multi-computer and multi-core (180-node PC-Cluster, 8-core nodes with 16 threads each one)	MPI and OpenMP	C	Not defined	[38]
CPU-GPU NSGA-II	Multi-computer (4-node PC-Cluster) and GPU	MPI and OpenCL	C++ (MPI) and C (OpenCL)	Linux CentOS	[39]

N islands correspond to Single-Objective Genetic Algorithm (SOGA) groups and the remaining one is a Multi-Objective Genetic Algorithm (MOGA) group. Then, each group is structured using the island model, following the design of MOGADES. Hence, there will be $N + 1$ groups connected by a star topology, where each group is formed by an island model using an unidirectional ring topology with random migration. Thus, the MOGA group runs MOGADES and the peripheral nodes, related to the SOGA groups, perform a parallel genetic algorithm (pGA) for single-objective optimization. Moreover, when extreme solutions are interchanged and they are better in the corresponding objective, one deme of the source node is migrated. This way, the most successful algorithm obtains more resources. DCMOGADES performed better than NSGA-II, SPEA2 and MOGADES on two bi-objective problems, producing widespread solutions. However, more evidence is required to validate its performance. One advantage of this scheme is that the number of demes is independent of the number of objectives.

Zaharie et al. [108] proposed a framework for the parallel execution of MOEAs, based on a layered structure, targetting different execution environments such as single computers, computer clusters and grid infraestructure. In the upper layer, a master-slave structure is implemented to independently evolve colonies of populations, each one being executed in a location of a grid environment. The slaves only communicate with the master node at the end of the evolutionary process to share the non-dominated individuals that they found. Each slave is in turn structured using the island model, where the islands implement sequential algorithms that explore the entire decision space or a fragment of it. From a practical point of view, this layer is executed on a cluster of computers so that each processor deals with the evolution of one or several subpopulations from a colony. The authors implemented the approach using eight independent pMOEAs (based on NSGA-II and island APDE [109]) running on a cluster, having different communication strategies. In the highest level, a master process, hosted on a grid, collects all the results and constructs the set of non-dominated solutions. The authors tested the proposal on the

ZDT test suite, observing that the best strategy was to use NSGA-II with a population of 200 individuals and a crossover probability of 0.9. Moreover, the collective step affects the speed up when the evaluation cost is negligible. The main drawback of this approach is that the user must define the configuration of each MOEA.

The design of digital circuits is a challenging application in which, besides functionality, it is required to minimize the area of the circuit and the power consumption. Hrbacek [38] proposed a multi-objective evolutionary approach where the trade-off between error and efficiency of the circuit is exploited. The proposal, denoted here as multi-level NSGA-II, combines an iteration-level design using the island model with a solution-level that aims to reduce the high computational cost of evaluating the objective functions of the generated digital circuits. To handle the design of digital circuits, the authors embedded a cartesian genetic programming (CGP) representation into NSGA-II. Due to the neutrality present in CGP, premature convergence is caused. In consequence, the authors introduced a new equivalence rank which allows to place equivalent solutions in a certain order that helps to preserve the neutrality character of CGP. Hence, each island executes a NSGA-II algorithm using the above mentioned modifications. The evaluation of each of the three objective functions (namely, the mean squared error, area of the circuit based on the approximate number of transistors, and latency) was performed in parallel using a decomposition of function approach. The authors analyzed the performance of the approach and the most relevant result was that the influence of the island model to reduce the average error of the circuit has a good impact when the number of islands increases. They also analyzed the effect of increasing the size of the offspring population and the number of generations executed. Regarding the design of two arithmetical circuits (a 4-bit multiplier and a 4-bit adder), the approach was able to generate designs that could reduce the overall error.

Escobar *et al.* [39] introduced an MOEA for feature selection in electroencephalogram (EEG) classification which takes advantage of the CPU-GPU technology. Specifically, this proposal deals with a multi-objective feature selection problem (MOF) in unsupervised classification of patterns characterized by a high number of features. Such a problem is tackled using as baseline NSGA-II that is adapted to be executed in a CPU-GPU environment. In the upper level, the iteration-level is implemented to split the population following the master-slave model. The communication between the master and the slaves is implemented using MPI to dynamically distribute the population, which reduces unbalanced loads. In consequence, the master asynchronously responds to the requests for subpopulations of each slave until there is no more work to do. Each slave evolves a subpopulation that encodes different feature selections. Each individual performs a k-means algorithm to evaluate two objective functions defined according to two clustering validation indices which correspond to the minimization and maximization of the intra-cluster and the inter-cluster distances. Due to these time-consuming functions, it is necessary to parallelize them. The nature of the k-means algorithm allows to exploit the data parallelism. Therefore, the solution-level uses the decomposition of data approach. The proposed approach was tested on 178 EEG patterns with 3600 features per pattern, varying the number of subpopulations from 1 to 32 subpopulations. Both a sequential and the CPU-GPU approaches were compared and the experimental results showed that they are similar in terms of hypervolume values, although the latter was faster as the number of subpopulations increased. With 32 subpopulations, it was possible to obtain speed ups of 70x.

3.5.1. Discussion

The design of multi-level approaches is a challenging and promising problem research direction. As shown in Figure 1, one can produce several combinations of parallel models, exploiting the advantages of them. Currently, due to the available hardware, the design of multi-level pMOEAs can be easily accomplished, combining GPUs, multi-core and multi-computer systems. For example, the Multi-level NSGA-II uses a 180-node PC-Cluster, where each computer has a multi-core architecture. This architecture encourages a massive parallelization. It is worth noting that both the Multi-level NSGA-II and the CPU-GPU

NSGA-II are iteration-level approaches that also support the parallelization of the calculation of the objective functions. Hence, these approaches have great potential to be used when dealing with expensive MOPs with heterogeneous objective functions. In contrast, DCMOGADES [107] and the proposal of Zaharie *et al* [108] implement a more common structure oriented to the parallelization of the population. In summary, multi-level parallelization is still a fertile research field where algorithms with different characteristics can be constructed. However, the few degrees of freedom available in these approaches is their main drawback. For instance, it is necessary to analyze the overhead related to the communication between the pieces of hardware. Furthermore, due to the utilization of two or more parallel schemes, this involves a study of how the inherent parameters affect the quality of the final approximation sets.

4. Potentially parallelizable MOEAs

In this section, we discuss two design strategies of multi-objective evolutionary algorithms: coevolution and decomposition. Both approaches are specially relevant due to the wide variety of algorithms based on them and their potential to be parallelized. On the one hand, coevolutionary MOEAs (CMOEAs) which are extensions of traditional MOEAs, have been effectively employed to solve large-scale MOPs [110]. Large-scale MOPs involve decision spaces with hundreds (or even thousands) of decision variables which makes them very complex and computationally expensive. On the other hand, decomposition-based MOEAs are a popular designing strategy where an MOP is decomposed into several single-objective optimization problems (SOPs) which are simultaneously solved by an MOEA [111, 112]. MOEAs based on these strategies have a great potential to be parallelized.

4.1. Coevolutionary MOEAs

According to Miguel Antonio and Coello [110], coevolutionary MOEAs are mainly divided into three main classes: cooperative CMOEAs, competitive

CMOEAs, and competitive-cooperative CMOEAs. From these classes, the cooperative CMOEAs (CCMOEAs) are noteworthy since they decompose either the decision or the objective spaces (the reader is referred to [110] which presents a survey of CCMOEAs). Hence, parallelization seems as a straightforward way to improve these algorithms. Regarding the decomposition of the decision space, CCMOEAs assign a species population to each subset of decision variables. Thus, a complete solution is partitioned into a given number of species populations. A CCMOEA makes the member of the species populations collaborate to evaluate the objective functions. Some representative sequential CCMOEAs can be found in [113, 114, 115, 116, 117, 6]. Due to the existence of multiple species populations in charge of subsets of decision variables, the parallelization could be a latent tool to improve these CCMOEAs. However, special attention should be paid to these MOPs since sometimes there are highly complex interdependencies which prevents their parallelization. In Section 3.2, we revised pPAES [47] which is an algorithmic-level parallel CCMOEA and we also analyzed the functioning of two iteration-level parallel CCMOEAs focused on the partitioning of the decision space [98, 97] in Section 3.3. In the case of CCMOEAs focused on the decomposition of the objective space, the species populations cooperate with each other to approximate the whole Pareto front. In this case, each objective function is assigned to a species population to be optimized (which contrasts with the decomposition-based MOEAs where the SOPs are formed by scalarizing functions). This scheme aims to improve the exploration of the objective space. Recent proposals are the following: [118, 119, 117, 120]. Due to the assigment of an objective function to a species population, the application of the master-slave or the island model seems direct. However, an important factor is the maintenance of diversity among the whole population since the optimization of the objective functions in isolation may cause diversity loss.

4.2. Decomposition-based MOEAs

The Multi-Objective Evolutionary Algorithm based on Decomposition [21] has deserved a lot of attention from the evolutionary multi-objective optimization community due to is efficiency and good performance. The underlying idea is the decomposition of the MOP into several SOPs by using scalarizing functions. MOEA/D simultaneously optimizes the SOPs to push the population closer to the Pareto front. Since MOEA/D manages multiple SOPs, this has been exploited to generate several parallel versions of MOEA/D. In this paper, we have analyzed several parallel versions of MOEA/D where most of them are based on the iteration-level approach: pCMOEA/D-DMA [100], PMEA [95], MP-MOEA/D [93], PaDe [92], GPU-MOEA/D-ACO [121], PQEA (which is a quantum-based version of MOEA/D) [89], and pMOEA/D [87]. Regarding the algorithmic-level approaches, to the best of our knowledge, MOEA/D$_{sp}^{pe}$ [58] is the only one under this parallelization level. In the specialized literature, there are a plethora of sequential approaches following the MOEA/D framework [111, 112]. Even though MOEA/D seems to be easily parallelizable due to its use of multiple SOPs, there are several subtleties that should be taken into account. First, much of the success of MOEA/D is due to its mechanism to propagate solutions in the underlying neighborhood structure. Second, an important aspect is the definition of the synchronicity or asynchronicity of the communication. Depending on which one is adopted, this impacts the quality of the final solution set as shown in [93, 92, 87].

5. Real-world Applications

Throughout the years, MOEAs have shown their value when solving real-world problems. These problems, coming from different engineering, industrial, and scientific areas, represent a challenge because they usually involve high dimensionality in both decision and objective spaces and the objective functions are computationally expensive. In consequence, parallelization techniques have emerged as an important tool to improve MOEAs and, thus, allowing them to effectively tackle these real-world MOPs. In Table 8, we provide a few representative examples of real-world problems, indicating the parallel level(s) and the corresponding parallel model to which they correspond in our taxonomy, as well

Table 8: Real-world applications of pMOEAs. The following terms are employed: master-slave model (MS), island model (IM), parallelism of population (PPop), decomposition of data (DD), decomposition of function(DF), synchronous (Sync), asynchronous (Async), global (G), partitioned (P), centralized (C), distributed (D), homogeneous (Ho), heterogeneous (He), static (St), dynamic (Dyn), and not applicable (N/A) when an approach does not use a parallel level.

Problem	Algorithmic-level	Iteration-level	Solution-level	Baseline algorithm	n	m	Ref.
X-ray spectroscopic analysis	N/A	PPop MS	N/A	NPGA	2	2	[122]
	N/A	PPop MS	N/A	NSGA-II	2	3	[123]
Diesel engine design	IM	N/A	N/A	MOGADES	12	3	[124]
	MS	N/A	N/A	NSGA-II	10	3	[55, 68]
Airfoil design	N/A	PPop MS	DD	NSGA	14	2	[36]
	N/A	PPop Diff	N/A	SPEA2	24	2	[125]
Portfolio selection	IM	N/A	N/A	NSGA-II, clustering	30	2	[49]
2D packing	IM	N/A		Memetic NSGA-II	100	2	[126]
Antenna positioning	IM	N/A	N/A	NSGA-II	349	2	[127]
Mobile ad-hoc networks	IM	N/A	N/A	METCO	5	3	[128, 129]
	N/A	PPop Diff	N/A	cMOGA	5	2, 3	[130], [131]
Calibration of hydrologic models	N/A	PPop MS	N/A	AMALGAM	75	3	[132]
Photoinjector beam design	IM	N/A	DFs	MODE	10	2	[133]
Data mining	N/A	PPop Parallelism of mechanisms MS	N/A	NSGA-II	17	2	[134]
	MS	PPop IM	N/A	NSGA-II, PDE	10, 14	3	[108]
Protein structure prediction	N/A	PPop MS	N/A	PAES, NSGA-II	1	3	[135, 136]
Job shop scheduling	N/A	PPop PP	N/A	Single-objective heuristics	50	2	[85]
Dynamic molecular alignment	N/A	PPop MS	N/A	SMS-EMOA	80	2	[3]
Vehicle routing	N/A	PPop IM	N/A	NSGA-II	100	2, 3	[137]
Mesh partitioning	IM	N/A	N/A	SPEA2	7	2	[138]
Interplaneraty trajectory design	IM	N/A	N/A	NSGA-II	15, 19	2	[139]
Phylogenetic inference	N/A	PPop MS	DF	PhyloMOEA	50, 250	2	[37]
Competitive facility location	N/A	PPop MS	N/A	Memetic NSGA-II	10	2	[140]

as the baseline MOEAs used to solve instances with n decision variables and m objective functions. We also include some combinatorial applications: job shop scheduling, interplanetary trajectory design, phylogenetic inference, block layout and the 2D packing problem.

The common characteristic in all these problems is that they are indeed difficult, having multiple local Pareto fronts, constraints, expensive function evaluations, non-uniform costs for the entire population, huge search spaces and, in some cases, NP-completeness. The use of pMOEAs has produced a considerable reduction in computational time, achieving almost linear [122, 123] or sub-linear [55, 68], [108, 37] speed ups, and producing, in some cases, an improvement in the quality of solutions [55, 68, 49].

A difficult problem that is usually tackled by pMOEAs is the multi-objective portfolio selection problem. Streichert *et al.* [49] solved this problem using a very interesting approach where the decision and objective spaces were partitioned using a clustering technique. Regarding the problem, even though it is not computationally expensive, the authors proposed to control the number of assets such that the number of local Pareto fronts could be handled by the available processors. Hence, this is an insight for the design of benchmark problems for pMOEAs where we can match the computational resources to the properties of the problem. Sometimes, the objetive functions of real-world problems do not have an analytical form as benchmark problems normally do. Thus, numerical models are necessary as in the case of the shape of a laser pulse for precision alignment of molecules [3]. Due to the high computational cost of this problem, Klinkenberg *et al.* proposed to use a surrogate model and, then, solve the problem, using a parallel SMS-EMOA. The use of surrogate models and parallel techniques represents a clear approach to reduce the runtime complexity of solving a real-world problem. However, due to the use of the surrogate model, precision aspects of the problem may be lost. NP-complete problems such as the antenna positioning problem (APP) are remarkable approaches where the advantages of pMOEAs arise. Segredo *et al.* [127] tackled a multiobjectivized APP using a parallel version of SGA-II. Moreover, the problem involved a dataset of

$n = 349$ candidate sites for the positioning of base stations. The use of NP-complete problems to test pMOEAs is a noteworthy aspect to the design of benchmark problems. In this case, the authors achieved superlinear speed ups in comparison with the performance of a sequential version of NSGA-II. Another important aspect to take into account is the use of datasets in bechmark problems. Currently, due to the success of machine learning techniques, datasets are available for different applications. Hence, the evolutionary multi-objective community could focus on these datasets to create new benchmark problems, simulating real-world problems.

The parallel architectures employed to solve real-world problems range from multi-core computers [49], multi-processor systems [127, 126] up to grids of computers as in [3]. Furthermore, the parallel libraries commonly used in these applications are MPI[7] [141], [142], [36], [37], PVM[8] [122] and Condor[9] [125] for computer clusters; OpenMP[10] [37] and light weight processes of UNIX for processor architectures; and gLite[11] [108], GridWay[12] and Globus Toolkit[13] [125] for grids.

6. Some Future Research Trends

Currently, there are numerous pMOEAs that have achieved outstanding quality results and that have helped to reduce the computational cost of sequential MOEAs. However, there are lots of room for improvement. Some possible paths for future research in this area are summarized in the following points:

---

[7] http://www.mpi-forum.org

[8] http://www.csm.ornl.gov/pvm

[9] http://research.cs.wisc.edu/htcondor

[10] http://openmp.org/wp

[11] http://glite.cern.ch

[12] http://www.gridway.org

[13] http://www.globus.org/toolkit

- A public-domain framework that includes the most representative pMOEAs is necessary for both researchers and practitioners. Although such a framework is a challenging project since it involves the management of different parallel technologies, it would be a very useful tool for performing comparative studies and/or solving real-world applications. Currently, the PlatEMO framework [143] offers the possibility to execute in parallel MOEAs that are related to an experimental setting. On the other hand, the Pymoo framework [144] allows the parallelization of MOEAs by using threads, processes, and distributed processes. Finally, the jMetal framework [145] offers four parallel MOEAs as part of its suite of algorithms.

- Acoording to the No-Free Lunch Theorems for search [146, 147], a pMOEA cannot show a good performance on all the possible MOPs. Hence, it is important to generate comparative studies among different models and pMOEAs, aiming to identify the classes of problems in which a particular type of parallel model or specific pMOEA has a particularly good or bad performance. In this light, Falcón-Cardona *et al.* [62] proposed an island-based multi-indicator algorithm where it is possible to observe how different indicator-based MOEAs complement each other to produce Pareto front approximations with better convergence and diversity properties. By following this research direction, researchers could improve the robustness of pMOEAs, i.e., its good performance under MOPs with different search difficulties and Pareto front shapes.

- In [87], it was shown that the real advantages of pMOEA/D were only observable when using MOPs to which a useless loop was added to emulate a time-consuming problem. This demonstrated that the use of classic benchmarks such as the ZDT, DTLZ, and WFG test suites is not enough to validate pMOEAs. Consequently, it is mandatory to develop test suites specifically designed to test the properties of pMOEAs not only considering search difficulties but also incorporating computationally expensive MOPs with heterogeneous times in their objective functions. In this regard, a possibility is to conform a test suite based on some expensive real-world problems tackled by the evolutionary multi-objective optimization (EMOO) community, following a crowdsourcing style.

- As can be seen in Tables 2 and 4, most of the pMOEAs are based on a small number of baseline MOEAs such as NSGA-II, MOEA/D, and SMS-EMOA. However, currently, there is a plethora of recently proposed MOEAs that outperform these MOEAs, having very good performance on many-objective optimization problems [34, 148, 149] or even in large-scale MOPs [6]. Hence, it is necessary to propose parallel versions of such new MOEAs to determine their quality properties.

- In our proposed taxonomy (see Fig. 3), it is possible to see that a pMOEA encompasses several properties such as the parallel mode, the type of communication among the nodes and the distribution of the nodes, among others. Currently, we do not completely understand the effect of each of these properties. In consequence, a study that shows how these properties improve or worsen the performance of a pMOEA is necessary. For instance, the study of the impact of different topologies and migration schemes on the performance of different pMOEAs was presented in [150, 151] and [139], respectively. More recently, Hernández-Gómez *et al.* studied the impact of the migration parameters of an island-based SMS-EMOA.

- Currently, algorithmic-level and iteration-level pMOEAs have been designed under three main parallel models: master-slave, island model, and diffusion. However, new parallel models could be proposed or taken from the parallel single-objective evolutionary algorithms. Regarding the latter case, one possibility is the predator-prey model that has been scarcely exploited to design MOEAs [85, 40]. On the other hand, the consumer-producer model should be deeply investigated and implemented on pMOEAs. According to Roy *et al.* [41], in such a model, there are multiple processors, each running a copy of an evolutionary algorithm. Unlike the island model, each processor is not confined to a set of individuals. Instead, there is a common pool of individuals from which each processor picks up individuals for computing the next generation. Additonally, hybrid models have been scarcely explored to design MOEAs and their utilization could exploit the best properties of the three main parallel models. Finally, in general, the EMOO community should look at parallel models proposed in other disciplines and further exploit those utilized for single-objective evolutionary algorithms.

- Commonly, pMOEAs have a trade-off between achieving good execution times and producing high quality results. This fact is clear when testing asynchronous pMOEAs which are faster than their synchronous counterparts, but they produce Pareto front approximations with less quality. However, Harada and Takadama [42] showed that the use of an adaptive semi-asynchronous communication strategy on a pMOEA can produce outcomes similar in quality to synchronous MOEAs (or even similar to sequential algorithms) but maintaining high speed ups. This is an insight that the design of self-adaptive pMOEAs (not only taking into account the communication strategy) could lead to high quality algorithms. A possible research path is the implementation of a machine learning-based method that predicts when to switch from synchronous to asynchrnous communication and for how long. This can be done by means of a hyper-heuristic.
- Another interesting research direction is the design of parallel MOEAs to solve constrained MaOPs [152]. Currently, most of the pMOEAs have been focused on the solution of unconstrained artificial MOPs (see Tables 2, 4, and 6) but real-world problems often contain complex constraints and many objectives. The parallelization of objective functions and constraints could encourage the design of more solution-level approaches which have been scarcely explored by the EMOO community.
- There are several other topics that are also worth investigating, including: theoretical aspects of pMOEAs (e.g., convergence and landscape analy-

sis), automated parameter tuning of pMOEAs, design of parallel hyper-heuristics for multi-objective optimization, design of parallel memetic multi-objective evolutionary algorithms, density estimators and archiving techniques specially tailored for pMOEAs, as well as performance indicators to assess performance of pMOEAs, among others.

7. Conclusions

The parallelization of MOEAs is recommended in problems that require a very large population size, such as large scale optimization or many-objective optimization. They are also advisable in difficult and complex problems where it becomes necessary to find solutions in a fast manner, as well as in dynamic problems, where an answer is required almost in real-time. pMOEAs are also a good choice in problems in which it is very difficult to maintain diversity, since some parallel models (e.g., the use of islands) are very suitable to enhance diversity. In this paper, we presented a comprehensive review of publications on parallel multi-objective evolutionary algorithms. A summary of popular models of parallelization, including some recent innovations, has been provided. We also proposed a taxonomy based on three main levels of parallelization, emphasizing the properties that each pMOEA could have. Finally, we described some possible future research paths on which researchers and practitioners could focus.

References

[1] C. A. Coello Coello, G. B. Lamont, D. A. Van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd Edition, Springer, New York, 2007, iSBN 978-0-387-33254-3.

[2] L. V. Santana-Quintero, A. Arias Montaño, C. A. Coello Coello, A Review of Techniques for Handling Expensive Functions in Evolutionary Multi-Objective Optimization, in: Y. Tenne, C.-K. Goh (Eds.), Computational

Intelligence in Expensive Optimization Problems, Springer, Berlin, Germany, 2010, pp. 29–59, iSBN 978-3-642-10700-9.

[3] J.-W. Klinkenberg, M. T. Emmerich, A. H. Deutz, O. M. Shir, T. Bäck, A Reduced-Cost SMS-EMOA Using Kriging, Self-Adaptation, and Parallelization, in: M. Ehrgott, B. Naujoks, T. J. Stewart, J. Wallenius (Eds.), Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, Springer, Lecture Notes in Economics and Mathematical Systems Vol. 634, Heidelberg, Germany, 2010, pp. 301–311.

[4] B. Li, J. Li, K. Tang, X. Yao, Many-Objective Evolutionary Algorithms: A Survey, ACM Computing Surveys 48 (1).

[5] R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, Test Problems for Large-Scale Multiobjective and Many-Objective Optimization, IEEE Transactions on Cybernetics PP (99) (2016) 1–14.

[6] L. M. Antonio, C. A. Coello Coello, Use of Cooperative Coevolution for Solving Large Scale Multiobjective Optimization Problems, in: 2013 IEEE Congress on Evolutionary Computation (CEC'2013), IEEE Press, Cancún, México, 2013, pp. 2758–2765, iSBN 978-1-4799-0454-9.

[7] E. Manoatl Lopez, L. Miguel Antonio, Carlos A. Coello Coello, A GPU-Based Algorithm for a Faster Hypervolume Contribution Computation, in: A. Gaspar-Cunha, C. H. Antunes, C. Coello Coello (Eds.), Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015, Springer. Lecture Notes in Computer Science Vol. 9019, Guimarães, Portugal, 2015, pp. 80–94.

[8] R. Hernández-Gómez, C. A. C. Coello, E. Alba, A Parallel Version of SMS-EMOA for Many-Objective Optimization Problems, in: J. Handl, E. Hart, P. R. Lewis, M. L.-I. nez, G. Ochoa, B. Paechter (Eds.), Parallel Problem Solving from Nature – PPSN XIV, 14th International Conference, Springer. Lecture Notes in Computer Science Vol. 9921, Edinburgh, UK, 2016, pp. 568–577, iSBN 978-3-319-45822-9.

[9] T. Glasmachers, Optimized Approximation Sets for Low-Dimensional Benchmark Pareto Fronts, in: T. Bartz-Beielstein, J. Branke, B. Filipič, J. Smith (Eds.), Parallel Problem Solving from Nature - PPSN XIII, 13th International Conference, Springer. Lecture Notes in Computer Science Vol. 8672, Ljubljana, Slovenia, 2014, pp. 569–578.

[10] H. Aguirre, A. Liefooghe, S. Verel, K. Tanaka, A Study on Population Size and Selection Lapse in Many-objective Optimization, in: 2013 IEEE Congress on Evolutionary Computation (CEC'2013), IEEE Press, Cancún, México, 2013, pp. 1507–1514, iSBN 978-1-4799-0454-9.

[11] S. G. Akl, The Design and Analysis of Parallel Algorithms, Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[12] G. Coulouris, J. Dollimore, T. Kindberg, G. Blair, Distributed Systems: Concepts and Design, 5th Edition, Addison-Wesley Publishing Company, USA, 2011.

[13] E.-G. Talbi, A unified view of parallel multi-objective evolutionary algorithms, Journal of Parallel and Distributed Computing 133 (2019) 349–358.

[14] A. Nebro, F. Luna, E.-G. Talbi, E. Alba, Parallel Multiobjective Optimization, in: E. Alba (Ed.), Parallel Metaheuristics, Wiley-Interscience, New Jersey, USA, 2005, pp. 371–394, iSBN 13-978-0-471-67806-9.

[15] F. Luna, E. Alba, Parallel Multiobjective Evolutionary Algorithms, in: J. Kacprzyk, W. Pedrycz (Eds.), Springer Handbook of Computational Intelligence, Springer, Berlin, Germany, 2015, pp. 1017–1031, iSBN 978-3-662-43504-5.

[16] A. López Jaimes, C. A. Coello Coello, Applications of Parallel Platforms and Models in Evolutionary Multi-Objective Optimization, in: A. Lewis, S. Mostaghim, M. Randall (Eds.), Biologically-Inspired Optimisation

Methods, Springer, Berlin, Germany, 2009, pp. 23–49, iSBN 978-3-642-01261-7.

[17] D. A. Van Veldhuizen, J. B. Zydallis, G. B. Lamont, Considerations in Engineering Parallel Multiobjective Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 144–173.

[18] F. Luna, A. J. Nebro, E. Alba, Parallel Evolutionary Multiobjective Optimization, in: N. Nedjah, E. Alba, L. de Macedo Mourelle (Eds.), Parallel Evolutionary Computations, Springer, Berlin Heidelberg, 2006, pp. 33–56.

[19] E.-G. Talbi, S. Mostaghim, T. Okabe, H. Ishibuchi, G. Rudolph, C. A. Coello Coello, Parallel Approaches for Multi-objective Optimization, in: J. Branke, K. Deb, K. Miettinen, R. Slowinski (Eds.), Multiobjective Optimization. Interactive and Evolutionary Approaches, Springer. Lecture Notes in Computer Science Vol. 5252, Berlin, Germany, 2008, pp. 349–372.

[20] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II, IEEE Transactions on Evolutionary Computation 6 (2) (2002) 182–197.

[21] Q. Zhang, H. Li, MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.

[22] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: Multiobjective selection based on dominated hypervolume, European Journal of Operational Research 181 (3) (2007) 1653–1669.

[23] K. Deb, R. B. Agrawal, Simulated Binary Crossover for Continuous Search Space, Complex Systems 9 (2) (1995) 115–148.

[24] N. Srinivas, K. Deb, Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms, Evolutionary Computation 2 (3) (1994) 221–248.

[25] E. Zitzler, L. Thiele, Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach, IEEE Transactions on Evolutionary Computation 3 (4) (1999) 257–271.

[26] J. D. Knowles, D. W. Corne, Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy, Evol. Comput. 8 (2) (2000) 149–172.

[27] E. Zitzler, M. Laumanns, L. Thiele, SPEA2: Improving the Strength Pareto Evolutionary Algorithm, Tech. Rep. 103, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH) Zurich, Gloriastrasse 35, CH-8092 Zurich, Switzerland (May 2001).

[28] E. Zitzler, S. Künzli, Indicator-based Selection in Multiobjective Search, in: X. Y. et al. (Ed.), Parallel Problem Solving from Nature - PPSN VIII, Springer-Verlag. Lecture Notes in Computer Science Vol. 3242, Birmingham, UK, 2004, pp. 832–842.

[29] D. ung H. Phan, J. Suzuki, R2-IBEA: R2 Indicator Based Evolutionary Algorithm for Multiobjective Optimization, in: 2013 IEEE Congress on Evolutionary Computation (CEC'2013), IEEE Press, Cancún, México, 2013, pp. 1836–1845, iSBN 978-1-4799-0454-9.

[30] M. Emmerich, N. Beume, B. Naujoks, An EMO Algorithm Using the Hypervolume Measure as Selection Criterion, in: C. A. C. Coello, A. H. Aguirre, E. Zitzler (Eds.), Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005, Springer. Lecture Notes in Computer Science Vol. 3410, Guanajuato, México, 2005, pp. 62–76.

[31] C. Igel, N. Hansen, S. Roth, Covariance Matrix Adaptation for Multiobjective Optimization, Evolutionary Computation 15 (1) (2007) 1–28.

[32] Q. Zhang, H. Li, MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition, IEEE Transactions on Evolutionary Computation 11 (6) (2007) 712–731.

[33] J. Bader, E. Zitzler, HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization, Evolutionary Computation 19 (1) (Spring, 2011) 45–76.

[34] K. Deb, H. Jain, An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints, IEEE Transactions on Evolutionary Computation 18 (4) (2014) 577–601.

[35] H. Jain, K. Deb, An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point Based Nondominated Sorting Approach, Part II: Handling Constraints and Extending to an Adaptive Approach, IEEE Transactions on Evolutionary Computation 18 (4) (2014) 602–622.

[36] N. Marco, S. Lanteri, J.-A. Desideri, J. Périaux, A Parallel Genetic Algorithm for Multi-Objective Optimization in Computational Fluid Dynamics, in: K. Miettinen, M. M. Mäkelä, P. Neittaanmäki, J. Périaux (Eds.), Evolutionary Algorithms in Engineering and Computer Science, John Wiley & Sons, Ltd, Chichester, UK, 1999, Ch. 22, pp. 445–456.

[37] W. Cancino, L. Jourdan, E.-G. Talbi, A. C. B. Delbem, Parallel Multi-Objective Approaches for Inferring Phylogenies, in: C. Pizzuti, M. D. Ritchie, M. Giacobini (Eds.), Evolutionary Computation, Machine Learning and Data Mining in Bioinformatics, 8th European Conference, EvoBIO 2010, Springer. Lecture Notes in Computer Science Vol. 6023, Istanbul, Turkey, 2010, pp. 26–37, iSBN 978-3-642-12210-1.

[38] R. Hrbacek, Parallel multi-objective evolutionary design of approximate circuits, in: Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation, GECCO'15, Association for Computing Machinery, New York, NY, USA, 2015, pp. 687–694. `doi:10.1145/2739480.2754785`.
URL `https://doi.org/10.1145/2739480.2754785`

[39] J. J. Escobar, J. Ortega, A. F. Díaz, J. González, M. Damas, Multi-objective feature selection for eeg classification with multi-level parallelism on heterogeneous cpu-gpu clusters, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO'18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1862–1869. `doi:10.1145/3205651.3208239`.
URL `https://doi.org/10.1145/3205651.3208239`

[40] C. Grimme, J. Lepping, A. Papaspyrou, Parallel Predator—Prey interaction for Evolutionary Multi-Objective Optimization, Natural Computing 11 (3) (2012) 519–533.

[41] G. Roy, H. Lee, J. L. Welch, Y. Zhao, V. Pandey, D. Thurston, A Distributed Pool Architecture for Genetic Algorithms, in: 2009 IEEE Congress on Evolutionary Computation (CEC'2009), IEEE Press, Trondheim, Norway, 2009, pp. 1177–1184.

[42] T. Harada, K. Takadama, A Study of Self-Adaptive Semi-Asynchronous Evolutionary Algorithm on Multi-Objective Optimization Problem, Association for Computing Machinery, New York, NY, USA, 2017, p. 1812–1819.
URL `https://doi.org/10.1145/3067695.3084221`

[43] J. Kamiura, T. Hiroyasu, M. Miki, S. Watanabe, MOGADES: Multi-Objective Genetic Algorithm with Distributed Environment Scheme, in: A. Abraham, B. Nath, M. Sambandham, P. Saratchandran (Eds.), Computational Intelligence and Applications. 2nd International Workshop on Intelligent Systems Design and Applications (ISDA 2002), Dynamic publishers, Atlanta, Georgia, USA, 2002, pp. 143–148, iSBN 0-9640398-0-X.

[44] T. Okuda, T. Hiroyasu, M. Miki, S. Watanabe, DCMOGA: Distributed Cooperation Model of Multi-Objective Genetic Algorithm, in: J. Knowles (Ed.), Proceedings of the PPSN/SAB Workshop on Multiobjective Problem Solving from Nature II (MPSN-II), Granada, Spain, 2002.

[45] H. Horii, M. Miki, T. Koizumi, N. Tsujiuchi, Asynchronous Migration of Island Parallel GA for Multi-Objective Optimization Problem, in: L. Wang, K. C. Tan, T. Furuhashi, J.-H. Kim, X. Yao (Eds.), Proceedings of the 4th Asia-Pacific Conference on Simulated Evolution and Learning (SEAL'02), Vol. 1, Nanyang Technical University, Orchid Country Club, Singapore, 2002, pp. 86–90.

[46] N. Xiao, M. P. Armstrong, A Specialized Island Model and Its Application in Multiobjective Optimization, in: E. C.-P. et al. (Ed.), Genetic and Evolutionary Computation (GECCO'2003). Proceedings, Part II, Springer. Lecture Notes in Computer Science Vol. 2724, Berlin, Germany, 2003, pp. 1530–1540, iSBN 978-3-540-40603-7.

[47] C. A. Coello Coello, M. Reyes Sierra, A Study of the Parallelization of a Coevolutionary Multi-Objective Evolutionary Algorithm, in: R. Monroy, G. Arroyo-Figueroa, L. E. Sucar, H. Sossa (Eds.), Proceedings of the Third Mexican International Conference on Artificial Intelligence (MICAI'2004), Springer Verlag. Lecture Notes in Artificial Intelligence Vol. 2972, Mexico City, Mexico, 2004, pp. 688–697.

[48] J. Branke, H. Schmeck, K. Deb, M. Reddy, Parallelizing Multi-Objective Evolutionary Algorithms: Cone Separation, in: 2004 Congress on Evolutionary Computation (CEC'2004), Vol. 2, IEEE Service Center, Portland, Oregon, USA, 2004, pp. 1952–1957.

[49] F. Streichert, H. Ulmer, A. Zell, Parallelization of Multi-objective Evolutionary Algorithms Using Clustering Algorithms, in: C. A. Coello Coello, A. Hernández Aguirre, E. Zitzler (Eds.), Evolutionary Multi-Criterion Optimization. Third International Conference, EMO 2005, Vol. 3410, Springer. Lecture Notes in Computer Science Vol. 3410, Guanajuato, México, 2005, pp. 92–107.

[50] A. López-Jaimes, C. C. Coello, MRMOGA: Parallel Evolutionary Multiobjective Optimization using Multiple Resolutions, in: 2005 IEEE

Congress on Evolutionary Computation (CEC'2005), Vol. 3, IEEE Service Center, Edinburgh, Scotland, 2005, pp. 2294–2301.

[51] A. Essabri, M. Gzara, T. Loukil, Parallel Multi-Objective Evolutionary Algorithm with Multi-Front Equitable Distribution, in: 2006 Fifth International Conference on Grid and Cooperative Computing (GCC'06), IEEE Computer Society Press, Hunan, China, 2006, pp. 241–244.

[52] C. León, G. Miranda, C. Segura, Parallel Hyperheuristic: A Self-Adaptive Island-Based Model for Multi-Objective Optimization, in: 2008 Genetic and Evolutionary Computation Conference (GECCO'2008), ACM Press, Atlanta, USA, 2008, pp. 757–758, iSBN 978-1-60558-131-6.

[53] Z. xin Wang, G. Ju, A parallel genetic algorithm in multi-objective optimization, in: 2009 Chinese Control and Decision Conference, IEEE, Guilin, China, 2009, pp. 3497–3501.

[54] T. Qiu, G. Ju, A selective migration parallel multi-objective genetic algorithm, in: 2010 Chinese Control and Decision Conference, IEEE, Xuzhou, China, 2010, pp. 463–467.

[55] M. Yagoubi, L. Thobois, M. Schoenauer, Asynchronous Evolutionary Multi-Objective Algorithms with Heterogeneous Evaluation Costs, in: 2011 IEEE Congress on Evolutionary Computation (CEC'2011), IEEE Service Center, New Orleans, Louisiana, USA, 2011, pp. 21–28.

[56] Y. Zhou, J. Wang, J. Chen, S. Gao, L. Teng, Ensemble of many-objective evolutionary algorithms for many-objective problems, Soft Computing 21 (2017) 2407–2419.

[57] C. Sanhueza, F. Jiménez, R. Berretta, P. Moscato, PasMoQAP: A Parallel Asynchronous Memetic Algorithm for Solving the Multi-Objective Quadratic Assignment Problem, in: 2017 IEEE Congress on Evolutionary Computation (CEC'2017), IEEE Press, San Sebastián, Spain, 2017, pp. 1103–1110, iSBN 978-1-5090-4601-0.

[58] W. Ying, S. Chen, B. Wu, Y. Xie, Y. Wu, Distributed Parellel MOEA/D on Spark, in: 2017 International Conference on Computing Intelligence and Information System (CIIS'2017), IEEE Press, Nanjing, China, 2017, pp. 18–23.

[59] H. Chen, X. Zhu, W. Pedrycz, S. Yin, G. Wu, H. Yan, PEA: Parallel Evolutionary Algorithm by Separating Convergence and Diversity for Large-Scale Multi-Objective Optimization, in: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS'2018), IEEE Computer Society, Vienna, Austria, 2018, pp. 223–232, iSBN 978-1-5386-6872-6.

[60] N. Kantour, S. Bouroubi, D. Chaabane, A parallel moea with criterion-based selection applied to the knapsack problem, Applied Soft Computing 80 (2019) 358 – 373. `doi:https://doi.org/10.1016/j.asoc.2019.04.005`.
URL `http://www.sciencedirect.com/science/article/pii/S1568494619301899`

[61] R. Hernández Gómez, C. A. Coello Coello, E. Alba, A Parallel Island Model for Hypervolume-Based Many-Objective Optimization, in: P. p. Thomas Bartz-Beielstein, Bogdan Filipič, E.-G. Talbi (Eds.), High-Performance Simulation-Based Optimization, Springer, Studies in Computational Intelligence Vol. 833, Cham, Switzerland, 2020, pp. 247–273, iSBN 978-3-030-18763-7.

[62] J. G. Falcón-Cardona, H. Ishibuchi, C. A. C. Coello, M. Emmerich, On the Effect of the Cooperation of Indicator-based Multi-Objective Evolutionary Algorithms, IEEE Transactions on Evolutionary Computation (2021) 1–15`doi:10.1109/TEVC.2021.3061545`.

[63] C. M. Fonseca, P. J. Fleming, Genetic Algorithms for Multiobjective Optimization: Formulation, Discussion and Generalization, in: S. Forrest

(Ed.), Proceedings of the Fifth International Conference on Genetic Algorithms, University of Illinois at Urbana-Champaign, Morgan Kauffman Publishers, San Mateo, California, 1993, pp. 416–423.

[64] E. Zitzler, K. Deb, L. Thiele, Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, Evolutionary Computation 8 (2) (2000) 173–195.

[65] S. Huband, P. Hingston, L. Barone, L. While, A Review of Multiobjective Test Problems and a Scalable Test Problem Toolkit, IEEE Transactions on Evolutionary Computation 10 (5) (2006) 477–506.

[66] V. L. Huang, P. N. Suganthan, J. J. Liang, Comprehensive learning particle swarm optimizer for solving multiobjective optimization problems, International Journal of Intelligent Systems 21 (2) (2006) 209–226.

[67] D. A. V. Veldhuizen, Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, Ph.D. thesis, Department of Electrical and Computer Engineering. Graduate School of Engineering. Air Force Institute of Technology, Wright-Patterson AFB, Ohio, USA (May 1999).

[68] M. Yagoubi, M. Schoenauer, Asynchronous Master/Slave MOEAs and Heterogeneous Evaluation Costs, in: 2012 Genetic and Evolutionary Computation Conference (GECCO'2012), ACM Press, Philadelphia, USA, 2012, pp. 1007–1014, iSBN: 978-1-4503-1177-9.

[69] S. Yang, M. Li, X. Liu, J. Zheng, A Grid-Based Evolutionary Algorithm for Many-Objective Optimization, IEEE Transactions on Evolutionary Computation 17 (5) (2013) 721–736.

[70] M. Li, S. Yang, X. Liu, Shift-Based Density Estimation for Pareto-Based Algorithms in Many-Objective Optimization, IEEE Transactions on Evolutionary Computation 18 (3) (2014) 348–365.

[71] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable Test Problems for Evolutionary Multiobjective Optimization, in: A. Abraham, L. Jain, R. Goldberg (Eds.), Evolutionary Multiobjective Optimization. Theoretical Advances and Applications, Springer, USA, 2005, pp. 105–145.

[72] C. A. Coello Coello, N. Cruz Cortés, Solving Multiobjective Optimization Problems using an Artificial Immune System, Genetic Programming and Evolvable Machines 6 (2) (2005) 163–190.

[73] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications, Ph.D. thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (November 1999).

[74] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, I. Stoica, Spark: Cluster computing with working sets, in: Proceedings of the 2nd USENIX Conference on Hot Topics in Cloud Computing, HotCloud'10, USENIX Association, USA, 2010, p. 10.

[75] R. Cheng, M. Li, Y. Tian, X. Zhang, S. Yang, Y. Jin, X. Yao, A benchmark test suite for evolutionary many-objective optimization, Complex & Intelligent Systems 3 (1) (2017) 67–81.

[76] A. Auger, J. Bader, D. Brockhoff, E. Zitzler, Theory of the Hypervolume Indicator: Optimal $\{\mu\}$-Distributions and the Choice of the Reference Point, in: FOGA '09: Proceedings of the tenth ACM SIGEVO workshop on Foundations of genetic algorithms, ACM, Orlando, Florida, USA, 2009, pp. 87–102.

[77] R. Hernández Gómez, C. A. Coello Coello, E. Alba Torres, A Multi-Objective Evolutionary Algorithm based on Parallel Coordinates, in: 2016 Genetic and Evolutionary Computation Conference (GECCO'2016), ACM Press, Denver, Colorado, USA, 2016, pp. 565–572, iSBN 978-1-4503-4206-3.

[78] D. Brockhoff, T. Wagner, H. Trautmann, On the Properties of the $R2$ Indicator, in: 2012 Genetic and Evolutionary Computation Conference (GECCO'2012), ACM Press, Philadelphia, USA, 2012, pp. 465–472, iSBN: 978-1-4503-1177-9.

[79] H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified Distance Calculation in Generational Distance and Inverted Generational Distance, in: A. Gaspar-Cunha, C. H. Antunes, C. C. Coello (Eds.), Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015, Springer. Lecture Notes in Computer Science Vol. 9019, Guimarães, Portugal, 2015, pp. 110–125.

[80] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca, Performance Assessment of Multiobjective Optimizers: An Analysis and Review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.

[81] O. Schütze, X. Esquivel, A. Lara, C. A. Coello Coello, Using the Averaged Hausdorff Distance as a Performance Measure in Evolutionary Multiobjective Optimization, IEEE Transactions on Evolutionary Computation 16 (4) (2012) 504–522.

[82] D. P. Hardin, E. B. Saff, Discretizing Manifolds via Minimum Energy Points, Notices of the AMS 51 (10) (2004) 1186–1194.

[83] H. Ishibuchi, Y. Setoguchi, H. Masuda, Y. Nojima, Performance of Decomposition-Based Many-Objective Algorithms Strongly Depends on Pareto Front Shapes, IEEE Transactions on Evolutionary Computation 21 (2) (2017) 169–190.

[84] H. Wang, L. Jiao, X. Yao, Two_Arch2: An Improved Two-Archive Algorithm for Many-Objective Optimization, IEEE Transactions on Evolutionary Computation 19 (4) (2015) 524–541. `doi:10.1109/TEVC.2014.2350987`.

[85] C. Grimme, J. Lepping, A. Papaspyrou, The Parallel Predator-Prey Model: A Step towards Practical Application, in: G. Rudolph, T. Jansen, S. Lucas, C. Poloni, N. Beume (Eds.), Parallel Problem Solving from Nature–PPSN X, Springer. Lecture Notes in Computer Science Vol. 5199, Dortmund, Germany, 2008, pp. 681–690.

[86] M. L. Wong, Parallel multi-objective evolutionary algorithms on graphics processing units, in: Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers, GECCO '09, Association for Computing Machinery, New York, NY, USA, 2009, p. 2515–2522. `doi:10.1145/1570256.1570354`.
URL `https://doi.org/10.1145/1570256.1570354`

[87] A. J. Nebro, J. J. Durillo, A study of the parallelization of the multi-objective metaheuristic moea/d, in: C. Blum, R. Battiti (Eds.), Learning and Intelligent Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2010, pp. 303–317.

[88] D. Liu, Y. Liu, Y. Liu, X. Zhao, A Parallelized Multi-Objective Particle Swarm Optimization Model to Design Soil Sampling Network, in: 2012 20th International Conference on Geoinformatics, IEEE Press, Hong Kong, 2012.

[89] X. Wei, S. Fujimura, Parallel quantum evolutionary algorithms with Client-Server model for multi-objective optimization on discrete problems, in: 2012 IEEE Congress on Evolutionary Computation (CEC'2012), IEEE Press, Brisbane, Australia, 2012, pp. 3183–3190.

[90] M. Depolli, R. Trobec, B. Filipič, Asynchronous master-slave parallelization of differential evolution for multi-objective optimization, Evolutionary Computation 21 (2) (2013) 261–291, pMID: 22452341. `doi:10.1162/EVCO\_a\_00076`.
URL `https://doi.org/10.1162/EVCO_a_00076`

[91] M. Z. de Souza, A. T. Ramirez Pozo, A GPU Implementation of MOEA/D-ACO for the Multiobjective Traveling Salesman Problem, in: 2014 Brazilian Conference on Intelligent Systems, IEEE, Sao Paulo, Brazil, 2014, pp. 324–329, iSBN 978-1-4799-5618-0.

[92] A. Mambrini, D. Izzo, Pade: A parallel algorithm based on the moea/d framework and the island model, in: T. Bartz-Beielstein, J. Branke, B. Filipič, J. Smith (Eds.), Parallel Problem Solving from Nature – PPSN XIII, Springer International Publishing, Cham, 2014, pp. 711–720.

[93] B. Derbel, A. Liefooghe, G. Marquet, E.-G. Talbi, A Fine-Grained Message Passing MOEA/D, in: 2015 IEEE Congress on Evolutionary Computation (CEC'2015), IEEE Press, Sendai, Japan, 2015, pp. 1837–1844, iSBN 978-1-4799-7492-4.

[94] S. Gupta, G. Tan, A Scalable Parallel Implementation of Evolutionary Algorithms for Multi-Objective Optimization on GPUs, in: 2015 IEEE Congress on Evolutionary Computation (CEC'2015), IEEE Press, Sendai, Japan, 2015, pp. 1567–1574, iSBN 978-1-4799-7492-4.

[95] Z.-J. Wang, Z.-H. Zhan, J. Zhang, Parallel multi-strategy evolutionary algorithm using massage passing interface for many-objective optimization, in: 2016 IEEE Symposium Series on Computational Intelligence (SSCI'2016), IEEE, Athens, Greece, 2016.

[96] E. Manoatl Lopez, C. A. Coello Coello, A Parallel Multi-objective Memetic Algorithm Based on the IGD+ Indicator, in: J. Handl, E. Hart, P. R. Lewis, M. L.-I. nez, G. Ochoa, B. Paechter (Eds.), Parallel Problem Solving from Nature – PPSN XIV, 14th International Conference, Springer. Lecture Notes in Computer Science Vol. 9921, Edinburgh, UK, 2016, pp. 473–482, iSBN 978-3-319-45822-9.

[97] A. Atashpendar, B. Dorronsoro, G. Danoy, P. Bouvry, A parallel cooperative coevolutionary SMPSO algorithm for multi-objective optimization,

in: 2016 International Conference on High Performance Computing & Simulation (HPCS), IEEE, Innsbruck, Austria, 2016, pp. 713–720, iSBN 978-1-5090-2089-8.

[98] B. Xu, Y. Zhang, D.-w. Gong, L. Wang, A Parallel Multi-Objective Cooperative Co-Evolutionary Algorithm with Changing Variables, Association for Computing Machinery, New York, NY, USA, 2017, p. 1888–1893.
URL `https://doi.org/10.1145/3067695.3084222`

[99] W.-J. Yu, J.-Z. Li, W.-N. Chen, J. Zhang, A parallel double-level multiobjective evolutionary algorithm for robust optimization, Applied Soft Computing 59 (2017) 258 – 275. `doi:https://doi.org/10.1016/j.asoc.2017.06.008`.
URL `http://www.sciencedirect.com/science/article/pii/S1568494617303514`

[100] M. Miyakawa, H. Sato, Y. Sato, A study for parallelization of multi-objective evolutionary algorithm based on decomposition and directed mating, in: Proceedings of the 2019 3rd International Conference on Intelligent Systems, Metaheuristics and Swarm Intelligence, ISMSI 2019, Association for Computing Machinery, New York, NY, USA, 2019, p. 25–29. `doi:10.1145/3325773.3325790`.
URL `https://doi.org/10.1145/3325773.3325790`

[101] M. T. Emmerich, A. H. Deutz, Test Problems Based on Lamé Superspheres, in: S. Obayashi, K. Deb, C. Poloni, T. Hiroyasu, T. Murata (Eds.), Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007, Springer. Lecture Notes in Computer Science Vol. 4403, Matshushima, Japan, 2007, pp. 922–936.

[102] L. Ke, Q. Zhang, R. Battiti, MOEA/D-ACO: A Multiobjective Evolutionary Algorithm using Decomposition and Ant Colony, IEEE Transactions on Cybernetics 43 (6) (2013) 1845–1859.

[103] E. Manoatl Lopez, C. A. Coello Coello, IGD$^+$-EMOA: A Multi-Objective Evolutionary Algorithm based on IGD$^+$, in: 2016 IEEE Congress on Evolutionary Computation (CEC'2016), IEEE Press, Vancouver, Canada, 2016, pp. 999–1006, iSBN 978-1-5090-0623-9.

[104] I. C. Parmee, A. H. Watson, Preliminary airframe design using co-evolutionary multiobjective genetic algorithms, in: Proceedings of the 1st Annual Conference on Genetic and Evolutionary Computation - Volume 2, GECCO'99, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1999, p. 1657–1665.

[105] N. Chen, W. Chen, Y. Gong, Z. Zhan, J. Zhang, Y. Li, Y. Tan, An evolutionary algorithm with double-level archives for multiobjective optimization, IEEE Transactions on Cybernetics 45 (9) (2015) 1851–1863.

[106] M. Miyakawa, H. Sato, Y. Sato, Directed mating in decomposition-based moea for constrained many-objective optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 721–728. `doi:10.1145/3205455.3205554`.
URL `https://doi.org/10.1145/3205455.3205554`

[107] T. Okuda, T. Hiroyasu, M. Miki, J. Kamiura, S. Watanabe, DCMOGADES: Distributed Cooperation Model of Multi-objective Genetic Algorithm with Distributed Scheme, in: Second International Workshop on Intelligent Systems Design and Application, Dynamic Publishers, Atlanta, Georgia, USA, 2002, pp. 143–148.

[108] D. Zaharie, D. Petcu, S. Panica, A Hierarchical Approach in Distributed Evolutionary Algorithms for Multiobjective Optimization, in: I. Lirkov, S. Margenov, J. Waśniewski (Eds.), Large-Scale Scientific Computing, 6th International Conference, LSSC 2007, Springer. Lecture Notes in Computer Science Vol. 4818, Sozopol, Bulgaria, 2008, pp. 516–523.

[109] D. Zaharie, D. Petcu, Adaptive Pareto Differential Evolution and Its Parallelization, in: R. Wyrzykowski, J. Dongarra, M. Paprzycki, J. Wasniewski (Eds.), Parallel Processing and Applied Mathematics, Springer, Lecture Notes in Computer Science, Vol. 3019, Heidelberg, Germany, 2004, pp. 261–268.

[110] L. Miguel Antonio, C. A. Coello Coello, Coevolutionary Multiobjective Evolutionary Algorithms: Survey of the State-of-the-Art, IEEE Transactions on Evolutionary Computation 22 (6) (2018) 851–865. `doi:10.1109/TEVC.2017.2767023`.

[111] A. Trivedi, D. Srinivasan, K. Sanyal, A. Ghosh, A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition, IEEE Transactions on Evolutionary Computation 21 (3) (2017) 440–462. `doi:10.1109/TEVC.2016.2608507`.

[112] Q. Xu, Z. Xu, T. Ma, A Survey of Multiobjective Evolutionary Algorithms Based on Decomposition: Variants, Challenges and Future Directions, IEEE Access 8 (2020) 41588–41614. `doi:10.1109/ACCESS.2020.2973670`.

[113] L. M. Antonio, C. A. C. Coello, M. A. R. Morales, S. G. Brambila, J. F. González, G. C. Tapia, Coevolutionary Operations for Large Scale Multiobjective Optimization, in: 2020 IEEE Congress on Evolutionary Computation (CEC), 2020, pp. 1–8. `doi:10.1109/CEC48606.2020.9185846`.

[114] M. Li, J. Wei, A Cooperative Co-Evolutionary Algorithm for Large-Scale Multi-Objective Optimization Problems, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 1716–1721. `doi:10.1145/3205651.3208250`.
URL `https://doi.org/10.1145/3205651.3208250`

[115] Z. Zhang, C. Zhong, Z. Xu, H. Teng, A Non-Dominated Sorting Cooperative Co-Evolutionary Differential Evolution Algorithm for Multi-

Objective Layout Optimization, IEEE Access 5 (2017) 14468–14477. `doi:10.1109/ACCESS.2017.2716111`.

[116] W. Zhao, S. Alam, H. A. Abbass, MOCCA-II: A multi-objective co-operative co-evolutionary algorithm, Applied Soft Computing 23 (2014) 407–416. `doi:https://doi.org/10.1016/j.asoc.2014.06.011`.

[117] L. M. Antonio, C. A. C. Coello, A non-cooperative game for faster convergence in cooperative coevolution for multi-objective optimization, in: 2015 IEEE Congress on Evolutionary Computation (CEC), 2015, pp. 109–116. `doi:10.1109/CEC.2015.7256881`.

[118] A. Menchaca-Méndez, E. Montero, L. M. Antonio, S. Zapotecas-Martínez, C. A. Coello Coello, M. Riff, A Co-Evolutionary Scheme for Multi-Objective Evolutionary Algorithms Based on ϵ -Dominance, IEEE Access 7 (2019) 18267–18283. `doi:10.1109/ACCESS.2019.2896962`.

[119] J. Wang, W. Zhang, J. Zhang, Cooperative Differential Evolution With Multiple Populations for Multiobjective Optimization, IEEE Transactions on Cybernetics 46 (12) (2016) 2848–2861. `doi:10.1109/TCYB.2015.2490669`.

[120] R. Wang, R. C. Purshouse, P. J. Fleming, Preference-inspired co-evolutionary algorithms using weight vectors, European Journal of Operational Research 243 (2) (2015) 423–441. `doi:https://doi.org/10.1016/j.ejor.2014.05.019`.

[121] M. Z. de Souza, A. T. R. Pozo, Parallel moea/d-aco on gpu, in: A. L. Bazzan, K. Pichara (Eds.), Advances in Artificial Intelligence – IBERAMIA 2014, Springer International Publishing, Cham, 2014, pp. 405–417.

[122] I. E. Golovkin, S. J. Louis, R. C. Mancini, Parallel Implementation of Niched Pareto Genetic Algorithm Code for X-ray Plasma Spectroscopy, in: Congress on Evolutionary Computation (CEC'2002), Vol. 2, IEEE Service Center, Piscataway, New Jersey, 2002, pp. 1820–1824.

[123] K. Xu, S. J. Louis, R. C. Mancini, A Scalable Parallel Genetic Algorithm for X-ray Spectroscopic Analysis, in: H.-G. B. et al. (Ed.), 2005 Genetic and Evolutionary Computation Conference (GECCO'2005), Vol. 1, ACM Press, New York, USA, 2005, pp. 811–816.

[124] T. Hiroyasu, M. Miki, J. Kamiura, S. Watanabe, H. Hiroyasu, MOGADES: Multi-Objective Genetic Algorithm with Distributed Environment Scheme, in: A. Abraham, L. Jain, R. Goldberg (Eds.), Evolutionary Multiobjective Optimization: Theoretical Advances And Applications, Springer-Verlag, London, 2005, pp. 201–227, iSBN 1-85233-787-7.

[125] V. Asouti, I. Kampolis, K. Giannakoglou, A Grid-Enabled Asynchronous Metamodel-Assisted Evolutionary Algorithm for Aerodynamic Optimization, Genetic Programming and Evolvable Machines 10 (4) (2009) 373–389.

[126] C. Segura, E. Segredo, C. León, Parallel Island-Based Multiobjectivised Memetic Algorithms for a 2D Packing Problem, in: 2011 Genetic and Evolutionary Computation Conference (GECCO'2011), ACM Press, Dublin, Ireland, 2011, pp. 1611–1618.

[127] E. Segredo, C. Segura, C. León, On the Comparison of Parallel Island-Based Models for the Multiobjectivised Antenna Positioning Problem, in: A. König, A. Dengel, K. Hinkelmann, K. Kise, R. J. Howlett, L. C. Jain (Eds.), Knowledge-Based and Intelligent Information and Engineering Systems, 15th International Conference, KES 2011, Springer. Lecture Notes in Computer Science Vol. 6881, Kaiserslautern, Germany, 2011, pp. 32–41.

[128] C. León, G. Miranda, C. Segura, METCO: A Parrallel Plugin-Based Framework for Multi-Objective Optimization, International Journal on Artificial Intelligence Tools 18 (04) (2009) 569–588.

[129] C. Segura, A. Cervantes, A. J. Nebro, M. D. J. íz Simón, E. Segredo, S. García, F. Luna, J. A. Gómez-Pulido, G. Miranda, C. Luque,

E. Alba, M. Ángel Vega-Rodríguez, C. León, I. M. Galván, Optimizing the DFCN Broadcast Protocol with a Parallel Cooperative Strategy of Multi-Objective Evolutionary Algorithms, in: M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, M. Sevaux (Eds.), Evolutionary Multi-Criterion Optimization. 5th International Conference, EMO 2009, Springer. Lecture Notes in Computer Science Vol. 5467, Nantes, France, 2009, pp. 305–319.

[130] E. Alba, B. Dorronsoro, F. Luna, P. Bouvry, A Cellular Multi-objective Genetic Algorithm for Optimal Broadcasting Strategy in Metropolitan MANETs, in: Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, IEEE Computer Society, Denver, Colorado, USA, 2005, iSBN 0-7695-2312-9.

[131] E. Alba, B. Dorronsoro, F. Luna, A. Nebro, P. Bouvry, L. Hogie, A cellular multi-objective genetic algorithm for optimal broadcasting strategy in metropolitan MANETs, Computer Communications 30 (4) (2007) 685–697.

[132] J. A. Vrugt, P. H. Stauffer, T. Wöhling, B. A. Robinson, V. V. Vesselinov, Inverse Modeling of Subsurface Flow and Transport Properties: A Review with New Developments, Vadose Zone 7 (2) (2008) 843–864.

[133] J. Qiang, Y. Chao, C. Mitchell, S. Paret, R. Ryne, A Parallel Multi-objective Differential Evolution Algorithm for Photoinjector Beam Dynamics Optimization, in: Proceedings of the 4^{th} International Particle Accelerator Conference (IPAC'2013), JACoW, Shanghai, China, 2013, pp. 1031–1033.

[134] M.-L. Wong, G. Cui, Data mining using parallel Multi-Objective Evolutionary algorithms on graphics hardware, in: 2010 IEEE Congress on Evolutionary Computation (CEC'2010), IEEE Press, Barcelona, Spain, 2010, pp. 1812–1819.

[135] J. C. Calvo, J. Ortega, M. Anguita, A Hybrid Scheme to Solve the Protein Structure Prediction Problem, in: M. P. Rocha, F. Fernández Riverola, H. Shatkay, J. M. Corchado (Eds.), Advances in Bioinformatics, 4th International Workshop on Practical Applications of Computational Biology and Bioinformatics 2010 (IWPACBB 2010), Advances in Intelligent and Soft Computing Vol. 74, Springer, Berlin, Germany, 2010, pp. 233–240, iSBN 978-3-642-13213-1.

[136] J. C. Calvo, J. Ortega, M. Anguita, Comparison of Parallel Multi-Objective Approaches to Protein Structure Prediction, The Journal of Supercomputing 58 (2) (2011) 253–260.

[137] I.-D. Psychas, M. Marinaki, Y. Marinakis, A Parallel Multi-Start NSGA II Algorithm for Multiobjective Energy Reduction Vehicle Routing Problem, in: A. Gaspar-Cunha, C. H. Antunes, C. C. Coello (Eds.), Evolutionary Multi-Criterion Optimization, 8th International Conference, EMO 2015, Springer. Lecture Notes in Computer Science Vol. 9018, Guimarães, Portugal, 2015, pp. 336–350.

[138] A. Rama Mohan Rao, Distributed Evolutionary Multi-Objective Mesh-Partitioning Algorithm for Parallel Finite Element Computations, Comput. Struct. 87 (23-24) (2009) 1461–1473.

[139] M. Märtens, D. Izzo, The Asynchronous Island Model and NSGA-II: Study of a New Migration Operator and Its Performance, in: Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation, GECCO '13, ACM, New York, NY, USA, 2013, pp. 1173–1180.

[140] A. Lančinskas, J. Žilinskas, Parallel Multi-objective Memetic Algorithm for Competitive Facility Location, in: R. Wyrzykowski, J. Dongarra, K. Karczewski, J. Waśniewski (Eds.), Parallel Processing and Applied Mathematics, 10th International Conference, PPAM 2013, Springer. Lecture Notes in Computer Science Vol. 8385, Warsaw, Poland, 2014, pp. 354–363, iSBN 978-3-642-55194-9.

[141] R. Mäkinen, P. Neittaanmäki, J. Periaux, M. Sefrioui, J. Toivanen, Parallel genetic solution for multiobjective MDO, in: A. Schiano, A. Ecer, J. Périaux, N. Satofuka (Eds.), Parallel CFD'96 Conference, Elsevier, Capri, 1996, pp. 352–359.

[142] B. R. Jones, W. A. Crossley, A. S. Lyrintzis, Aerodynamic and Aeroacoustic Optimization of Airfoils via a Parallel Genetic Algorithm, in: Proceedings of the 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, AIAA-98-4811, AIAA, St. Louis, Missouri, USA, 1998, pp. 1088–1096.

[143] Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: A MATLAB platform for evolutionary multi-objective optimization, IEEE Computational Intelligence Magazine 12 (4) (2017) 73–87.

[144] J. Blank, K. Deb, Pymoo: Multi-objective optimization in python, IEEE Access 8 (2020) 89497–89509.

[145] J. J. Durillo, A. J. Nebro, jmetal: A java framework for multi-objective optimization, Advances in Engineering Software 42 (10) (2011) 760–771. `doi:https://doi.org/10.1016/j.advengsoft.2011.05.014`.
URL `https://www.sciencedirect.com/science/article/pii/S0965997811001219`

[146] D. H. Wolpert, W. G. Macready, No Free Lunch Theorems for Optimization, IEEE Transactions on Evolutionary Computation 1 (1) (1997) 67–82.

[147] D. W. Corne, J. D. Knowles, No Free Lunch and Free Leftovers Theorems for Multiobjective Optimisation Problems, in: C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, L. Thiele (Eds.), Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, Springer. Lecture Notes in Computer Science. Volume 2632, Faro, Portugal, 2003, pp. 327–341.

[148] R. Hernández Gómez, C. A. Coello Coello, A Hyper-Heuristic of Scalarizing Functions, in: 2017 Genetic and Evolutionary Computation Conference (GECCO'2017), ACM Press, Berlin, Germany, 2017, pp. 577 –584, iSBN 978-1-4503-4920-8.

[149] J. G. Falcón-Cardona, C. A. C. Coello, M. Emmerich, CRI-EMOA: A Pareto-Front Shape Invariant Evolutionary Multi-Objective Algorithm, in: K. Deb, E. Goodman, C. A. C. Coello, K. Klamroth, K. Miettinen, S. Mostaghim, P. Reed (Eds.), Evolutionary Multi-Criterion Optimization, 10th International Conference, EMO 2019, Springer. Lecture Notes in Computer Science Vol. 11411, East Lansing, Michigan, USA, 2019, pp. 307–318, iSBN 978-3-030-12597-4.

[150] E. Szlachcic, W. Zubik, Parallel distributed genetic algorithm for expensive multi-objective optimization problems, in: R. Moreno-Díaz, F. Pichler, A. Quesada-Arencibia (Eds.), Computer Aided Systems Theory - EUROCAST 2009, Springer Berlin Heidelberg, Berlin, Heidelberg, 2009, pp. 938–946.

[151] M. Garza-Fabre, C. A. C. C. Gregorio Toscano-Pulido, E. Rodríguez-Tello, Effective Ranking + Speciation = Many-Objective Optimization, in: 2011 IEEE Congress on Evolutionary Computation (CEC'2011), IEEE Service Center, New Orleans, Louisiana, USA, 2011, pp. 2115–2122.

[152] R. Jiao, S. Zeng, C. Li, S. Yang, Y. S. Ong, Handling constrained many-objective optimization problems via problem transformation, IEEE Transactions on Cybernetics (2020) 1–14`doi:10.1109/TCYB.2020.3031642`.