Journal of Parallel and Distributed Computing 126 (2019) 13–33

Contents lists available at ScienceDirect

J. Parallel Distrib. Comput.

journal homepage: www.elsevier.com/locate/jpdc

Parallel multi-swarm PSO strategies for solving many objective optimization problems

Arion de Campos Jr. [a], Aurora T.R. Pozo [b], Elias P. Duarte Jr. [b,*]

[a] Department of Informatics, State University of Ponta Grossa (UEPG), Ponta Grossa, Brazil
[b] Department of Informatics, Federal University of Parana (UFPR), Curitiba, Brazil

HIGHLIGHTS

- This work presents parallel strategies for Particle Swarm Optimization (PSO) based on multiple swarms to solve Many-Objectives Problems.
- The multiple swarms co-evolve in parallel and interact by means of migration, different policies for triggering migration are proposed and evaluated.
- An extensive experimental evaluation of the algorithms is presented, in particular we investigated the impact of the proposed methods on the convergence and diversity of PSO search in many objective scenarios.

ARTICLE INFO

Article history:
Received 29 September 2017
Received in revised form 29 October 2018
Accepted 20 November 2018
Available online 3 December 2018

Keywords:
Evolutionary computing
Multi-swarm PSO
Many-objective optimization problems

ABSTRACT

In this work we present two parallel PSO strategies based on multiple swarms to solve MaOPs (Many-Objective Optimization Problems). The first strategy is based on Pareto dominance and the other is based on decomposition. Multiple swarms execute on independent processors and communicate using broadcast on a fully connected network. We investigate the impact of using both synchronous and asynchronous communication strategies for the decomposition-based approach. Experimental results were obtained for several benchmark problems. It is possible to conclude that the parallelization has a positive effect on the convergence and diversity of the optimization process for problems with many objectives. However, there is no single strategy that is the best results for all classes of problems. In terms of scalability, for higher numbers of objectives the parallel algorithms based on decomposition are always either the best or present comparable results with the Pareto approach. There are exceptions, but only when the problem itself has discontinuities on the Pareto Front.

1. Introduction

Multi-Objective Optimization Problems (MOPs) [15] involve the simultaneous optimization of two or more objectives. MOPs present several challenges when the number of objectives is greater than three and in this case they are called Many-Objective Optimization Problems (MaOPs) [1]. Optimization problems with multiple objectives usually do not have a single best solution, as frequently the objectives present conflicts among themselves. The goal is thus to find a good set of "balanced" solutions, i.e. a set of solutions that present a good trade-off taking into consideration all the objectives.

Particle Swarm Optimization (PSO) [32] has been successfully applied to solve multi-objective problems [15,31,45]. PSO is a population-based stochastic optimization technique inspired on the social behavior of groups of animals. Multi-Objective PSO (MOPSO) techniques are based on Pareto dominance, decomposition, indicators, and reference points. In this work we focus on Pareto dominance and decomposition. Strategies based on Pareto dominance usually employ an external archive (or repository) to store non-dominated solutions found throughout the optimization. As the process evolves and reaches new solutions, it is checked whether they can be used to update the repository. A solution can be used if it is non-dominated with respect to the solutions already in the repository [13].

Strategies based on decomposition follow a different approach. In order to pursue an approximation to the Pareto front, the MOP is decomposed into a number of scalar mono-objective subproblems [8,66]. Methods for constructing aggregation functions can be adopted in which the collective objective is an aggregation of all functions. In this case, there is a single best individual for each subproblem. Several PSO approaches based on decomposition have been proposed [3,46,47,64]. Although these strategies represent significant contributions to the field, most do not scale when the number of objectives grow, i.e. they usually cannot be efficiently applied to solve problems with *many* objectives.

* Corresponding author.
E-mail addresses: arion@uepg.br (A.d. Campos), aurora@inf.ufpr.br (A.T.R. Pozo), elias@inf.ufpr.br (E.P. Duarte).

https://doi.org/10.1016/j.jpdc.2018.11.008
0743-7315/

In this work we propose parallel strategies to solve MaOPs using multiple swarms that run on independent but interconnected processors. The multiple swarms co-evolve in parallel and interact by means of policies for migrating solutions. Communication among swarms is based on broadcast. We propose two parallel PSO strategies for solving MaOPs: the first is based on the Pareto dominance and the second on decomposition. The first strategy employs a set of repositories of solutions (particles), each of which is kept independently by each swarm. At each step, new nondominated solutions of each swarm are added to the local repository. Swarms exchange new nondominated solutions asynchronously. In the second solution each swarm evolves a number of scalar subproblems and shares best individuals (particles) with the other swarms. This strategy relies on mechanisms employed by mono-objective optimization for the migration of individual particles [9].

Experimental results are presented for the DTLZ family of many-objective benchmarking problems [19]. We executed experiments in which the proposed parallel algorithms were applied to problems with 2, 3, 5, 10, 15, and 20 objectives. In order to obtain reference baseline results, each problem was first solved using two sequential optimization algorithms, the first based on Pareto dominance and the other on decomposition. The results were evaluated using a set of quality indicators and were statistically analyzed. We investigated how the proposed methods impact the convergence and diversity of MOPSO search in many objective scenarios. We also present a comparison with the NSGA-II algorithm [18] and MOEA/DD. Furthermore, considering the obtained results, we report an analysis of the overall performance of the algorithms. Results show that there is no single best solution for every case. Depending on the characteristics of the problem to be solved, different optimization strategies present superior performance.

The remainder of this paper is organized as follows. Section 2 presents an overview of particle swarm optimization, including strategies for solving many objective problems and those based on multiple swarms. Section 3 describes our proposed strategies for running multi-swarm PSO to solve MaOPs. Section 4 presents our empirical evaluation, and includes descriptions of the simulation environment, benchmark functions, parameter settings and results, including a comparison with the NSGA-II algorithm. Finally, Section 6 concludes the paper and presents future research directions.

2. Multi-Objective PSO (MOPSO)

Particle Swarm Optimization (PSO) is a population-based stochastic optimization technique developed by Eberhart and Kennedy in 1995 [32]. PSO was developed inspired by the social behavior of groups of animals, like flocks of birds and schools of fish. PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithm (GA): the system is initialized with a population of random solutions and searches for optima by updating generations. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In the PSO algorithm each individual is called a particle and behaves like a bird (or fish) in the flock (school) searching for food or fleeing from a predator. Each particle has a position and a velocity used to explore the search space of the problem. A position represents a potential solution to the problem. A particle learns from its own experiences (cognitive component), and also learns from the group (social component). Usually, the individual and social components are called *pbest* (personal best) and *gbest* (global best) respectively. In order to evaluate a solution, PSO iteratively computes the fitness for each particle.

In the PSO algorithm, at each step after finding the *pbest* and *gbest*, every particle updates its velocity and position. Thus the particle "moves" in the search space. Each particle i has a position $\vec{x}(t) \in \Re^n$ at a given time t. This time t corresponds to an iteration. The new position of particle i at time $t+1$ (next iteration), is computed by adding the velocity ($\vec{v}(t) \in \Re^n$) to the position $x(t)$, as shown in Eq. (1).

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \tag{1}$$

The velocity of particle i at time $t+1$ is updated according to Eq. (2).

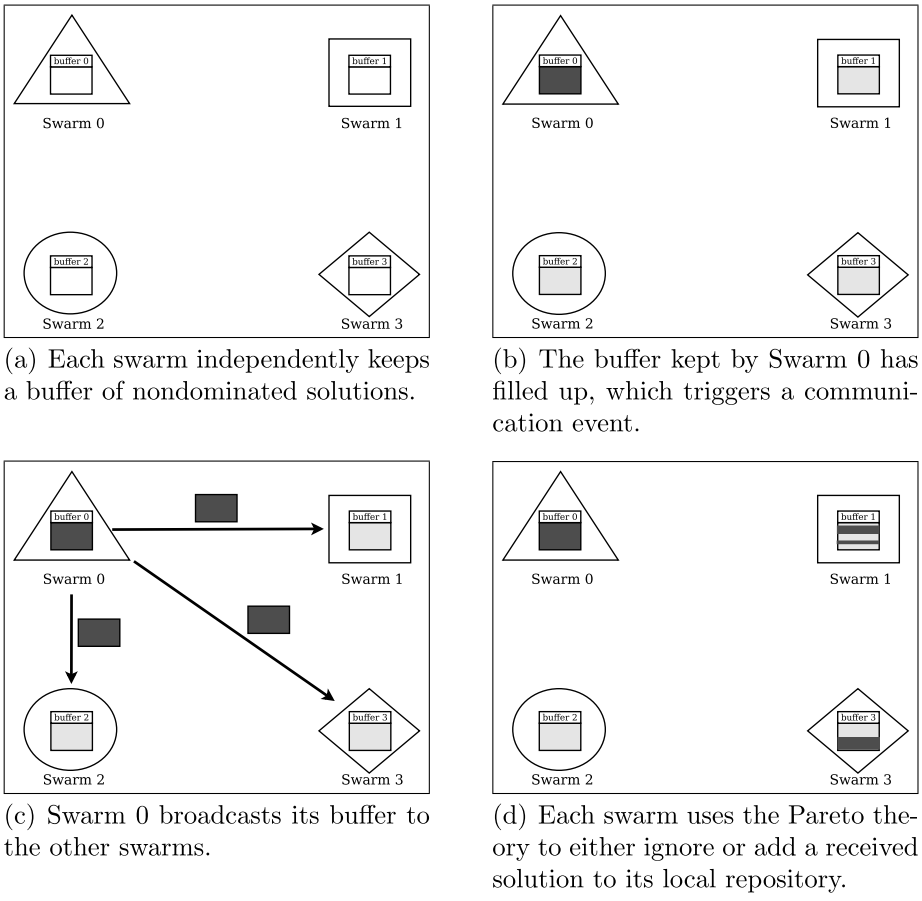
$$\begin{aligned}\vec{v}_i(t+1) = \varpi \cdot \vec{v}_i(t) + C1 \cdot rand() \cdot (\vec{pbest}_i(t) - \vec{x}_i(t)) \\ + C2 \cdot rand() \cdot (\vec{gbest}(t) - \vec{x}_i(t))\end{aligned} \tag{2}$$

In Eq. (2), C_1 and C_2 are constants that define the learning factor and *rand*() is a random number between 0 and 1. $\vec{v}_i$ is the particle velocity; $\vec{pbest}_i$ is the best solution found by particle i itself; $\vec{x}_i$ corresponds to the position of the particle being manipulated, and *gbest* is the best solution found by the whole group of particles. w denotes the inertia weight, i.e. the tendency of a particle to keep the same velocity. At the end of a number of iterations or some other stop criterion, the best solution is presented as the result.

Although PSO was originally proposed to solve single objective problems (also called *mono*-objective problems) [6], PSO strategies to solve multi-objective problems (MOPs) [15] have also been proposed and are called MOPSO (Multi-Objective PSO). Multi-objective problems usually do not have a single best solution. The goal is to find a good set of "balanced" solutions, i.e. a set of solutions that present a good *trade-off* taking into consideration all the objectives. Since problems with multiple objectives have a set of optimal solutions, in order to solve those problems the PSO algorithm must be modified by incorporating a selection mechanism based on Pareto optimality and also adopting a diversity preservation mechanism that avoids the convergence to a single solution [13].

In order to allow MOPSO to obtain balanced solutions a popular approach is to use an external repository or archive. This repository stores nondominated solutions found by the optimization process and keeps Pareto solutions generated up to a certain generation. As the process evolves, new solutions are devised and considered for updating the repository: each new solution must be analyzed for nondominance with respect to the solutions currently in the repository [13]. Furthermore, each particle in the population must employ a leader selection method to choose one of the archive members as its *gbest*. The sigma distance has been successfully used to select the global best particle, as has the Crowding Distance [7] to manage the archive size [10]. A mechanism for constraining the accumulated velocity in each dimension can also be used [13].

Another different strategy is adopted by the dMOPSO algorithm proposed by Martinez and Coello [64]. This algorithm decomposes a MOP into N mono-objective optimization problems. These subproblems are solved simultaneously by evolving a population of solutions. In each generation, the population is composed by the best solution found so far. Thus, a set of approximate solutions to the Pareto optimal front is reached by optimizing each subproblem (scalar aggregate function) individually, instead of using the Pareto dominance relationship. This approach uses a weighted vector to define a scalar function that defines a search direction. There are several methods for constructing such functions, dMOPSO uses the Penalty Boundary Intersection (PBI) approach proposed by Zhang and Li [66]. For each cycle, the best positions of each particle are employed to find the best solutions for each subproblem. Therefore, the set of best global solutions *gbest* naturally emerges. This set contains the solution that minimizes each subproblem and is updated at each cycle accordingly. Reported results [64] indicate that dMOPSO is a competitive algorithm that outperforms others in several benchmark tests.



(a) Each swarm independently keeps a buffer of nondominated solutions.

(b) The buffer kept by Swarm 0 has filled up, which triggers a communication event.

(c) Swarm 0 broadcasts its buffer to the other swarms.

(d) Each swarm uses the Pareto theory to either ignore or add a received solution to its local repository.

Fig. 1. Multiple swarms co-evolve in parallel and interact by means of policies for migrating buffers of nondominated solutions.

3. The proposed multi-swarm MOPSO strategies

In this section we describe the proposed parallel strategies based on multiple swarms that cooperate to solve problems with many objectives. By adopting multiple processors to run the swarms independently, the optimization process is executed in parallel. This represents a feasible strategy for optimizing functions with a high number of objectives [16,57]. We assume that swarms are running on independent processors on a fully connected network. Thus, each swarm can communicate directly with any other swarm and there is no need to employ intermediates in this communication. However, each population independently performs the optimization process, in an attempt to prevent the convergence to suboptimal solutions [25,42].

The two MOPSO strategies are presented next. They are based on (1) the Pareto dominance relationship; and (2) decomposition.

3.1. Multi-Swarm MOPSO based on Pareto dominance

In the proposed strategy based on Pareto dominance the multiple swarms share buffers of new non-dominated solutions. Swarms execute Algorithm 1, which calls the procedures specified in Algorithm 2 for all strategies. Each swarm evolves its particles using the global leader chosen in the local repository. If a swarm improves its results and updates the local repository then the buffer of solutions is updated. When this buffer is full, a communication event is triggered and the buffer is broadcast to the other swarms. Based on the Pareto theory, the destination swarm compares the received buffer with its own solutions kept in its local repository. Each solution in the buffer can be in one of the following situations (also illustrated in Fig. 1):

1. a received solution **dominates** a local repository solution → the received solution is saved in the repository and all dominated solutions are deleted from the repository;
2. a received solution **is dominated** by a local repository solution → the received solution is ignored;
3. there is **no dominance relationship** among solutions → if the destination repository is not full, the received solution is inserted. If the repository is full, then the Crowding Distance is used to decide which solutions are kept.

Thus the MOPSO strategy based on Pareto dominance is defined as follows:

(a) The information exchanged is a "buffer": a buffer of the nondominated solutions (particles) is kept independently by each swarm. At each step, new nondominated solutions of the swarm are added to the buffer. When this buffer is full it is broadcast to the other swarms and reset as empty.

(b) The communication strategy is asynchronous: the buffer kept by each swarm is shared only when it is full. Thus, we avoid excessive messages. In our case we employed a buffer size that is determined by the swarm population size.

(c) Broadcast is employed to share information: the nondominated solutions (kept in the buffer) are sent from a swarm to all other swarms through the network.

(d) Updating the repository: a buffer received from another swarm is evaluated taking into account the set of nondominated solutions kept at the local repository. If the received particles are nondominated solutions, then the local repository is updated.

The optimization completes when the number of iterations executes reaches a predefined maximum value. The same criterion was adopted for the baseline executions.

3.2. Multi-swarm MOPSO based on decomposition

The proposed strategy is based on decomposition sharing the best particles for each subproblem. Swarms execute Algorithm 1 and the procedures in Algorithm 2. Swarms evolve their particles using the global leader (*gbest*). The global leaders in turn are updated taking into account the best particles. Two communication policies are employed. In the *synchronous* communication policy, at each iteration every swarm shares its local *gbest* with the other swarms. In the *asynchronous* policy, at each iteration each swarm checks whether the local *current gbest* has been modified (improved) and in this case the communication occurs. Thus, at each iteration, a communication event occurs only if the best solution associated with each subproblem is new. This strategy is illustrated in Fig. 2 and is described next.

Algorithm 1 Parallel Multi-Swarm Algorithm

```
for each swarm i do
  run Initialize-swarm; // in Algorithm 2
repeat
  for each swarm i do in parallel
    run Swarm-Update // in Algorithm 2
    run Swarm-Exchange // in Algorithm 2
  end for
until until the maximum number of iterations is reached
run Output-Solutions // in Algorithm 2
```

The MOPSO strategy based on decomposition is defined as follows:

- **(a) The information exchanged among swarms consists of the set of best solutions from each swarm:** each swarm maintains a *gbest* for each subproblem. At each step, new solutions are generated from the evolution of the particles. For each subproblem, the particle with the best fitness value is defined as the corresponding *gbest* for the subproblem.
- **(b) Communication strategies: (a) synchronous**: at each iteration the set of best solutions (one *gbest* associated with each subproblem) is shared with the other swarms; **(b) asynchronous:** the *gbest* of each subproblem is sent only if it has improved.
- **(c) Broadcast is employed to share information**: the set of *gbest's* is sent from each swarm to all other swarms through the network;
- **(d) The *gbest* of each subproblem is updated:** the set of solutions received from other swarms is evaluated by checking each *gbest* associated with each subproblem. If the incoming particles represent solutions with better fitness, then for each subproblem, the corresponding *gbest* is updated.

The next section presents experimental results of the performance of the proposed multi-swarm optimization strategies to solve MaOPs.

4. Empirical evaluation

This Section presents the evaluation of the proposed parallel PSO strategies. In the first subsection we present the quality indicators and statistical tests employed. In the next subsection parameters and the methodology we adopted are described. The comparison between the sequential and parallel approaches is presented in the third subsection. Finally, we compare the parallel strategies with the NSGA-II algorithm.

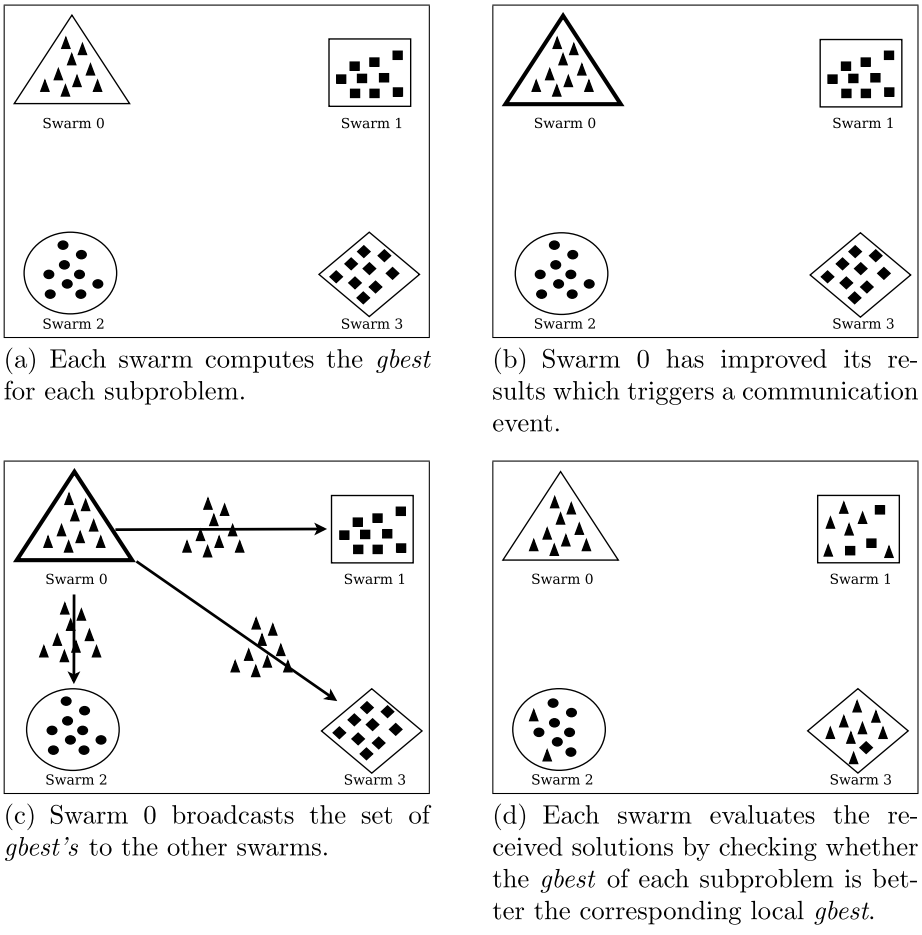
Algorithm 2 The Procedures Defined for All Strategies

```
Initialize-Swarm:
  Strategy Based on Pareto Dominance
      Initialize particles, pbest, the repository, and the buffer;
  Strategy Based on Decomposition
      Initialize particles, pbest, gbest, and the weight vectors for the
  subproblems;
Swarm-Update:
Strategy Based on Pareto Dominance
      select the gbest from repository i;
      given the gbest, update the particle;
      evaluate the particle;
      update pbest;
      depending on the Pareto dominance:
          update the repository and the buffer;
Strategy Based on Decomposition
      update particle;
      evaluate particle;
      update pbest and gbest for each subproblem;
Swarm-Exchange:
  Strategy Based on Pareto Dominance
       if buffer of swarm i is full
            then a communication event is triggered:
  broadcast(buffer i)
            empty buffer i
       when a buffer is received:
       depending on the Pareto dominance
            update the repository
  Strategy Based on Decomposition and Asynchronous
       if gbest was improved
            then a communication event is triggered:
  broadcast(gbest)
  Strategy Based on Decomposition and Synchronous
       at each iteration
            share one gbest from each subproblem with the other
  swarms
       when a gbest if is received:
            if incoming particles represent solutions with better
  fitness
                 then update the local gbest
Output-Solutions:
Strategy Based on Pareto Dominance
      return all repositories
Strategy Based on Decomposition
      return the gbest of all subproblems
```

Five algorithms are compared, these algorithms are referred to as follows:

1. RefPar: sequential reference algorithm, based on dominance among solutions;
2. RefDec: sequential reference algorithm, based on MOP decomposition into single-objective subproblems;
3. BPar: parallel algorithm, based on dominance among solutions;
4. BDec: synchronous version of the parallel algorithm, based on MOP decomposition into single-objective subproblems;
5. AsBDec: asynchronous version of the parallel algorithm, based on MOP decomposition into single-objective subproblems.

In order to evaluate the performance of the proposed algorithms we employed the full set of seven benchmark problems of the well-known DTLZ MaOPs family [19]. These benchmark problems present several features that allow a comprehensive evaluation of MOEAs. Among these features we can highlight: their



(a) Each swarm computes the *gbest* for each subproblem.

(b) Swarm 0 has improved its results which triggers a communication event.

(c) Swarm 0 broadcasts the set of *gbest's* to the other swarms.

(d) Each swarm evaluates the received solutions by checking whether the *gbest* of each subproblem is better the corresponding local *gbest*.

Fig. 2. Multiple swarms co-evolve in parallel and interact by means of policies for migrating best solutions.

design is clear and simple; they present scalability both in terms of the number of decision variables (n) and the number of objectives (M); and it is also possible to adjust the level of difficulty to obtain the true Pareto front. The algorithms were used to solve problems with 2, 3, 5, 10, 15 and 20 objectives.

The DTLZ family is composed of seven problems with distinct properties.[1] For each problem, the true Pareto front is known and the search complexity is expressed by variable k. The size of the vector solution is $n = M + k - 1$, where M is the number of objectives. DTLZ problems are minimization functions in the form $\min f(\overrightarrow{X})$. In these problems, $\overrightarrow{X_M}$ is a vector with size $|\overrightarrow{X_M}| = k$ containing the last values of $\overrightarrow{X}$. Results must be evaluated using quality indicators both in terms of convergence and diversity.

The implementation of PSO with multiple swarms for solving MaOPs from the DTLZ family was done using the SMPL (SiMulation Programming Language) [43]. SMPL is a simulation library for the **C** programming language based on events. In the following subsection we describe the quality indicators and statistical tests adopted.

[1] Actually, the number of DTLZ problems varies according to different authors. For instance, Huband et al. [28], state that there are nine DTLZ problems specified in the original technical report. A more recent conference paper version of the technical report only specifies seven problems. Namely, problems DTLZ6, DTLZ7 and DTLZ8 from the original technical report are renamed as DTLZ5, DTLZ6 and DTLZ7, in the conference paper (DTLZ5 and DTLZ9 from the original report were dropped).

4.1. Quality indicators & statistical tests

The purpose of the empirical evaluation is to investigate the performance of the proposed strategies in terms of convergence and diversity, as well as the scalability with respect to the number of objectives. It is not trivial to choose metrics to evaluate and in particular to compare results obtained from the execution of algorithms for solving MaOPs [1]. We employed a set of indicators that encompass different aspects of the behavior of the algorithms [70]. These quality indicators are described next.

The **Generational Distance**$_p$(GDp) [52] measures the convergence of the solution set and determines how far the Pareto-obtained front is from the Pareto-optimal (real Pareto) front. The **Inverse Generational Distance**$_p$ (IGDp) [52] measures how far the Pareto-optimal front is from the Pareto-obtained front (this is useful when the points obtained are good solutions but do not cover the entire front). **Spacing** [51] gives a measure of the range of the variance between neighboring solutions in the front (zero indicates that all solutions are equally distributed in the objective space). Finally, the **hypervolume** proposed by Zitzler [68] considers the volume of the covered area by the solutions in the objective space. Results with lower values are better for GDp, IGDp and the spacing indicators; for the hypervolume, higher values are better.

There are several statistical tests that can be employed to evaluate the performance of MOEAs [14]. In this work, we employed the Friedman statistical test [20] and the Wilcoxon test [24].

Friedman is a nonparametric test employed to assess differences among several related samples. The null hypothesis for the

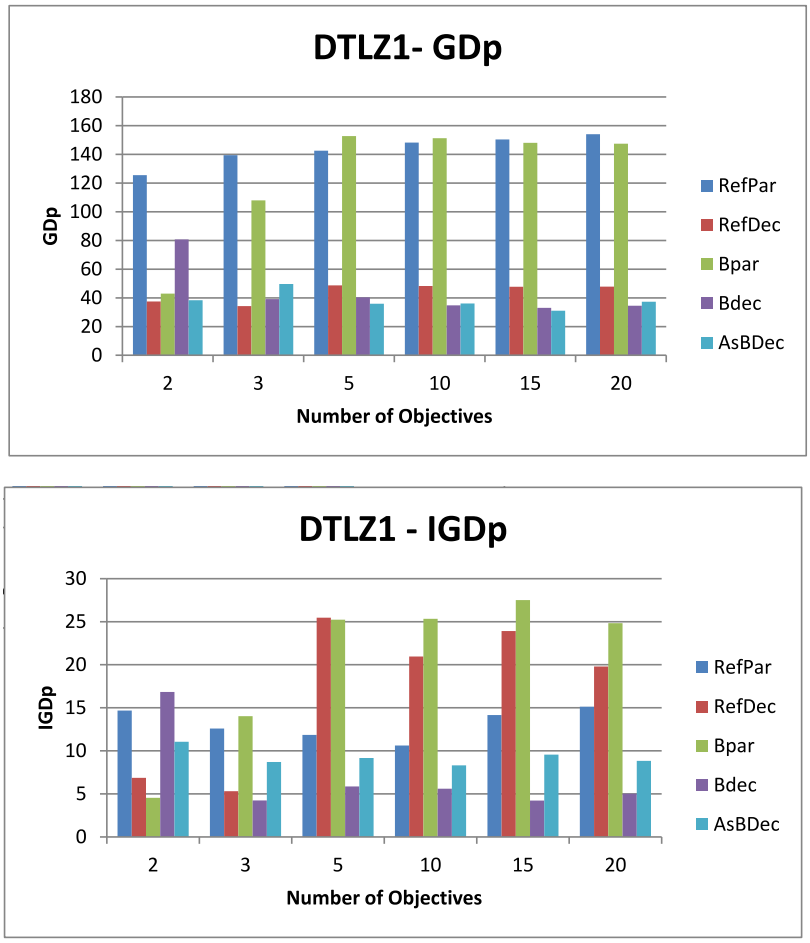


Fig. 3. Average results for the GDp and IGDp metrics for the DTLZ1 problem. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Friedman test is that there are no differences between the results. If the calculated probability is lower than the significance level, the null-hypothesis is rejected. In this case, it can be concluded that at least two of the results are significantly different from each other. If the null-hypothesis is rejected, post-hoc tests for pairwise multiple comparisons are employed to determine which algorithm presents better performance. Using this test it is possible to ensure that differences observed in the data are not casual.

Wilcoxon also is a nonparametric test. However it is a test that compares two paired groups. The test essentially calculates the difference between each set of pairs and analyzes these differences. We use this test specifically to compare the obtained results between synchronous and asynchronous versions of our parallel version of MOPSO based on decomposition. The hypothesis test examines two opposing hypotheses about the values: the null-hypothesis states that there is no statistical difference between the performance of the two versions. On the other hand, the alternative hypothesis states the opposite.

4.2. Parameters & methodology

All algorithms were compared: the proposed parallel strategy based on Pareto dominance, the parallel versions of the decomposition-based strategy (synchronous and asynchronous) and the sequential algorithms. We employed the sigma method for selecting leaders [45] and repository management was based on the CD (Crowding Distance) [50].

For all configurations, each particle consisted of n randomly initialized values in the scope of each function. The total number of variables is $n + k = M - 1$, where M is the number of objectives, 2, 3, 5, 10, 15 and 20 objectives were evaluated. As suggested by Deb et al. [19], for DTLZ1, we set $k = 5$. For all other problems, $k = 10$. The other parameters were set as traditionally recommended in the literature [21]. The inertia weight (w) was initialized at 0.9 and was gradually lowered to 0.4. Constants C1 and C2 were set at 2. The *pbest* is updated only when a better solution is found, following [45].

The sequential algorithms (Pareto MOPSO and dMOPSO) employed 256 particles for DTLZ1 and DTLZ3 and 128 particles to solve the other problems. Although 128 particles proved to be enough for most of the problems, for the DTLZ1 and DTLZ3 problems it was necessary to use a larger number of particles to better represent the obtained Pareto fronts.

Each parallel algorithm was executed with eight swarms. In a previous work [9] we have shown that this number of swarms produced the best results. The total number of particles was distributed among the 8 swarms as follows: 32 particles per swarm for the DTLZ1 and DTLZ3 functions; 16 particles per swarm for the other functions. In both sequential and parallel versions the size of the repository for the algorithm based on Pareto dominance was constant, equal to the number of particles employed to solve each problem. For the algorithms based on decomposition, the number of subproblems was constant, also equal to the number of particles employed to solve each function.

The weights are initialized as follows:

- S is the number of swarms, N the number of particles on each swarm, M is the number of objectives of the MOP;

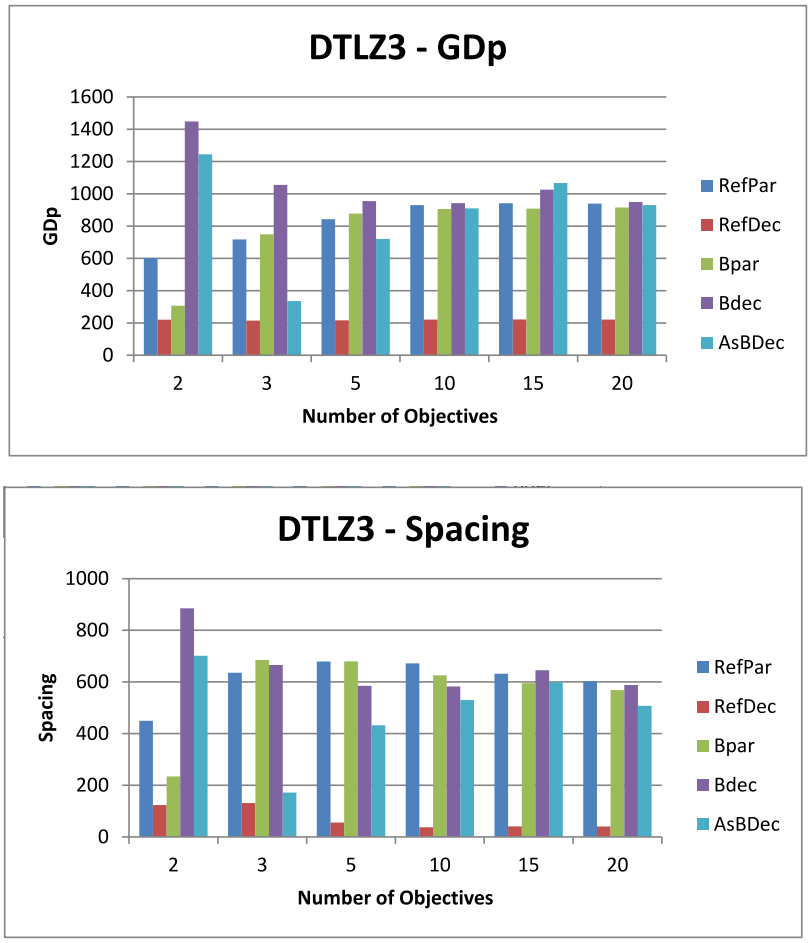


Fig. 4. Average results for the GDp and spacing metrics for the DTLZ3 problem.

- $W = S * N$ uniformly distributed weights are generated: $W_i = (w_j, \ldots, w_M)$, for $i = 1, \ldots, W$ and $j = 1, \ldots, M$. Satisfying the restrictions $w_j \geq 0$ and $\sum_{j=1}^{M} w_j = 1$;
- The weight vectors are sorted in ascending order considering the objective M. After, every swarm receives N different weight vectors. Observe that the weights used by the swarms are different.

When the communication strategy is asynchronous, the buffer is updated whenever a better solution is found; however, instead of sending messages every time this happens, a message is sent only when the buffer is full. Thus, we avoid excessive messages. In our case the buffer size is equal to the swarm population size.

Both parallel and sequential algorithms were executed for up to 100 iterations. The number of iterations was selected according to earlier results that confirm the quick convergence of PSO [9,12,54]. Other parameters, including the population sizes, were also chosen based on those earlier results.

Table 1 summarizes the parameters.

Next, we present and discuss the results for each benchmark function and compare the results obtained by the five algorithms. Each algorithm was executed for 30 independent runs for each number of objectives. 30 different seeds were employed for the 30 independent runs, thus minimizing the possibility that the algorithms produce good or bad results due to the initial state. For each execution the final content of the repository or the set of *gbest* was stored. As shown below, we have computed the average (A) and standard deviation (SD) computed from samples obtained from 30 runs of each function. In order to compare the results, the Friedman test was applied with a significance level of 5%. The R toolkit[2] for statistical computing was employed. When the test shows a significant statistical difference, after post-hoc tests for pairwise multiple comparisons, the best averages are highlighted in bold.

4.3. Results

In this section we present results obtained from the execution of all algorithms for DTLZ1 (Table A.4), DTLZ2 (Table A.5), DTLZ3 (Table A.6), DTLZ4 (Table A.7), DTLZ5 (Table A.8), DTLZ6 (Table A.9), and DTLZ7 (Table A.10). As mentioned above, the performance indicators consisted of the following metrics: GDp, IGDp, spacing and hypervolume.[3] The figures that show the hypervolume were normalized based on the maximum hypervolume value for each number of objectives. These Tables are presented in Appendix A.

The DTLZ1 problem is characterized by having a large number of local Pareto fronts. These tend to attract suboptimal solutions before they reach the real Pareto front.

For 2 and 3 objectives, the RefDec algorithm is the one that stands out with the best results for all indicators. However, according to the statistical tests for all cases, at least one parallel version has shown competitive results. For 5, 10, 15 and 20 objectives,

[2] http://www.r-project.org/.

[3] According to Yang and Ding [61], the runtime of this metric is exponential on the dimension of the objective space. Hence, due to time constraints, for DTLZ's [1,2,3,4,6], we obtained hypervolume indicators from 2 to 15 objectives, while for DTLZ's [5,7] we obtained hypervolume indicators from 2 to 10 objectives.

Table 1
Empirical evaluation: parameters (NA: does Not Apply).

	DTLZ1 (k=5) and DTLZ3 (k=10)			DTLZ[2,4,5,6,7] (k=10)		
Algorithm	Particles	Rep. size	Subproblems	Particles	Rep. size	Subproblems
RefPar	256	256	NA	128	128	NA
RefDec	256	NA	256	128	NA	128
BPar	32/swarm	32	NA	16/swarm	16/swarm	NA
BDec	32/swarm	NA	32/swarm	16/swarm	NA	16/swarm
AsBDec	32/swarm	NA	32/swarm	16/swarm	NA	16/swarm

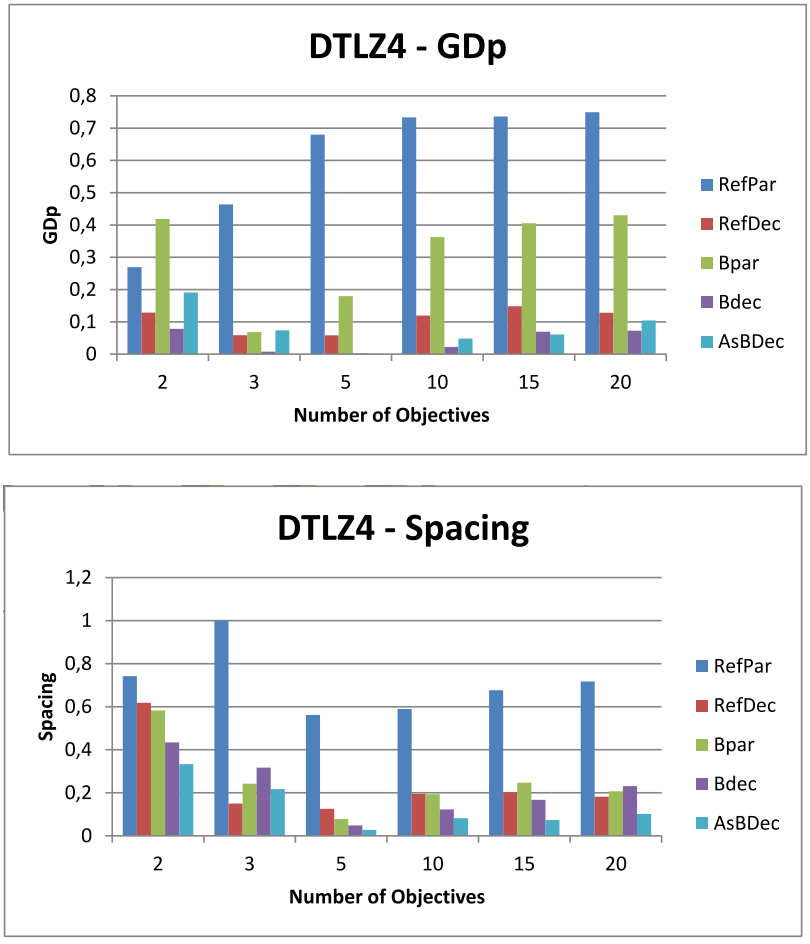


Fig. 5. Average results for the GDp and spacing metrics for the DTLZ4 problem.

parallel versions of the algorithms based on decomposition present both good convergence and diversity, according to the GDp and IGDp metrics. The same happens to the hypervolume metric for the BDec algorithm.

Fig. 3 illustrates the average results obtained by the algorithms with respect to the GDp and IGDp indicators. Considering the scalability of algorithms, it is possible to observe that the parallel versions based on decomposition (green and brown lines) are the only strategies that have the best average performance for both metrics. A first look at the results obtained for DTLZ1 leads to the conclusion that no algorithm is clearly superior to the others for all performance indicators and every number of objectives. However, it is possible to claim that algorithms based on decomposition perform better than those based on dominance relationships.

The DTLZ2 problem is suitable for assessing the convergence of the algorithms as the number of objectives increases. The indicators show that the parallel versions based on decomposition have good scalability, especially the BDec algorithm. For any number of objectives, this algorithm presented good results for GDp, IGDp and hypervolume, simultaneously. Results for the spacing metric should be taken with caution, as this metric can produce good diversity results for approximation fronts that are far from the true front. The sequential version based on dominance relationship (RefPar) showed the lowest results for nearly all indicators for every number of objectives.

By combining the characteristics of both DTLZ1 and DTLZ2, the DTLZ3 problem is a good choice to investigate the ability of the algorithms converge to the real Pareto front. For two objectives, both the BPar and RefDec algorithms stand out for all indicators. However, as the number of objectives increases, algorithms based on decomposition present superior performance in terms of both convergence and diversity. It is evident that for this problem the perceived superiority of this type of algorithm is due to the decomposition of the MOP into subproblems each of which is solved by a single large swarm.

In Fig. 4 it is possible to observe that the RefDec algorithm presents good scalability, as its performance remains good as the

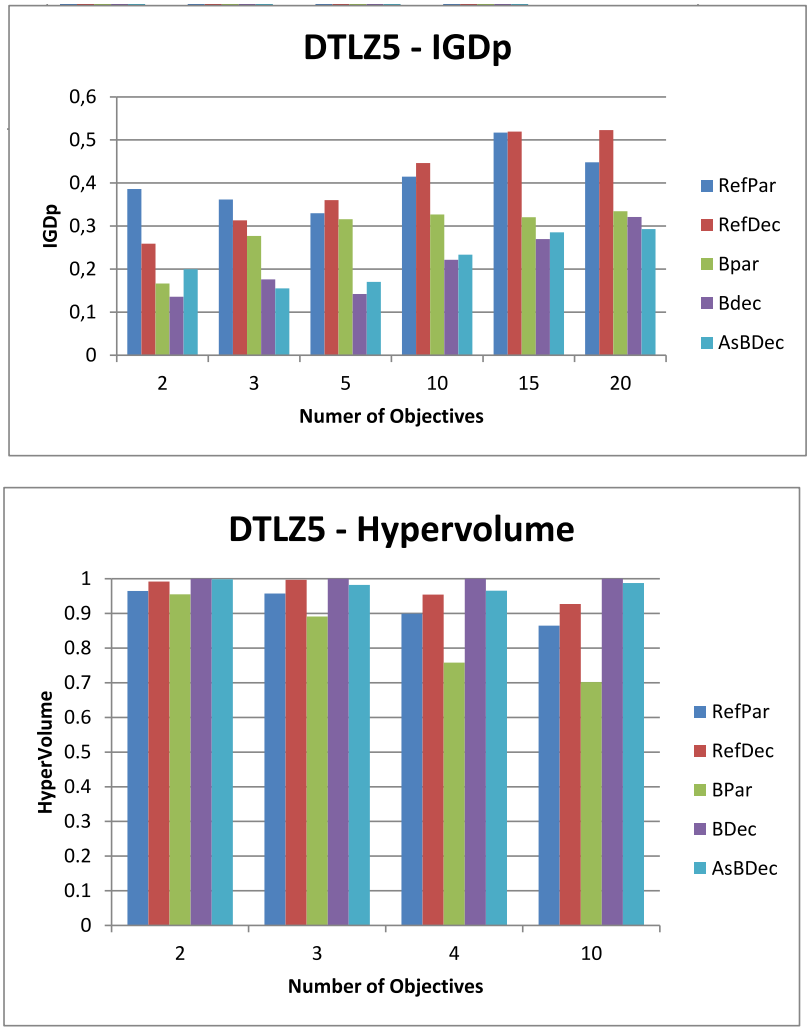


Fig. 6. Average results for the IGDp and hypervolume metrics for the DTLZ5 problem.

number of objectives increases. Results present not only good convergence according to the GDp metric and also feature well-distributed solutions along the front, as indicated by the spacing metric.

The DTLZ4 problem tends to generate points in a particular region of the Pareto front. For this reason, it is a good choice to evaluate the distribution of the solutions. For two objectives, the parallel and synchronous algorithm based on decomposition (BDec) performs well for all indicators. A similar performance is observed for the sequential algorithm based on decomposition, although the spacing metric indicates that the solutions do not show good diversity. As the number of objectives increases, the versions based on decomposition yield the best results in terms of convergence and distribution (according to the GDp and spacing metrics, respectively). The performance of these parallel versions is shown in Fig. 5.

For the dominance-based algorithms, the parallel version (BPar) is superior than the sequential version (RefPar) for the hypervolume metric for two objectives. However, as the number of objectives increases, this relationship is inverted.

For the DTLZ5 problem, the real Pareto front is characterized as the border of a spherical unit, a densely populated curve in a M-dimensional space of objectives. Considering the GDp indicator, it is evident that the parallel algorithm based on Pareto dominance (BPar) is superior, this is confirmed by the statistical test. A similar conclusion can be drawn for the spacing metric. The RefPar algorithm presents good results in some cases.

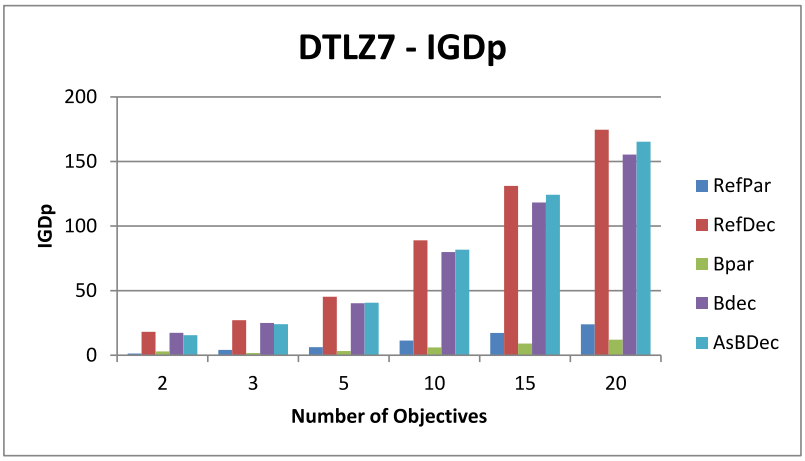
Considering the IGDp indicator, the performance of the parallel algorithms based on decomposition proved to be good, although the parallel algorithm BPar also presented good results for 2, 15 and 20 objectives. Fig. 6(a) shows that the performance achieved by the parallel versions is the best of all alternatives, even as the number of objectives increases.

In terms of the convergence (Table A.8), for this problem the parallel algorithms are clearly superior. For GDp, BPar presents the best results. Considering the hypervolume metric, it can be seen from Fig. 6(b) that the parallel algorithms based on decomposition present good performance for many objectives.

The DTLZ6 problem is a variation of the previous problem and prioritizes the evaluation of the convergence of the algorithms. Considering the GDp, IGDp and spacing metrics it is evident that the algorithms based on decomposition are superior, in particular the sequential version.

For this problem, we note that the strategy of decomposing the MOP into subproblems, but maintaining a single and large swarm is a good alternative. In terms of the hypervolume metric, it is difficult to pick an algorithm that stands out in relation to the others, especially as the number of objectives increases. The statistical test showed that for this metric, all algorithms have equivalent performance for 10 and 15 objectives.

Finally, the DTLZ7 problem shows the ability of algorithms to maintain individuals in different regions of the real Pareto front. For this problem, the algorithms based on dominance



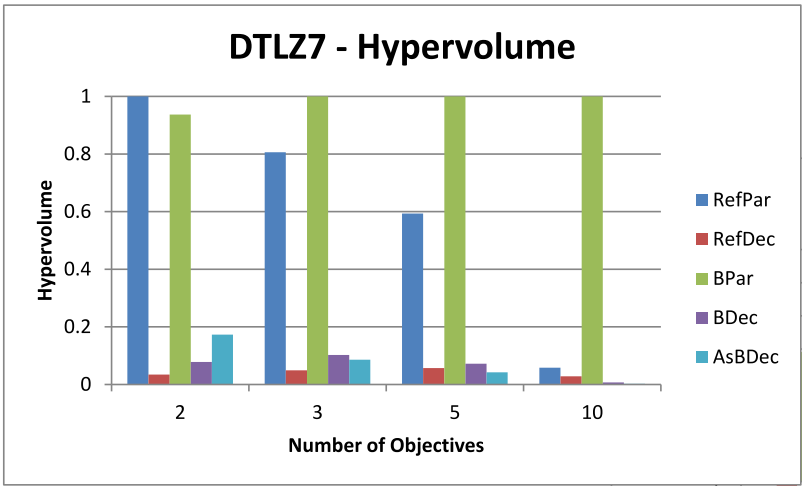


Fig. 7. Average results for the IGDp and Hypervolume indicators for the DTLZ7 problem.

Table 2
Wilcoxon test for communication strategies based on decomposition (HV=Hypervolume).

DTLZ1			DTLZ2					
Indicator	p-value	Better	Indicator	p-value	Better			
GDp	0.84	Async	GDp	0.56	Async			
IGDp	0.43	Sync	IGDp	0.56	Async			
Spacing	0.21	Async	Spacing	0.15	Sync			
HV	0.12	Async	HV	0.18	Sync			
DTLZ3			DTLZ4					
Indicator	p-value	Better	Indicator	p-value	Better			
GDp	0.15	Async	GDp	0.15	Sync			
IGDp	0.68	Sync	IGDp	0.84	Sync			
Spacing	**0.03**	**Async**	Spacing	**0.03**	**Async**			
HV	0.81	Sync	HV	0.06	Sync			
DTLZ5			DTLZ6			DTLZ7		
Indicator	p-value	Better	Indicator	p-value	Better	Indicator	p-value	Better
GDp	0.56	Async	GDp	0.31	Async	GDp	**0.03**	**Async**
IGDp	0.56	Sync	IGDp	0.43	Async	IGDp	0.31	Sync
Spacing	0.31	Sync	Spacing	0.06	Async	Spacing	**0.03**	**Async**
HV	0.12	Sync	HV	0.31	Sync	HV	0.87	Sync

relationships proved to be the best. Both the parallel version (BPar) and the sequential version (RefPar) have the best values for all metrics.

The performance of the algorithms for the DTLZ7 problem is shown in Fig. 7. This figure depicts the results obtained for the IGDp and hypervolume indicators. In both situations, it is noticeable that the performance of the BPar parallel algorithm is the best, especially for the hypervolume and when the number of objectives increases.

For this problem, it is clear that to decompose a MOP into subproblems that are solved by a single sequential swarm is a poor strategy.

Influence of the communication policy for MOPSO based on decomposition

We employed the Wilcoxon test with significance level equal to 5% and using the R statistics toolkit[4] to compare the performance of two communication policies.

Table 2 shows the obtained *p-value* reported by toolkit for each benchmark considering each indicator. Based on indicator results, the best communication strategy is selected. For instances that had a significant statistical difference according to the test, the communication strategy is highlighted in bold.

For DTLZ {1,2,3,5} problems, the test did not accuse any difference between communication strategies for any indicator. For DTLZ 3, 4, only the Spacing indicator showed significant advantage for the asynchronous communication method. For DTLZ 7, the asynchronous communication method outperforms the synchronous method for the GDp and Spacing indicators.

Table 3 summarizes the results. In this table, each time that the algorithm outperforms or equals the performance of the others according to one indicator, the corresponding cell value is increased by one. Thus, for instance if the value is equal to 4, this means that the algorithm has better values for all four indicators: GD, IG, Space and Hypervolume.

4.4. Comparison with NSGA-II and MOEA/DD

In this subsection, we compare our parallel approaches to NSGA-II and MOEA/DD algorithms. The same methodology presented in the last section was employed. The specific parameters of NSGA-II were set as follows. Crossover probability: 0.9, crossover distribution index: 20.0, crossover: SBX mutation probability: (1.0 / (number of variables)), mutation distribution index: 20.0, mutation: polynomial mutation and selection: binary tournament selection. For MOEA/DD the specific parameters were set as: Crossover probability: 1.0, crossover distribution index: 30.0, crossover: SBX mutation probability: (1.0 / (number of variables)), mutation distribution index: 20.0, mutation: polynomial mutation as described in [33].

The obtained results from the comparison with NSGA-II are shown for each problem: DTLZ1 (Table A.11), DTLZ2 (Table A.12), DTLZ3 (Table A.13), DTLZ4 (Table A.14), DTLZ5 (Table A.15), DTLZ6 (Table A.16), and DTLZ7 (Table A.17). The best averages are highlighted in bold.

For the DTLZ {1,2,3,5,6} problems, observing the results obtained (Figs. 8–10), it is possible to point out that NSGA-II presented better results for up to five objectives. But, when the number of objectives increases the parallel versions outperform NSGA-II. The only exception is DTLZ7 (Table A.17), where NSGA-II is the best algorithm for all numbers of objectives.

The obtained results from the comparison with MOEA/DD are shown in Table A.18 and Figs. 11 to 24. It can be seen that for the GDp metric, our algorithms are competitive for all problems, from DTLZ1 to DTLZ7. For the IGDp metric, our algorithms present competitive results for 4 of the DTLZ problems: DTLZ2, DTLZ3, DTLZ5 and DTLZ6. These results confirm that the parallel algorithms proposed are a competitive approach for solving many objective problems.

Table 3
Summary of the comparison results — sequential vs. parallel approaches.

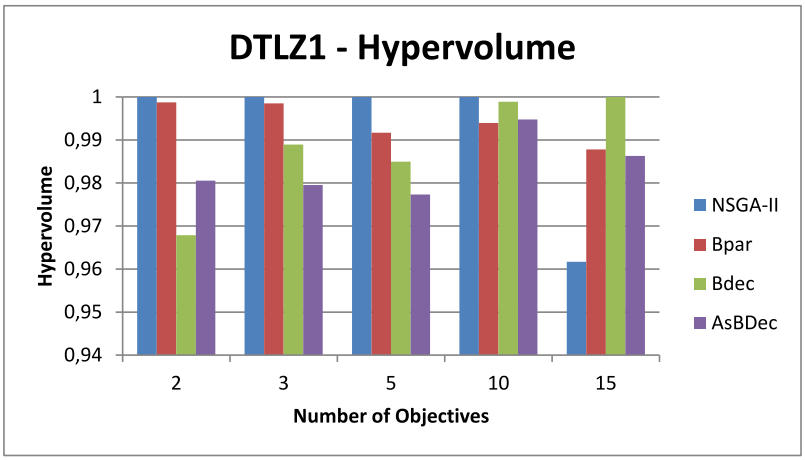
Test	DTLZ1				
# Obj.	RefPar	RefDec	BPar	BDec	AsBDec
2	1	4	3	0	2
3	1	4	1	3	2
5	1	2	0	2	3
10	2	2	0	2	2
15	1	3	0	3	2
20	0	2	0	2	2
Test	DTLZ2				
# Obj.	RefPar	RefDec	BPar	BDec	AsBDec
2	0	1	1	3	1
3	0	1	1	3	2
5	2	0	0	3	2
10	0	1	0	3	2
15	0	1	0	3	2
20	0	2	1	2	2
Test	DTLZ3				
# Obj.	RefPar	RefDec	BPar	BDec	AsBDec
2	0	4	4	0	0
3	0	4	0	0	2
5	1	4	0	0	0
10	0	4	0	1	0
15	0	4	0	0	0
20	0	3	0	0	0
Test	DTLZ4				
# Obj.	RefPar	RefDec	BPar	BDec	AsBDec
2	0	3	1	4	2
3	2	3	2	2	2
5	1	2	0	2	2
10	0	2	0	2	2
15	0	3	0	2	2
20	0	3	1	2	2
Test	DTLZ5				
# Obj.	RefPar	RefDec	BPar	BDec	AsBDec
2	0	0	3	2	2
3	0	1	2	1	2
5	1	0	2	2	1
10	1	0	2	2	2
15	1	0	3	1	1
20	1	0	3	1	1
Test	DTLZ6				
# Obj.	RefPar	RefDec	BPar	BDec	AsBDec
2	0	4	1	0	2
3	0	4	0	0	3
5	1	4	0	0	0
10	1	4	1	1	2
15	1	4	1	2	2
20	0	3	0	1	1
Test	DTLZ7				
# Obj.	RefPar	RefDec	BPar	BDec	AsBDec
2	3	1	3	0	0
3	3	1	2	2	2
5	3	1	3	0	0
10	2	1	3	1	1
15	2	0	2	0	1
20	2	0	2	0	1

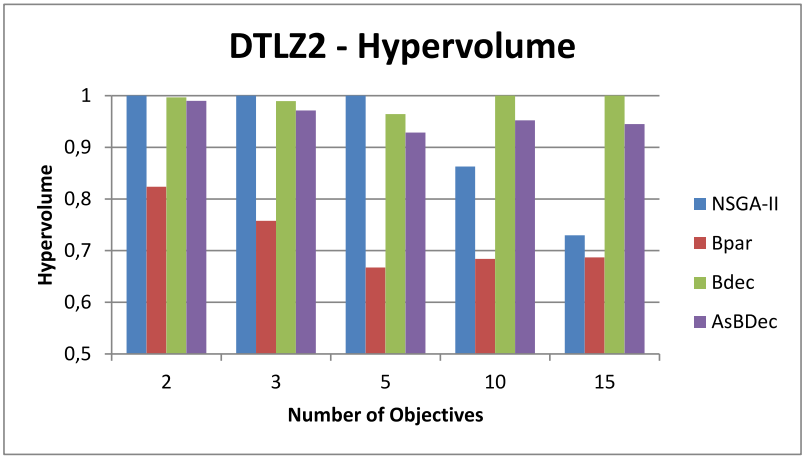
4.5. Discussion

The first conclusion that we can draw from the observation of the empirical results is that there is no single strategy that produces the best results for all classes of problems. Thus, it is important to evaluate in which situations specific approaches gave the good or poor results in relation to the others. These cases are summarized below:

- For up to 3 objectives, keeping a single population proved to be a good strategy. However, for more objectives, parallel strategies show superior performance.
- In terms of the scalability of a MOEA, for most problems the parallel algorithms based on decomposition should produce the best results, because algorithms based on decomposition are more able to "escape" from local optima. This affirmation is supported by the comparison between parallel algorithms based on Pareto and those based on decomposition. The

[4] http://www.r-project.org/.





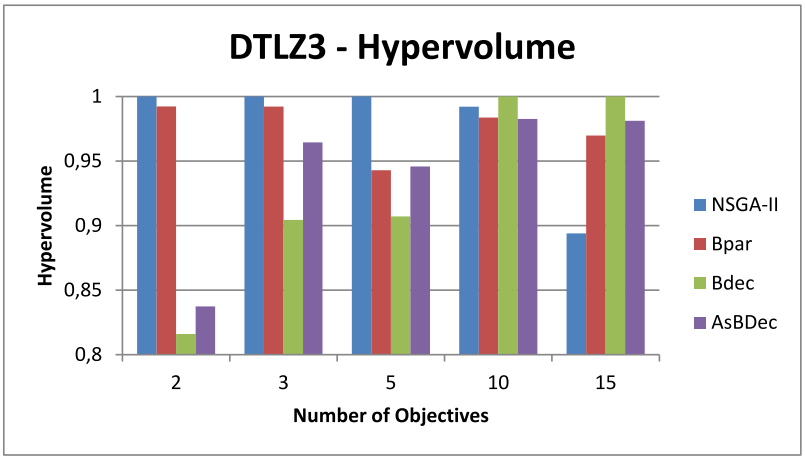


Fig. 8. Average results for the Hypervolume indicator for the DTLZ's {1,2,3} problems.

results for this comparison are shown in Tables A.19 to A.25 (Appendix A), obtained with the same methodology used along the research. From these results, it is possible to observe that for higher numbers of objectives the decomposition approach is always either the best or presents comparable results with the Pareto approach. The only exception is for DTLZ7, where the decomposition approaches do not present good results due to the discontinuity of the Pareto Front.

- Problems investigating the ability of the algorithm to generate well distributed solutions show that the parallel strategies presented the best results. However, when the focus is on the convergence of the solutions – for instance, for the DTLZ4 test – the sequential strategy based on decomposition appears to be the best choice.
- Changes based on the same problem can have different effects on different strategies. Sequential strategies presented superior performance for the DTLZ3 problem, while parallel strategies were the best for DTLZ5. Note that both are based on the DTLZ2 problem, where the scalability of algorithms is evaluated in terms of the convergence.
- In general, algorithms based on decomposition presented competitive results for most problems. However, strategies based on dominance relationships are the best to keep individuals in different regions of a disjoint Pareto front.

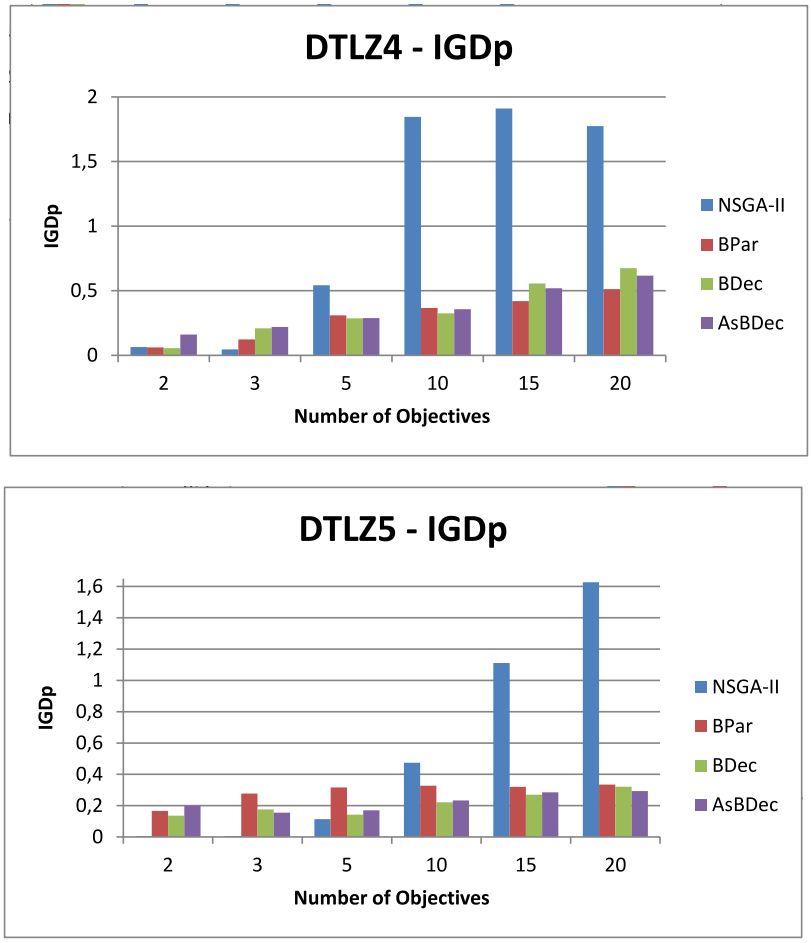


Fig. 9. Average results for the IGDp indicator for the DTLZ's {4,5} problems.

- For problems where the Pareto front is known to be disconnected, using parallel particles in different swarms proved to be the best strategy. The DTLZ7 problem clearly illustrates this case.
- Considering the communication policies for the model based on decomposition, statistical tests show no significant difference. However, in the experiments, it was possible to observe that most of the time there was useful information to communicate.

5. Related work

MOPSO has been widely used to solve several multi-objective optimization problems. Different Multi-Objective Evolutionary Algorithms (MOEAs), including algorithms based on MOPSO, have been shown to be notably effective, addressing MOPs in a suitable way and providing sets of good solutions for the problems.

However, in spite of the good results of MOEAs, these algorithms scale poorly when the number of objective functions increases, and the algorithms usually encounter difficulties with more than 3 objective functions [29,53]. One of the main challenges that are faced by MOEAs with many objectives is the deterioration of the searching ability. This deterioration mainly occurs because of an increase in the number of non-dominated solutions with the number of objectives, and consequently, there is no pressure towards the Pareto front. The Many-Objective Optimization field is a research area that has a focus on overcoming these limitations [2].

There are various algorithms for MaOPs in the literature, that use different strategies to deal with the challenges of this kind of problems. These MOEAs can be divided into different classes [35], described next.

The first class covers algorithms that use Pareto-based dominance as the first criteria and adopt customized diversity-based approaches to weaken the adverse impact of diversity maintenance. In [36], the authors propose the shift-based density estimation (SDE) strategy, this strategy has been efficiently applied to three popular Pareto-based algorithms. The SDE strategy has been found to be very promising in dealing with various many-objective problems [34,35]. In the recently proposed, knee point-based algorithm (KnEA) [67], a secondary selection scheme based on the knee point is used to enhance the selection pressure. Other approaches use reference points, for example NSGA-III [17] employs a set of reference vectors to assist the algorithm to select solutions which are close to those points. Several refinements of these ideas behind NSGA-III have been proposed such as in [55,63] and [11]. VaEA [60]. Among these, the Vector angle based many-objective Evolutionary Algorithm (VaEA) [60] uses the maximum-vector-angle-first principle to keep a good distribution among solutions. Furthermore, another principle called worse-elimination is adopted to conditionally replace bad solutions in terms of the convergence.

The second class covers indicator-based algorithms that use the value of a performance indicator to guide the search process, such as the IBEA algorithm [69]. For instance, several MOEAs based on the hypervolume (HV) metric have been proposed, such as the SMS-EMOA and the fast hypervolume based evolutionary algorithm (HypE) [4] [5], however their major disadvantages are the high overhead for computing the HV values especially for

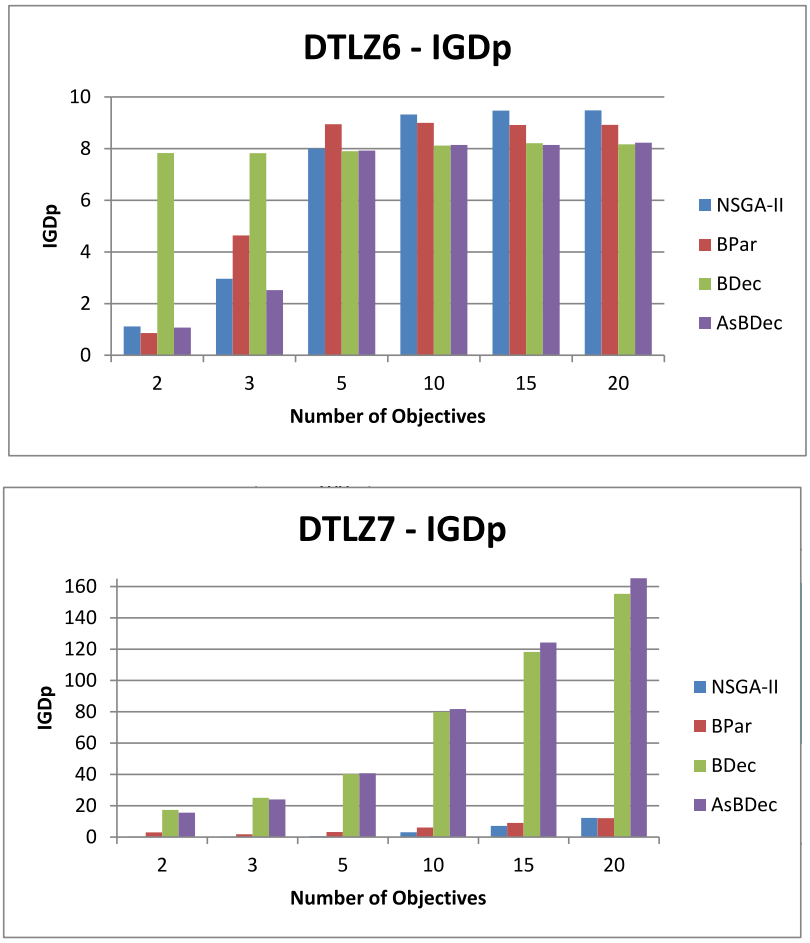


Fig. 10. Average results for the IGDp indicator for the DTLZ's {6,7} problems.

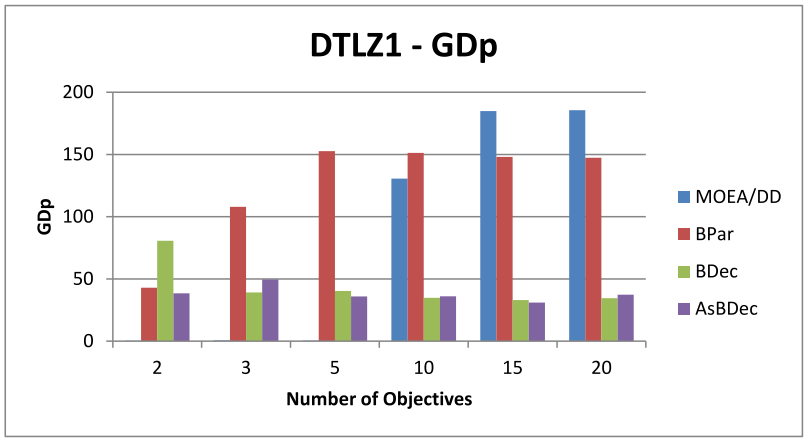


Fig. 11. Average results for the GDp indicator for the DTLZ1 problem (MOEA/DD vs. parallel approaches).

solving MaOPs. Recently, an Inverse Generational Distance Plus (IGD+) indicator-based evolutionary algorithm (IGD+- EMOA) was proposed in [41] for addressing MaOPs with no more than 8 objectives.

The third class covers the algorithms that modify the traditional dominance relation to enhance the selection pressure towards the Pareto front. Many algorithms have been proposed like the ϵ-dominance method [27], the fuzzy dominance methods [26]. Recently, an effective θ-dominance based evolutionary algorithm was proposed based on the so called θ-dominance relation [63].

The fourth class covers indicator-based algorithms that use the value of a performance indicator to guide the search process, such as the IBEA algorithm [69]. For instance, several MOEAs based on the hypervolume (HV) metric were proposed, such as the SMS-EMOA and the fast Hypervolume based Evolutionary algorithm (HypE) [4] [5], however their major disadvantages are the high overhead for computing the HV values especially for solving MaOPs. Recently, an Inverse Generational Distance Plus (IGD+) indicator-based evolutionary algorithm (IGD+- EMOA) was proposed in [41] for addressing MaOPs with no more than 8 objectives.

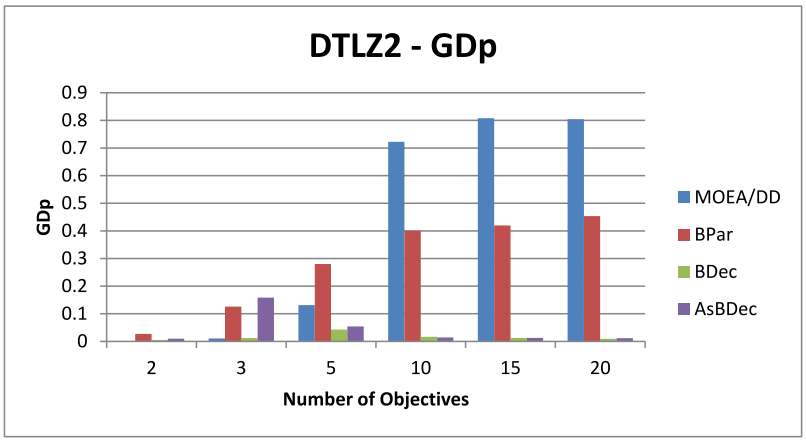


Fig. 12. Average results for the GDp indicator for the DTLZ2 problem (MOEA/DD vs. parallel approaches).

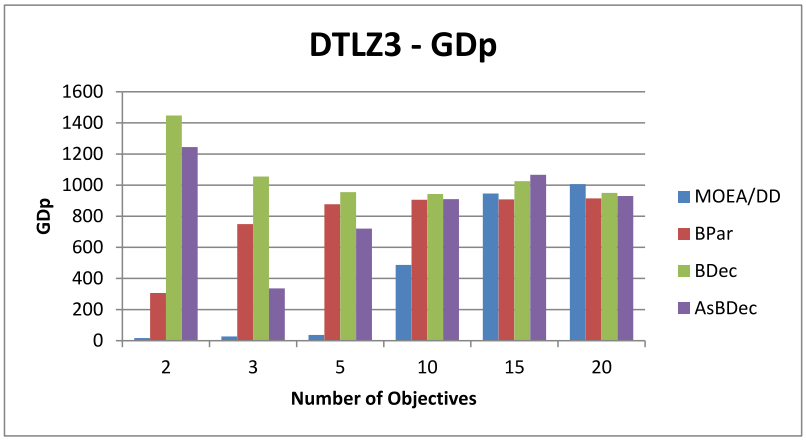


Fig. 13. Average results for the GDp indicator for the DTLZ3 problem (MOEA/DD vs. parallel approaches).

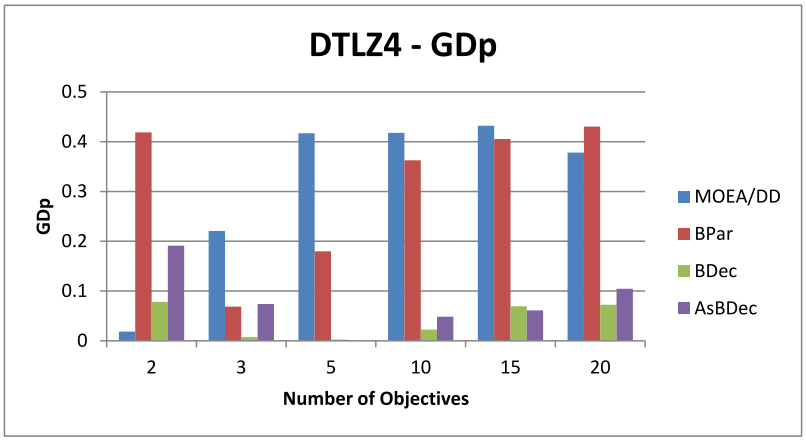


Fig. 14. Average results for the GDp indicator for the DTLZ4 problem (MOEA/DD vs. parallel approaches).

The fifth class covers MOEAs based on preference [49] and mainly includes three categories: a priori algorithms where the preference information is specified before the search; interactive algorithms where the decision maker provides the information interactively. In posteriori algorithms the information is introduced after the search. PICEA-g [59] is a well-known posteriori algorithm.

Finally, the last class covers hybrid strategies. For instance, the Two Arch2 [58], HEA-DP [65] and BCE [38] share similar characteristics for combining multiple strategies. All of them keep multiple populations (or archives) that are updated using different criteria based on quality measures. Furthermore, the algorithms select parents from different populations. The MOEA/DD [33] algorithm combines advantages of dominance and decomposition-based approaches. Other related methods include: MnRP-BILDE [48] based on objective reduction; SPEA/R [30] based on reference direction; and RPEA [39] based on reference points-based approach.

There are also algorithms that do not fit on these classes like the AnD [40] algorithm which only makes use of two strategies, the angle-based selection strategy and the shift based density estimation strategy for the environmental selection process. The

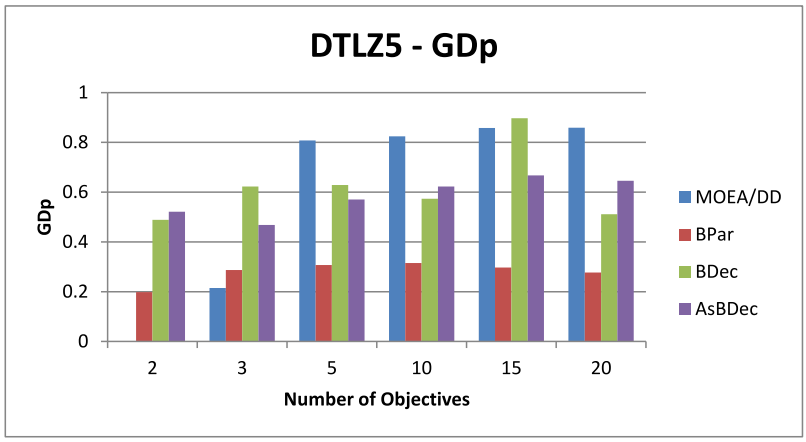


Fig. 15. Average results for the GDp indicator for the DTLZ5 problem (MOEA/DD vs. parallel approaches).

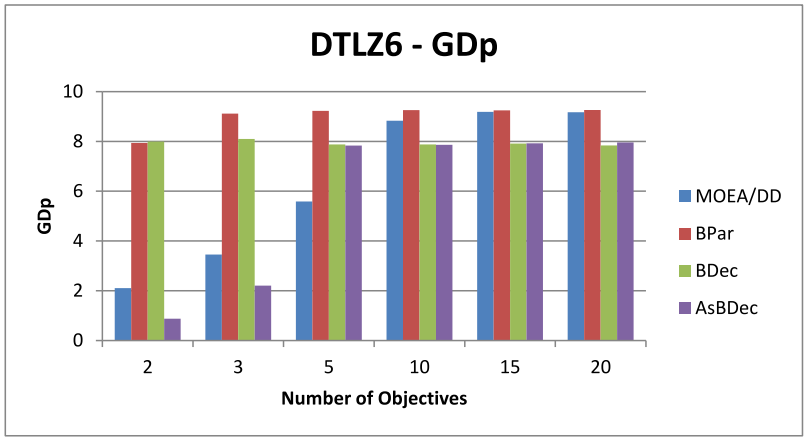


Fig. 16. Average results for the GDp indicator for the DTLZ6 problem (MOEA/DD vs. parallel approaches).

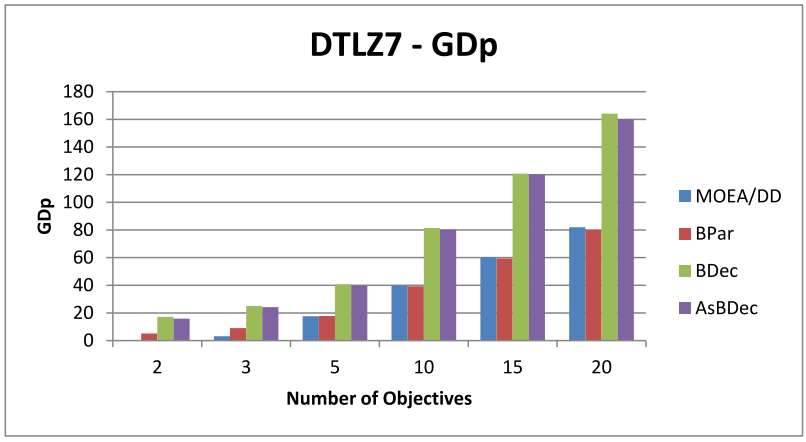


Fig. 17. Average results for the GDp indicator for the DTLZ7 problem (MOEA/DD vs. parallel approaches).

GrEA algorithm [62] introduces grid-dominance and three grid-based criteria that are applied to maintain both the convergence to and diversity of the solutions. The Bi-Goal Evolution (BiGE) algorithm [37] converts a given multi-objective optimization problem into a bi-goal (objective) optimization problem regarding convergence and diversity, and then handles this problem using the Pareto dominance relation in the bi-goal domain.

The research on MOPSO follows the ideas presented for MOEAs, thus, there are MOPSO based on: Pareto dominance [13,56], decomposition [3,46,47,64], indicators [23] and reference points [22]. In addition to these algorithms, multi-swarm techniques have been also applied to MaOPs. In [44] a multi-swarm approach is proposed that combines different swarms executing separately using different archiving methods.

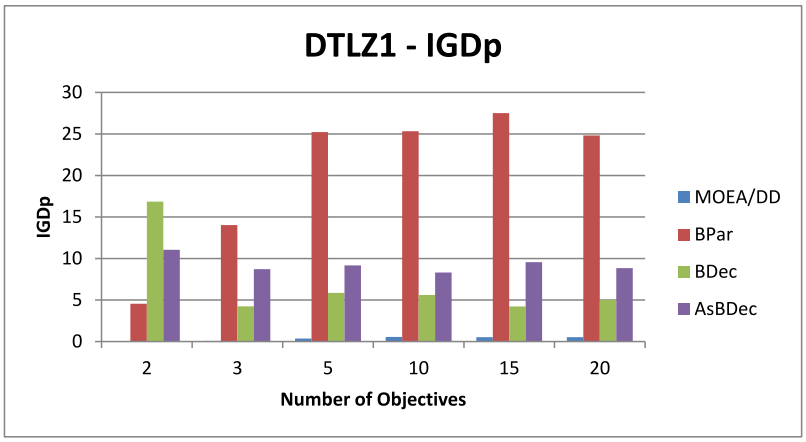


Fig. 18. Average results for the IGDp indicator for the DTLZ1 problem (MOEA/DD vs. parallel approaches).

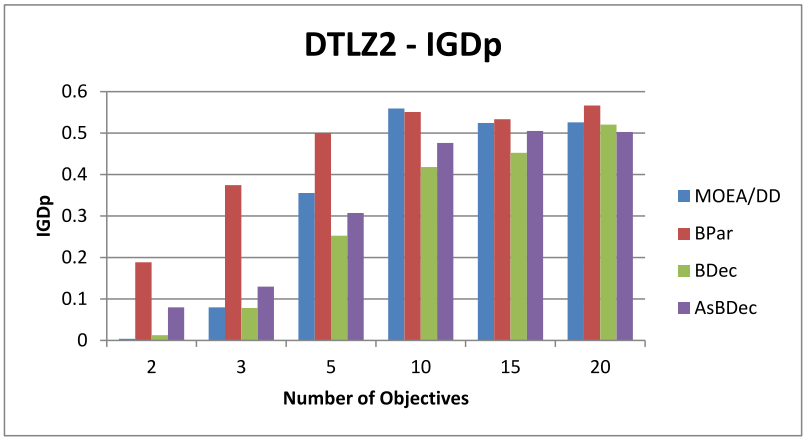


Fig. 19. Average results for the IGDp indicator for the DTLZ2 problem (MOEA/DD vs. parallel approaches).

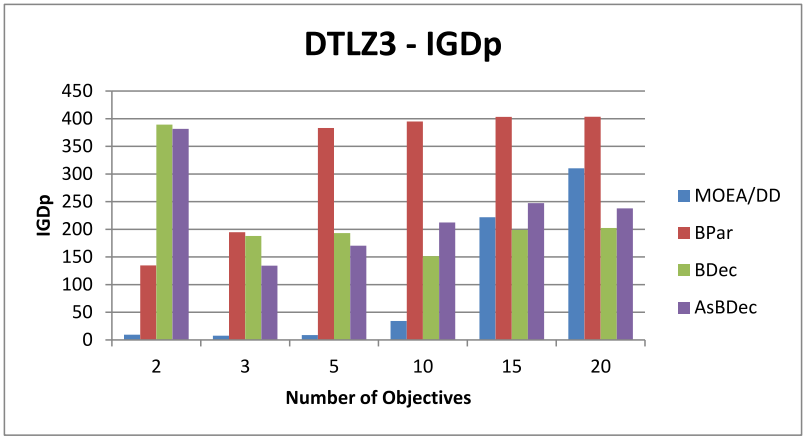


Fig. 20. Average results for the IGDp indicator for the DTLZ3 problem (MOEA/DD vs. parallel approaches).

Despite the existence of different strategies that address Many-Objective Optimization, it is possible to say that no single MOEA can outperform the others for all problems [35]. Furthermore, algorithms perform differently depending on the problem characteristics and the number of objectives.

In the present work, we join the strengths of several MOEAs in a collaborative framework exploring parallel strategies.

6. Conclusion

In this work we proposed two parallel strategies for the solution of MaOPs using multiple independent swarms that communicate using broadcast. In the first parallel strategy particles are grouped into independent subsets and each swarm keeps a repository with the best individuals of each subpopulation. Asynchronous

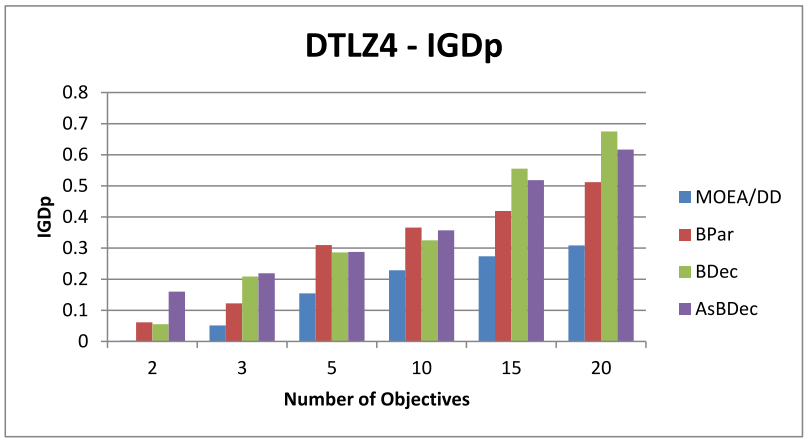


Fig. 21. Average results for the IGDp indicator for the DTLZ4 problem (MOEA/DD vs. parallel approaches).

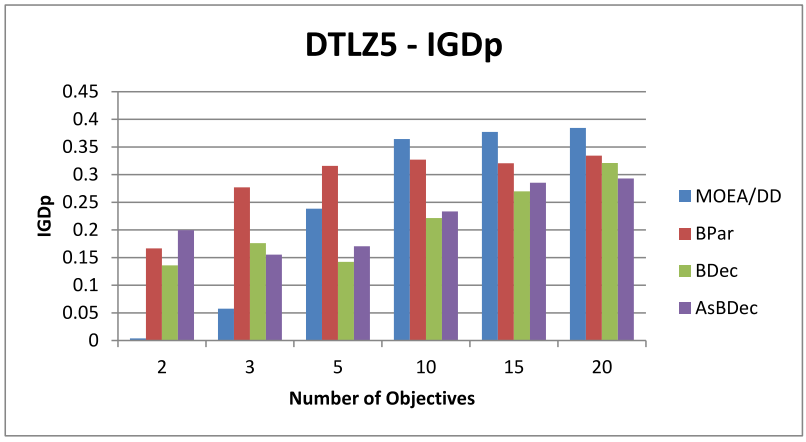


Fig. 22. Average results for the IGDp indicator for the DTLZ5 problem (MOEA/DD vs. parallel approaches).

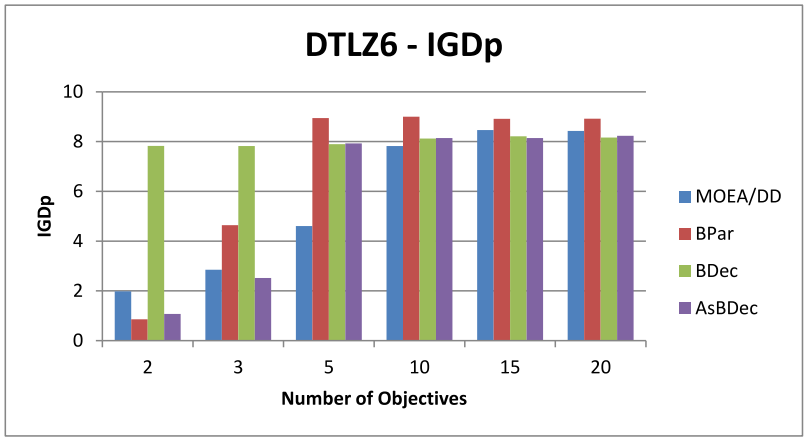


Fig. 23. Average results for the IGDp indicator for the DTLZ6 problem (MOEA/DD vs. parallel approaches).

communication methods were defined that are triggered by the improvement of the particles. This strategy allows the swarms to share individuals that help explore promising regions of the search space. The second parallel strategy involves the use of the decomposition technique to solve problems with multiple objectives. The multi-objective problems are divided into several single-objective subproblems, which allows the fitness to be evaluated for each subproblem. The two strategies were evaluated under both the synchronous and asynchronous communication models.

We employed the traditional DTLZ benchmark problems and results were with those of a sequential version. The conclusion is that the parallel algorithms have a positive effect on the convergence and diversity of the optimization process for problems with many objectives. Depending on the characteristics of each problem, different optimization strategies presented the best performance. Nevertheless, as the number of objectives increases, the parallel algorithms based on decomposition frequently outperform the Pareto approach. However, it is important to highlight that

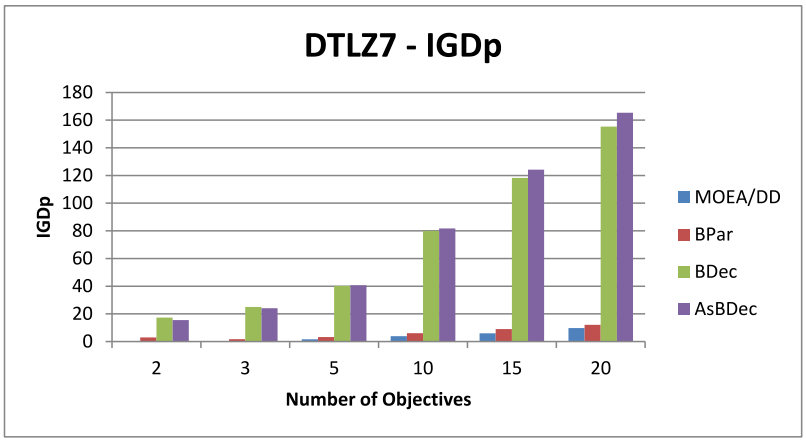


Fig. 24. Average results for the IGDp indicator for the DTLZ7 problem (MOEA/DD vs. parallel approaches).

specific problem features such as discontinuities of the Pareto Front reverse the situation so that the Pareto approach becomes the best choice.

Future work includes allowing different swarms to employ different optimization strategies. Swarms will still cooperate in order to solve the problem as a whole. The cooperation should employ different communication schemes for running parallel and distributed optimization, such as peer-to-peer networks, parallel computing platforms including GPU and multicore processors. Moreover, the design can focus on specialized swarms to cover part of the Pareto front, promoting convergence and resulting in an effective combination to exploit the whole Pareto front. Finally, future work should also include the investigation of other criteria besides the crowding distance for maintaining the diversity of the population.

Acknowledgments

This work was partially funded by the Brazilian Research Council (CNPq) grants 306103/2015-0 and 311451/2016-0, and by the Ministry of Education CAPES grant 88881.172797/2018-01.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.jpdc.2018.11.008.

References

[1] S.F. Adra, P.J. Fleming, Diversity management in evolutionary many-objective optimization, Trans. Evol. Comput. 15 (2) (2011) 183–195.
[2] S.F. Adra, P.J. Fleming, Diversity management in evolutionary many-objective optimization, IEEE Trans. Evol. Comput. 15 (2) (2011) 183–195.
[3] N. Al Moubayed, A. Petrovski, J. McCall, D2mopso: multi-objective particle swarm optimizer based on decomposition and dominance, in: Proceedings of the 12th European conference on Evolutionary Computation in Combinatorial Optimization, in: EvoCOP'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 75–86.
[4] J. Bader, E. Zitzler, Hype: An algorithm for fast hypervolume-based many-objective optimization, Evol. Comput. 19 (1) (2011) 45–76.
[5] N. Beume, B. Naujoks, M. Emmerich, Sms-emoa: Multiobjective selection based on dominated hypervolume, European J. Oper. Res. 181 (3) (2007) 1653–1669.
[6] M. Bonyadi, Z. Michalewicz, Analysis of stability, local convergence, and transformation sensitivity of a variant of particle swarm optimization algorithm, IEEE Trans. Evol. Comput. PP (99) (2015) 1–1.
[7] A. Britto, A. Pozo, Using archiving methods to control convergence and diversity for many-objective problems in particle swarm optimization, in: Evolutionary Computation (CEC), 2012 IEEE Congress on, 2012, pp. 1–8.
[8] X. Cai, Y. Li, Z. Fan, Q. Zhang, An external archive guided multiobjective evolutionary algorithm based on decomposition for combinatorial optimization, IEEE Trans. Evol. Comput. 19 (4) (2015) 508–523.
[9] A. de Campos, A.T. Pozo, E.P. Duarte, Evaluation of asynchronous multi-swarm particle optimization on several topologies, Concurr. Comput.: Pract. Exper. 25 (8) (2013) 1057–1071.
[10] A.B. de Carvalho, A. Pozo, Using different many-objective techniques in particle swarm optimization for many objective problems: An empirical study, Int. J. Comput. Inf. Syst. Ind. Manage. Appl. 3 (2011) 96–197.
[11] R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, A reference vector guided evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 20 (5) (2016) 773–791.
[12] C.W. Cleghorn, A.P. Engelbrecht, Particle swarm convergence: An empirical investigation, in: 2014 IEEE Congress on Evolutionary Computation (CEC), 2014, pp. 2524–2530.
[13] C.A. Coello Coello, G.B. Lamont, D.A.V. Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems (Genetic and Evolutionary Computation), Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
[14] C. Coello Coello, G. Lamont, D. van Veldhuizen, Evolutionary Algorithms for Solving Multi-Objective Problems, second ed., in: Genetic and Evolutionary Computation, Springer, Berlin, Heidelberg, 2007.
[15] C. Coello Coello, M. Lechuga, Mopso: a proposal for multiple objective particle swarm optimization, in: Evolutionary Computation, 2002. CEC '02. Proceedings of the 2002 Congress on, Vol. 2, 2002, pp. 1051–1056.
[16] I. De Falco, A.D. Cioppa, G.A. Trunfio, Large scale optimization of computationally expensive functions: An approach based on parallel cooperative coevolution and fitness metamodeling, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, in: GECCO '17, ACM, New York, NY, USA, 2017, pp. 1788–1795.
[17] 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 Trans. Evol. Comput. 18 (4) (2014) 577–601.
[18] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, IEEE Trans. Evol. Comput. 6 (2) (2002) 182–197.
[19] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable multi-objective optimization test problems, in: Congress on Evolutionary Computation (CEC 2002), IEEE Press, 2002, pp. 825–830.
[20] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[21] R. Eberhart, Y. Shi, Comparing inertia weights and constriction factors in particle swarm optimization, in: Evolutionary Computation, 2000. Proceedings of the 2000 Congress on, Vol. 1, 2000, pp. 84 –88.
[22] E.M.N. Figueiredo, T.B. Ludermir, C.J.A.B. Filho, Many objective particle swarm optimization, Inform. Sci. 374 (2016) 115–134.
[23] I.C. Garcia, C.A.C. Coello, A. Arias-Montano, Mopsohv: A new hypervolume-based multi-objective particle swarm optimizer, in: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2014, Beijing, China, July 6-11, 2014, 2014, pp. 266–273.
[24] S. García, D. Molina, M. Lozano, F. Herrera, A study on the use of non-parametric tests for analyzing the evolutionary algorithms' behaviour: a case study on the cec'2005 special session on real parameter optimization, J. Heuristics 15 (6) (2009) 617–644.
[25] M. Hajjem, H. Bouziri, E.-G. Talbi, K. Mellouli, Parallel ant colony optimization for evacuation planning, in: Proceedings of the Genetic and Evolutionary Computation Conference Companion, in: GECCO '17, ACM, New York, NY, USA, 2017, pp. 51–52.
[26] Z. He, G.G. Yen, J. Zhang, Fuzzy-based pareto optimality for many-objective evolutionary algorithms, IEEE Trans. Evol. Comput. 18 (2) (2014) 269–285.
[27] A.G. Hernndez-Daz, L.V. Santana-Quintero, C.A.C. Coello, J. Molina, Pareto-adaptive -dominance, Evol. Comput. 15 (4) (2007) 493–517, PMID: 18021017.

[28] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (5) (2006) 477–506.
[29] H. Ishibuchi, N. Tsukamoto, Y. Nojima, Evolutionary many-objective optimization: A short review, in: Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on, 2008, pp. 2419–2426.
[30] S. Jiang, S. Yang, A strength pareto evolutionary algorithm based on reference direction for multiobjective and many-objective optimization, IEEE Trans. Evol. Comput. 21 (3) (2017) 329–346.
[31] O.R.C. Jr., A. Pozo, Using hyper-heuristic to select leader and archiving methods for many-objective problems, in: Evolutionary Multi-Criterion Optimization - 8th International Conference, EMO 2015, Guimarães, Portugal, March 29 -April 1, 2015. Proceedings, Part I, 2015, pp. 109–123.
[32] J. Kennedy, R. Eberhart, Particle swarm optimization, in: Neural Networks, 1995. Proceedings., IEEE International Conference on, Vol. 4, 1995, pp. 1942 –1948.
[33] K. Li, K. Deb, Q. Zhang, S. Kwong, An evolutionary many-objective optimization algorithm based on dominance and decomposition, IEEE Trans. Evol. Comput. 19 (5) (2015) 694–716.
[34] M. Li, C. Grosan, S. Yang, X. Liu, X. Yao, Multiline distance minimization: A visualized many-objective test problem suite, IEEE Trans. Evol. Comput. 22 (1) (2018) 61–78.
[35] K. Li, R. Wang, T. Zhang, H. Ishibuchi, Evolutionary many-objective optimization: A comparative study of the state-of-the-art, IEEE Access 6 (2018) 26194–26214.
[36] M. Li, S. Yang, X. Liu, Shift-based density estimation for pareto-based algorithms in many-objective optimization, IEEE Trans. Evol. Comput. 18 (3) (2014) 348–365.
[37] M. Li, S. Yang, X. Liu, Bi-goal evolution for many-objective optimization problems, Artificial Intelligence 228 (2015) 45–65.
[38] M. Li, S. Yang, X. Liu, Pareto or non-pareto: Bi-criterion evolution in multiobjective optimization, IEEE Trans. Evol. Comput. 20 (5) (2016) 645–665.
[39] Y. Liu, D. Gong, X. Sun, Y. Zhang, Many-objective evolutionary optimization based on reference points, Appl. Soft Comput. 50 (2017) 344–355.
[40] Z.-Z. Liu, Y. Wang, P.-Q. Huang, A Many-Objective Evolutionary Algorithm with Angle-Based Selection and Shift-Based Density Estimation, ArXiv e-prints, 2017.
[41] E.M. Lopez, C.A.C. Coello, Igd-emoa: A multi-objective evolutionary algorithm based on igd, in: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 999–1006.
[42] M. Løvbjerg, T.K. Rasmussen, T. Krink, Hybrid particle swarm optimiser with breeding and subpopulations, in: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2001, Morgan Kaufmann, 2001, pp. 469–476.
[43] M.H. MacDougall, Simulating Computer Systems: Techniques and Tools, MIT Press, Cambridge, MA, USA, 1987.
[44] J.L. Matos, A. Britto, Multi-swarm algorithm based on archiving and topologies for many-objective optimization, in: 2017 IEEE Congress on Evolutionary Computation (CEC), 2017, pp. 1877–1884.
[45] S. Mostaghim, J. Teich, Strategies for finding good local guides in multi-objective particle swarm optimization (mopso), in: Swarm Intelligence Symposium, 2003. SIS '03. Proceedings of the 2003 IEEE, 2003, pp. 26–33.
[46] N.A. Moubayed, A. Petrovski, J.A.W. McCall, A novel smart multi-objective particle swarm optimisation using decomposition, in: PPSN (2), 2010, pp. 1–10.
[47] W. Peng, Q. Zhang, A decomposition-based multi-objective particle swarm optimization algorithm for continuous optimization problems, in: Granular Computing, 2008. GrC 2008. IEEE International Conference on, 2008, pp. 534–537.
[48] N. Pholdee, S. Bureerat, A. Yildiz, Hybrid real-code population-based incremental learning and differential evolution for many-objective optimisation of an automotive floor-frame, Int. J. Veh. Des. 73 (2017) 20–53.
[49] F. di Pierro, S.T. Khu, D.A. Savic, An investigation on preference order ranking scheme for multiobjective evolutionary optimization, IEEE Trans. Evol. Comput. 11 (1) (2007) 17–45.
[50] C.R. Raquel, P.C. Naval Jr., An effective use of crowding distance in multiobjective particle swarm optimization, in: Proceedings of the 2005 conference on Genetic and evolutionary computation, in: GECCO '05, ACM, New York, NY, USA, 2005, pp. 257–264.
[51] J.R. Schott, Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization (Ph.D. thesis), Massachusetts Institute of Technology, 1995.
[52] O. Schütze, X. Esquivel, A. Lara, C.A.C. Coello, Using the averaged hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 16 (4) (2012) 504–522.
[53] O. Schutze, A. Lara, C.A. Coello Coello, On the influence of the number of objectives on the hardness of a multiobjective optimization problem, IEEE Trans. Evol. Comput. 15 (4) (2011) 444–455.
[54] Y. Shi, R.C. Eberhart, Empirical study of particle swarm optimization, in: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), Vol. 3, 1999, p. 1950, http://dx.doi.org/10.1109/CEC.1999.785511.
[55] Y. Sun, G.G. Yen, Z. Yi, Reference line-based estimation of distribution algorithm for many-objective optimization, Knowl.-Based Syst. 132 (2017) 129–143.
[56] M. Torres, B. Baran, Optimizacion de enjambre de particulas para problemas de muchos objetivos, in: 2015 Latin American Computing Conference (CLEI), 2015, pp. 1–11.
[57] G.A. Trunfio, P. Topa, J. Ws, A new algorithm for adapting the configuration of subcomponents in large-scale optimization with cooperative coevolution, Inform. Sci. 372 (Supplement C) (2016) 773–795.
[58] H. Wang, L. Jiao, X. Yao, Two arch2: An improved two-archive algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 19 (4) (2015) 524–541.
[59] R. Wang, R.C. Purshouse, P.J. Fleming, Preference-inspired coevolutionary algorithms for many-objective optimization, IEEE Trans. Evol. Comput. 17 (4) (2013) 474–494.
[60] Y. Xiang, Y. Zhou, M. Li, Z. Chen, A vector angle-based evolutionary algorithm for unconstrained many-objective optimization, IEEE Trans. Evol. Comput. 21 (1) (2017) 131–152.
[61] Q. Yang, S. Ding, Novel algorithm to calculate hypervolume indicator of pareto approximation set, in: D.-S. Huang, L. Heutte, M. Loog (Eds.), Advanced Intelligent Computing Theories and Applications. With Aspects of Contemporary Intelligent Computing Techniques, in: Communications in Computer and Information Science, vol. 2, Springer Berlin Heidelberg, 2007, pp. 235–244.
[62] S. Yang, M. Li, X. Liu, J. Zheng, A grid-based evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 17 (5) (2013) 721–736.
[63] Y. Yuan, H. Xu, B. Wang, X. Yao, A new dominance relation-based evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 20 (1) (2016) 16–37.
[64] S. Zapotecas Martínez, C.A. Coello Coello, A multi-objective particle swarm optimizer based on decomposition, in: Proceedings of the 13th annual conference on Genetic and evolutionary computation, in: GECCO '11, ACM, New York, NY, USA, 2011, pp. 69–76.
[65] Y.H. Zhang, Y.J. Gong, J. Zhang, Y. b. Ling, A hybrid evolutionary algorithm with dual populations for many-objective optimization, in: 2016 IEEE Congress on Evolutionary Computation (CEC), 2016, pp. 1610–1617.
[66] Q. Zhang, H. Li, Moea/d: A multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
[67] X. Zhang, Y. Tian, Y. Jin, A knee point-driven evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 19 (6) (2015) 761–776.
[68] E. Zitzler, Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications (Ph.D. thesis), ETH Zurich, Switzerland, 1999.
[69] E. Zitzler, S. Künzli, Indicator-based selection in multiobjective search, in: X. Yao, E.K. Burke, J.A. Lozano, J. Smith, J.J. Merelo-Guervós, J.A. Bullinaria, J.E. Rowe, P. Tiňo, A. Kabán, H.-P. Schwefel (Eds.), Parallel Problem Solving from Nature - PPSN VIII, Springer Berlin Heidelberg, Berlin, Heidelberg, 2004, pp. 832–842.
[70] E. Zitzler, L. Thiele, M. Laumanns, C. Fonseca, V. da Fonseca, Performance assessment of multiobjective optimizers: an analysis and review, IEEE Trans. Evol. Comput. 7 (2) (2003) 117–132.

Arion de Campos Jr. is an Adjunct Professor at State University of Parana at Ponta Grossa, Brazil. He obtained his Ph.D. in Computer Science from the Federal University of Parana (UFPR), Brazil, 2014, M.Sc. in Computer Science from University of Sao Paulo (USP), Brazil, 2001, and B.Sc. in Informatics from the State University of Ponta Grossa (UEPG), Brazil, 1998. He is currently the Dean of the Computer Science Department at UEPG. His main research interests include Evolutionary Computing and Many-Objective Optimization Problems.

Aurora Trinidad Ramirez Pozo is a Full Professor of Computer Science at Federal University of Parana, Brazil, and chair of the Bio-inspired Computation Laboratory (C-Bio). She received her Ph.D. degree in Electrical Engineering from Federal University of Santa Catarina, Brazil. She received a M.S. in Electrical Engineering from Federal University of Santa Catarina, Brazil, in 1991. Prof. Aurora's research interests are in evolutionary computation, data mining and complex problems. She has served on several Editorial Boards, and chaired and served on the TPC of several conferences and workshops in her fields of interest. Prof. Aurora has published more than 100 peer-reviewed papers, and has supervised more than 50 students, including graduate and undergraduate levels. She is a member of the Brazilian Computer, the IEEE and the ACM Society.

Elias P. Duarte Jr. is a Full Professor of Computer Science at Federal University of Parana (UFPR), Curitiba, Brazil, where he is a faculty member since 1991. He chaired the Brazilian National Laboratory on Computer Networks (2010–2012), the Graduate Program in Computer Science of UFPR (2006–2008), and was a member of Advisory Board of RNP — the Brazilian National Academic Network (2012–2016). He has a Ph.D. degree in Computer Science from Tokyo Institute of Technology, Japan, 1997, M.Sc. Telecommunications from the Polytechnical University of Madrid, Spain, 1991, and B.Sc. and M.Sc. in Computer Science from Federal University of Minas Gerais, Brazil, 1987 and 1991, respectively. Research interests include Computer Networks and Distributed Systems, their Dependability and Algorithms. Prof. Elias has published nearly 200 peer-reviewed papers, and has supervised nearly 130 students, both on the graduate and undergraduate levels. He has served on several Editorial Boards, and chaired and served on the TPC of several conferences and workshops in his fields of interest. He also chaired the Special Interest Group on Fault Tolerant Computing of the Brazilian Computing Society (2005–2007). He is a member of the Brazilian Computer Society and a Senior Member of the IEEE.