Clustering Based Parallel Many-Objective Evolutionary Algorithms Using the Shape of the Objective Vectors

Christian von Lücken[1(✉)], Carlos Brizuela[2], and Benjamin Barán[1]

[1] Facultad Politécnica, Universidad Nacional de Asunción, San Lorenzo, Paraguay
clucken@pol.una.py
[2] CICESE Research Center, Ensenada, México

Abstract. Multi-objective Evolutionary Algorithms (MOEA) are used to solve complex multi-objective problems. As the number of objectives increases, Pareto-based MOEAs are unable to maintain the same effectiveness showed for two or three objectives. Therefore, as a way to ameliorate this performance degradation several authors proposed preference-based methods as an alternative to Pareto based approaches. On the other hand, parallelization has shown to be useful in evolutionary optimizations. A central aspect for the parallelization of evolutionary algorithms is the population partitioning approach. Thus, this paper presents a new parallelization approach based on clustering by the shape of objective vectors to deal with many-objective problems. The proposed method was compared with random and k-means clustering approaches using a multi-threading framework in parallelization of the NSGA-II and six variants using preference-based relations for fitness assignment. Executions were carried-out for the DTLZ problem suite, and the obtained solutions were compared using the generational distance metric. Experimental results show that the proposed shape-based partition achieves competitive results when comparing to the sequential and to other partitioning approaches.

Keywords: Multi-Objective Evolutionary Algorithms · Many-objective optimization · Parallel evolutionary algorithms

1 Introduction

Multi-objective evolutionary algorithms (MOEAs) are well-suited for solving several problems requiring simultaneous optimization of two or three conflicting objectives [3,4]. In general, MOEAs differ in the fitness assignment method, but some of the most successful of them, such as the NSGA-II [5], use the Pareto dominance concept as the foundation to guide the search towards the optimal solution set.

In the last few years, several researchers have pointed out convergence difficulties that Pareto-based MOEAs face when solving many-objective problems,

A. Gaspar-Cunha et al. (Eds.): EMO 2015, Part II, LNCS 9019, pp. 50–64, 2015.
DOI: 10.1007/978-3-319-15892-1_4

i.e. problems having four or more conflicting objectives [17]. The main source of these difficulties comes from that the proportion of non-dominated individuals in an evolutionary population tends to one as the number of objectives increases. Therefore, for a growing number of objectives, it becomes increasingly difficult to discriminate among solutions to assign fitness values using only the dominance relation [4,12]. In order to distinguish among solutions, some authors propose to replace the Pareto dominance relation by preference relations that use additional information such as the number of objectives for which one solution is better than another [9,12], the size of improvement [12,20], or the number of subspaces in which a given solution remains non-dominated [8].

Even though there are other approaches to deal with many-objective problems using MOEAs [17], the use of alternative relations have shown to be able to improve the quality of the obtained Pareto set approximations without requiring to combine or reduce the number of objectives which, in several cases, may not be adequate or even possible. Also, alternative relations are relatively easy to incorporate into existing (Pareto-based) MOEAs with a minimal computational overhead.

On the other hand, parallelization of existing MOEAs have proven to be an effective mechanism to improve the quality of the obtained approximations in multi-objective problems of increasing difficulty [3,21]. However, to the best of our knowledge, parallelization of methods based on preference relations were not applied and tested in many-objective problems. Moreover, parallel MOEAs (pMOEAs) were mainly developed and tested in distributed memory parallel systems [15]; however, nowadays availability of lower cost shared memory multicore platforms requires a review of the parallelization methods in the new existing environment in order to leverage the computational power that these platforms may offer.

The island model is the most popular parallelization paradigm for MOEAs, it consists of a number of subpopulations or islands evolving independently provided with a mechanism to interchange individuals exploring for global optima. Using multiple subpopulations has been identified as a simple way to increase the chance of finding better solutions for a problem [2,3]. In general, in the island model, the interchange of individuals is carried out by a migration operator that selects solutions to migrate from one subpopulation to another. The migration strategy definition includes to specify the elapsed time between migrations, as well as the selection criterion and the number of elements to migrate.

Alternatively, instead of conceiving a migration strategy, other island-based methods split the evolutionary population into multiple subpopulations that, after evolving independently, are combined again into a single population, which also can evolve in a single process for a number of iterations, then the division process is repeated [1,13,18,19]. In this work, since division is usually used to create partitions with similar individuals, these methods are generically called clustering based parallel MOEA. Population partitions with similar individuals induce a mating restriction, i.e. recombination of similar individuals, that has shown to be useful to improve the performance of evolutionary algorithms [4]. As

it is noted in [2], the main drawback of these pMOEAs based on repeated partitioning of a global population is that the division procedure introduces a strong dependency among the computing units. However, the expected improvement provided by working with multiple subpopulations running in different processors may justify the use of these methods. Most of clustering based pMOEAs were implemented into distributed parallel systems, thus, requiring additional data communication to send subpopulations to the process in charge of join them in a whole population and redistribute it again. However, in nowadays multicore systems with shared memory systems to gather subpopulations requires a reduced communication and synchronization time.

This paper studies clustering based parallelization of NSGA-II variants using preference relations for many-objective problems. In this case, the clustering methods serves to search into different regions simultaneously using preference based MOEAs, while, in each subpopulation, the preference relations are used to improve the rank over similar individuals, which may be useful to improve the overall search. Moreover, a new clustering method based on the shape of the objective vectors is proposed (see Section 3).

In order to validate the proposal in the field of many objective optimization, an empirical comparison of the proposed method against other alternatives for population division for pMOEAs was produced considering the same parallel framework. Using this framework, the NSGA-II and six of its variants based on preference relations were parallelized with three options to divide the population: at random, using the k-means clustering algorithm, and using the proposed shape-based clustering algorithm. Sequential and parallel implementations were used to solve the suite of DTLZ problems with 10 objectives and the obtained results compared regarding the Generational Distance metric [4].

2 Clustering Based Parallelization of MOEAs

2.1 Parallel MOEAs Using Clustering

In general, the island model is used to develop pMOEAs as parallel extensions of existing mono-objective or multi-objective evolutionary algorithms. Besides the algorithm considered for parallelization, island based pMOEAs differ in the mechanism used to interchange information among subpopulations searching for the global Pareto front. As previously explained, an alternative to exchange individuals among islands is to iteratively divide a global population into subpopulations and gather them to repeat the cycle again. In this case, methods may also alternate between the execution of sequential iterations, considering the whole population, with parallel iterations. Some methods to divide the global population into subpopulations considered here as relevant related works are:

- population partitioning based on sorting of a objective function [13,18],
- population partitioning based on the cone separation approach [1],
- population partitioning based on the k-means clustering algorithm [19].

Parallelization approaches based on objective function sorting use the value of one selected criterion to divide and distribute the population in different islands. Thus, each island works on a subpopulation with individuals that are similar regarding the chosen sorting objective. After some evaluations, subpopulations are gathered again and the process is repeated until a stop condition is met. Examples of these approaches the Divided Range Multi-objective Genetic Algorithm [13] and the Parallel Single Front Genetic Algorithm [18].

The Cone Separation Approach [1] parallelizes MOEAs by dividing the search space and mapping different search areas to different processors. In this case, objective values are normalized, then, considering a given reference point, the search space is divided in regular partitions or regions called cones that can be assigned to different processors. As the searching process progresses, the search space partitioning is adapted at regular intervals by normalizations and readjustment of the region assigned to each processor. Region restrictions are applied to each subpopulation and individuals not meeting constraints in a subpopulation migrate to the one where they do not violate the constraints. Migrated individuals are added to the receiving population without deletion. This way, there is not a centralized process that divides the population iteratively but a decentralized one; however, the method is conceptually equal to the methods based on repeated division of a global population.

Streichert et al. [19] use the k-Means clustering algorithm to divide the search space of a given optimization problem in suitable partitions without a priori knowledge about the search space topology. The k-Means procedure is applied over the current Pareto Front of the whole population to produce partitions to be distributed to the available processors or islands. In case the size of Pareto Front being smaller than the number of processors, next levels of Pareto fronts are also used for clustering. Each processor runs NSGA-II until the number of generations reaches a number of iterations, then, solutions are gathered in a master procedure where search space is partitioned and distributed again.

2.2 Preference Relations in MOEAs for Many-Objective Problems

The NSGA-II is a well known MOEA that showed an excellent performance in several multi-objective problems, thus, several researchers considered the NSGA-II [5] as the base algorithm to implement and validate their proposed relations and algorithmic approaches both in multi-objective as well as many-objective problem domains. The NSGA-II assigns the fitness of solutions based on two values: its non-dominance ranking and its crowding distance. Variants of the NSGA-II can be produced by considering alternative methods to calculate fitness. In this paper, the following operators and ranking methods are used to modify the fitness assignment procedure of the NSGA-II in a minimization context, for space limitation reasons the details are not explained here but we refer the interested reader to the corresponding references:

1. **Favour relation [9]:** this relation counts the number of objectives in which a given solution outperforms another. Given a multi-objective problem minimizing a function $\mathbf{F}(\mathbf{x}) = (f_1(\mathbf{x}), \ldots, f_m(\mathbf{x}))$ with m objectives, and let $\mathbf{x}$

and $\mathbf{x}'$ be two vectors in the set of feassible solutions $\mathcal{X}_f$, it is said that $\mathbf{x}$ is favoured than $\mathbf{x}'$, denoted as $\mathbf{x} \prec_{favour} \mathbf{x}'$, if and only if

$$n_b(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}')) > n_b(\mathbf{F}(\mathbf{x}'), \mathbf{F}(\mathbf{x})) \quad (1)$$

where:

$$n_b(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}')) = |\{f_i(\mathbf{x}) \text{ s.t. } f_i(\mathbf{x}) < f_i(\mathbf{x}')\}| \quad (2)$$

In [9], the favour relation is proposed to be used with the Satisfiability Class Ordering classification (SCO) procedure [9] to sort solutions.

2. $\epsilon-$**Preferred Relation [20]:** the $\epsilon-$Preferred relation compares solutions by counting the number of times a solution exceeds user defined limits for each dimension (ϵ_i) and, in case of a tie, it uses the favour relation. Given two solutions $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{X}_f$, $\mathbf{F}(\mathbf{x}) = \mathbf{y} = (y_1, \ldots, y_m)$, $\mathbf{F}(\mathbf{x}') = \mathbf{y}' = (y'_1, \ldots, y'_m)$, it is said that $\mathbf{x}$ is $\epsilon-$preferred than $\mathbf{x}'$, denoted as $\mathbf{x} \prec_{\epsilon-preferred} \mathbf{x}'$, iff $\mathbf{x} \prec_{\epsilon-exceed} \mathbf{x}' \vee (\mathbf{x}' \not\prec_{\epsilon-exceed} \mathbf{x} \wedge \mathbf{x} \prec_{favour} \mathbf{x}')$, where $\mathbf{x} \prec_{\epsilon-exceed} \mathbf{x}'$ implies that:

$$|\{i : y_i < y'_i \wedge |y_i - y'_i| > \epsilon_i\}| > |\{i : y'_i < y_i \wedge |y'_i - y_i| > \epsilon_i\}|$$

As in [9], in [20] the SCO algorithm is used to rank solutions.

3. **Preference Ordering based on order of efficiency (PO_{k}) [8]:** a solution $\mathbf{x}$ is considered to be efficient of order k if it is Pareto optimal in the $\binom{m}{\mathsf{k}}$ subspaces of the objective space taking into account only k out of m objectives at a time. The order of efficiency of a solution $\mathbf{x}$, denoted by $K(\mathbf{x})$ is the minimum k value for which $\mathbf{x}$ is efficient.
4. $-\epsilon$**-DOM [16]:** the $-\epsilon$-DOM distance replaces the NSGA-II crowding distance. The $-\epsilon$-DOM distance of a solution $\mathbf{x}$ is the smallest value such that if subtracted from all objectives of $\mathbf{F}(\mathbf{x}')$, makes $\mathbf{x}$ dominated.
5. $(1-k)-$**dominance relation [12]:** this relation counts the objectives in which a solution is better or equal than another one. Let $\mathbf{x}$ and $\mathbf{x}' \in \mathcal{X}_f$, $\mathbf{F}(\mathbf{x}) = \mathbf{y}$, $\mathbf{F}(\mathbf{x}') = \mathbf{y}'$, it is said that $\mathbf{x}$ $(1-k)-$dominates $\mathbf{x}'$ iff : $n_e(\mathbf{y}, \mathbf{y}') < m$ and $n_b(\mathbf{y}, \mathbf{y}') \geq \frac{m-n_e}{k+1}$, where m is the number of objectives, $0 \leq k \leq 1$, n_b is as in Eq. (2), and $n_e(\mathbf{F}(\mathbf{x}), \mathbf{F}(\mathbf{x}'))$ is $|\{f_i(\mathbf{x}) \text{ s.t. } f_i(\mathbf{x}) = f_i(\mathbf{x}')\}|$.
6. **Fuzzy** $(1-k_F)-$**dominance relation [12]:** the fuzzy extension of $(1-k)-$dominance is defined by determining membership functions μ_b^i , μ_e^i and μ_w^i for each objective function i.

2.3 A Framework for Clustering Based Parallelization of MOEAs Based on Preference Relations

To study clustering based pMOEAs, Algorithm 1 presents a framework for pMOEAs that use iterative partitioning of a global population for multi-threading systems. Using this framework, by setting the corresponding parameters it is possible to study different options for pMOEAs based on clustering.

Algorithm 1 starts reading its parameters; the MOEA $\mathcal{M}$ to be considered for evolutions and its parameters, the partition method (PM), the number of

Algorithm 1. A framework for clustering based pMOEAs

```
Read parameters
Set t = 0
Create an initial random global evolutionary population P_t
while t < it_g do                          ▷ it_g is the total number of evolutionary steps performed
    for it_s iterations do                 ▷ it_s: iterations considering a single population
        Evolve P_t in P_{t+1} using M      ▷ M is the MOEA to be used
        t = t + 1
    end for
    In τ parallel threads
    t' = 0
    for it_p iterations do                 ▷ it_p: iterations considering subpopulations in parallel
        if t' mod it_c = 0 then
            Split P_t in P_t^1, ..., P_t^τ using partition method PM
        end if
        Evolve P_{t+t'}^{Id} in P_{t+t'+1}^{Id} using M
        t' = t' + 1
    end for
    End parallel
    t = t + it_p
end while
Save non-dominated solutions from P_t
```

islands (τ), the total number of evolutionary steps (it_g), the number of single thread iterations (it_s), the number of parallel iterations (it_p) and the number of iterations before clustering (it_c). Next, the global number of iterations t is set to 0, and the global population P_t is created at random. After initialization, while $t < it_g$, it_s evolutionary iterations are executed in a single thread considering the evolutionary population as a whole. Then, τ threads are created, one for each island, and parallel execution starts. The next step is to split $P(t)$ using the procedure PM in τ subpopulations ($P_t^1 \ldots, P_t^\tau$), this procedure, repeated each it_c iterations, may be implemented in parallel or as a single thread. Each thread has an identifier Id, thus at each thread Id, evolution of P_t^{Id} occurs during it_p iterations. When iterations in all threads end, the global count of iterations t is updated to $t + it_p$ and the cycle continues until the stop condition is met. Finally, the final set of solutions is saved.

3 Partition Approach Based on the Shape of Solutions

In [10] it is presented a similarity measure over Euclidean spaces for high dimensional vectors. To calculate this measure, a real vector $\mathbf{y} = (y_1, \ldots, y_M) \in \mathcal{R}^m$ is divided in a pair $(s(\mathbf{y}), \pi(\mathbf{y}))$, where $s(\mathbf{y})$ is the ordered version (weak) of $\mathbf{y}$ elements, and $\pi(\mathbf{y})$ is the permutation of indexes $\{1, \ldots, m\}$ that produce the sorting which is called as the shape part. The distance between two vectors $\mathbf{y}$ and $\mathbf{y}'$ is defined in [10] by a combination of the distance between $s(\mathbf{y})$ and $s(\mathbf{y}')$, and $\pi(\mathbf{y})$ and $\pi(\mathbf{y}')$. The shape of a vector can be formally defined as follows:

Definition 1. *Shape of a vector in $\mathcal{R}^m$: Given a vector $\mathbf{y} \in \mathcal{R}^m$, a permutation $\pi(\mathbf{y}) = \{\pi_1, \ldots, \pi_m\}$, $\pi_i \in \{1, \ldots, m\}$, is the shape of $\mathbf{y}$ iff:*

$$y_{\pi_i} \leq y_{\pi_j}, \forall i < j$$

As an example, let $\mathbf{y} = [0.1, 0.5, 0.3]$ and $\mathbf{y}' = [0.3, 0.1, 0.5]$, then, their shapes are $\pi(\mathbf{y}) = [1, 3, 2]$ and $\pi(\mathbf{y}') = [2, 1, 3]$.

This work proposes the shape of a vector to divide the population in groups of solutions having a similar shape. The number of different shapes grows factorially with the number of objectives, i.e. for three objectives there are 6 different shapes, but for 5 objectives there are 120 possible shapes. Therefore, it is not practical for a large number of objectives to split the solutions according to all possible shapes, but by means of a clustering method considering the similarity between the shapes of objective vectors. There are several measures that can be used to measure the distance between permutations. In this work, the Spearman's rho distance, defined as follows, is considered [11].

Definition 2. *Spearman's rho distance: Given permutations π and π', and interpreting π_i as the position of element i in π, the Spearman's rho distance is defined as:*

$$\rho(\pi, \pi') = (\sum_{i=1}^{M} |\pi_i - \pi'_i|^2)^{(1/2)} \tag{3}$$

Algorithm 2 presents a clustering procedure using the shape of objective functions. First, a set $\{C_1, C_2, \ldots, C_\tau\}$ of τ clusters are initialized at empty. Then, for each element $\mathbf{x}$ in the population to be classified P, a normalized objective value $\hat{\mathbf{F}}(\mathbf{x})$ is calculated with its corresponding shape $\pi_{\mathbf{x}} = \pi(\hat{\mathbf{F}}(\mathbf{x}))$. Thereafter, τ different shapes are randomly selected from the set of shapes of the normalized objective values into a set $\mathcal{S}$. If eventually there are less than τ different shapes, $\mathcal{S}$ is completed by repeated elements. An index Id corresponding to each cluster is assigned to each shape in $\mathcal{S}$. Next, for each $\mathbf{x}$ in P, a set $\mathcal{S}'_{\mathbf{x}}$ containing the indexes of the shapes in $\mathcal{S}$ having the minimal Spearman's rho distance to $\pi_{\mathbf{x}}$ is created. The set $\mathcal{S}'_{\mathbf{x}}$ is used to select the index Id of the cluster in which $\mathbf{x}$ will be included. If the cardinality of $\mathcal{S}'_{\mathbf{x}}$ is one, the index is the value of the unique element in $\mathcal{S}'_{\mathbf{x}}$; otherwise, one of the indexes in $\mathcal{S}'_{\mathbf{x}}$ is selected at random. Finally, $\mathbf{x}$ is included in the cluster C_{Id}. The procedure finishes when all individuals are assigned to a cluster.

Algorithm 2. Clustering algorithm using the shape of objective functions

Initialize $\{C_1, C_2, \ldots, C_\tau\}$ clusters, s.t. $C_{Id} = \emptyset$ for $Id \in [1, \ldots, \tau]$
For each $\mathbf{x} \in P$ obtain its normalized objective value $\hat{\mathbf{F}}(\mathbf{x})$ and shape $\pi_{\mathbf{x}} = \pi(\hat{\mathbf{F}}(\mathbf{x}))$
Randomly select τ different shapes $\mathcal{S} = \{sh_1, sh_2, \ldots, sh_\tau\}$ from shapes of $\mathbf{x} \in P$
for each $\mathbf{x} \in P$ **do**
  $\mathcal{S}' = \{\ Id \mid sh_{Id} \in \min_{sh \in Sh}\{\rho(\pi_{\mathbf{x}}, sh)\}\}$
  if($|\mathcal{S}'| == 1$) **then** Id is the element of $\mathcal{S}'$
  else select Id at random from $\mathcal{S}'$
  end if
  Set $C_{Id} = C_{Id} \cup \mathbf{x}$
end for

4 Experimental Comparison

4.1 Experimental Setup and Metrics

The main focus of this work is to compare the shape-based clustering method to parallelize MOEAs based on preference relations with their sequential counterparts and other population division methods over a set of many-objective optimization problems. Thus, using the framework presented in Subsection 2.3, three partition methods were implemented to parallelize the original NSGA-II and six variants of it considering the relations explained in Subsection 2.2. The considered partition methods are: the proposed clustering method based on the shape of objective vectors (SH), a random (RN) partition and a k-means based partitioning (KM). The test problems used in this work are the DTLZ1 to DTLZ7 problems with 10 objectives [6]. The programs were implemented in C language and the OpenMP library.

For each problem 10 runs were executed for sequential and parallel programs using 2, 4, and 8 threads. The choice of the number of runs was made taking into account other works such as [8,14], and the available time to execute the programs and analyse the results (note that there is a total of 4900 executions). The experimental computational platform was a machine provided with two Intel Xeon quad-core Processors E5640 (12M Cache, 2.66 GHz, 5.86 GT/s) and 16 GB of main memory running the GNU/Linux operating system. The programs consider the following common parameters: population size 400, it_g is 400, binary coding of 32 bits per variable, one point crossover probability of 0.8, mutation probability of 0.002. For the ϵ-Preferred relation, the ϵ value is 0.0001; for the $(1-k)-$dominance relation, k is 0.5; and, for the $(1-k_F)-$dominance relation, k_F is also 0.5 and a fuzzy trapezoidal rule is used ($a = -0.001$, $b = 0$, $c = 0$, $d = 0.001$) [12]. The k and k_F values were selected taking into account the test cases in [12], while the values used for ϵ and the fuzzy trapezoidal rule were selected on experimental basis, however, no fine tuning of the above parameters was considered. For parallel methods $it_s = 0$, it_p is set to 400, it_c is 1. The sequential version is also implemented using the framework, but, in this case, $\tau = 1$ and $it_p = 0$.

The obtained results were evaluated using the Generational Distance (GD) and Spread (Δ) metrics. GD measures the average distance between obtained solutions in objective space and the true Pareto Front of the problem, while Spread evaluates the extent of the Pareto Front covered by the obtained set of solutions. Since GD requires a reference $\mathcal{PF}^\star$ to be computed, and equations to produce $\mathcal{PF}^\star$ are known, a set of 2000 optimal solutions was determined analytically. Both metrics are expected to be minimized.

4.2 Experimental Results

Table 1 and Table 2 show the average GD and Spread values performing 10 runs of implemented combinations, for each DTLZ problem. In these tables, for each problem, there are 10 rows corresponding to each execution type: one

for the sequential execution (Seq) and 3 for each partition method used for parallelization considering 2, 4, and 8 threads (τ value), while each column is for different MOEAs tested in this work. The values in parenthesis show the ranking obtained by each execution type of the given MOEA for each problem. Note that, until the value is represented by using a reduced number of digits, rankings are calculated using the machine numeric representation. Also, the best value for each column (execution method) is boldfaced. In order to summarize the results, the two last columns indicate the average of the rankings of each row, and regarding the data in this column, the last column labelled "Final" shows the overall rank obtained by each execution type for each problem.

The results in Table 1 and Table 2 may be used to determine the combination of partition method, MOEA and number of threads, performing the best for each problem and metric. However, in this work, the detailed results are not analyzed but the general and average behaviour of the studied partition methods in order to show how clustering may serve as a basis to develop pMOEAs for many-objective problems. At first glance, Table 1 shows that in almost all cases, for each MOEA considered, at least one parallel implementation evaluates better than its corresponding sequential counterparts for the GD metric. Considering the seven DTLZ problems and seven implemented MOEAs, only in 4 out of 49 results obtained by sequential implementations are not improved by parallel executions for GD. The result is remarkable since parallel and sequential implementations were executed using the same number of iterations, and, therefore, the same number of objective function evaluations. Therefore, the source of the benefit is provided by the interactions among individuals in subpopulations and not by executing more evaluations in the same execution time.

Figure 1 and Figure 2 show the final ranking of each combination of partition method and number of threads by problem using data in Table 1 and Table 2, respectivelly. The labels indicate the τ value followed by the partition method, i.e. 2-KM is for implementations using 2 threads and k-means. As it can be noted, in Figure 1 the smaller bars are for the SH implementations. In fact, in 5 out of 7 problems, an SH implementation obtains the best rank value for GD, in one problem a method based on k-means and in another case a method based on random partitioning. Also, the figure indicates that the best performance is for 8-SH obtaining the best ranking positions for almost all problems. In fact, the worst value obtained by the 8-SH is 5, in problem DTLZ6, whereas the other partitioning options receive in at least one problem an overall ranking greater than 5. From a visual inspection of Figure 2, it is not possible to determine which implementation alternative may be considered as the best, however it appears that sequential implementation obtain the best ranking in three problems, and that the 8-KM obtains, in general, the worse values.

Figure 3 presents the ranking distribution for GD metric considering the 49 implementations of each sequential and parallel MOEAs with different number of threads (7 MOEAs times 7 DTLZ problems). According to this figure the results of shape-based implementations concentrates in the first ranks while they have the fewer number of implementations in position 10. Thus, using the Figure 3 as

Table 1. Average values for GD metric with 10 objectives

Test	τ	Part	ϵ−DOM	ϵ−Pref	$(1-k)$	$(1-k_F)$	Favour	NSGAII	PO_k	Ranking Avg.	Ranking Final
DTLZ1	1	0	1.05E-1(6)	1.28E+0(10)	2.11E-1(4)	1.99E-1(10)	2.50E-1(2)	3.40E+0(6)	**8.77E-2(1)**	5.57	5
	2	KM	1.53E-1(10)	2.14E-1(3)	2.31E-1(7)	1.95E-1(9)	3.88E-1(6)	2.85E+0(5)	1.27E-1(6)	6.57	10
		RN	9.24E-2(3)	3.87E-1(6)	2.33E-1(8)	1.47E-1(5)	**2.39E-1(1)**	5.31E+0(8)	1.21E-1(4)	5	4
		SH	8.24E-2(2)	6.13E-1(9)	2.00E-1(3)	1.54E-1(6)	2.65E-1(3)	3.75E+0(7)	1.69E-1(10)	5.71	6
	4	KM	9.26E-2(4)	1.75E-1(2)	3.21E-1(10)	1.43E-1(4)	4.12E-1(7)	1.14E+0(2)	1.10E-1(3)	4.57	2
		RN	9.34E-2(5)	3.40E-1(5)	2.34E-1(9)	1.60E-1(7)	3.07E-1(5)	5.47E+0(9)	1.25E-1(5)	6.43	9
		SH	1.14E-1(7)	4.62E-1(8)	2.15E-1(5)	1.19E-1(3)	4.88E-1(8)	2.58E+0(4)	1.62E-1(9)	6.29	8
	8	KM	1.27E-1(9)	4.22E-1(7)	1.56E-1(2)	1.02E-1(2)	5.11E-1(9)	**1.09E+0(1)**	1.08E-1(2)	4.57	3
		RN	**5.35E-2(1)**	2.36E-1(4)	2.28E-1(6)	1.67E-1(8)	2.75E-1(4)	1.11E+1(10)	1.61E-1(8)	5.86	7
		SH	1.22E-1(8)	**1.35E-1(1)**	**1.46E-1(1)**	**9.89E-2(1)**	6.98E-1(10)	1.15E+0(3)	1.47E-1(7)	**4.43**	**1**
DTLZ2	1	0	6.63E-3(9)	6.18E-5(7)	2.02E-3(10)	1.38E-3(8)	2.90E-5(7)	4.92E-2(8)	5.83E-3(10)	8.43	10
	2	KM	5.90E-3(7)	3.21E-4(8)	6.42E-4(8)	1.33E-3(4)	4.89E-5(8)	4.59E-2(6)	2.28E-3(8)	7	8
		RN	6.84E-3(10)	3.62E-5(6)	1.47E-5(5)	1.96E-3(10)	1.97E-5(5)	4.88E-2(7)	4.86E-3(9)	7.43	9
		SH	5.12E-3(4)	2.55E-5(5)	1.83E-5(6)	1.38E-3(7)	1.66E-5(4)	4.58E-2(5)	2.02E-3(6)	5.29	4
	4	KM	5.07E-3(3)	1.08E-3(9)	1.15E-3(9)	1.33E-3(3)	6.76E-4(9)	4.16E-2(3)	5.47E-4(3)	5.57	7
		RN	6.00E-3(8)	1.92E-5(4)	**1.32E-5(1)**	1.36E-3(6)	1.54E-5(2)	4.97E-2(9)	2.22E-3(7)	5.29	5
		SH	**4.60E-3(1)**	1.86E-5(3)	1.38E-5(3)	1.73E-3(9)	1.98E-5(6)	4.33E-2(4)	8.21E-4(5)	4.43	2
	8	KM	5.68E-3(6)	1.54E-3(10)	4.42E-4(7)	**1.25E-3(1)**	1.51E-3(10)	**3.47E-2(1)**	3.55E-4(2)	5.29	6
		RN	5.37E-3(5)	1.65E-5(2)	1.41E-5(4)	1.36E-3(5)	1.65E-5(3)	5.12E-2(10)	7.92E-4(4)	4.71	3
		SH	4.73E-3(2)	**1.54E-5(1)**	1.38E-5(2)	1.32E-3(2)	**1.42E-5(1)**	3.85E-2(2)	3.32E-4(1)	**1.57**	**1**
DTLZ3	1	0	6.82E-1(2)	6.01E+0(9)	5.57E+0(10)	1.14E+0(10)	2.70E+0(5)	2.84E+0(6)	6.41E-1(3)	6.43	7
	2	KM	7.14E-1(4)	6.90E+0(10)	3.22E+0(9)	1.09E+0(7)	3.53E+0(7)	3.06E+0(7)	6.64E-1(4)	6.86	9
		RN	7.70E-1(8)	3.97E+0(4)	2.42E+0(6)	1.08E+0(6)	3.75E+0(9)	4.02E+0(8)	7.02E-1(5)	6.57	8
		SH	7.95E-1(9)	4.60E+0(7)	2.79E+0(8)	9.53E-1(5)	3.70E+0(8)	2.50E+0(5)	7.84E-1(7)	7	10
	4	KM	7.35E-1(5)	4.75E+0(8)	2.02E+0(5)	9.26E-1(3)	3.89E+0(10)	1.30E+0(2)	8.51E-1(9)	6	5
		RN	6.95E-1(3)	4.39E+0(6)	1.82E+0(3)	1.13E+0(8)	2.63E+0(4)	5.34E+0(9)	7.41E-1(6)	5.57	4
		SH	7.54E-1(6)	4.16E+0(5)	2.72E+0(7)	9.44E-1(4)	**2.01E+0(1)**	1.92E+0(4)	8.90E-1(10)	5.29	3
	8	KM	**6.16E-1(1)**	**2.51E+0(1)**	1.84E+0(4)	**7.12E-1(1)**	3.52E+0(6)	**1.14E+0(1)**	**5.50E-1(1)**	2.14	1
		RN	9.96E-1(10)	3.00E+0(3)	**1.40E+0(1)**	1.14E+0(9)	2.12E+0(2)	1.87E+1(10)	8.07E-1(8)	6.14	6
		SH	7.68E-1(7)	2.90E+0(2)	1.77E+0(2)	7.77E-1(2)	2.59E+0(3)	1.85E+0(3)	6.27E-1(2)	3	2
DTLZ4	1	0	6.74E-3(9)	2.55E-2(10)	4.62E-2(10)	1.83E-3(3)	4.16E-3(6)	6.79E-2(7)	1.13E-2(4)	7	10
	2	KM	**3.31E-3(1)**	3.58E-3(3)	9.01E-3(8)	3.14E-3(6)	7.02E-3(10)	6.77E-2(6)	1.81E-2(9)	6.14	7
		RN	6.99E-3(10)	6.32E-3(8)	7.61E-3(7)	**1.22E-3(1)**	4.27E-3(7)	7.01E-2(8)	1.39E-2(7)	6.86	9
		SH	5.13E-3(5)	3.70E-3(4)	2.94E-3(4)	2.36E-3(4)	2.64E-3(2)	6.73E-2(5)	1.27E-2(6)	4.29	2
	4	KM	3.78E-3(2)	3.04E-3(2)	1.38E-2(9)	3.56E-3(7)	3.39E-3(5)	6.36E-2(3)	2.11E-2(10)	5.43	5
		RN	5.85E-3(7)	4.78E-3(7)	1.51E-3(1)	2.81E-3(5)	5.28E-3(9)	7.21E-2(9)	1.24E-2(5)	6.14	8
		SH	4.40E-3(4)	2.64E-3(1)	3.68E-3(5)	4.53E-3(9)	**2.28E-3(1)**	6.57E-2(4)	1.08E-2(3)	**3.86**	**1**
	8	KM	5.61E-3(6)	3.93E-3(5)	4.29E-3(6)	7.66E-3(10)	2.88E-3(3)	5.87E-2(1)	1.49E-2(8)	5.57	6
		RN	6.55E-3(8)	9.10E-3(9)	1.51E-3(2)	1.76E-3(2)	3.12E-3(4)	7.35E-2(10)	9.02E-3(2)	5.29	4
		SH	4.17E-3(3)	4.52E-3(6)	2.29E-3(3)	3.65E-3(8)	4.54E-3(8)	6.00E-2(2)	**8.36E-3(1)**	4.43	3
DTLZ5	1	0	3.85E-2(2)	1.34E-5(4)	1.25E-1(7)	**1.30E-5(1)**	1.38E-1(10)	1.20E-1(9)	**1.14E-1(1)**	4.86	2
	2	KM	4.02E-2(9)	1.95E-5(9)	1.27E-1(8)	1.38E-5(5)	1.29E-1(9)	1.17E-1(5)	1.20E-1(3)	6.86	10
		RN	3.95E-2(7)	**1.26E-5(1)**	1.25E-1(5)	1.32E-5(3)	1.28E-1(4)	1.18E-1(7)	1.25E-1(4)	4.43	1
		SH	3.85E-2(3)	1.28E-5(3)	1.25E-1(6)	1.56E-5(7)	1.28E-1(7)	1.18E-1(6)	1.25E-1(5)	5.29	4
	4	KM	3.86E-2(4)	1.46E-5(7)	1.28E-1(9)	1.58E-5(9)	1.28E-1(6)	1.14E-1(3)	1.17E-1(2)	5.71	7
		RN	4.04E-2(10)	1.27E-5(2)	1.25E-1(4)	1.32E-5(2)	1.28E-1(5)	1.18E-1(8)	1.25E-1(6)	5.29	5
		SH	3.91E-2(6)	1.42E-5(6)	1.25E-1(2)	1.56E-5(8)	1.28E-1(8)	1.14E-1(4)	1.25E-1(7)	5.86	8
	8	KM	**3.81E-2(1)**	2.04E-5(10)	1.32E-1(10)	3.24E-3(10)	1.27E-1(2)	**1.02E-1(1)**	1.29E-1(10)	6.29	9
		RN	4.01E-2(8)	1.40E-5(5)	**1.25E-1(1)**	1.36E-5(4)	**1.25E-1(1)**	1.28E-1(10)	1.25E-1(8)	5.29	6
		SH	3.90E-2(5)	1.81E-5(8)	1.25E-1(3)	1.52E-5(6)	1.27E-1(3)	1.11E-1(2)	1.25E-1(9)	5.14	3
DTLZ6	1	0	1.75E-1(7)	2.98E+0(7)	**6.36E-4(1)**	2.69E+0(5)	2.73E-3(7)	4.57E-1(5)	4.19E-3(7)	5.57	7
	2	KM	1.75E-1(8)	2.63E+0(3)	1.93E-2(10)	3.13E+0(10)	1.23E-2(10)	4.53E-1(2)	5.95E-3(8)	7.29	10
		RN	1.64E-1(3)	2.76E+0(5)	7.55E-4(5)	2.77E+0(7)	3.48E-4(6)	4.65E-1(6)	4.14E-3(6)	5.43	6
		SH	1.66E-1(4)	3.00E+0(8)	6.51E-4(2)	2.93E+0(8)	3.35E-4(5)	**4.53E-1(1)**	2.94E-3(3)	**4.43**	**1**
	4	KM	1.87E-1(9)	2.40E+0(2)	1.32E-2(9)	2.75E+0(6)	9.25E-3(8)	4.56E-1(3)	1.03E-2(9)	6.57	8
		RN	1.58E-1(2)	3.08E+0(9)	8.53E-4(6)	2.20E+0(2)	**2.39E-4(1)**	4.69E-1(8)	3.79E-3(5)	4.71	4
		SH	1.70E-1(5)	2.72E+0(4)	6.79E-4(4)	2.98E+0(9)	2.48E-4(4)	4.57E-1(4)	2.55E-3(2)	4.57	3
	8	KM	1.88E-1(10)	**1.83E+0(1)**	1.27E-2(8)	2.24E+0(3)	9.49E-3(9)	4.67E-1(7)	1.24E-2(10)	6.86	9
		RN	**1.52E-1(1)**	2.90E+0(6)	1.51E-3(7)	**2.11E+0(1)**	2.44E-4(3)	4.75E-1(9)	3.40E-3(4)	4.43	2
		SH	1.71E-1(6)	3.14E+0(10)	6.69E-4(3)	2.35E+0(4)	2.40E-4(2)	4.75E-1(10)	**1.55E-3(1)**	5.14	5
DTLZ7	1	0	8.44E-3(10)	9.11E-5(8)	6.28E-3(10)	1.34E-3(7)	2.93E-4(7)	4.67E-2(8)	7.10E-3(10)	8.57	10
	2	KM	6.32E-3(6)	5.35E-5(7)	2.14E-3(9)	1.34E-3(5)	3.08E-4(8)	4.32E-2(6)	3.15E-3(6)	6.71	9
		RN	7.93E-3(9)	4.53E-5(6)	2.70E-5(6)	1.31E-3(4)	2.37E-5(3)	4.67E-2(7)	5.43E-3(9)	6.29	7
		SH	4.40E-3(2)	4.39E-5(5)	2.67E-5(5)	1.34E-3(6)	2.51E-5(4)	4.30E-2(5)	3.86E-3(7)	4.86	3
	4	KM	4.75E-3(4)	1.43E-3(9)	1.07E-3(7)	1.25E-3(2)	8.48E-4(9)	3.76E-2(3)	8.80E-4(3)	5.29	5
		RN	7.68E-3(8)	3.27E-5(3)	2.46E-5(2)	1.62E-3(9)	2.56E-5(5)	4.97E-2(10)	4.12E-3(8)	6.43	8
		SH	4.65E-3(3)	3.64E-5(4)	**2.44E-5(1)**	1.35E-3(8)	2.79E-5(6)	3.97E-2(4)	1.44E-3(4)	4.29	2
	8	KM	4.80E-3(5)	2.03E-3(10)	1.66E-3(8)	**1.23E-3(1)**	1.28E-3(10)	**3.15E-2(1)**	**6.09E-4(1)**	5.14	4
		RN	6.92E-3(7)	2.93E-5(2)	2.56E-5(3)	1.88E-3(10)	**1.98E-5(1)**	4.94E-2(9)	1.60E-3(5)	5.29	6
		SH	**4.22E-3(1)**	**2.67E-5(1)**	2.57E-5(4)	1.30E-3(3)	2.12E-5(2)	3.60E-2(2)	6.60E-4(2)	**2.14**	**1**

Table 2. Average values for Spread metric with 10 objectives

Test	τ	Part	ϵ−DOM	ϵ−Pref	$(1-k)$	$(1-k_F)$	Favour	NSGAII	PO_k	Ranking Avg.	Ranking Final
DTLZ1	1	0	5.95E-1(3)	1.07E+0(6)	**1.02E+0(1)**	9.08E-1(8)	1.06E+0(5)	5.47E-1(7)	**5.81E-1(1)**	4.43	4
		KM	6.07E-1(5)	1.06E+0(5)	1.09E+0(6)	8.35E-1(2)	1.04E+0(2)	5.31E-1(6)	7.28E-1(2)	4.00	2
	2	RN	5.75E-1(2)	1.09E+0(8)	1.04E+0(2)	**8.33E-1(1)**	1.05E+0(4)	4.96E-1(4)	8.30E-1(4)	3.57	1
		SH	**5.59E-1(1)**	**1.00E+0(1)**	1.11E+0(9)	9.11E-1(9)	**1.03E+0(1)**	4.66E-1(2)	1.06E+0(7)	4.29	3
		KM	6.42E-1(7)	1.08E+0(7)	1.06E+0(4)	9.75E-1(10)	1.05E+0(3)	6.53E-1(9)	7.47E-1(3)	6.14	8
	4	RND	6.78E-1(8)	1.02E+0(2)	1.06E+0(3)	8.78E-1(6)	1.09E+0(8)	4.79E-1(3)	9.28E-1(5)	5.00	5
		SH	6.03E-1(4)	1.06E+0(4)	1.06E+0(5)	8.40E-1(3)	1.11E+0(9)	5.04E-1(5)	1.11E+0(8)	5.43	6
		KM	7.18E-1(9)	1.16E+0(10)	1.09E+0(7)	8.87E-1(7)	1.08E+0(7)	6.70E-1(10)	9.88E-1(6)	8.00	9
	8	RND	7.32E-1(10)	1.05E+0(3)	1.11E+0(8)	8.72E-1(5)	1.07E+0(6)	**4.12E-1(1)**	1.23E+0(9)	6.00	7
		SH	6.12E-1(6)	1.14E+0(9)	1.18E+0(10)	8.48E-1(4)	1.21E+0(10)	5.85E-1(8)	1.52E+0(10)	8.14	10
DTLZ2	1	0	4.82E-1(3)	**1.00E+0(1)**	1.04E+0(2)	7.34E-1(3)	**1.00E+0(1)**	2.77E-1(10)	**6.92E-1(1)**	3.00	1
		KM	4.86E-1(4)	1.03E+0(3)	1.22E+0(4)	7.31E-1(2)	1.03E+0(2)	2.44E-1(4)	8.58E-1(3)	3.14	2
	2	RN	5.45E-1(7)	1.00E+0(2)	**1.00E+0(1)**	7.51E-1(7)	**1.00E+0(1)**	2.73E-1(9)	7.58E-1(2)	4.14	6
		SH	4.81E-1(2)	1.00E+0(2)	**1.00E+0(1)**	7.61E-1(9)	**1.00E+0(1)**	2.51E-1(7)	1.09E+0(4)	3.71	3
		KM	5.38E-1(6)	1.10E+0(4)	1.34E+0(5)	7.51E-1(6)	1.14E+0(3)	2.35E-1(2)	1.20E+0(6)	4.57	8
	4	RND	5.77E-1(9)	1.00E+0(2)	**1.00E+0(1)**	7.52E-1(8)	**1.00E+0(1)**	2.53E-1(8)	1.17E+0(5)	4.86	10
		SH	**4.78E-1(1)**	1.00E+0(2)	**1.00E+0(1)**	7.82E-1(10)	**1.00E+0(1)**	2.47E-1(5)	1.45E+0(8)	4.00	5
		KM	5.72E-1(8)	1.31E+0(5)	1.17E+0(3)	7.49E-1(5)	1.23E+0(4)	**2.33E-1(1)**	1.32E+0(7)	4.71	9
	8	RND	6.54E-1(10)	1.00E+0(2)	**1.00E+0(1)**	**7.30E-1(1)**	**1.00E+0(1)**	2.49E-1(6)	1.56E+0(9)	4.29	7
		SH	4.92E-1(5)	1.00E+0(2)	**1.00E+0(1)**	7.38E-1(4)	**1.00E+0(1)**	2.39E-1(3)	1.63E+0(10)	3.71	4
DTLZ3	1	0	5.84E-1(5)	1.00E+0(2)	**1.00E+0(1)**	7.37E-1(5)	**1.00E+0(1)**	4.83E-1(8)	**5.76E-1(1)**	3.29	1
		KM	5.92E-1(6)	1.00E+0(4)	1.01E+0(3)	7.48E-1(7)	1.01E+0(5)	5.11E-1(9)	7.41E-1(3)	5.29	8
	2	RN	6.40E-1(7)	1.00E+0(5)	1.01E+0(4)	7.71E-1(9)	1.03E+0(6)	4.61E-1(6)	6.20E-1(2)	5.57	9
		SH	**5.47E-1(1)**	1.00E+0(3)	1.01E+0(6)	7.12E-1(2)	1.00E+0(4)	4.69E-1(7)	9.17E-1(5)	4.00	3
		KM	5.74E-1(3)	1.03E+0(6)	1.05E+0(7)	7.16E-1(3)	1.05E+0(7)	4.24E-1(4)	1.01E+0(6)	5.14	6
	4	RND	6.95E-1(8)	**1.00E+0(1)**	1.00E+0(2)	7.44E-1(6)	1.00E+0(2)	4.28E-1(5)	8.98E-1(4)	4.00	4
		SH	5.83E-1(4)	**1.00E+0(1)**	1.00E+0(2)	7.25E-1(4)	1.00E+0(2)	3.87E-1(3)	1.28E+0(8)	3.43	2
		KM	7.91E-1(9)	1.13E+0(7)	1.16E+0(8)	**6.93E-1(1)**	1.22E+0(8)	5.66E-1(10)	1.13E+0(7)	7.14	10
	8	RND	9.72E-1(10)	**1.00E+0(1)**	1.00E+0(2)	8.09E-1(10)	1.00E+0(2)	3.67E-1(2)	1.28E+0(9)	5.14	7
		SH	5.63E-1(2)	**1.00E+0(1)**	1.01E+0(5)	7.70E-1(8)	1.00E+0(3)	**3.66E-1(1)**	1.56E+0(10)	4.29	5
DTLZ4	1	0	1.65E+0(8)	1.21E+0(6)	1.29E+0(6)	1.29E+0(4)	1.00E+0(2)	2.49E-1(10)	1.17E+0(4)	5.71	6
		KM	1.67E+0(10)	1.40E+0(8)	1.46E+0(8)	1.54E+0(7)	1.19E+0(7)	2.24E-1(2)	9.47E-1(2)	6.29	7
	2	RN	1.59E+0(3)	1.05E+0(2)	1.22E+0(4)	**1.23E+0(1)**	1.09E+0(3)	2.40E-1(9)	1.22E+0(6)	4.00	2
		SH	1.64E+0(5)	1.16E+0(5)	**1.02E+0(1)**	1.26E+0(3)	**1.00E+0(1)**	2.28E-1(5)	1.20E+0(5)	3.57	1
		KM	1.65E+0(7)	1.44E+0(9)	1.53E+0(10)	1.67E+0(10)	1.29E+0(9)	**2.22E-1(1)**	**9.46E-1(1)**	6.71	8
	4	RND	**1.50E+0(1)**	1.09E+0(3)	1.25E+0(5)	1.36E+0(5)	1.15E+0(5)	2.33E-1(8)	1.24E+0(7)	4.86	4
		SH	1.61E+0(4)	**1.03E+0(1)**	1.13E+0(3)	1.50E+0(6)	1.16E+0(6)	2.27E-1(4)	1.32E+0(8)	4.57	3
		KM	1.66E+0(9)	1.61E+0(10)	1.52E+0(9)	1.66E+0(9)	1.57E+0(10)	2.24E-1(3)	1.14E+0(3)	7.57	10
	8	RND	1.56E+0(2)	1.09E+0(4)	1.30E+0(7)	1.25E+0(2)	1.14E+0(4)	2.30E-1(6)	1.36E+0(9)	4.86	5
		SH	1.64E+0(6)	1.24E+0(7)	1.04E+0(2)	1.57E+0(8)	1.23E+0(8)	2.30E-1(7)	1.44E+0(10)	6.86	9
DTLZ5	1	0	4.60E-1(8)	1.00E+0(2)	1.02E+0(5)	**1.00E+0(1)**	1.01E+0(7)	3.94E-1(9)	1.06E+0(7)	5.57	8
		KM	4.91E-1(10)	1.00E+0(5)	1.14E+0(8)	1.00E+0(2)	1.19E+0(8)	3.12E-1(4)	1.10E+0(8)	6.43	9
	2	RN	4.41E-1(5)	1.00E+0(3)	1.02E+0(3)	**1.00E+0(1)**	1.00E+0(6)	3.56E-1(7)	1.03E+0(4)	4.14	6
		SH	4.50E-1(7)	1.00E+0(2)	1.02E+0(4)	**1.00E+0(1)**	1.00E+0(5)	3.12E-1(5)	1.01E+0(3)	3.86	5
		KM	4.38E-1(3)	**1.00E+0(1)**	1.14E+0(9)	**1.00E+0(1)**	1.21E+0(10)	3.07E-1(3)	1.21E+0(10)	5.29	7
	4	RND	4.34E-1(2)	1.00E+0(3)	1.02E+0(2)	**1.00E+0(1)**	1.00E+0(2)	3.43E-1(6)	1.03E+0(5)	3.00	1
		SH	4.47E-1(6)	1.00E+0(6)	1.05E+0(7)	**1.00E+0(1)**	**1.00E+0(1)**	2.85E-1(2)	1.01E+0(2)	3.57	3
		KM	4.71E-1(9)	1.00E+0(4)	1.16E+0(10)	1.04E+0(3)	1.20E+0(9)	4.40E-1(10)	1.14E+0(9)	7.71	10
	8	RND	4.38E-1(4)	1.00E+0(4)	**1.01E+0(1)**	**1.00E+0(1)**	1.00E+0(3)	3.68E-1(8)	**1.00E+0(1)**	3.14	2
		SH	**4.31E-1(1)**	1.00E+0(7)	1.04E+0(6)	**1.00E+0(1)**	1.00E+0(4)	**2.80E-1(1)**	1.03E+0(6)	3.71	4
DTLZ6	1	0	6.57E-1(5)	**9.95E-1(1)**	1.02E+0(7)	1.01E+0(8)	1.01E+0(7)	4.00E-1(10)	1.02E+0(7)	6.43	7
		KM	6.50E-1(4)	1.01E+0(8)	1.37E+0(10)	9.97E-1(2)	1.29E+0(10)	3.13E-1(5)	1.05E+0(8)	6.71	8
	2	RN	7.05E-1(8)	9.99E-1(4)	1.01E+0(2)	1.00E+0(5)	1.00E+0(6)	3.69E-1(9)	1.01E+0(4)	5.43	6
		SH	6.72E-1(6)	9.96E-1(2)	1.01E+0(5)	1.00E+0(6)	1.00E+0(5)	3.20E-1(7)	1.00E+0(2)	4.71	3
		KM	6.46E-1(2)	1.03E+0(9)	1.25E+0(9)	1.04E+0(10)	1.27E+0(9)	2.75E-1(2)	1.10E+0(9)	7.14	10
	4	RND	7.66E-1(9)	1.00E+0(7)	**1.00E+0(1)**	**9.97E-1(1)**	1.00E+0(3)	3.21E-1(8)	1.01E+0(6)	5.00	5
		SH	6.49E-1(3)	1.00E+0(5)	1.01E+0(6)	9.98E-1(3)	1.00E+0(4)	2.82E-1(3)	**9.94E-1(1)**	3.57	1
		KM	**6.34E-1(1)**	1.08E+0(10)	1.25E+0(8)	1.04E+0(9)	1.27E+0(8)	**2.57E-1(1)**	1.19E+0(10)	6.71	9
	8	RND	8.10E-1(10)	9.99E-1(3)	1.01E+0(3)	1.00E+0(4)	1.00E+0(2)	3.15E-1(6)	1.01E+0(5)	4.71	4
		SH	6.76E-1(7)	1.00E+0(6)	1.01E+0(4)	1.01E+0(7)	**1.00E+0(1)**	2.85E-1(4)	1.00E+0(3)	4.57	2
DTLZ7	1	0	5.02E-1(3)	**1.00E+0(1)**	1.05E+0(4)	7.22E-1(5)	**1.00E+0(1)**	2.79E-1(10)	**5.99E-1(1)**	3.57	1
		KM	**4.86E-1(1)**	1.00E+0(3)	1.27E+0(5)	7.14E-1(4)	1.02E+0(4)	2.53E-1(7)	7.87E-1(3)	3.86	3
	2	RN	5.78E-1(8)	1.00E+0(4)	**1.00E+0(1)**	7.08E-1(3)	1.00E+0(2)	2.57E-1(8)	6.87E-1(2)	4.00	4
		SH	5.11E-1(5)	1.00E+0(2)	**1.00E+0(1)**	7.28E-1(7)	1.00E+0(2)	2.46E-1(5)	8.24E-1(4)	3.71	2
		KM	5.36E-1(6)	1.07E+0(8)	1.29E+0(6)	**6.95E-1(1)**	1.23E+0(6)	2.35E-1(2)	8.58E-1(5)	4.86	5
	4	RND	5.66E-1(7)	1.00E+0(5)	1.00E+0(3)	7.25E-1(6)	1.00E+0(2)	2.59E-1(9)	8.61E-1(6)	5.43	8
		SH	4.89E-1(2)	1.00E+0(7)	**1.00E+0(1)**	7.32E-1(9)	1.00E+0(3)	2.40E-1(4)	1.13E+0(8)	4.86	6
		KM	6.03E-1(10)	1.26E+0(9)	1.35E+0(7)	6.98E-1(2)	1.22E+0(5)	**2.35E-1(1)**	1.04E+0(7)	5.86	9
	8	RND	6.01E-1(9)	1.00E+0(6)	**1.00E+0(1)**	7.30E-1(8)	1.00E+0(2)	2.49E-1(6)	1.36E+0(9)	5.86	10
		SH	5.06E-1(4)	1.00E+0(6)	1.00E+0(2)	7.32E-1(10)	1.00E+0(2)	2.39E-1(3)	1.53E+0(10)	5.29	7

a complement to the average ranking values in the last two columns of Table 1, we can state that parallelization using shape-based clustering can be considered in general as the best parallelization option for GD improvement for the set of problems and MOEAs considered.

To statistically support the claims about the convenience of parallelization using shape-based clustering, Table 3 presents the number of wins among the implementations indicated in each column regarding implementations in each row for the Sign test pairwise comparisons [7] the obtained ranking using the GD and Spread metrics. In case that detected differences exists these are indicated in parenthesis, and the data is boldfaced. The values 32, 34, 36, and 37 are considered as the critical number of wins needed to achieve levels of significance of 0.1, 0.05, 0.02, and 0.01, respectively. In case of ties, the count is split evenly between the pair of compared algorithms. This table shows that the 8-SH has a significant improvement over the parallelization options using less than 8 threads. Also, as it is indicated, parallel approaches based on shape clustering clearly improve the results of the sequential implementations. For the Δ metric, there is a similar number of wins among sequential and paralelization approaches based on random and shape based partitioning, thus it can not be stated that one algorithm performs better than another considering this metric. However, for the Δ metric, all alternatives are better than the k-means parallelization using 8 threads.

Finally, Figure 3 also shows that considering the number of threads, the execution using 8 threads (clusters) clearly obtains a large number of GD rankings one and two than the other alternatives. The results suggest that using a large number of clusters may aid to improve the GD results of parallel executions using clustering based pMOEAs.

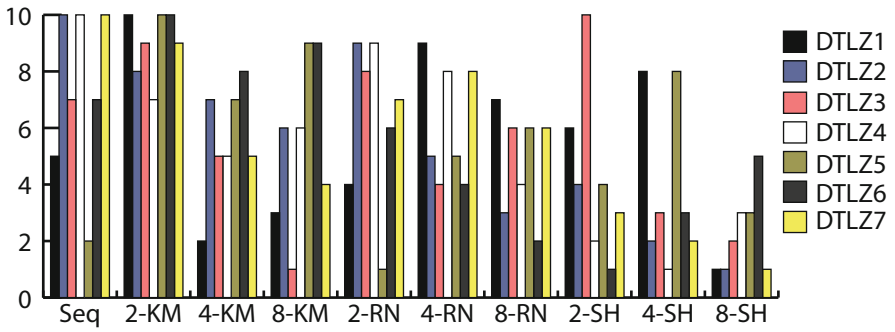

Fig. 1. Final GD ranking of each implementation option by problem

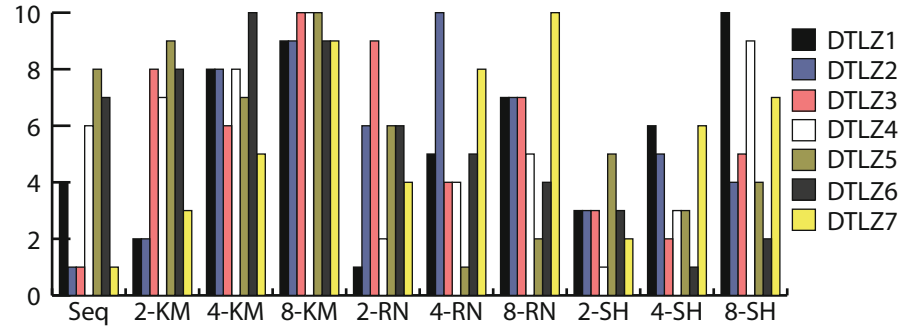

Fig. 2. Final Δ ranking of each implementation option by problem

Table 3. Number of wins and detected differences for Sing test for pairwise distribution of each implementation approach considering ranking values in Table 1 and Table 2

Generational distance(GD)										
	Seq	2-KM	2-RN	2-SH	4-KM	4-RN	4-SH	8-KM	8-RN	8-SH
Seq	24.5	24	30	**35(0.05)**	27	29	**34(0.05)**	30	31	**36(0.02)**
2-KM	25	24.5	28	**34(0.05)**	30	28	**36(0.02)**	**33(0.1)**	30	**39(0.01)**
2-RN	19	21	24.5	29	26	24	**32(0.1)**	29	27	**38(0.01)**
2-SH	14	15	20	24.5	23	22	29	24	25	**37(0.01)**
4-KM	22	19	23	26	24.5	25	23	28	26	**33(0.1)**
4-RN	20	21	25	27	24	24.5	28	26	26	**33(0.1)**
4-SH	15	13	17	20	26	21	24.5	23	23	**35(0.05)**
8-KM	19	16	20	25	21	23	26	24.5	25	27
8-RN	18	19	22	24	23	23	26	24	24.5	31
8-SH	13	10	11	12	16	16	14	22	18	24.5
Spread (Δ)										
	Seq	2-KM	2-RN	2-SH	4-KM	4-RN	4-SH	8-KM	8-RN	8-SH
Seq	24.5	15	26	29.5	15.5	24	28	9	25	21
2-KM	**34(0.05)**	24.5	29	**32(0.1)**	18	29	**34(0.05)**	12	28	25
2-RN	23	20	24.5	29	19.5	22	24.5	9	22	21.5
2-SH	19.5	17	20	24.5	14.5	20.5	24.5	12	23	17.5
4-KM	**33.5(0.1)**	31	29.5	**34.5(0.05)**	24.5	29.5	30.5	16	27.5	**32.5(0.1)**
4-RN	25	20	27	28.5	19.5	24.5	23.5	11	21	22
4-SH	21	15	24.5	24.5	18.5	25.5	24.5	15	22	15.5
8-KM	**40(0.01)**	**37(0.01)**	**40(0.01)**	**37(0.01)**	**33(0.1)**	**38(0.01)**	**34(0.05)**	24.5	**32.5(0.1)**	**34(0.05)**
8-RN	24	21	27	26	21.5	28	27	16.5	24.5	19.5
8-SH	28	24	27.5	31.5	16.5	27	**33.5(0.1)**	15	29.5	24.5

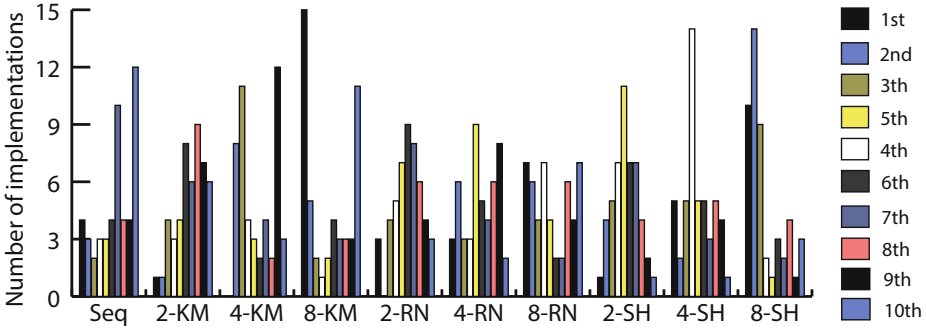

Fig. 3. Ranking distribution of GD by partition method and number of threads

5 Conclusions and Future Work

In evolutionary algorithms dealing with the simultaneous optimization of more than 4 objectives, the use of parallelization may be useful to improve their performances. One key aspect of this parallelization is the correct partitioning of the population into subpopulations that are to be distributed among processors. In this paper, we have proposed a method to do such partitioning based on what is known as the shape of the objective vector. We present an experimental comparison of the performance of the resulting parallel MOEAs and their sequential counterpart as well as a comparison of different partition methods.

The obtained results have shown that, in most of the studied cases, at least one parallel MOEAs outperform their sequential counterparts (45 out of 49 cases). Comparison results have also shown that, for the considered experimental setting and metrics, the parallel implementations based on the clustering of solutions using the shape of objective vectors obtains the best average rank in almost all considered problems (5 out of 7 cases).

Future work is aimed at extending the evaluation considering the hypervolume metric and increasing the number of objectives.

References

1. Branke, J., Schmeck, H., Deb, K., Reddy, M.: Parallelizing multi-objective evolutionary algorithms: cone separation. In: 2004 Congress on Evol. Comput., vol. 2, pp. 1952–1957. IEEE, Portland (2004)
2. Cheshmehgaz, H.R., Haron, H., Sharifi, A.: The review of multiple evolutionary searches and multi-objective evolutionary algorithms. Artificial Intelligence Review, 1–33 (2013)
3. Coello Coello, C.A., Lamont, G., Van Veldhuizen, D.: Evolutionary Algorithms for Solving Multi-Objective Problems, 2nd edn. Springer, New York (2007). ISBN: 978-0-387-33254-3
4. Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley (2001)
5. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. on Evol. Comput. (2002)
6. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: Proc. of the 2002 Congr. on Evol. Comput. (2002)
7. Derrac, J., García, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm and Evolutionary Computation **1**(1), 3–18 (2011)
8. di Pierro, F., Khu, S., Savić, D.: An investigation on Preference Order ranking scheme for multiobjective evolutionary optimization. IEEE Trans. on Evol. Comput. (2007)
9. Drechsler, N., Drechsler, R., Becker, B.: Multi-objective optimisation based on relation favour. In: First Int. Conf. on Evol. Multi-Criterion Optim. Springer (2001)
10. Egecioglu, Ö.: Parametric approximation algorithms for high-dimensional euclidean similarity. In: Siebes, A., De Raedt, L. (eds.) PKDD 2001. LNCS (LNAI), vol. 2168, pp. 79–90. Springer, Heidelberg (2001)

11. Fagin, R., Kumar, R., Sivakumar, D.: Comparing top k lists. In: Proc. of the 14th Annual ACM-SIAM Symp. on Discrete Algorithms. p. 36. SIAM (2003)
12. Farina, M., Amato, P.: On the Optimal Solution Definition for Many-criteria Optimization Problems. In: Proc. of the NAFIPS-FLINT Int. Conf. 2002, pp. 233–238. IEEE Service Center (2002)
13. Hiroyasu, T., Miki, M., Watanabe, S.: The new model of parallel genetic algorithm in multi-objective optimization problems: divided range multi-objective genetic algorithm. In: 2000 Congress on Evol. Comput., vol. 1, pp. 333–340. IEEE, New Jersey (2000)
14. Ishibuchi, H., Tsukamoto, N., Nojima, Y.: Evolutionary many-objective optimization: A short review. In: 2008 IEEE Congr. on Evol. Comput. (2008)
15. Jaimes, A.L., Coello Coello, C.A.: Applications of parallel platforms and models in evolutionary multi-objective optimization. In: Lewis, A., Mostaghim, S., Randall, M. (eds.) Biologically-Inspired Optimisation Methods. SCI, vol. 210, pp. 23–49. Springer, Heidelberg (2009)
16. Köppen, M., Yoshida, K.: Substitute distance assignments in NSGA-II for handling many-objective optimization problems. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 727–741. Springer, Heidelberg (2007)
17. von Lücken, C., Barán, B., Brizuela, C.: A survey on multi-objective evolutionary algorithms for many-objective problems. Comput. Optim. and Appl. **1**(1), 1–50 (2014)
18. Negro, F.D.T., Ortega, J., Ros, E., Mota, S., Paechter, B., Martín, J.M.: PSFGA: parallel processing and evolutionary comput. for multiobjective optim. Parallel Comput. (2004)
19. Streichert, F., Ulmer, H., Zell, A.: Parallelization of multi-objective evolutionary algorithms using clustering algorithms. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 92–107. Springer, Heidelberg (2005)
20. Sülflow, A., Drechsler, N., Drechsler, R.: Robust multi-objective optimization in high dimensional spaces. In: Obayashi, S., Deb, K., Poloni, C., Hiroyasu, T., Murata, T. (eds.) EMO 2007. LNCS, vol. 4403, pp. 715–726. Springer, Heidelberg (2007)
21. Talbi, E.-G., Mostaghim, S., Okabe, T., Ishibuchi, H., Rudolph, G., Coello Coello, C.A.: Parallel approaches for multiobjective optimization. In: Branke, J., Deb, K., Miettinen, K., Słowiński, R. (eds.) Multiobjective Optimization. LNCS, vol. 5252, pp. 349–372. Springer, Heidelberg (2008)