Selection Mechanisms based on the Maximin Fitness Function to solve Multi-Objective Optimization Problems

Adriana Menchaca-Mendez

CINVESTAV-IPN,
Departamento de Computación
Av. IPN 2508. San Pedro Zacatenco
México D.F. 07300, MÉXICO,
email: adriana.menchacamendez@gmail.com

Carlos A. Coello Coello[1]

CINVESTAV-IPN,
Departamento de Computación
Av. IPN 2508. San Pedro Zacatenco
México D.F. 07300, MÉXICO,
email: ccoello@cs.cinvestav.mx

Abstract

In this paper, we study three selection mechanisms based on the maximin fitness function and we propose another one. These selection mechanisms give rise to the following MOEAs: "MC-MOEA", "MD-MOEA", "MH-MOEA" and "MAH-MOEA". We validated them using standard test functions taken from the specialized literature, having from three up to ten objective functions. We compare these four MOEAs among them and also with respect to MOEA/D (which is based on decomposition), and to SMS-EMOA (which is based on the hypervolume indicator). Our preliminary results indicate that "MD-MOEA" and "MAH-MOEA" are promising alternatives for solving MOPs with either low or high dimensionality.

Keywords: Multi-objective evolutionary algorithms; selection operators; maximin fitness function

1. Introduction

In real-world applications there are many problems which involve the simultaneous optimization of multiple objective functions [1], which are normally in conflict with each other. They are called "*Multi-objective Optimization Problems (MOPs)*". Because of the conflicting nature of the objectives to be optimized, the notion of optimality refers in this case to finding the best possible trade-offs among the objectives (i.e., we aim to find solutions for which no objective can be improved without worsening another). Consequently, when solving MOPs we do not aim to find a single optimal solution but a set of them, which constitute the so-called Pareto optimal set, whose image is known as the Pareto front.

The use of evolutionary algorithms for solving MOPs has become very popular in the last few years [2], giving rise to the so-called Multi-Objective Evolutionary Algorithms (MOEAs).[2] MOEAs have two main goals: (i) to find solutions that are, as close as possible, to the true Pareto front and, (ii) to produce solutions that are spread along the Pareto front as uniformly as possible. We can talk of two types of MOEAs, if we classify them based on their selection mechanism: (i) those that incorporate the concept of Pareto optimality into their selection mechanism, and (ii) those that do not use Pareto dominance to select individuals. The use of Pareto-based selection has several limitations from which, its poor scalability with respect to the number of objective functions is, perhaps, the most remarkable. The quick increase in the number of non-dominated solutions as we increase the number of objective functions, rapidly dilutes the effect of the selection mechanism of a MOEA [9].

Here, we are interested in the maximin fitness function (MFF) [10, 11] which can act as a selection mechanism of type (ii) and it has interesting properties. Furthermore, computing the MFF is computationally efficient because its com-

[2] Although this paper focuses on MOEAs, there are many other multi-objective metaheuristics currently available (for example, multi-objective ant colony optimizers [3, 4], multi-objective particle swarm optimizers [5], multi-objective firefly algorithms [6], multi-objective flower pollination algorithms [7], and multi-objective harmony search algorithms [8] just to name a few). However, their discussion is beyond the scope of this paper.

plexity is linear with respect to the number of objective functions. Nevertheless, the use of the MFF also has some disadvantages, but there have been some proposals to address them.

Thus, the main goal of this paper is to provide an in-depth study about the MFF and its proposed variations, so that we can identify its main advantages and possible limitations. Such a study aims to provide more information about the sort of instances in which it is advisable to use any of the proposed MFF-based MOEAs, as well as those cases in which their use may present some difficulties.

The remainder of this paper is organized as follows. Section 2 states the problem of our interest. The previous related work about MOEAs based on MFF is presented in Section 3. MFF is described in detail in Section 4. Section 5 describes three MOEAs based on MFF (MC-MOEA, MD-MOEA and MH-MOEA) and we also propose a new version of MH-MOEA called MAH-MOEA. Our experimental results are presented in Section 6. Finally, we provide our conclusions and some possible paths for future work in Section 7.

2. Problem Statement

We are interested in the general *multiobjective optimization problem (MOP)*, which is defined as follows: Find $\vec{x}^* = [x_1^*, x_2^*, \ldots, x_n^*]^T$ which optimizes

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \ldots, f_k(\vec{x})]^T \tag{1}$$

such that $\vec{x}^* \in \Omega$, where $\Omega \subset R^n$ defines the feasible region of the problem. Assuming minimization problems, we have the following definitions.

Definition 1. *We say that a vector $\vec{x} = [x_1, \ldots, x_n]^T$ dominates vector $\vec{y} = [y_1, \ldots, y_n]^T$, denoted by $\vec{x} \prec \vec{y}$, if and only if $f_i(\vec{x}) \leq f_i(\vec{y})$ for all $i \in \{1, ..., k\}$ and there exists an $i \in \{1, \ldots, k\}$ such that $f_i(\vec{x}) < f_i(\vec{y})$.*

Definition 2. *We say that a vector $\vec{x} = [x_1, \ldots, x_n]^T$ is weakly non-dominated if there does not exist any $\vec{y}$ such that $f_i(\vec{y}) < f_i(\vec{x})$ for all $i \in \{1, ..., k\}$.*

Definition 3. *A point* $\vec{x}^* \in \Omega$ *is Pareto optimal if there does not exist any* $\vec{x} \in \Omega$ *such that* $\vec{x} \prec \vec{x}^*$.

Definition 4. *A point* $\vec{x} \in \Omega$ *is weakly Pareto optimal if there does not exist another point* $\vec{y} \in \Omega$ *such that* $f_i(\vec{y}) < f_i(\vec{x})$ *for all* $i \in \{1, ..., k\}$.

Definition 5. *For a given MOP,* $\vec{f}(\vec{x})$*, the Pareto optimal set is defined as:* $\mathcal{P}^* = \{\vec{x} \in \Omega | \neg\exists \vec{y} \in \Omega : \vec{y} \prec \vec{x}$.

Definition 6. *Let* $\vec{f}(\vec{x})$ *be a given MOP and* $\mathcal{P}^*$ *the Pareto optimal set. Then, the Pareto Front is defined as:* $\mathcal{PF}^* = \{\vec{f}(\vec{x}) \mid \vec{x} \in \mathcal{P}^*\}$.

3. Related Work

The maximin fitness function (MFF) was originally proposed by Balling in [10] and it has been incorporated in genetic algorithms [11, 12, 13, 14, 15], particle swarm optimizers [16, 17], ant colony optimizers [18] and differential evolution [19].

The early proposals based on MFF only considered MOPs with low dimensionality (two objective functions) and did not adopt a technique to improve the distribution based on the idea that MFF penalizes clustering. It was until 2012 [19] that a more in-depth study of MOEAs based on MFF was undertaken. The authors of this study found two important disadvantages when MFF is used to select individuals:

1. MFF prefers weakly non-dominated individuals over dominated individuals and this causes a loss in the diversity of the population, especially, in MOPs in which one objective function is easier to solve than the others.
2. The second disadvantage has to do with the poor diversity obtained in objective function space. Although MFF penalizes clustering between solutions, it is possible that many individuals have the same fitness and then we cannot know which individual should be selected.

In recent years, some proposals to address the two above disadvantages have been made [19, 13, 14, 15]. In the following sections we will provide an in-depth analysis of such proposals.

4. Maximin Fitness Function

The maximin fitness function (MFF) works as follows. Let's consider a MOP with K objective functions and an evolutionary algorithm whose population size is P. Let f_k^i be the normalized value of the kth objective for the ith individual in a particular generation. Assuming minimization problems, we have that the jth individual weakly dominates the ith individual if: $min_k(f_k^i - f_k^j) \geq 0$. The ith individual, in a particular generation, will be weakly dominated by another individual, in the generation, if: $max_{j \neq i}(min_k(f_k^i - f_k^j)) \geq 0$. Then, the maximin fitness of individual i is defined as:

$$fitness^i = max_{j \neq i}(min_k(f_k^i - f_k^j)) \tag{2}$$

where the min is taken over all objective functions, and the max is taken over all individuals in the population, except for the same individual i. From eq. (2), we can say the following: (i) Any individual whose maximin fitness is greater than zero is a dominated individual; (ii) any individual whose maximin fitness is less than zero is a non-dominated individual; (iii) finally, any individual whose maximin fitness is equal to zero is a weakly-dominated individual. Some interesting properties of MFF are the following:

1. MFF penalizes clustering of non-dominated individuals. See Figure 1(b).
2. The maximin fitness of dominated individuals is a metric of the distance to the non-dominated front. See Figure 1(c).
3. The max function in the maximin fitness of a dominated individual is always controlled by a non-dominated individual and is indifferent to clustering. The max function in the maximin fitness of a non-dominated individual may be controlled by a dominated or a non-dominated individual. See Figure 1(c).

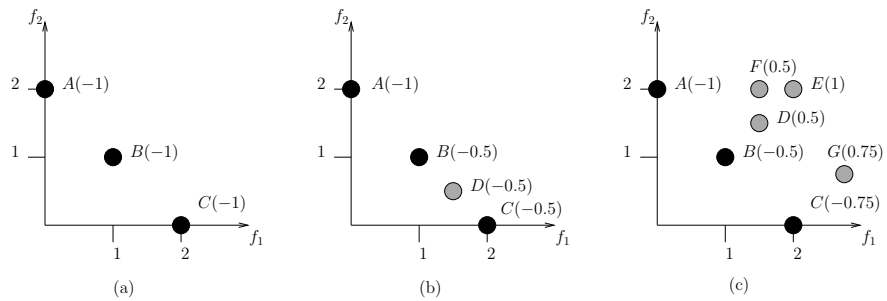


Figure 1: Properties of maximin fitness function. In (b), we can see that if we incorporate individual D, individuals B, C and D are penalized because they are close from each other. In (c), we can see that the fitness of individuals D, E and F is controlled by the non-dominated individual B, and their fitness is a metric of the distance to the individual B. The same occurs with individual G but its fitness is controlled by the non-dominated individual C. Also, we can see that the fitness of individual B is affected by the dominated individual D because they are close and the fitness of individual C is affected by the dominated individual G.

The author of MFF proposed in [11] the following modified MFF:

$$fitness^i = max_{j \neq i, j \in \mathcal{ND}}(min_k(f_k^i - f_k^j)) \tag{3}$$

where $\mathcal{ND}$ is the set of non-dominated individuals. Using eq. (3) to assign the fitness of each individual, we guarantee that the fitness of a non-dominated individual is controlled only by its non-dominance and then, we only penalize clustering between non-dominated individuals. For example, if we use the modified MFF in Figure 1(c), individual B would not be penalized and it would retain a fitness value equal to -1.

It is interesting to observe that MFF allows to design, in an easy way, an interactive method to solve MOPs when the decision maker can define preferences. For example, at each iteration of the algorithm we can present to the decision maker the set of non-dominated solutions, and then he/she can choose which solutions will be considered to calculate the fitness of each solution in the population. Then, we use the following equation to assign fitness:

$$fitness^i = max_{j \neq i, j \in \mathcal{A}}(min_k(f_k^i - f_k^j)) \tag{4}$$

where $\mathcal{A}$ is the set of non-dominated individuals which were chosen by the decision maker.

4.1. Disadvantages of the Maximin Fitness Function

A MOEA based on Differential Evolution and MFF was proposed in [19]. In that work, two important disadvantages of the MFF were identified. The principal disadvantage arises from the following question: **Is it better to prefer weakly non-dominated individuals than dominated individuals?** The answer was that it is not good to prefer weakly non-dominated individuals (even if they are weakly non-dominated by any dominated individual). As an example to illustrate this claim, the ZDT2 test problem was used:

$$f_1(\vec{x}) = x_1; \quad f_2(\vec{x}) = g(\vec{x})\left(1 - (x_1/g(\vec{x}))^2\right); \quad g(\vec{x}) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i; \quad (5)$$

If we use MFF into an evolutionary algorithm to solve ZDT2, we would assign the fitness of each individual using MFF, and then we would sort the individuals according to their fitness values. Then, we will obtain many (perhaps even only) weakly Pareto points because f_1 is easier to optimize than f_2 and then, we quickly obtain weakly non-dominated solutions at one extreme of the Pareto front. Figure 2(a) shows that if we use Differential Evolution and MFF, we only obtain weakly Pareto points. Figure 2(b) shows that if we use a Genetic Algorithm and MFF, the convergence to the Pareto optimal front is slow because we obtain many weakly Pareto points during the search. In [16], the authors proposed a MOEA based on a particle swarm optimizer and MFF, and also reported problems in ZDT2. In order to address this problem, the following constraint was proposed in [19]: *Any individual that we want to select must not be similar (in objective function space) to another (already selected) individual.* The process to verify similarity between individuals is shown in Algorithm 1.

By adding this constraint, we can find the true Pareto front of ZDT2 when we use a MOEA based on Differential Evolution, see Figure 2(c). Also, we speed up convergence when we use a MOEA based on a Genetic Algorithm,

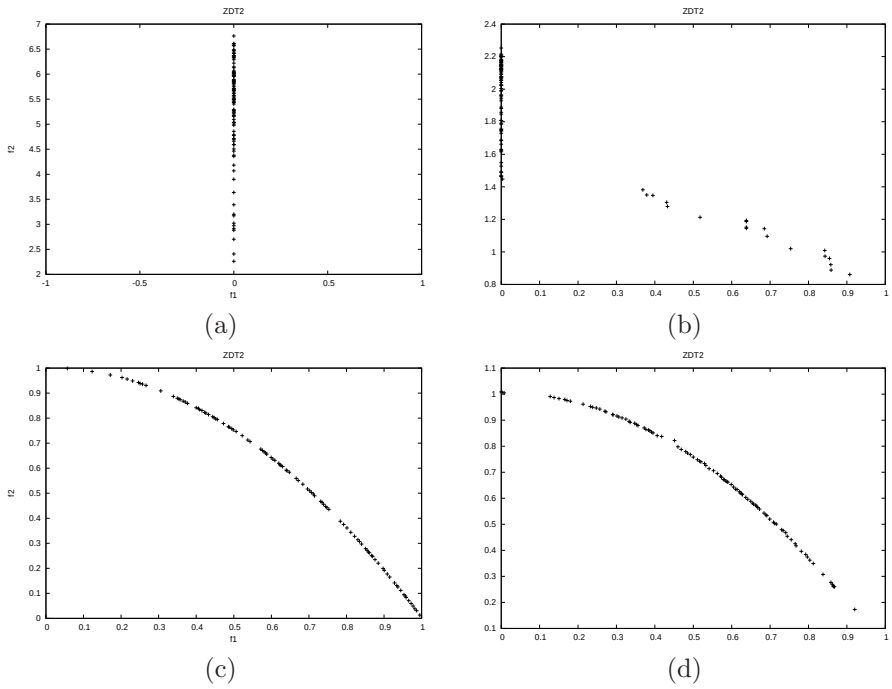


Figure 2: In (a), we used a MOEA based on Differential Evolution and the maximin fitness function. In (b), we used a MOEA based on a Genetic Algorithm and the maximin fitness function. In cases (c) and (d), we used the same MOEAs adopted in (a) and (b), respectively, but using the constraint to verify similarity. In all cases, we used a population size of 100 individuals. In cases (a) and (c), we iterated for 100 generations. Finally, in cases (b) and (d), we iterated for 150 generations.

Input : $\mathcal{P}$ (population), x (individual such that $x \notin \mathcal{P}$) and ϵ (minimum difference between components).
Output: Returns 1, if the individual x is similar to any individual in the population $\mathcal{P}$; otherwise, returns 0.

```
foreach y ∈ P do
    foreach objective function "k" do
        if |x.f̃[k] − y.f̃[k]| < ε then
            return 1;
        end
    end
end
return 0;
```

Algorithm 1: Verify similarity

see Figure 2(d). The complete selection mechanism using MFF and the above constraint proposed in [19] is shown in Algorithm 2. One could think that we can use MFF simply without selecting solutions whose maximin fitness value is equal to zero (because they are weakly dominated). However, it is important to note that the above constraint avoids that we select both: (i) solutions which are weakly dominated by non-dominated solutions and (ii) solutions which are weakly dominated by dominated solutions. For example, let's assume that we want to select five individuals in Figure 1(c). If we only use MFF, then we select individuals A, C, B, D, F. If we use MFF and the above constraint, then we sort them according to their fitness values: A(-1), C(-0.75), B(-0.5), D(0.5), F(0.5), G(0.75) and E(1). Finally, we select individuals A, C, B, D and we consider the individual F but we do not select it because is similar to individual D (in objective function f_1) which had been already selected. Consequently, we select individual G.

```
Input  : P (population), N (number of individuals that we want to choose such that
         N < |P|) and ε (minimum difference between objectives).
Output: S (selected individuals).
/*Sorting with respect to the maximin fitness values                          */
AssignFitness(P);
Sort(P);
/*Fill up the new population with the best copies according to the maximin fitness
  values, verifying that no solution is similar to one that had been previously
  selected                                                                    */
S ← ∅;
foreach y ∈ P do
    if |S| < N and VerifySimilarity(y, S, ε) = 0 then
        S ← S ∪ y;
    end
end
/*Choose the remaining individuals considering only the maximin fitness values */
if |S| < N then
    foreach y ∈ P such that y has not been selected and |S| < N do
        S ← S ∪ y;
    end
end
return S;
```

Algorithm 2: Maximin Selection

The second disadvantage has to do with the approximate Pareto optimal front and its distribution. In [19], the authors showed that the maximin fitness has difficulties in some cases. For example, in Figure 1(b), individuals B, C and D have the same maximin fitness value. Therefore, we cannot know which of

those three is the best individual that should be part of the next generation. In orded to address this disadvantage, several approaches have been proposed. In [19], the authors proposed to combine MFF with a clustering technique. In [13], the authors studied if it was better to use the original MFF or its modified version. In [14], the authors proposed to combine MFF with a technique based on Euclidean distances which has as its aim to improve the distribution of solutions. Finally, in [15], the authors proposed combining MFF with the hypervolume indicator. In the following section, we will analyze these proposals in more detail.

5. Selection Mechanisms based on MFF

5.1. MFF and a Clustering Technique

In [19], the authors proposed a selection mechanism based on MFF and a clustering technique to select solutions from a set of non-dominated solutions. Such mechanism works as follows. If we want to select N individuals from a population of non-dominated individuals called $\mathcal{ND}$, then, we choose the best N individuals with respect to their maximin fitness, and we use them as centers of the clusters. Then, we proceed to place each individual in its nearest cluster. Finally, for each of the resulting clusters, we recompute the center, and we choose the individual closest to it. This procedure is shown in Algorithm 3. With this technique, if we return to Figure 1(b) and we assume that we want to choose two individuals, we can see that, regardless of the individual (B, C or D) that we choose as an initial center of the cluster, we always obtain two clusters: one of them contains individual A, and the other one contains individuals B, C and D. After applying this procedure, we always choose individuals A and C. See Figure 3. It is important to note that clustering selection does not iterate many times to improve the distribution of the centers because we choose the initial centers regarding the maximin fitness and we only want to do a small correction based on the idea that MFF penalizes clustering.

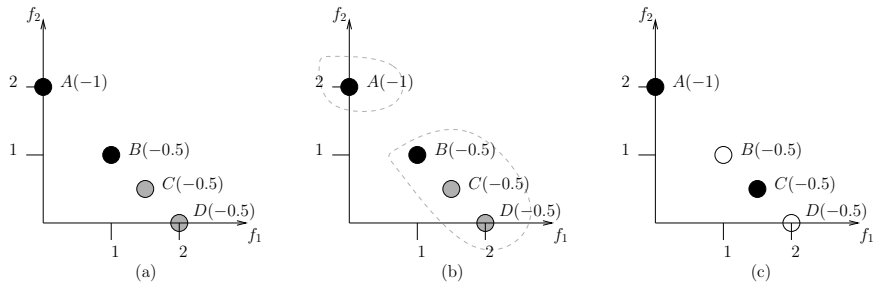


Figure 3: Let's assume that we want to select two individuals. If we use only the maximin fitness function and we assume that A, B, C and D is the ordering of the solutions after sorting them with respect to their fitness value, then we select individuals A and B and individuals C and D are not considered (see (a)). This is clearly not a good selection procedure. If we use the clustering technique proposed in [19], we take A and B as initial centers of the clusters and we obtain two clusters: the first one only has A and the second has B, C and D, see (b). When we recalculate the centers of the clusters and choose the closest solution to the centers, we select A and C, see (c).

Input : $\mathcal{ND}$ (population of non-dominated individuals) and N (number of individuals that we want to choose such that $N < |\mathcal{P}|$).
Output: $\mathcal{S}$ (selected individuals).
`/*Choose the best` N `individuals, according to maximin fitness, as centers of the clusters` $\mathcal{C}$ `*/`
AssignFitness($\mathcal{ND}$);
$Sort(\mathcal{ND})$;
for $j \leftarrow 1$ **to** N **do**
  $\mu_j = y_j$ such that $y_j \in \mathcal{ND}$;
  $\mathcal{C}_j = \{\emptyset\}$;
end
`/*Do one iteration of clustering */`
foreach $y \in \mathcal{ND}$ **do**
  if μ_j *is closest to* y **then**
    $\mathcal{C}_j \leftarrow \mathcal{C}_j \cup y$;
  end
end
`/*Obtain the new centers of the clusters */`
for $j \leftarrow 1$ **to** N **do**
  $\mu_j \leftarrow \frac{1}{|\mathcal{C}_j|} \sum_{y_i \in \mathcal{C}_j} y_i$;
end
`/*Select individuals who are closest to the centers of the clusters */`
$\mathcal{S} \leftarrow \emptyset$;
for $j \leftarrow 1$ **to** N **do**
  if $y_i \mid y_i \in \mathcal{C}_j$ *is the nearest to the center* μ_j **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup y_i$;
  end
end
return $\mathcal{S}$;

Algorithm 3: Clustering Selection (setting the centers using maximin)

It is necessary to consider that if we want to select from a set which contains dominated solutions, this selection mechanism is not effective. For example, in Figure 1(c), if we want to select three individuals, the clustering technique selects individuals A, D and C, penalizing individual B. This is clearly not good because individual B dominates individual D. Therefore, the complete selection mechanism that the authors proposed in [19] is a combination of Algorithms 2 and 3. If we want to select N individuals from a population $\mathcal{P}$, first, we obtain the set of non-dominated solutions which will be called "$\mathcal{ND}$". Then, if the number of non-dominated solutions is greater than N (i.e., $|\mathcal{ND}| > N$), we use Algorithm 3; otherwise, we use Algorithm 2.

Although with this selection mechanism the authors were able to address some difficulties of MFF, it still has some disadvantages. For example, if we see Figure 4, and we assume that we want to select six individuals, if we only use MFF to select, we would choose individuals A, B, C, D, E and F. If we use the selection mechanism based on MFF and the above clustering technique, we would choose individuals A, B, C, D, E and K. None of these two results is correct. This is because MFF penalizes all solutions (G, H, ..., O) and prefers to select solutions in other parts of the Pareto front, leaving big gaps in the front.

Finally, a study about the impact of using the original MFF or its modified version was done in [13]. The main conclusions of this study were that there is no significant impact. In this paper, we always use the modified version of MFF for all the MOEAs presented.

5.2. MFF and Euclidean Distances

In [14], the authors proposed to combine MFF with a technique based on Euclidean distances to improve the distribution of the solutions in objective function space. They explained that they used Euclidean distances because the aim was that the solutions were uniformly distributed. Such a selection mechanism works as follows:

Let's assume that we want to select N individuals from a population called $\mathcal{P}$. First, we assign fitness to each individual using the modified version of MFF.

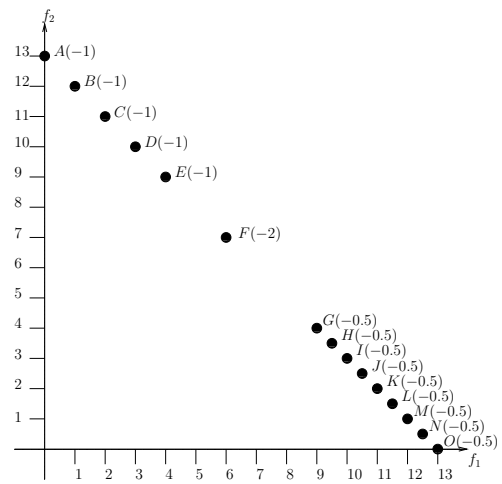


Figure 4: Maximin fitness function penalizes all solutions G, H, ... O. This is undesirable, because it leaves gaps when we select individuals.

Then, we proceed to select individuals according to their fitness value, verifying similarity between the selected individuals (see Algorithm 1). We put the selected individuals in the set called $\mathcal{S}$. If we already selected the N individuals but there are still non-dominated individuals which have not participated in the selection process, then we proceed to do the following. For each non-dominated individual y who has not participated in the selection process (because its fitness value is low), we obtain its nearest neighbor from $\mathcal{S}$ ($s_{nearest}$) and we choose a random individual from $\mathcal{S}$ (s_{random}, such that $s_{nearest} \neq s_{random}$). We assume that the probability of choosing an individual in a crowded region is higher than the probability of choosing an individual in an unexplored region. Then, y will compete with s_{random} and $s_{nearest}$ to survive. We use $s_{nearest}$ with the idea of improving the diversity locally: If we move $s_{nearest}$ to y, do we increase the distance with respect to its nearest neighbor in $\mathcal{S}$? And, we use s_{random} because we consider the scenario in which the solution $s_{nearest}$ is in an unexplored region and, therefore, it is not a good idea to delete $s_{nearest}$ or y. Therefore, first, y competes with the randomly chosen solution s_{random}: If the Euclidean distance from y to its nearest neighbor in $\mathcal{S}$ is greater than the Euclidean distance from s_{random} to its nearest neighbor in $\mathcal{S}$, we replace s_{random} with y. If

y loses the competition, then y competes with its nearest neighbor to survive. If the Euclidean distance from y to its nearest neighbor in $\mathcal{S}$ (without considering $s_{nearest}$) is greater than the Euclidean distance from $s_{nearest}$ to its nearest neighbor in $\mathcal{S}$, then we replace $s_{nearest}$ with y. It is important to mention that if all the objectives are equally important, we need to calculate the Euclidean distance on the normalized values of the objective functions. The complete selection mechanism is shown in Algorithm 4.

```
Input  : P (population), N (number of individuals to choose N < ||P||).
Output: S (selected individuals).
/*Sorting with respect to the maximin fitness                                  */
AssignFitness(P);
Sort(P);
ND ← The non-dominated solutions in P;
/*Fill up the new population with the best copies according to the maximin fitness,
  verifying that there is not a similar one                                    */
S ← ∅;
foreach y ∈ P do
    if |S| < N and VerifySimilarity(y, S, ε) = 0 then
        S ← S ∪ y;
    end
end
if |S| ≤ N then
    /*Choose the remaining individuals considering only the maximin fitness    */
    foreach y ∈ P such that y has been not selected do
        S ← S ∪ y;
    end
else
    /*Improve diversity according to the Euclidean distances between solutions. */
    foreach y ∈ ND who has not participated in the selection process do
        if VerifySimilarity(y, S, ε) = 0 then
            s_nearest ← The nearest neighbor of y in S;
            dy1 ← Distance from y to s_nearest;
            s_random ← Obtain a random individual from S such that s_nearest ≠ s_random;
            dsrandom ← Distance from s_random to its nearest neighbor in S;
            if dy1 > dsrandom then
                Replace s_random with y;
            else
                dsnearest ← Distance from s_nearest to its nearest neighbor in S;
                dy2 ← Distance from y to its nearest neighbor in S without regarding
                s_nearest;
                if dy2 > dsnearest then
                    Replace s_nearest with y;
                end
            end
        end
    end
end
return S;
```

Algorithm 4: Maximin-Euclidean Selection

Figure 5 shows the selection process using MFF and Euclidean distances. Since individuals C and D are not considered in (a), in (b), C competes with A and B, and C replaces B. In (c), D competes with A and C, and D replaces C. With this selection mechanism, if we return to Figure 4, we can avoid that the

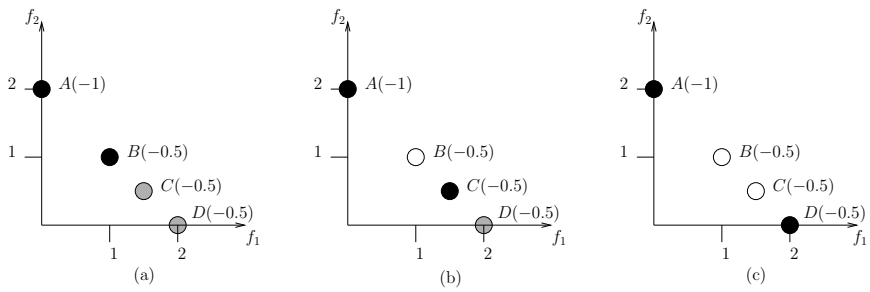


Figure 5: Let's assume that we want to select two individuals. If we use the technique based on MFF and Euclidean distances, first we select A and B ($\mathcal{S} = A, B$), see (a). After that, we consider individual C; its nearest neighbor is B and we choose A as a random solution. First, C competes with A and C loses because the distance from A to B is greater than the distance from C to B. Then, C competes with B and C wins because the distance from C to A is greater than the distance from B to A, see (b). Finally, we consider D, and D loses with A but it wins with C. Then, we select A and D, see (c).

approximate Pareto front has big gaps. Because of that, all individuals G, H, $\cdots$, O have the same fitness value.

5.3. MFF and the Hypervolume indicator

There are different indicators to assess the quality of the approximate Pareto optimal set generated by a MOEA. However, the *hypervolume indicator* (I_H) is the only unary indicator which is strictly "Pareto compliant[3]" [20]. Furthermore, I_H rewards convergence towards the Pareto front as well as the maximum spread of the solutions obtained. For these reasons, many MOEAs based on it have been proposed [21, 22, 23, 24, 25, 26, 27, 28]. However, this indicator has an important disadvantage: its high computational cost (the problem of computing I_H is **#P-hard**[4] [29]). In [15], the authors proposed a selection mechanism that combines MFF and I_H. Their idea is to use MFF as the main selection mechanism and I_H is used only to correct the possible errors produced when selecting with MFF. One interesting thing of this selection mechanism is

[3]An indicator $I : \Omega \rightarrow \mathbb{R}$ is **Pareto compliant** if for all $\mathcal{A}, \mathcal{B} \subseteq \Omega : \mathcal{A} \preceq \mathcal{B} \Rightarrow I(\mathcal{A}) \geq I(\mathcal{B})$ assuming that greater indicator values correspond to higher quality, where $\mathcal{A}$ and $\mathcal{B}$ are approximations of the Pareto optimal set, Ω is the feasible region and $\mathcal{A} \preceq B$ means that every point $\vec{b} \in \mathcal{B}$ is weakly dominated by at least one point $\vec{a} \in \mathcal{A}$.

[4]I_H cannot be computed exactly in polynomial time in the number of objective functions unless $P = NP$.

that, to the author's best knowledge, it is the only one based on I_H that is known to work with a population-based scheme. This is probably because MFF determines the order in which each individual competes to survive using I_H and also uses the competition scheme proposed in [28] in which each individual only competes with two other individuals of the population. Therefore, the original combinatorial problem no longer exists.

The selection mechanism proposed in [15] works as follows: If we want to select N individuals from a population $\mathcal{P}$, we assign first a fitness value to each individual using the modified MFF. Then, we proceed to select the individuals according to their fitness, verifying similarity between selected individuals, see Algorithm 1. If we consider all individuals in the population and we do not select N individuals, we select the remaining individuals considering only the maximin fitness. If we already selected the N individuals but there are still non-dominated individuals in $\mathcal{P}$ who have not participated in the selection process, then, we proceed to use the contribution to I_H as follows: Let $\mathcal{S}$ be the set of current selected individuals. Then, for each non-dominated individual y who has not participated in the selection process, we obtain its nearest neighbor in $\mathcal{S}$ (we call it $y_{nearest}$) and we choose a random individual called y_{random} such that $y_{nearest} \neq y_{random}$. Finally, we calculate the contribution to I_H of y, $y_{nearest}$ and y_{random}. If y has a better contribution than $y_{nearest}$ or y_{random}, then y replaces the individual with the worst contribution ($y_{nearest}$ or y_{random}). The full selection mechanism is shown in Algorithm 5.

Figure 6 shows the selection process using MFF and I_H. Since individuals C and D are not considered, in (a), C competes with A and B, and C replaces B. In (b), D competes with A and C, and it loses. Also, with this selection mechanism, if we return to Figure 4, we can avoid that there are big gaps in the front.

As we mentioned before, calculating I_H or its contribution is a **#P-hard** problem. Therefore, although with the selection mechanism based on MFF and I_H, we can reduce the number of times that we need to calculate the contribution to I_H, if we want to solve MOPs with many objective functions, e.g., more than

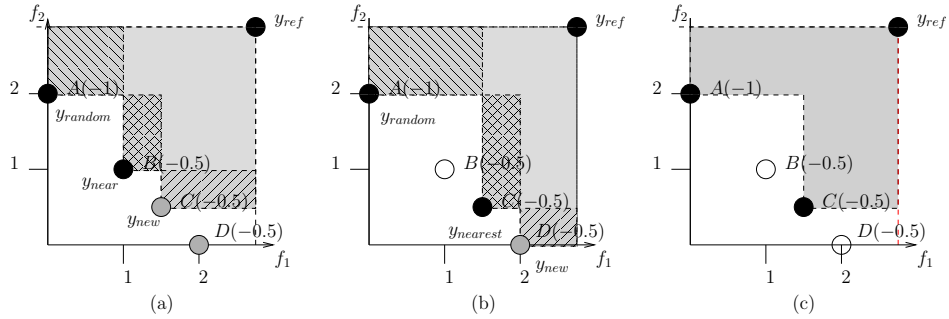


Figure 6: Let's assume that we want to select two individuals. If we use the technique based on MFF and I_H, first we select A and B ($\mathcal{S} = A, B$). After that, we consider individual C; its nearest neighbor is B and we choose A as a random solution. Individual B is eliminated because it has the worst contribution, see (b). Finally, we consider D; its nearest neighbor is C and we choose A as a random solution. Individual D is eliminated because it has the worst contribution, see (b). Finally, we choose individuals A and C, see (c).

Input : $\mathcal{P}$ (population), n (number of individuals to choose $N < \|\mathcal{P}\|$).
Output: $\mathcal{S}$ (selected individuals).

AssignFitness($\mathcal{P}$);
Sort($\mathcal{P}$);
$\mathcal{ND} \leftarrow$ The non-dominated solutions in $\mathcal{P}$;
$\mathcal{S} \leftarrow \emptyset$;
/*Fill up the new population with the best copies according to the maximin fitness, verifying that there is not a similar one */
$\mathcal{S} \leftarrow \emptyset$;
foreach $y \in \mathcal{P}$ **do**
  if $|\mathcal{S}| < N$ *and VerifySimilarity(y, $\mathcal{S}$, ϵ) = 0* **then**
    $\mathcal{S} \leftarrow \mathcal{S} \cup y$;
  end
end
if $|\mathcal{S}| \leq N$ **then**
  /*Choose the remaining individuals considering only the maximin fitness */
  foreach $y \in \mathcal{P}$ *such that y has been not selected* **do**
    $\mathcal{S} \leftarrow \mathcal{S} \cup y$;
  end
else
  /*Improve the diversity according to the contribution to I_H */
  foreach $y \in \mathcal{ND}$ *who had not participated in the selection process* **do**
    if *VerifySimilarity(y, $\mathcal{S}$, ϵ) = 0* **then**
      $y_{nearest} \leftarrow$ The nearest neighbor of y in $\mathcal{S}$;
      $y_{random} \leftarrow$ A randomly selected individual in $\mathcal{S}$ such that $y_{nearest} \neq y_{random}$;
      /*Calculate the contributions to the hypervolume */
      $C_{nearest} \leftarrow C_H(y_{nearest}, \mathcal{S})$;
      $C_{random} \leftarrow C_H(y_{random}, \mathcal{S})$;
      $C_y \leftarrow C_H(y, \mathcal{S})$;
      /*Remove the individual with the worst contribution */
      $worst \leftarrow$ Individual with the worst contribution (y, $y_{nearest}$ or y_{random});
      if $worst = y_{nearest}$ *or* $worst = y_{random}$ **then**
        Replace $worst$ with y;
      end
    end
  end
end
return $\mathcal{S}$;

Algorithm 5: Maximin-Hypervolume Selection

six, this MOEA is not practical.

In [30], the authors studied the competition scheme proposed in [28] and also studied different ways to approximate I_H or its contribution. Finally, they showed that approximating the contribution to I_H by adopting the technique proposed by Bringmann and Friedrich in [31] within the selection mechanism proposed by Menchaca and Coello in [28], produces good results. For this reason, in this work, we propose to use a version of the selection mechanism based on MFF and I_H which approximates the contributions to I_H, using the technique proposed by Bringmann and Friedrich.

6. Experimental Results

Each of the four selection mechanism described above were incorporated into a MOEA that uses the crossover and mutation operators of NSGA-II to create new individuals, giving rise to the four following MOEAs: "MC-MOEA: Maximin-Clustering Multi-Objective Evolutionary Algorithm", "MD-MOEA: Maximin-Distances Multi-Objective Evolutionary Algorithm", "MH-MOEA: Maximin-Hypervolume Multi-Objective Evolutionary Algorithm" and "MAH-MOEA: Maximin-Approximated Hypervolume Multi-Objective Evolutionary Algorithm". These MOEAs work as follows: If the size of the population is P, then we create P new individuals. We use a binary tournament to select the parents. At each tournament, two individuals are randomly selected and the one with the higher maximin fitness value is chosen. After that, we combine the population of parents and offspring to obtain a population of size $2P$. Then, we use one of the four available selection mechanisms to choose the P individuals that will take part of the following generation. This process is repeated for a certain (pre-defined) number of generations.

For our experiments, we used the DTLZ [32] test problems,[5] Table 1 shows some features of each test problem. We used MOPs with up to ten objective

[5] In the technical report [33], we present a comparative study using the WFG [34] test suite, too.

functions. We used $k = 5$ for DTLZ1, DTLZ3 and DTLZ6 and $k = 10$ for the remaining DTLZ test problems. We adopted the parameters suggested by the authors of NSGA-II: $p_c = 0.9$ (crossover probability), $p_m = 1/n$ (mutation probability), where n is the number of decision variables. For the crossover and mutation operators, we adopted $\eta_c = 15$ and $\eta_m = 20$, respectively. Our maximum number of fitness function evaluations was set to 50,000 (we used a population size of 100 individuals and we iterated for 500 generations). In the case of MAH-MOEA, we used 10^4 as our number of samples.

Table 1: Features of the test problems adopted. An objective function is **separable** if it can be optimized by considering each parameter in turn, independently of one another, and the resultant set of globally optimal parameter vectors is the cross-product of the optimal sets for each individually optimized parameter. In the multi-objective sense, this means that the ideal points for separable objectives can be determined considering only one parameter at a time. An objective function is **multimodal** when it has multiple local optima and it is **unimodal** when it has a single optimum. We consider that a problem is multimodal if it has at least one multimodal objective function.

MOP	Separability	Modality	Geometry
DTLZ1	separable	multimodal	linear
DTLZ2	separable	unimodal	concave
DTLZ3	separable	multimodal	concave
DTLZ4	separable	unimodal	concave
DTLZ5	?	unimodal	degenerate
DTLZ6	?	unimodal	degenerate
DTLZ7	separable	unimodal	disconnected

6.1. Performance Indicators

To assess performance, we adopted the following indicators:

- **Hypervolume indicator** (I_H). It is defined as the size of the space covered by the Pareto optimal solutions. I_H rewards both convergence towards the Pareto front as well as the maximum spread of the solutions obtained. To calculate I_H, we normalized the approximations of the Pareto optimal set, generated by the MOEAs, and we used $y_{ref} = [y_1, \cdots, y_k]$ such that $y_i = 1.1$ is adopted as our reference point. The normalization was performed considering all approximations generated by the different MOEAs (i.e., we place, in one set, all non-dominated solutions found by the MOEAs which are being compared and from this set we calculate the maximum and minimum for each objective function).

- **Two Set Coverage** (I_{SC}). We decided to use this indicator with the aim of assessing the convergence of the MOEAs. I_{SC} was proposed by Zitzler et al. [35] and it is a binary Pareto compliant indicator. Let $\mathcal{A}, \mathcal{B}$ be two approximations of the Pareto optimal set, I_{SC} is defined as follows:

$$I_{SC}(\mathcal{A}, \mathcal{B}) = \frac{|\vec{b} \in \mathcal{B} \text{ such that } \exists \vec{a} \in \mathcal{A} \text{ with } \vec{a} \prec \vec{b}|}{|\mathcal{B}|}$$

  If all points in $\mathcal{A}$ dominate or are equal to all points in $\mathcal{B}$, then by definition $I_{SC} = 1$. $I_{SC} = 0$ implies that no element in $\mathcal{B}$ is dominated by any element of $\mathcal{A}$. In general, both $I_{SC}(\mathcal{A}, \mathcal{B})$ and $I_{SC}(\mathcal{B}, \mathcal{A})$ have to be considered.

- **Spacing** (I_S). It was proposed by Schott [36]. It measures the spread of solutions in the approximate Pareto optimal front. This indicator is defined as follows:

$$I_S(A) = \sqrt{\frac{1}{|A| - 1} \sum_{i=1}^{|A|} \left(\overline{d} - d_i\right)^2}$$

  where: $d_i = \min_{j, j \neq i} \sum_k |f_k^i - f_k^j|$ and $\overline{d} = \frac{1}{|A|} \sum_{i=1}^{|A|} d_i$, k is the number of objective functions, $i, j = 1, \cdots |A|$. When $I_S = 0$ all the solutions in A are uniformly spread.

It is important to keep in mind that we can obtain different results if we use different indicators since each indicator can measure a different feature of a multi-objective problem. Even if they measure the same feature, the use of different indicators can provide different results, e.g., the hypervolume indicator assesses both convergence and spread of solutions, and the $R2$-indicator also assesses both features but the optimal distribution for the $R2$-indicator depends of the convex weights that it adopts. If the convex weights are uniformly distributed, then the optimal distribution of these two indicators is different if the Pareto front is not linear. In our case, we chose the hypervolume indicator because, as it is known, this is the only unary indicator that is known to

be strictly Pareto compliant [20]. Additionally, we chose the two set coverage indicator and the spacing indicator because the first one assesses convergence and it is also Pareto compliant and the second one assesses distribution but its optimal distribution is uniform. In this way, our comparison among MOEAs can be performed in a fair manner. However, it is worth noticing that the use of the spacing indicator has to be considered in combination with the two set coverage indicator because a set of solutions that presents a uniform distribution is considered appropriate only if it constitutes a good approximation to the true Pareto optimal front (i.e., convergence has precedence over distribution when assessing performance of a MOEA).

6.2. Comparison of MOEAs based on MFF

In this section, we compare the four MOEAs based on MFF: MC-MOEA, MD-MOEA, MH-MOEA and MAH-MOEA. Table 2 shows the results with respect to I_H for the DTLZ test problems with up to six objective functions. In this table, we can see that MC-MOEA ranked fourth in all twenty-eight cases; MD-MOEA ranked third in twenty-four cases, second in two cases and first in two cases; MH-MOEA ranked first in twenty-five cases and only in three cases ranked second; finally, MAH-MOEA ranked second in twenty-three cases, third in four cases and first in one case. Table 8(a) shows the results of the statistical analysis that we made to validate our experiments, for which we used Wilcoxon's rank sum. In this case, we decided to compare the fourth place with the third place (MC-MOEA and MD-MOEA, respectively), the third place with the second place (MD-MOEA and MAH-MOEA, respectively) and the second place with the first place (MAH-MOEA and MH-MOEA, respectively). For MC-MOEA and MD-MOEA, we can see that in twenty-six cases we can reject the null hypothesis (medians are equal) and only for DTLZ6 with four objective functions and DTLZ1 with five objective functions we can say that these two algorithms have a similar behavior. For MD-MOEA and MAH-MOEA, we can see that in twenty-five cases we can reject the null hypothesis and only for three problems both algorithms have a similar behavior. Finally, for MAH-

MOEA and MH-MOEA, we have that for sixteen cases we can reject the null hypothesis and for twelve cases we can say that both algorithms have a similar behavior. This result is interesting because we can say that MAH-MOEA is really competitive with respect to MH-MOEA.

Since MD-MOEA outperformed MC-MOEA in all cases, we can say that the technique based on Euclidean distances was able to correct some disadvantages of the technique based on clustering, e.g., it can avoid that the approximate Pareto front has big gaps. However, both MOEAs (MC-MOEA and MD-MOEA) have difficulties when the MOP has a degenerate Pareto front (see problems DTLZ5 and DTLZ6). We think that this problem arises because these two selection mechanisms have as their aim to distribute the solutions uniformly and then, it is hard for the MOEA to converge to a front with a lower dimensionality than the dimensionality of the problem. However, we can see that MH-MOEA and MAH-MOEA were able to correct this disadvantage. This is because the aim of these selection mechanisms is to maximize I_H and the maximum I_H corresponds to a distribution into the degenerate Pareto front.

An interesting thing is that MAH-MOEA, the version of MH-MOEA that approximates the contribution to I_H, obtained results very close to MH-MOEA but at a lower computational cost (see Tables 2, 3 and 8(a)). This is an important result because, as we know, MOEAs based on the use of the exact I_H values are not practical when we want to solve MOPs with more than five or six objective functions. In order to address this disadvantage, some authors have proposed different techniques to approximate I_H or its contribution. However, the quality of the solutions obtained by these MOEAs considerably degrades in most cases, unlike MAH-MOEA which does not lose much quality due to two reasons: First, it approximates the contribution to I_H in the competition scheme proposed in [13] as the authors suggested in [30]. And second, it produces a ranking using MFF to perform an initial selection and then it uses the contribution to I_H only to correct the possibles errors in this first selection procedure, i.e., I_H is not used as the primary selection mechanism.

From Tables 2 and 3, we can say that the best option to solve MOPs with low

and high dimensionality (in objective function space) is MAH-MOEA. However, if we need to obtain the approximate Pareto optimal set in the shortest time possible, MD-MOEA is a good option but we should be careful when dealing with MOPs having degenerate Pareto fronts.

It is important to note that if we use I_H to compare the different MOEAs, then it is evident that MOEAs based on this indicator have advantages over those which don't adopt it, because the aim of the former type of MOEAs is to maximize I_H. For this reason, we decided to use two other indicators to compare the approximate Pareto optimal sets obtained by the MOEAs. We adopted the two set coverage indicator (I_{SC}) to measure convergence to the Pareto front and the spacing indicator (I_S) to measure distribution of the solutions found. Since we can use these two indicators to evaluate approximations which involve any number of objective functions, we decided to use up to ten objective functions. However, in this comparison we only considered MD-MOEA and MAH-MOEA due to two reasons: First, Table 2 shows clearly that MD-MOEA obtained better results than MC-MOEA. And second, although MH-MOEA is better than MAH-MOEA, it cannot be used to solve MOPs with more than six objective functions (its running time is too high, and it would require weeks or even months to complete all the required experiments).

Table 4 shows the results for the DTLZ test problems with respect to I_{SC} and we can see that in fifty-four cases the solutions found by MAH-MOEA were able to cover a larger percentage of the solutions found by MD-MOEA than the percentage of solutions found by MAH-MOEA which are covered by at least one solution found by MD-MOEA. However, only in the DTLZ6 test problem we can assure that MAH-MOEA is better than MD-MOEA because only in this problem the percentage of solutions found by MAH-MOEA which are covered by at least one solution found by MD-MOEA is zero or close to zero and the percentage of solutions found by MD-MOEA which are covered by at least one solution found by MAH-MOEA is close to one. Table 5 shows the results regarding I_S and we can observe that MD-MOEA ranked second in thirty-nine cases and first in seventeen cases. With these two tables, we can corroborate

the results found when we use I_H: MAH-MOEA is better than MD-MOEA in most cases. Finally, Figures 7 and 8 show the Pareto fronts obtained by the four algorithms in their median with respect to the hypervolume indicator in some of the test problems adopted. Here, we can see again that MAH-MOEA is the best MOEA because it found well-distributed Pareto fronts, and its results are very similar with respect to MH-MOEA. Also, we can see that MD-MOEA is better than MC-MOEA and it is competitive with respect MH-MOEA and MAH-MOEA (only in DTLZ6 it obtained a worse distribution).

As final conclusions of this section, we can say that MD-MOEA and MAH-MOEA are the best options to solve MOPs with high and low dimensionality in objective function space. Although MAH-MOEA is better than MD-MOEA according to I_H, regarding I_{SC} and I_S they are competitive. Also, MD-MOEA is much faster than MAH-MOEA. However, it is important to be careful when we use MD-MOEA because it has difficulties to solve certain types of MOPs, e.g., those with a degenerate Pareto front.

6.3. MOEAs based on MFF vs MOEAs not based on MFF

In this section, we compare MD-MOEA and MAH-MOEA with respect to two well-known MOEAs: The first one is MOEA/D. We chose this MOEA because it has been a viable alternative to deal with many-objective optimization problems in recent years. Also, its computational cost is very low. MOEA/D [37] decomposes the MOP into N scalar optimization subproblems and then it solves these subproblems simultaneously using an evolutionary algorithm. For our experiments, we used the version in which MOEA/D adopts PBI (Penalty Boundary Intersection) to decompose the MOP. We decided to use PBI because the resulting optimal solutions with PBI are normally much better distributed than those obtained by the Tchebycheff approach [37]. To generate the convex weights we used the technique proposed in [38] and after that, we applied clustering (k-means) to obtain a specific number of weights.

The second one is SMS-EMOA [24]. We chose this MOEA because it is the most popular hypervolume-based MOEA. SMS-EMOA creates an initial

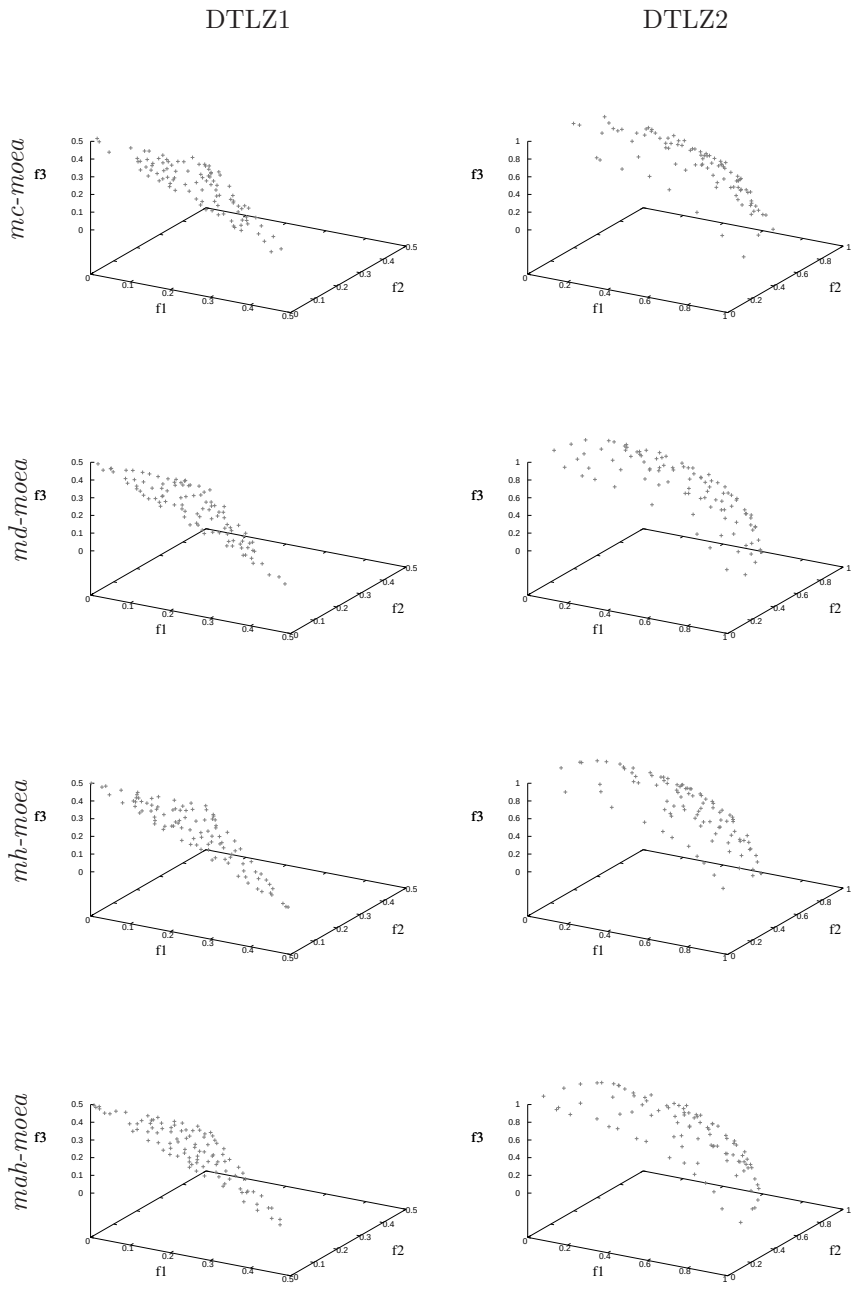


Figure 7: Pareto fronts obtained by the four MOEAs (MC-MOEA, MD-MOEA, MH-MOEA and MAH-MOEA) in the median (with respect to the hypervolume indicator) of their thirty independent runs for the test problems DTLZ1 and DTLZ2.

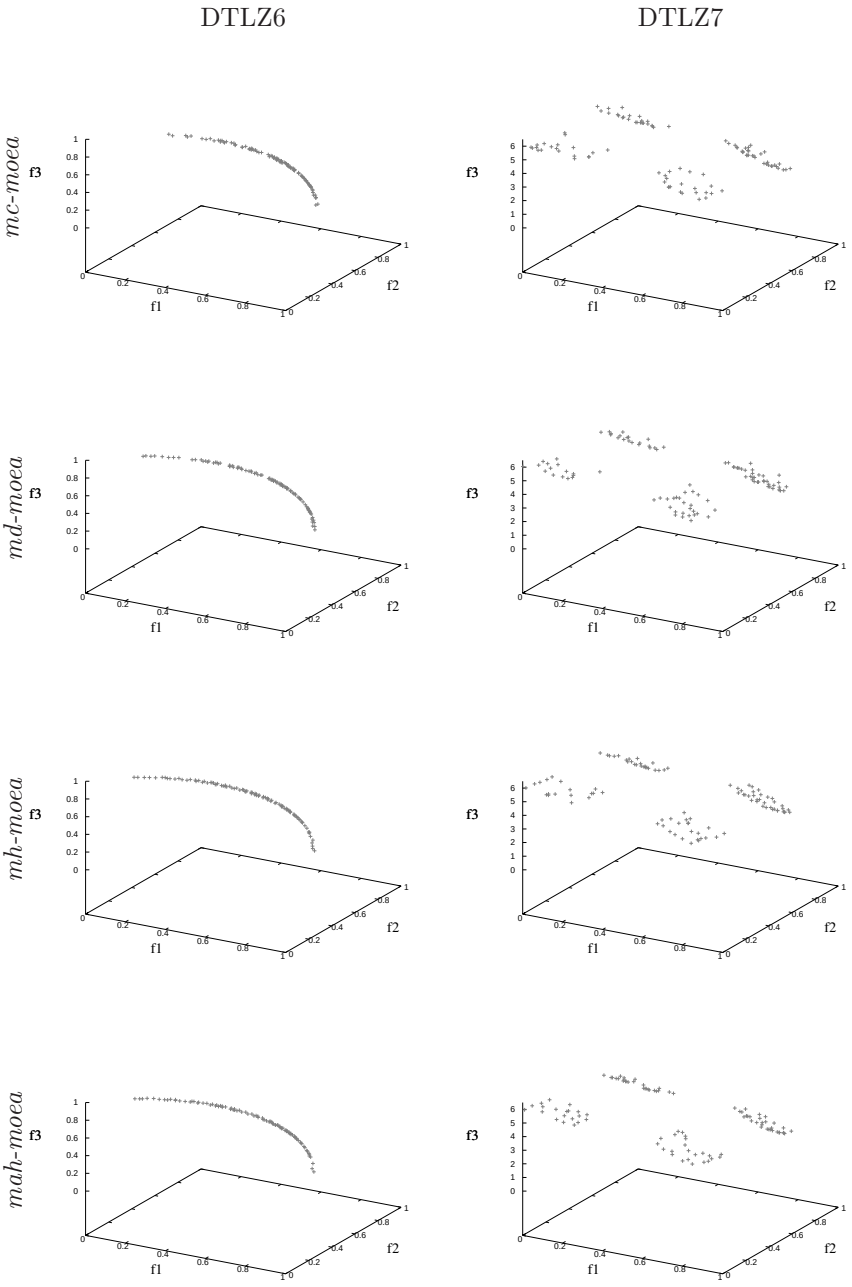


Figure 8: Pareto fronts obtained by the four MOEAs (MC-MOEA, MD-MOEA, MH-MOEA and MAH-MOEA) in the median (with respect to the hypervolume indicator) of their thirty independent runs for the test problems DTLZ6 and DTLZ7.

population and then, it generates only one solution per iteration. After that, it applies Pareto ranking. When the last front has more than one solution, SMS-EMOA calculates the contribution to I_H of each individual in the last front and it eliminates the individual with the worst contribution. Beume et al. [25] proposed not to use the contribution to I_H when in the Pareto ranking we obtain more than one front. In that case, they proposed to use the number of solutions which dominate to one solution (the solution that is dominated by more solutions is removed). In this work, we used the version proposed by Beume et al. but instead of calculating the exact contribution to I_H, we approximate it using the same technique that we adopted for MAH-MOEA.

Since these four MOEAs use the same operators to create new individuals (they use the same crossover and mutation operators adopted by NSGA-II), the comparison of selection mechanisms is fair. For MOEA/D and SMS-EMOA, we also adopted the parameters suggested by the authors of NSGA-II: $p_c = 0.9$, $p_m = 1/n$, where n is the number of decision variables, $\eta_c = 15$ and $\eta_m = 20$. In the case of MOEA/D, we used a neighborhood with size equal to 20 and in the case of SMS-EMOA we used 10^4 as our number of samples.

Before we perform the comparison, it is important to mention that both MOEA/D and SMS-EMOA have important disadvantages. SMS-EMOA is impractical to solve MOPs with many objective function because calculating I_H or its contribution involves a very high computational cost. In this work, we use a version that approximates the contribution to I_H. However, as we will see later on, the competition scheme used by SMS-EMOA is not efficient and therefore, the running time of this version of SMS-EMOA is also high. On the other hand, MOEA/D needs to generate a set of well-distributed convex weights and this task becomes more difficult as we increase the number of objective functions.

Regarding I_H and considering the DTLZ test problems (see Table 6) MD-MOEA ranked second in eleven cases, third in nine cases, fourth in five cases and first in three cases. MAH-MOEA ranked first in fourteen cases, second in ten cases, third in three cases and fourth in one case. MOEA/D ranked fourth in nineteen cases, third in five cases, second in three cases and first in

one case. Finally, SMS-EMOA ranked third in eleven cases, first in ten cases, second in four cases and fourth in three cases. Table 8(b) shows the results of the statistical analysis that we conducted to validate our experiments, for which we used Wilcoxon's rank sum and I_H. In this case, we decided to compare the MOEAs based on MFF (MD-MOEA and MAH-MOEA) with respect to MOEA/D and SMS-EMOA. For MD-MOEA and MOEA/D, we can say that only in one problem they have a similar behavior and in the twenty-seven remaining problems the null hypothesis ("medians are equal") can be rejected. The same occurs with MAH-MOEA and MOEA/D, since in only one problem they have a similar behavior. In the case of MD-MOEA and SMS-EMOA only in two problems both algorithms have a similar behavior and in the twenty-six remaining problems the null hypothesis can be rejected. Finally, with respect to MAH-MOEA and SMS-EMOA only in two cases they have a similar behavior and in the twenty-six remaining problems the null hypothesis can be rejected. From these results, we can say that MAH-MOEA is the best algorithm, followed by SMS-EMOA in the second place, MD-MOEA in the third place and MOEA/D in the fourth place. Another interesting thing is that MAH-MOEA is much faster than SMS-EMOA. It is also worth noticing that MD-MOEA is ranked second with respect to the running time but it is not much slower than MOEA/D which is in the first place. See Table 7.

As conclusions of this section, we can say that MOEAs based on MFF are a good option to solve MOPs with low and high dimensionality because they can outperform well-known MOEAs such as SMS-EMOA and MOEA/D, e.g., both MD-MOEA and MAH-MOEA outperformed MOEA/D in the set of test problems adopted and MAH-MOEA also outperformed SMS-EMOA in these test problems. In addition, both MD-MOEA and MAH-MOEA are much faster than SMS-EMOA and MD-MOEA is not much slower than MOEA/D.

7. Conclusions and Future Work

In this paper, we have studied three selection mechanisms based on MFF. The first one combines MFF with a clustering technique, the second one combines MFF with a technique based on Euclidean distances and the third one combines MFF with I_H. Since calculating I_H or its contribution is a $\#P$-hard problem, we propose to approximate the contribution to I_H as the authors suggested in [30]. Each of the four selection mechanisms was incorporated into a MOEA that uses simulated binary crossover (SBX) and parameter-based mutation (PM), giving rise the following MOEAs: "Maximin-Clustering Multi-Objective Evolutionary Algorithm (MC-MOEA)", "Maximin-Distances Multi-Objective Evolutionary Algorithm (MD-MOEA)", "Maximin-Hypervolume Multi-Objective Evolutionary Algorithm (MH-MOEA)" and "Maximin-Approximated Hypervolume Multi-Objective Evolutionary Algorithm (MAH-MOEA)". According to our experimental results, the best algorithm is MAH-MOEA because it obtains results with a high quality and it can also be used in MOPs with many objective functions (in this work we tested it with up to ten objective functions). MAH-MOEA is followed by MD-MOEA, in terms of performance. MD-MOEA obtained good results in most problems, but it has difficulties in MOPs with degenerate Pareto fronts. We think that this is due to the fact that the aim of the selection mechanism used by MD-MOEA is to obtain a uniform distribution. Consequently, it is hard for MD-MOEA to converge to a Pareto front with a dimensionality lower than the dimensionality of the MOP. If the time to obtain the approximate Pareto optimal set is an important factor, MD-MOEA is the best option because it obtains competitive results with respect to MAH-MOEA but at a much lower computational cost.

Besides, in this work we compare MD-MOEA and MAH-MOEA, with respect to two well-known MOEAs: MOEA/D and SMS-EMOA (in a version that approximates the contribution to I_H). These MOEAs use a selection mechanism based on decomposition and another based on I_H, respectively. Our results showed that both MD-MOEA and MAH-MOEA outperformed MOEA/D in the

DTLZ test problems and MAH-MOEA also outperformed SMS-EMOA. With respect to the running time, both algorithms (MD-MOEA and MAH-MOEA) are efficient because MAH-MOEA is much faster than SMS-EMOA and it also obtained good results (it outperformed SMS-EMOA in the DTLZ test problems) and MD-MOEA is not much slower than MOEA/D, while obtaining better results. Therefore, we can say that MD-MOEA and MAH-MOEA are a good option to solve MOPs with low and high dimensionality because they obtain approximations of the Pareto optimal set with a high quality and the computational cost of both MOEAs is affordable. Additionally, the running time of MD-MOEA is quite good. Also, these MOEAs do not need additional information such as MOEA/D that requires a set of well-distributed convex weights.

Another interesting feature of MOEAs based on MFF is that they can be used to solve MOPs in an interactive way when the decision maker defines his/her preferences. For example, a MOEA based on MFF can present at each generation a set of non-dominated solutions to the user so that he/she can choose the solutions which will be considered when calculating the maximin fitness of each indidividual.

As part of our future work, we want to study the constraints used to avoid selecting weakly dominated solutions because we think that this is one of the reasons for which SMS-EMOA obtains better results than our MAH-MOEA in some problems. For example, the constraint used in the MOEAs presented here does not check if the selected solution is worse than the new solution that we want to select. However, we cannot select such a solution, because it is similar to the solution that had been previously selected.

Acknowledgements

The first author acknowledges support from CONACyT through a scholarship to pursue graduate studies at the Computer Science Department of CINVESTAV-IPN. The last author gratefully acknowledges support from CONACyT project no. 221551. The second author also acknowledges support from a

Cátedra Marcos Moshinsky 2014.

References

[1] C. A. Coello Coello, G. B. Lamont (Eds.), Applications of Multi-Objective Evolutionary Algorithms, World Scientific, Singapore, 2004, iSBN 981-256-106-4.

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

[3] M. López-Ibáñez, T. Stützle, The Automatic Design of Multi-Objective Ant Colony Optimization Algorithms, IEEE Transactions on Evolutionary Computation 16 (6) (2012) 861–875.

[4] M. López-Ibáñez, T. Stützle, The impact of design choices of multi-objective ant colony optimization algorithms on performance: An experimental study on the biobjective TSP, in: M. Pelikan, J. Branke (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2010, ACM Press, New York, NY, USA, 2010, pp. 71–78.

[5] M. Reyes-Sierra, C. A. Coello Coello, Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art, International Journal of Computational Intelligence Research 2 (3) (2006) 287–308.

[6] X.-S. Yang, Multiobjective firefly algorithm for continuous optimization, Engineering with Computers 29 (2) (2013) 175–184.

[7] X.-S. Yang, M. Karamanoglu, X. He, Multi-objective Flower Algorithm for Optimization, in: 13th Annual International Conference on Computational Science (ICCS), Vol. 18 of Procedia Computer Science, Elsevier Science, Barcelona, Spain, 2013, pp. 861–868.

[8] S. Sivasubramani, K. Swarup, Multi-objective harmony search algorithm for optimal power flow problem, International Journal of Electrical Power & Energy Systems 33 (3) (2011) 745–752.

[9] M. Farina, P. Amato, On the Optimal Solution Definition for Many-criteria Optimization Problems, in: Proceedings of the NAFIPS-FLINT International Conference'2002, IEEE Service Center, Piscataway, New Jersey, 2002, pp. 233–238.

[10] R. Balling, Pareto sets in decision-based design, Journal of Engineering Valuation and Cost Analysis 3 (2000) 189–198.

[11] R. Balling, S. Wilson, The Maximin Fitness Function for Multi-objective Evolutionary Computation: Application to City Planning, in: L. Spector, E. D. Goodman, A. Wu, W. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, E. Burke (Eds.), Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001), Morgan Kaufmann Publishers, San Francisco, California, 2001, pp. 1079–1084.

[12] R. Balling, The Maximin Fitness Function; Multiobjective City and Regional Planning, in: C. M. Fonseca, P. J. Fleming, E. Zitzler, K. Deb, L. Thiele (Eds.), Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003, Springer. Lecture Notes in Computer Science. Volume 2632, Faro, Portugal, 2003, pp. 1–15.

[13] A. Menchaca-Mendez, C. A. C. Coello, Selection Operators Based on Maximin Fitness Function for Multi-Objective Evolutionary Algorithms, in: R. C. Purshouse, P. J. Fleming, C. M. Fonseca, S. Greco, J. Shaw (Eds.), Evolutionary Multi-Criterion Optimization, 7th International Conference, EMO 2013, Springer. Lecture Notes in Computer Science Vol. 7811, Sheffield, UK, 2013, pp. 215–229.

[14] A. Menchaca-Mendez, C. A. Coello Coello, MD-MOEA : A New MOEA based on the Maximin Fitness Function and Euclidean Distances between Solutions, in: 2014 IEEE Congress on Evolutionary Computation

(CEC'2014), IEEE Press, Beijing, China, 2014, pp. 2148–2155, iSBN 978-1-4799-1488-3.

[15] A. Menchaca-Mendez, C. A. Coello Coello, MH-MOEA: A New Multi-Objective Evolutionary Algorithm Based on the Maximin Fitness Function and the Hypervolume Indicator, in: T. Bartz-Beielstein, J. Branke, B. Filipič, J. Smith (Eds.), Parallel Problem Solving from Nature PPSN XIII, 13th International Conference, Springer. Lecture Notes in Computer Science Vol. 8672, Ljubljana, Slovenia, 2014, pp. 652–661.

[16] X. Li, Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function, in: K. D. et al. (Ed.), Genetic and Evolutionary Computation–GECCO 2004. Proceedings of the Genetic and Evolutionary Computation Conference. Part I, Springer-Verlag, Lecture Notes in Computer Science Vol. 3102, Seattle, Washington, USA, 2004, pp. 117–128.

[17] X. Li, J. Branke, M. Kirley, On Performance Metrics and Particle Swarm Methods for Dynamic Multiobjective Optimization Problems, in: 2007 IEEE Congress on Evolutionary Computation (CEC'2007), IEEE Press, Singapore, 2007, pp. 576–583.

[18] H. Li, X. Huang, Q. Feng, Optimizing expressway maintenance planning by coupling ant algorithm and geography information system transportation in hubei province, china, in: Geoscience and Remote Sensing Symposium (IGARSS), 2011 IEEE International, 2011, pp. 2977 –2979. doi:10.1109/IGARSS.2011.6049841.

[19] A. Menchaca-Mendez, C. A. Coello Coello, Solving Multi-Objective Optimization Problems using Differential Evolution and a Maximin Selection Criterion, in: 2012 IEEE Congress on Evolutionary Computation (CEC'2012), IEEE Press, Brisbane, Australia, 2012, pp. 3143–3150.

[20] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, V. G. da Fonseca, Performance Assessment of Multiobjective Optimizers: An Analysis and

Review, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 117–132.

[21] J. Knowles, D. Corne, Properties of an Adaptive Archiving Algorithm for Storing Nondominated Vectors, IEEE Transactions on Evolutionary Computation 7 (2) (2003) 100–116.

[22] S. Huband, P. Hingston, L. White, L. Barone, An Evolution Strategy with Probabilistic Mutation for Multi-Objective Optimisation, in: Proceedings of the 2003 Congress on Evolutionary Computation (CEC'2003), Vol. 3, IEEE Press, Canberra, Australia, 2003, pp. 2284–2291.

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

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

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

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

[27] S. Mostaghim, J. Branke, H. Schmeck, Multi-Objective Particle Swarm Optimization on Computer Grids, in: D. Thierens (Ed.), 2007 Genetic and Evolutionary Computation Conference (GECCO'2007), Vol. 1, ACM Press, London, UK, 2007, pp. 869–875.

[28] A. Menchaca-Mendez, C. A. Coello Coello, A New Selection Mechanism Based on Hypervolume and its Locality Property, in: 2013 IEEE Congress on Evolutionary Computation (CEC'2013), IEEE Press, Cancún, México, 2013, pp. 924–931, iSBN 978-1-4799-0454-9.

[29] K. Bringmann, T. Friedrich, Approximating the volume of unions and intersections of high-dimensional geometric objects, Computational Geometry 43 (67) (2010) 601 – 610.

[30] A. Menchaca-Mendez, C. A. Coello Coello, An alternative hypervolume-based selection mechanism for multi-objective evolutionary algorithms, Tech. Rep. EVOCINV-01-2015, Evolutionary Computation Group at CINVESTAV, Departamento de Computación, CINVESTAV-IPN, Mexico (May 2015).

[31] K. Bringmann, T. Friedrich, Approximating the least hypervolume contributor: NP-hard in general, but fast in practice, Theoretical Computer Science 425 (2012) 104–116.

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

[33] A. Menchaca-Mendez, C. A. Coello Coello, Muti-Objective Evolutionary Algorithms based on the Maximin Fitness Function to solve Many-Objective Optimization Problems, Tech. Rep. EVOCINV-02-2015, Evolutionary Computation Group at CINVESTAV, Departamento de Computación, CINVESTAV-IPN, Mexico (October 2015).

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

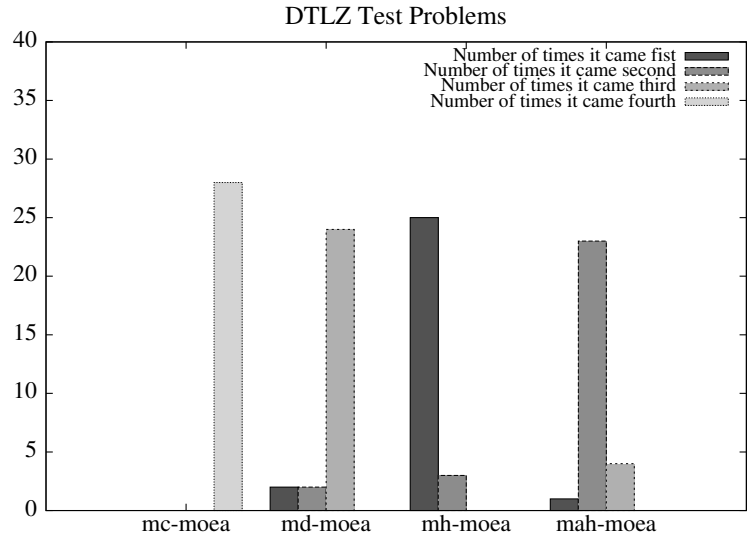
[35] E. Zitzler, K. Deb, L. Thiele, Comparison of Multiobjective Evolutionary Algorithms: Empirical Results, Evolutionary Computation 8 (2) (2000) 173–195.

[36] J. R. Schott, Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization, Master's thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, Massachusetts (May 1995).

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

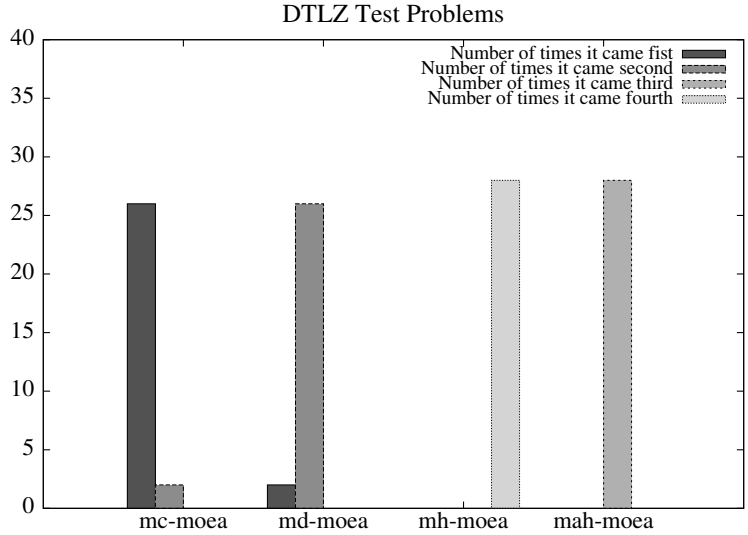
[38] I. Das, J. E. Dennis, Normal-boundary intersection: A new method for generating the pareto surface in nonlinear multicriteria optimization problems, SIAM J. on Optimization 8 (3) (1998) 631–657.

Table 2: Results obtained in the DTLZ test problems with up to six objective functions. We compare MC-MOEA, MD-MOEA, MH-MOEA and MAH-MOEA using the hypervolume indicator I_H. We show average values over 30 independent runs. The values in parentheses correspond to the standard deviations.



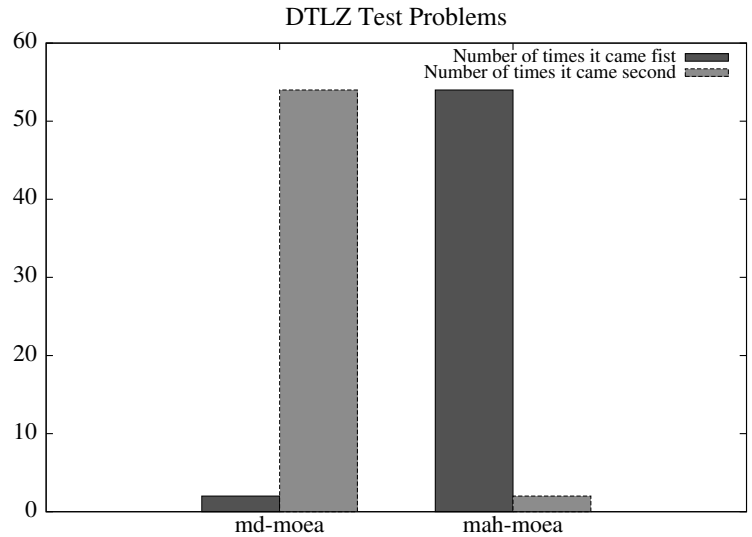
$\vec{f}$	mc-moea I_H	md-moea I_H	mh-moea I_H	mah-moea I_H
DTLZ1 (3)	1.1593 (0.013)	1.1823 (0.003)	**1.1897 (0.001)**	1.1896 (0.001)
DTLZ2 (3)	0.6964 (0.008)	0.7319 (0.003)	**0.7535 (0.002)**	0.7525 (0.002)
DTLZ3 (3)	1.3043 (0.021)	**1.3281 (0.000)**	1.3265 (0.000)	1.3263 (0.000)
DTLZ4 (3)	0.7015 (0.008)	0.7304 (0.006)	0.7491 (0.004)	**0.7492 (0.004)**
DTLZ5 (3)	0.2552 (0.005)	0.2651 (0.000)	**0.2661 (0.001)**	0.2661 (0.000)
DTLZ6 (3)	1.0453 (0.067)	1.0989 (0.008)	**1.0998 (0.007)**	1.0975 (0.009)
DTLZ7 (3)	0.5377 (0.040)	0.5519 (0.041)	**0.5645 (0.043)**	0.5638 (0.043)
DTLZ1 (4)	1.4241 (0.016)	1.4375 (0.005)	**1.4471 (0.002)**	1.4444 (0.002)
DTLZ2 (4)	1.0480 (0.019)	1.1249 (0.010)	**1.1672 (0.003)**	1.1647 (0.005)
DTLZ3 (4)	1.4630 (0.001)	**1.4638 (0.000)**	1.4638 (0.000)	1.4638 (0.000)
DTLZ4 (4)	1.1455 (0.016)	1.1912 (0.010)	**1.2261 (0.006)**	1.2252 (0.006)
DTLZ5 (4)	0.3928 (0.017)	0.4668 (0.023)	**0.5492 (0.005)**	0.5469 (0.004)
DTLZ6 (4)	1.2518 (0.045)	1.2742 (0.024)	**1.3617 (0.003)**	1.3600 (0.003)
DTLZ7 (4)	0.6659 (0.014)	0.7062 (0.016)	**0.7305 (0.019)**	0.7262 (0.018)
DTLZ1 (5)	1.5861 (0.069)	1.6011 (0.003)	**1.6061 (0.001)**	1.6031 (0.001)
DTLZ2 (5)	1.3004 (0.025)	1.4063 (0.015)	**1.4680 (0.004)**	1.4519 (0.006)
DTLZ3 (5)	1.6089 (0.002)	1.6098 (0.000)	**1.6098 (0.000)**	1.6098 (0.000)
DTLZ4 (5)	1.4341 (0.018)	1.4801 (0.011)	**1.5182 (0.006)**	1.5096 (0.007)
DTLZ5 (5)	0.7294 (0.019)	0.8122 (0.033)	**0.9249 (0.007)**	0.9180 (0.015)
DTLZ6 (5)	0.4890 (0.074)	1.3396 (0.063)	**1.5522 (0.002)**	1.5450 (0.005)
DTLZ7 (5)	0.7562 (0.026)	0.8534 (0.009)	**0.8716 (0.015)**	0.8699 (0.010)
DTLZ1 (6)	1.7002 (0.088)	1.7652 (0.004)	**1.7688 (0.001)**	1.7669 (0.001)
DTLZ2 (6)	1.6061 (0.027)	1.6757 (0.014)	**1.7264 (0.004)**	1.7114 (0.006)
DTLZ3 (6)	1.7707 (0.001)	1.7710 (0.000)	**1.7713 (0.000)**	1.7712 (0.000)
DTLZ4 (6)	1.6978 (0.024)	1.7234 (0.008)	**1.7495 (0.002)**	1.7407 (0.003)
DTLZ5 (6)	0.8283 (0.030)	0.9563 (0.067)	**1.1347 (0.019)**	1.1170 (0.041)
DTLZ6 (6)	0.5754 (0.050)	1.0454 (0.174)	**1.7039 (0.004)**	1.6984 (0.004)
DTLZ7 (6)	0.7382 (0.054)	0.9319 (0.022)	**0.9438 (0.020)**	0.9380 (0.026)

Table 3: Results obtained in the DTLZ test problems with up to six objective functions. We compare MC-MOEA, MD-MOEA, MH-MOEA and MAH-MOEA with respect to the running time required by each MOEA to obtain the approximation of the Pareto optimal set. The results are in seconds. We show average values over 30 independent runs. The values in parentheses correspond to the standard deviations.



$\vec{f}$	mc-moea time	md-moea time	mh-moea time	mah-moea time
DTLZ1 (3)	**0.6277 (0.010)**	0.8317 (0.025)	341.8550 (48.468)	26.5683 (6.888)
DTLZ2 (3)	**0.9857 (0.011)**	1.7247 (0.016)	362.5860 (46.285)	80.2447 (11.191)
DTLZ3 (3)	**0.6150 (0.016)**	0.7897 (0.028)	410.8940 (51.899)	26.4793 (5.165)
DTLZ4 (3)	**1.0137 (0.009)**	1.7187 (0.018)	441.8840 (25.976)	77.7587 (9.585)
DTLZ5 (3)	**0.8377 (0.009)**	1.2467 (0.019)	362.4503 (36.636)	9.6463 (2.308)
DTLZ6 (3)	**0.7157 (0.032)**	1.0763 (0.020)	358.8433 (17.170)	26.3760 (12.040)
DTLZ7 (3)	**0.9210 (0.018)**	1.0917 (0.036)	370.5477 (18.106)	28.6033 (7.032)
DTLZ1 (4)	**0.8150 (0.013)**	1.0227 (0.015)	350.6833 (13.917)	17.1963 (2.158)
DTLZ2 (4)	**1.2033 (0.015)**	2.0807 (0.009)	380.9720 (12.960)	56.9457 (9.412)
DTLZ3 (4)	**0.8083 (0.013)**	1.0830 (0.029)	403.1007 (26.826)	23.3403 (2.858)
DTLZ4 (4)	**1.2520 (0.011)**	2.0533 (0.020)	424.2623 (37.408)	52.1267 (9.269)
DTLZ5 (4)	**1.2420 (0.017)**	2.3523 (0.028)	418.8673 (56.864)	50.0157 (6.454)
DTLZ6 (4)	**0.9953 (0.011)**	1.6473 (0.020)	386.8503 (13.442)	49.7007 (6.381)
DTLZ7 (4)	**1.1347 (0.019)**	1.1903 (0.026)	354.2237 (11.950)	16.5777 (2.984)
DTLZ1 (5)	**0.9823 (0.015)**	1.1693 (0.037)	495.2057 (40.617)	12.5180 (1.662)
DTLZ2 (5)	**1.3807 (0.017)**	2.3043 (0.014)	893.2087 (23.893)	29.8747 (3.466)
DTLZ3 (5)	**0.9920 (0.009)**	1.2967 (0.028)	558.4480 (23.573)	17.2507 (2.455)
DTLZ4 (5)	**1.4673 (0.022)**	2.2767 (0.021)	843.5720 (16.920)	28.0337 (3.063)
DTLZ5 (5)	**1.4577 (0.011)**	2.4603 (0.021)	1516.0527 (91.143)	44.2683 (5.359)
DTLZ6 (5)	**1.3483 (0.016)**	2.2447 (0.044)	980.2290 (39.726)	46.6733 (5.942)
DTLZ7 (5)	1.3380 (0.016)	**1.3070 (0.035)**	324.5770 (35.547)	7.3807 (1.137)
DTLZ1 (6)	**1.1350 (0.017)**	1.3033 (0.070)	3137.1263 (406.976)	10.5600 (1.946)
DTLZ2 (6)	**1.5407 (0.046)**	2.4600 (0.023)	7760.6740 (199.641)	19.9033 (1.739)
DTLZ3 (6)	**1.1487 (0.010)**	1.4473 (0.030)	3859.2973 (340.664)	15.7413 (2.575)
DTLZ4 (6)	**1.6273 (0.014)**	2.4343 (0.026)	6662.2467 (190.364)	18.3560 (2.040)
DTLZ5 (6)	**1.6177 (0.025)**	2.6093 (0.028)	7964.5500 (1610.847)	37.3323 (5.122)
DTLZ6 (6)	**1.5197 (0.018)**	2.8480 (0.049)	5459.3457 (126.423)	44.0187 (5.048)
DTLZ7 (6)	1.5173 (0.017)	**1.4347 (0.047)**	618.1323 (159.834)	3.7347 (0.632)

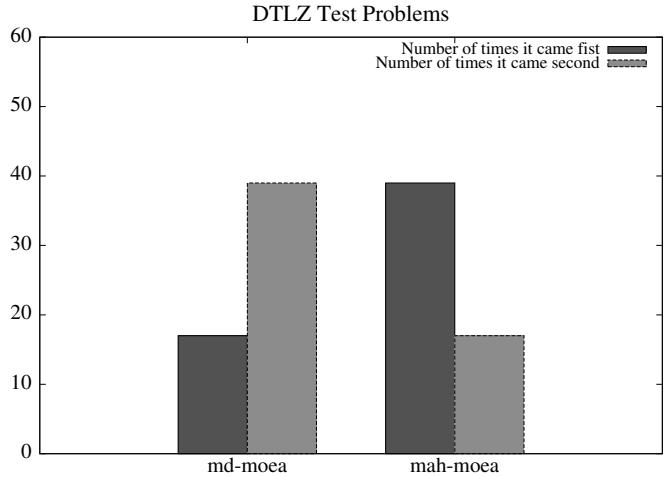
Table 4: Results obtained in the DTLZ test problems with up to ten objective functions. We compare MD-MOEA and MAH-MOEA with respect to I_{SC}. In this case, $\mathcal{A}$ is the set composed by all solutions found by MD-MOEA considering all 30 independent runs and $\mathcal{B}$ is the set composed by all solutions found by MAH-MOEA considering all 30 independent runs.



$\vec{f}$	$I_{SC}(\mathcal{A},\mathcal{B})$	$I_{SC}(\mathcal{B},\mathcal{A})$
DTLZ1 (3)	0.0040	**0.0577**
DTLZ2 (3)	0.0457	**0.4613**
DTLZ3 (3)	0.0320	**0.1180**
DTLZ4 (3)	0.0453	**0.4700**
DTLZ5 (3)	0.1610	**0.4800**
DTLZ6 (3)	**0.6693**	0.6390
DTLZ7 (3)	0.0903	**0.2870**
DTLZ1 (4)	0.0077	**0.0283**
DTLZ2 (4)	0.0227	**0.2827**
DTLZ3 (4)	0.0153	**0.0773**
DTLZ4 (4)	0.0213	**0.2647**
DTLZ5 (4)	0.0317	**0.4537**
DTLZ6 (4)	0.0137	**0.9117**
DTLZ7 (4)	0.0640	**0.2723**
DTLZ1 (5)	0.0110	**0.0277**
DTLZ2 (5)	0.0153	**0.1567**
DTLZ3 (5)	0.0063	**0.0893**
DTLZ4 (5)	0.0160	**0.1377**
DTLZ5 (5)	0.0193	**0.5630**
DTLZ6 (5)	0.0040	**0.9790**
DTLZ7 (5)	0.0433	**0.1357**
DTLZ1 (6)	0.0103	**0.0553**
DTLZ2 (6)	0.0067	**0.1380**
DTLZ3 (6)	0.0090	**0.0880**
DTLZ4 (6)	0.0067	**0.1143**
DTLZ5 (6)	0.0270	**0.5767**
DTLZ6 (6)	0.0000	**0.9867**
DTLZ7 (6)	0.0280	**0.0650**

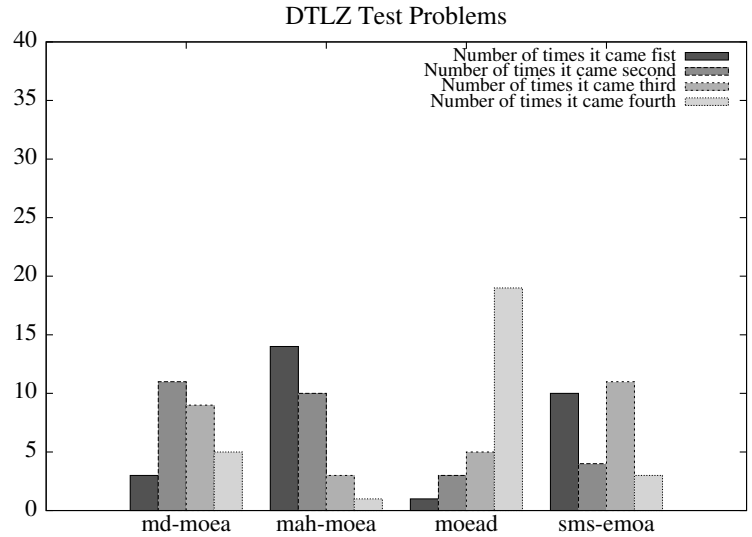
$\vec{f}$	$I_{SC}(\mathcal{A},\mathcal{B})$	$I_{SC}(\mathcal{B},\mathcal{B})$
DTLZ1 (7)	0.0090	**0.3713**
DTLZ2 (7)	0.0013	**0.1653**
DTLZ3 (7)	0.0097	**0.2893**
DTLZ4 (7)	0.0023	**0.0840**
DTLZ5 (7)	0.0260	**0.5730**
DTLZ6 (7)	0.0000	**0.9887**
DTLZ7 (7)	0.0287	**0.0373**
DTLZ1 (8)	0.0253	**0.8963**
DTLZ2 (8)	0.0000	**0.1813**
DTLZ3 (8)	0.0060	**0.7860**
DTLZ4 (8)	0.0013	**0.0617**
DTLZ5 (8)	0.0227	**0.5470**
DTLZ6 (8)	0.0000	**0.9867**
DTLZ7 (8)	**0.0173**	0.0140
DTLZ1 (9)	0.1243	**0.9667**
DTLZ2 (9)	0.0000	**0.3167**
DTLZ3 (9)	0.0033	**0.9230**
DTLZ4 (9)	0.0023	**0.0337**
DTLZ5 (9)	0.0303	**0.4677**
DTLZ6 (9)	0.0000	**0.9887**
DTLZ7 (9)	0.0070	**0.0073**
DTLZ1 (10)	0.0773	**0.9837**
DTLZ2 (10)	0.0000	**0.7277**
DTLZ3 (10)	0.0017	**0.9673**
DTLZ4 (10)	0.0000	**0.0370**
DTLZ5 (10)	0.0387	**0.2883**
DTLZ6 (10)	0.0000	**0.9817**
DTLZ7 (10)	0.0033	**0.0037**

Table 5: Results obtained in the DTLZ test problems with up to ten objective functions. We compare MD-MOEA and MAH-MOEA with respect to I_S. We show average values over 30 independent runs. The values in parentheses correspond to the standard deviations.



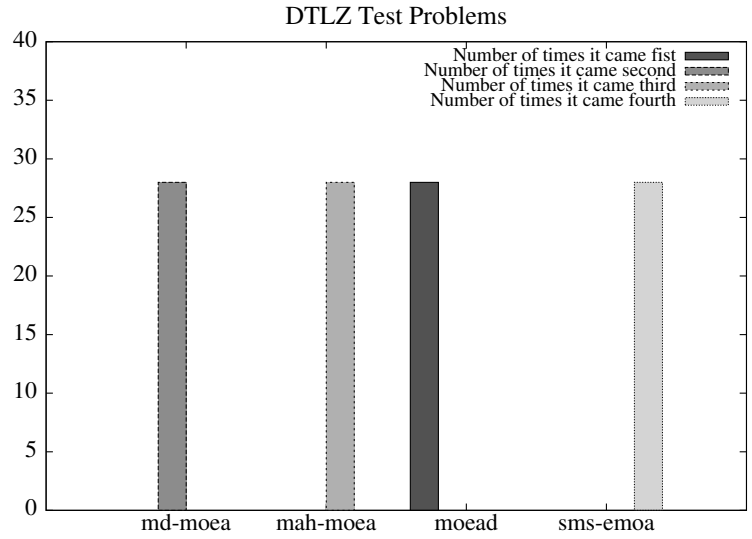
$\vec{f}$	md-moea I_S	mah-moea I_S	$\vec{f}$	md-moea I_S	mah-moea I_S
DTLZ1(3)	0.0383(0.051)	**0.0227(0.014)**	DTLZ1(7)	4.7387(8.298)	**0.1142(0.170)**
DTLZ2(3)	**0.0494(0.003)**	0.0519(0.005)	DTLZ2(7)	**0.1752(0.018)**	0.2028(0.027)
DTLZ3(3)	39.7739(16.843)	**0.0835(0.114)**	DTLZ3(7)	17.6441(27.499)	**0.6413(1.399)**
DTLZ4(3)	**0.0485(0.004)**	0.0534(0.005)	DTLZ4(7)	**0.1736(0.021)**	0.2011(0.018)
DTLZ5(3)	**0.0107(0.001)**	0.0149(0.002)	DTLZ5(7)	0.3055(0.034)	**0.2296(0.046)**
DTLZ6(3)	0.0477(0.050)	**0.0244(0.015)**	DTLZ6(7)	0.7690(0.102)	**0.3620(0.061)**
DTLZ7(3)	**0.0515(0.012)**	0.0548(0.011)	DTLZ7(7)	0.2959(0.081)	**0.2948(0.077)**
DTLZ1(4)	0.3313(1.026)	**0.0716(0.139)**	DTLZ1(8)	44.9282(15.636)	**3.1587(3.930)**
DTLZ2(4)	**0.0821(0.010)**	0.0976(0.007)	DTLZ2(8)	**0.2151(0.023)**	0.2337(0.021)
DTLZ3(4)	30.1241(24.519)	**0.1443(0.152)**	DTLZ3(8)	121.0272(70.941)	**2.0211(4.517)**
DTLZ4(4)	**0.0852(0.008)**	0.0979(0.007)	DTLZ4(8)	**0.1982(0.024)**	0.2265(0.020)
DTLZ5(4)	0.1303(0.028)	**0.0871(0.009)**	DTLZ5(8)	0.3297(0.035)	**0.2416(0.040)**
DTLZ6(4)	0.2187(0.036)	**0.1289(0.020)**	DTLZ6(8)	0.8535(0.075)	**0.3994(0.049)**
DTLZ7(4)	0.1223(0.015)	**0.1222(0.018)**	DTLZ7(8)	0.3472(0.080)	**0.3425(0.078)**
DTLZ1(5)	0.1476(0.311)	**0.1434(0.437)**	DTLZ1(9)	53.2934(10.620)	**9.9927(4.748)**
DTLZ2(5)	**0.1128(0.011)**	0.1406(0.012)	DTLZ2(9)	0.2740(0.076)	**0.2477(0.026)**
DTLZ3(5)	5.9726(12.907)	**0.2318(0.270)**	DTLZ3(9)	163.9581(73.838)	**0.8544(0.974)**
DTLZ4(5)	**0.1160(0.013)**	0.1379(0.011)	DTLZ4(9)	**0.2206(0.023)**	0.2521(0.024)
DTLZ5(5)	0.2209(0.029)	**0.1379(0.021)**	DTLZ5(9)	0.3472(0.040)	**0.2594(0.045)**
DTLZ6(5)	0.4020(0.044)	**0.2472(0.058)**	DTLZ6(9)	0.9396(0.103)	**0.4521(0.052)**
DTLZ7(5)	0.1837(0.033)	**0.1735(0.030)**	DTLZ7(9)	0.5877(0.906)	**0.5839(0.907)**
DTLZ1(6)	0.5147(1.209)	**0.1094(0.109)**	DTLZ1(10)	56.1186(11.811)	**13.1724(5.369)**
DTLZ2(6)	**0.1471(0.015)**	0.1782(0.016)	DTLZ2(10)	0.4853(0.102)	**0.2714(0.028)**
DTLZ3(6)	8.5028(15.487)	**0.6253(2.037)**	DTLZ3(10)	186.9971(41.579)	**3.5598(8.554)**
DTLZ4(6)	**0.1416(0.012)**	0.1777(0.018)	DTLZ4(10)	**0.2555(0.034)**	0.2882(0.033)
DTLZ5(6)	0.2670(0.033)	**0.1933(0.026)**	DTLZ5(10)	0.3324(0.049)	**0.2802(0.052)**
DTLZ6(6)	0.6232(0.066)	**0.3207(0.070)**	DTLZ6(10)	0.9776(0.088)	**0.4803(0.053)**
DTLZ7(6)	**0.2478(0.047)**	0.2520(0.055)	DTLZ7(10)	0.9527(1.374)	**0.8445(1.303)**

Table 6: Results obtained in the DTLZ test problems with up to six objective functions. We compare MD-MOEA, MAH-MOEA, MOEA/D and SMS-EMOA using the hypervolume indicator I_H. We show average values over 30 independent runs. The values in parentheses correspond to the standard deviations.



$\vec{f}$	md-moea I_H	mah-moea I_H	moead I_H	sms-emoa I_H
DTLZ1 (3)	1.1823 (0.003)	1.1896 (0.001)	1.2683 (0.001)	**1.3128 (0.001)**
DTLZ2 (3)	0.7319 (0.003)	0.7525 (0.002)	0.7203 (0.000)	**0.7645 (0.002)**
DTLZ3 (3)	1.3281 (0.000)	1.3263 (0.000)	1.3307 (0.000)	**1.3309 (0.000)**
DTLZ4 (3)	0.7304 (0.006)	0.7492 (0.004)	0.7255 (0.000)	**0.7699 (0.002)**
DTLZ5 (3)	0.2651 (0.000)	0.2661 (0.000)	0.2467 (0.001)	**0.2662 (0.000)**
DTLZ6 (3)	**1.0989 (0.008)**	1.0975 (0.009)	0.5129 (0.022)	0.5979 (0.012)
DTLZ7 (3)	0.5519 (0.041)	**0.5638 (0.043)**	0.4509 (0.027)	0.5366 (0.060)
DTLZ1 (4)	1.4375 (0.005)	1.4444 (0.002)	1.4134 (0.003)	**1.4621 (0.001)**
DTLZ2 (4)	1.1249 (0.010)	**1.1647 (0.005)**	0.9237 (0.001)	1.0700 (0.004)
DTLZ3 (4)	**1.4638 (0.000)**	1.4638 (0.000)	1.4638 (0.000)	1.4630 (0.001)
DTLZ4 (4)	1.1912 (0.010)	**1.2252 (0.006)**	0.9169 (0.001)	1.0645 (0.005)
DTLZ5 (4)	0.4668 (0.023)	0.5469 (0.004)	0.4921 (0.004)	**0.5618 (0.004)**
DTLZ6 (4)	1.2742 (0.024)	**1.3600 (0.003)**	1.2083 (0.006)	1.3085 (0.007)
DTLZ7 (4)	0.7062 (0.016)	**0.7262 (0.018)**	0.3875 (0.022)	0.6922 (0.043)
DTLZ1 (5)	1.6011 (0.003)	1.6031 (0.001)	1.6095 (0.000)	**1.6103 (0.000)**
DTLZ2 (5)	1.4063 (0.015)	**1.4519 (0.006)**	1.0345 (0.004)	1.2717 (0.017)
DTLZ3 (5)	**1.6098 (0.000)**	1.6098 (0.000)	1.6096 (0.000)	1.5998 (0.007)
DTLZ4 (5)	1.4801 (0.011)	**1.5096 (0.007)**	1.0317 (0.005)	1.2778 (0.016)
DTLZ5 (5)	0.8122 (0.033)	0.9180 (0.015)	0.7215 (0.022)	**0.9540 (0.006)**
DTLZ6 (5)	1.3396 (0.063)	**1.5450 (0.005)**	1.3358 (0.012)	1.4298 (0.009)
DTLZ7 (5)	0.8534 (0.009)	**0.8699 (0.010)**	0.1285 (0.089)	0.7547 (0.021)
DTLZ1 (6)	1.7652 (0.004)	1.7669 (0.001)	**1.7696 (0.000)**	1.7675 (0.009)
DTLZ2 (6)	1.6757 (0.014)	**1.7114 (0.006)**	1.0435 (0.010)	1.4695 (0.020)
DTLZ3 (6)	1.7710 (0.000)	**1.7712 (0.000)**	1.7702 (0.000)	1.7640 (0.007)
DTLZ4 (6)	1.7234 (0.008)	**1.7407 (0.003)**	1.0187 (0.007)	1.4427 (0.018)
DTLZ5 (6)	0.9563 (0.067)	1.1170 (0.041)	0.8622 (0.014)	**1.2291 (0.006)**
DTLZ6 (6)	1.0454 (0.174)	**1.6984 (0.004)**	1.4940 (0.021)	1.6855 (0.005)
DTLZ7 (6)	0.9319 (0.022)	**0.9380 (0.026)**	0.0255 (0.005)	0.7375 (0.009)

Table 7: Results obtained in the DTLZ test problems with up to six objective functions. We compare MD-MOEA, MAH-MOEA, MOEA/D and SMS-EMOA with respect to the running time required by each MOEA to obtain the approximation of the Pareto optimal set. The results are in seconds. We show average values over 30 independent runs. The values in parentheses correspond to the standard deviations.



$\vec{f}$	md-moea time	mah-moea time	moead time	sms-emoa time
DTLZ1 (3)	0.8317 (0.025)	26.5683 (6.888)	**0.5180 (0.009)**	1368.8483 (56.345)
DTLZ2 (3)	1.7247 (0.016)	80.2447 (11.191)	**0.5732 (0.008)**	3186.9297 (33.957)
DTLZ3 (3)	0.7897 (0.028)	26.4793 (5.165)	**0.5092 (0.008)**	2516.9973 (131.555)
DTLZ4 (3)	1.7187 (0.018)	77.7587 (9.585)	**0.5839 (0.003)**	3209.5767 (28.945)
DTLZ5 (3)	1.2467 (0.019)	9.6463 (2.308)	**0.5773 (0.023)**	485.1490 (13.942)
DTLZ6 (3)	1.0763 (0.020)	26.3760 (12.040)	**0.5055 (0.025)**	1872.8680 (494.697)
DTLZ7 (3)	1.0917 (0.036)	28.6033 (7.032)	**0.5600 (0.034)**	3047.7933 (245.354)
DTLZ1 (4)	1.0227 (0.015)	17.1963 (2.158)	**0.5386 (0.013)**	953.9040 (72.115)
DTLZ2 (4)	2.0807 (0.009)	56.9457 (9.412)	**0.6048 (0.010)**	2696.8443 (28.356)
DTLZ3 (4)	1.0830 (0.029)	23.3403 (2.858)	**0.5312 (0.010)**	1652.6900 (191.963)
DTLZ4 (4)	2.0533 (0.020)	52.1267 (9.269)	**0.6281 (0.007)**	3186.0707 (23.889)
DTLZ5 (4)	2.3523 (0.028)	50.0157 (6.454)	**0.5903 (0.001)**	1559.1663 (56.480)
DTLZ6 (4)	1.6473 (0.020)	49.7007 (6.381)	**0.5942 (0.008)**	1758.4097 (30.011)
DTLZ7 (4)	1.1903 (0.026)	16.5777 (2.984)	**0.6512 (0.008)**	3240.0063 (293.247)
DTLZ1 (5)	1.1693 (0.037)	12.5180 (1.662)	**0.5706 (0.005)**	1029.0453 (92.775)
DTLZ2 (5)	2.3043 (0.014)	29.8747 (3.466)	**0.6248 (0.001)**	1879.0867 (19.569)
DTLZ3 (5)	1.2967 (0.028)	17.2507 (2.455)	**0.5796 (0.016)**	1617.6280 (75.710)
DTLZ4 (5)	2.2767 (0.021)	28.0337 (3.063)	**0.6812 (0.001)**	2191.7757 (21.185)
DTLZ5 (5)	2.4603 (0.021)	44.2683 (5.359)	**0.6527 (0.040)**	1788.9270 (49.232)
DTLZ6 (5)	2.2447 (0.044)	46.6733 (5.942)	**0.6370 (0.006)**	1957.4123 (25.557)
DTLZ7 (5)	1.3070 (0.035)	7.3807 (1.137)	**0.6935 (0.005)**	2822.8597 (171.838)
DTLZ1 (6)	1.3033 (0.070)	10.5600 (1.946)	**0.6036 (0.016)**	1523.3607 (158.557)
DTLZ2 (6)	2.4600 (0.023)	19.9033 (1.739)	**0.6551 (0.005)**	1605.2567 (13.806)
DTLZ3 (6)	1.4473 (0.030)	15.7413 (2.575)	**0.5999 (0.013)**	1751.4160 (130.305)
DTLZ4 (6)	2.4343 (0.026)	18.3560 (2.040)	**0.7307 (0.001)**	1827.7093 (18.135)
DTLZ5 (6)	2.6093 (0.028)	37.3323 (5.122)	**0.7436 (0.001)**	1815.6567 (68.869)
DTLZ6 (6)	2.8480 (0.049)	44.0187 (5.048)	**0.6892 (0.005)**	2128.5127 (27.407)
DTLZ7 (6)	1.4347 (0.047)	3.7347 (0.632)	**0.7247 (0.004)**	2522.0583 (54.170)

Table 8: Statistical analysis using Wilcoxon's rank sum. For this, we used I_H, see Table 2 for (a) and see Table 6 for (b). P is the probability of observing the given result (the null hypothesis is true). Small values of P cast doubt on the validity of the null hypothesis. $H = 0$ indicates that the null hypothesis ("medians are equal") cannot be rejected at the 5% level. $H = 1$ indicates that the null hypothesis can be rejected at the 5% level.

$\vec{f}$	mc-moea & md-moea $P(H)$	md-moea & mah-moea $P(H)$	mah-moea & mh-moea $P(H)$
DTLZ1 (3)	0.000000 (1)	0.000000 (1)	0.464273 (0)
DTLZ2 (3)	0.000000 (1)	0.000000 (1)	0.203559 (0)
DTLZ3 (3)	0.000000 (1)	0.000000 (1)	0.100764 (0)
DTLZ4 (3)	0.000000 (1)	0.000000 (1)	0.958731 (0)
DTLZ5 (3)	0.000000 (1)	0.000000 (1)	0.125965 (0)
DTLZ6 (3)	0.000000 (1)	0.784460 (0)	0.180900 (0)
DTLZ7 (3)	0.000002 (1)	0.000001 (1)	0.510598 (0)
DTLZ1 (4)	0.000168 (1)	0.000000 (1)	0.000002 (1)
DTLZ2 (4)	0.000000 (1)	0.000000 (1)	0.048413 (1)
DTLZ3 (4)	0.000000 (1)	0.000586 (1)	0.428630 (0)
DTLZ4 (4)	0.000000 (1)	0.000000 (1)	0.355472 (0)
DTLZ5 (4)	0.000000 (1)	0.000000 (1)	0.005570 (1)
DTLZ6 (4)	0.067869 (0)	0.000000 (1)	0.046756 (1)
DTLZ7 (4)	0.000000 (1)	0.000001 (1)	0.010763 (1)
DTLZ1 (5)	0.055546 (0)	0.002624 (1)	0.000000 (1)
DTLZ2 (5)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ3 (5)	0.000007 (1)	0.111927 (0)	0.183242 (0)
DTLZ4 (5)	0.000000 (1)	0.000000 (1)	0.000015 (1)
DTLZ5 (5)	0.000000 (1)	0.000000 (1)	0.010315 (1)
DTLZ6 (5)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ7 (5)	0.000000 (1)	0.000001 (1)	0.464273 (0)
DTLZ1 (6)	0.000038 (1)	0.006096 (1)	0.000000 (1)
DTLZ2 (6)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ3 (6)	0.039873 (1)	0.003337 (1)	0.000004 (1)
DTLZ4 (6)	0.000001 (1)	0.000000 (1)	0.000000 (1)
DTLZ5 (6)	0.000000 (1)	0.000000 (1)	0.005570 (1)
DTLZ6 (6)	0.000000 (1)	0.000000 (1)	0.000001 (1)
DTLZ7 (6)	0.000000 (1)	0.153667 (0)	0.428963 (0)

(a)

$\vec{f}$	md-moea & moead $P(H)$	md-moea & sms-emoa $P(H)$	mah-moea & moead $P(H)$	mah-moea & sms-emoa $P(H)$
DTLZ1 (3)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ2 (3)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ3 (3)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ4 (3)	0.000068 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ5 (3)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.332841 (0)
DTLZ6 (3)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ7 (3)	0.000000 (1)	0.239850 (0)	0.000000 (1)	0.239850 (0)
DTLZ1 (4)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ2 (4)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ3 (4)	0.000252 (1)	0.000000 (1)	0.135171 (0)	0.000001 (1)
DTLZ4 (4)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ5 (4)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ6 (4)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ7 (4)	0.000000 (1)	0.491783 (0)	0.000000 (1)	0.000000 (1)
DTLZ1 (5)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ2 (5)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ3 (5)	0.000232 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ4 (5)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ5 (5)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ6 (5)	0.264326 (0)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ7 (5)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ1 (6)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000030 (1)
DTLZ2 (6)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ3 (6)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ4 (6)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ5 (6)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ6 (6)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)
DTLZ7 (6)	0.000000 (1)	0.000000 (1)	0.000000 (1)	0.000000 (1)

(b)