Selection Operators based on Maximin Fitness Function for Multi-Objective Evolutionary Algorithms

Adriana Menchaca-Mendez and Carlos A. Coello Coello*

CINVESTAV-IPN (Evolutionary Computation Group)
Departamento de Computación
México D.F. 07300, MÉXICO
adriana.menchacamendez@gmail.com, ccoello@cs.cinvestav.mx

Abstract. We analyze here some properties of the maximin fitness function, which has been used by several researchers, as an alternative to Pareto optimality, for solving multi-objective optimization problems. As part of this analysis, we identify some disadvantages of the maximin fitness function and then propose mechanisms to overcome them. This leads to several selection operators for multi-objective evolutionary algorithms which are further analyzed. We incorporate them into an evolutionary algorithm, giving rise to the so-called Maximin-Clustering Multi-Objective Evolutionary Algorithm (MC-MOEA) approach. Our proposed approach is validated using standard test problems taken from the specialized literature, having from two to eight objectives. Our preliminary results indicate that our proposed approach is a good alternative to solve multi-objective optimization problems having both low dimensionality (two or three) and high dimensionality (more than three) in objective function space.

1 Introduction

The use of evolutionary algorithms for solving multi-objective optimization problems (MOPs) has become very popular in the last few years [7]. When designing multi-objective evolutionary algorithms (MOEAs), there are two main types of approaches that are normally used as selection mechanism: (i) those that incorporate the concept of Pareto optimality, and (ii) those that do not use Pareto dominance to select individuals.

In this work, we are interested in the *maximin fitness function* [2] (belonging to the type (ii)). This technique assigns a fitness to each individual in the population. Such fitness value encompasses Pareto dominance (we can know which individuals are non-dominated), distance to the non-dominated individuals, and clustering between individuals (it penalizes individuals that are too close from each other). This scheme has the advantage of requiring very simple operations to calculate the fitness and is, thus, computationally efficient (its complexity is

* The second author acknowledges support from CONACyT project no. 103570.

linear with respect to the number of objectives). A preliminary study allowed us to design some selection operators based on the maximin fitness function which are incorporated into a MOEA that uses a simulated binary crossover (SBX) and a polynomial mutation operator (PM), giving rise to the main proposal of this paper, which is called: *Maximin-Clustering Multi-Objective Evolutionary Algorithm (MC-MOEA)*. The proposed approach is validated with several standard test problems using the hypervolume and the additive epsilon-indicator. Our proposed MC-MOEA approach is compared with respect to the NSGA-II (which is a very competitive Pareto-based MOEA), with respect to SMS-EMOA (which is a hypervolume-based MOEA), and with respect to a version of SMS-EMOA that uses Monte Carlo simulation to approximate the exact hypervolume (we called it APP-SMS-EMOA) [1]. Our preliminary results indicate that our proposed approach is a viable alternative, particularly when dealing with a high number of objectives, since it produces results that are similar in quality to those obtained with SMS-EMOA (low dimensionality) and APP-SMS-EMOA (high dimensionality), but at a very low computational cost.

The remainder of this paper is organized as follows. The maximin fitness function is described and studied in Section 2. Section 3 presents the proposed mechanisms to improve the maximin fitness function and describes in detail three selection operators based on it. In Section 4 we present a full description of our proposed MC-MOEA approach. Our experiments and the results obtained are shown in Section 5. Finally, we provide our conclusions and future work in Section 6.

2 Maximin Fitness Function

The maximin fitness function was proposed by Richard Balling and Scott Wilson in [2],[4] and, it works as follows. Let's consider a MOP with K objectives and an evolutionary algorithm whose population size is P. Let f_k^i be the normalized value of the k^{th} objective for the i^{th} individual in a particular generation. Assuming minimization problems, we have that the j^{th} individual weakly dominates the i^{th} individual if:

$$min_k(f_k^i - f_k^j) \geq 0 \tag{1}$$

The i^{th} individual, in a particular generation, will be weakly dominated by another individual, in the generation, if:

$$max_{j \neq i}(min_k(f_k^i - f_k^j)) \geq 0 \tag{2}$$

Then, the maximin fitness function of individual i is defined as:

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

[1] We approximate the hypervolume using the approach proposed in HyPE [1].

where the *min* is taken over all the objectives from 1 to K, and the *max* is taken over all the individuals in the population from 1 to P, except for the same individual i. From eq. (3), we can say the following:

1. Any individual whose maximin fitness is greater than zero is a dominated individual,
2. Any individual whose maximin fitness is less than zero is a non-dominated individual.
3. Finally, any individual whose maximin fitness is equal to zero is a weakly-dominated individual.

2.1 Reviewing the properties of the maximin fitness function

Let's review the properties of the maximin fitness function as presented in [3]:

1. The maximin fitness function penalizes clustering of non-dominated individuals. In the limit, the maximin fitness of duplicate non-dominated individuals is zero. See Figure 1.
2. The maximin fitness function rewards individuals at the middle of convex non-dominated fronts, see Figure 2. Also, it rewards individuals at the extremes of concave non-dominated fronts, see Figure 3. The maximin fitness function is a continuous function of objective values.
3. The maximin fitness of dominated individuals is a metric of the distance to the non-dominated front. See Figure 4.
4. 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 4.

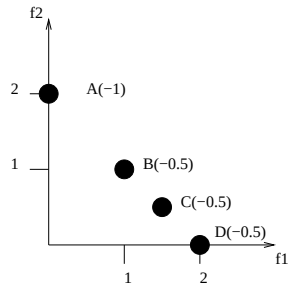

Fig. 1. We can see that the maximin fitness function penalizes individuals B, C and D because they are close from each other. It also rewards individual A, because it is far away from the other individuals.

Analyzing Property 1, we can see that although the maximin fitness function penalizes the clustering between individuals, it has the following disadvantage.

In Figure 1, we can observe that individuals B, C and D have the same maximin fitness. Then, if we use the maximin fitness function, we can not know which of the three is the best individual to form part of the next generation.

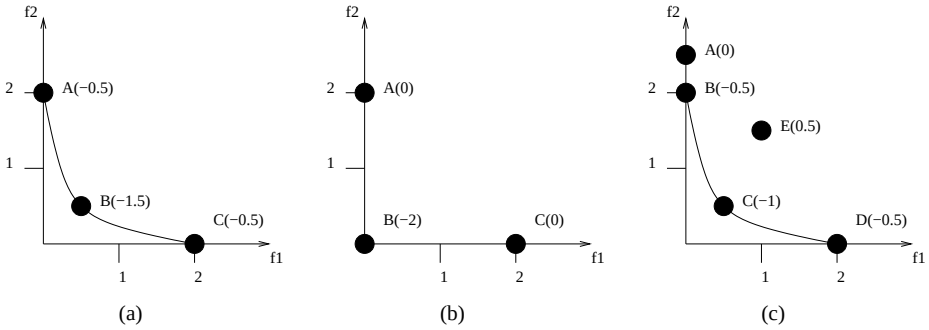

Fig. 2. In all cases, we can see that the maximin fitness function rewards individuals at the middle of convex non-dominated fronts. In (c), individual A has a maximin fitness equal to zero because it is a weakly dominated solution, and individual E has a positive maximin fitness equal to 0.5 because it is a dominated solution.

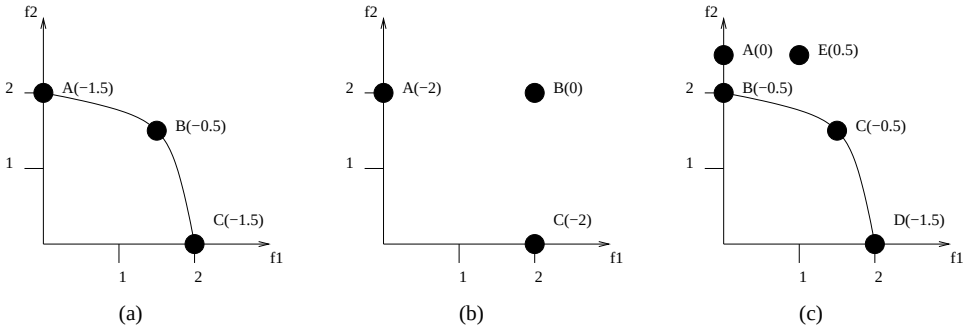

Fig. 3. In all cases, we can see that the maximin fitness function rewards individuals at the extremes of concave non-dominated fronts. In (c), individual A has a maximin fitness equal to zero because it is a weakly dominated solution, and individual E has a positive maximin fitness equal to 0.5 because it is a dominated solution.

To review Property 4, let's see Figure 4. In this case, we can see that the fitness of the non-dominated individual B is affected by the dominated individual D. Then, the maximin fitness function penalizes non-dominated individuals if they are close to another individual (no matter whether or not it is a dominated solution). The author of the maximin fitness function proposed in [4] the following modified maximin fitness function:

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

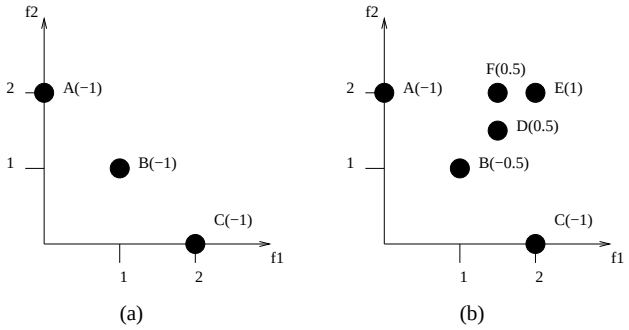

Fig. 4. In (b), we can see that the fitness of individuals D, E and F is controlled by the non-dominated individual B, and the value of their fitness is a metric of the distance to the individual B. Also, we can see that the fitness of B is affected by the dominated individual D.

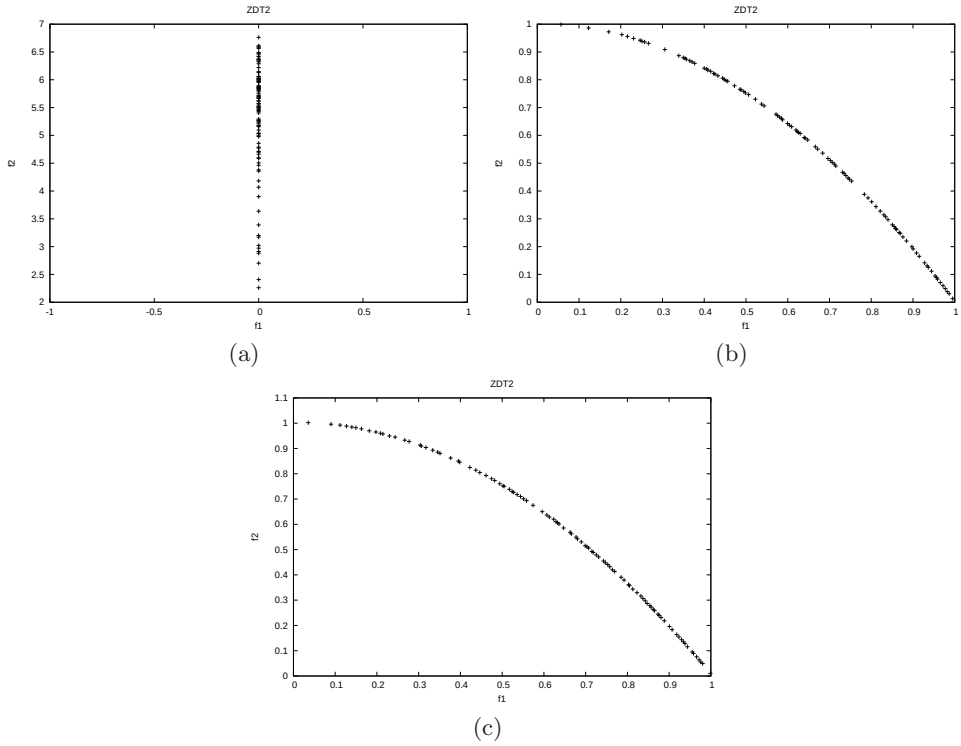

Fig. 5. In all cases, we use an evolutionary algorithm based on Differential Evolution coupled to any of the 3 selection mechanisms proposed here. In (a), we use only the maximin fitness to select individuals. In case (b), we use the maximin fitness and the constraint that prevents us from selecting similar individuals (in objective function space). Finally, in (c) we use the full selection operator proposed in this work.

where P is the set of non-dominated individuals. Using eq. (4) to assign the fitness of each individual, we guarantee that the fitness of a non-dominated individual is controlled only by non-dominated individuals and then we only penalize clustering between non-dominated individuals.

On the other hand, it is important to analyze if it is better to prefer weakly dominated individuals than dominated individuals. In the study that we will include next, we show that it is not good to prefer weakly dominated individuals or individuals which are close to being weakly dominated (even if they are weakly dominated by any dominated individual). For example, in Figure 2 (c), solution A is a weakly dominated individual and solution E is a dominated individual. To guarantee convergence to the Pareto optimal set, we must choose individual E. Otherwise, it is possible that the evolutionary algorithm converges to a weak Pareto optimal solution. Problem ZDT2 is an example of this:

$$\begin{aligned} f_1(\boldsymbol{x}) &= x_1 \\ f_2(\boldsymbol{x}) &= g(\boldsymbol{x})\left(1-(x_1/g(\boldsymbol{x}))^2\right) \\ g(\boldsymbol{x}) &= 1+\frac{9}{n-1}\sum_{i=2}^{n} x_i) \end{aligned} \tag{5}$$

where $x_i \in [0,1]$, and the problem has 30 decision variables. If we set the fitness of each individual with the maximin fitness function, into an evolutionary algorithm, and after sorting the individuals with respect to their fitness, we perform selection. Then, at the end of the generations, we obtain only weakly Pareto points, see Figure 5 (a). This happens because f_1 is easier to optimize than f_2 and then, we quickly obtain weakly dominated solutions in this extreme of the Pareto front.

3 Selection operators based on the maximin fitness function

Considering the properties of the maximin fitness function and its disadvantages, we propose here three possibles selection operators.

3.1 Operator I

In order to deal with the problem of the weakly dominated individuals, we proposed in [10] the following constraint: Any individual that we want to select must not be similar (in objective space) to another (selected) individual. The process to verify similarity between individuals is shown in Algorithm 1 and the full selection process is shown in Algorithm 2. So, we avoid selecting solutions that are weakly dominated by non-dominated solutions (see individual A in Figure 2 (c)) or solutions which are weakly dominated by dominated solutions (see individual F in Figure 4 (b)). In Figure 5 (b), we can observe that by imposing this constraint, we can find the true Pareto front of the ZDT2 function.

```
Input  : min_dif (Minimum difference), x (individual), Y (population), P (population
         size), K (number of objectives).
Output: Returns 1, if the individual x is similar to any individual in the population Y;
         otherwise, returns 0.
for i ← 1 to P do
    for k ← 1 to K do
        if |x.f[k] − Y[i].f[k]| < min_dif then
            return 1;
        end
    end
end
return 0;
```

Algorithm 1: IsSimilarToAny

In order to deal with the disadvantage of Property 1, we proposed in [10] a technique based on maximin fitness and clustering. Such a technique works as follows. If we want to select S individuals from a population of size P, then, we choose the best S individuals with respect to their maximin fitness, and use them as centers of their clusters. Then, we proceed to place each individual in the nearest cluster. Finally, for each of the resulting clusters, we recompute the center, and choose the individual closest to it. It is important to note that we don't iterate many times to improve the distribution of the centers, we only execute one time the correction. This procedure is shown in Algorithm 3.

```
Input  : X (Population), P (population size), K (number of objectives), S (the number of
         individuals to choose) and min_dif (minimum difference between objectives).
Output: Y (Selected individuals).
s ← 1, i ← 1;
/*Sorting with respect the maximin fitness of each individual                    */
X_sorted ← Sort(X);
/*Fill up the new population with the best copies according to maximin fitness,
  verifying that there is not a similar one                                      */
while s ≤ S AND i ≤ P do
    while IsSimilarToAny(min_dif, X_sorted[i], Y, s, K) = 1 AND i ≤ P do
        i ← i + 1;
    end
    if i ≤ P then
        Y[s] ← X_sorted[i];
        s ← s + 1;
    end
end
/*Fill up the new population with the best copies according to maximin fitness   */
i ← 1;
while s ≤ S do
    if X_sorted[i] has not been selected then
        Y[s] ← X_sorted[i];
        s ← s + 1;
    end
    i ← i + 1;
end
return Y;
```

Algorithm 2: Maximin-Selection

Input : X (Population), $NonDom$ (number of non-dominated individuals), K (number of objectives) and S (number of individuals to choose).
Output: Y (individuals selected).
/*Choose the best S individuals, according to maximin fitness, as centers of the clusters C */
$X_{sorted} \leftarrow Sort(X)$, $C_j = \{X_{sorted}[j]\}$;
/*Do one iteration of clustering */
for $i \leftarrow S+1$ **to** $NonDom$ **do**
  if $X_{sorted}[i]$ *is closer to* C_j **then**
    $C_j \leftarrow C_j \cup X_{sorted}[i]$;
  end
end
/*Obtain the new centers of the clusters */
for $j \leftarrow 1$ **to** S **do**
  $\mu_j \leftarrow \frac{1}{|C_j|} \sum_{X[i] \in C_j} X[i]$;
end
/*Select to individuals who are closest to the centers of the clusters */
for $j \leftarrow 1$ **to** S **do**
  if $X[i] \mid X[i] \in C_j$ *is the nearest to the center* μ_j **then**
    $Y[j] \leftarrow X[i]$;
  end
end
Returns Y;

Algorithm 3: Maximin-Clustering Selection

With the maximin-clustering technique, if we return to Figure 1 and 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. It is important to note that the above technique, which is used to improve the distribution of the selected individuals, is only effective in cases when all individuals are non-dominated. For example, if we analyze Figure 4 (b), and we want to select three individuals, our technique selects individuals A, D and C, penalizing individual B. This is clearly not good because individual B dominates individual D. In Figure 5 (c), we can see that if we use the maximin-clustering technique, we obtain a better distribution of solutions. In Algorithm 4, we describe the full selection operator.

Input : X (Current population), P (population size), K (number of objectives) and S (number of individuals to choose).
Output: Y (individuals selected).
MaximinFitnessFunction(X, P, K);
if *The number of nondominated individuals is greater to* S **then**
  $Y \leftarrow$ Maximin-Clustering Selection(X_{sorted}, P, K, S);
else
  $Y \leftarrow$ MaximinSelection(X_{sorted}, P, K, S);
end
Returns Y;

Algorithm 4: Operator I

3.2 Operator II

The clustering technique, that we propose to address the disadvantage of Property 1, makes the correction of the centers only once. Then, if we choose more efficiently the initial centers, we hope to obtain a better distribution of solutions along the Pareto front. Considering Property 4, we believe that it is a good idea to use the maximin fitness function at the beginning of the search and use the modified fitness function when we have many non-dominated individuals [2] because in this part of the search process, we are interested in obtaining a better distribution between non-dominated individuals and, the modified maximin fitness guarantees to penalize clustering only between non-dominated individuals. We show the full selection operator in Algorithm 5.

Input : X (Current population), P (population size), K (number of objectives) and S (number of individuals to choose).
Output: Y (individuals selected).
if *The number of nondominated individuals is greater to* S **then**
  ModifiedMaximinFitnessFunction(X, P, K);
  $Y \leftarrow$ Maximin-Clustering Selection(X_{sorted}, P, K, S);
else
  MaximinFitnessFunction(X, P, K);
  $Y \leftarrow$ MaximinSelection(X_{sorted}, P, K, S);
end
Returns Y;

Algorithm 5: Operator II

3.3 Operator III

For the last operator, we decided to apply the modified maximin fitness function only when we use our clustering technique, because at the beginning of the search process we need to penalize the clustering between individuals regardless of their dominance. This is because we want to explore all the search space and, therefore, we decided to use the original maximin fitness function at the beginning of the evolutionary process. However, in the third operator, we propose to use the modified maximin fitness function since the beginning of the search in order to analyze the behavior of the operator when we use the modified maximin fitness all the time. The third operator is described in Algorithm 6.

4 Maximin-Clustering Multi-Objective Evolutionary Algorithm

In order to compare the three operators based on the maximin fitness function, we designed a multi-objective evolutionary algorithm using a simulated binary

[2] Considering a $(\mu + \mu)$ selection scheme, we say that we have many non-dominated individuasl if more than μ individuals are non-dominated.

Input : X (Current population), P (population size), K (number of objectives) and S (number of individuals to choose).
Output: Y (individuals selected).
ModifiedMaximinFitnessFunction(X, P, K);
if *The number of nondominated individuals is greater to* S **then**
  $Y \leftarrow$ Maximin-Clustering Selection(X_{sorted}, P, K, S);
else
  $Y \leftarrow$ MaximinSelection(X_{sorted}, P, K, S);
end
Returns Y;

Algorithm 6: Operator III

crossover (SBX) and a polynomial mutation operator (PM) to create new individuals combined with the previously described selection operators as follows: If the size of the population is P, then we create P new individuals. The parents are selected as follows: We use a binary tournament. At each tournament, two individuals are randomly selected and the one with the higher 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 selection operators to choose the P individuals that will take part of the following generation.

5 Experimental results

Aiming to validate the selection mechanism of our proposed approach with respect to other types of mechanisms, we chose the following MOEAs: NSGA-II [8] (based on Pareto dominance) and (SMS-EMOA) [5] (based on the hypervolume performance measure [12] combined with the non-dominated sorting procedure adopted in NSGA-II). Due to the high computational cost required to calculate the hypervolume, we decided to use also an approximate calculation of the hypervolume. For this sake, we used the source code of HyPE available in the public domain [1] adopting 10^3 as our number of samples.

5.1 Experiments

For all our experiments, we used the two following sets of problems: The first consists of five bi-objective test problems taken from the Zitzler-Deb-Thiele (ZDT) test suite [13]. The second consisted of seven problems having three or more objectives, taken from the Deb-Thiele-Laumanns-Zitzler (DTLZ) suite [9]. For the DTLZ test problems, we used $k = 5$ for DTLZ1 and DTLZ6 and $k = 10$ for the remaining test problems and, three, four, five, six, seven and eight objetive functions (i.e., $M = 3$, 4, 5, 6, 7 and 8). For each test problem, we performed 30 independent runs. For all algorithms, we adopted the parameters suggested by the authors of NSGA-II: crossover probability $p_c = 0.9$, mutation probability $p_m = 1/n$, where n is the number of decision variables. Both for the crossover and mutation operators, we adopted $\eta_c = 15$ and $\eta_m = 20$, respectively. For our proposed selection operators we used $min\_dif = 0.0001$ in all cases. All approaches performed the same number of objective function evaluations. For the

ZDT test problems, we performed 20,000 evaluations (we used a population of 100 individuals and we iterated for 200 generations). For the DTLZ test problems we performed 125,000 evaluations (we used a population of 250 individuals and we iterated for 500 generations). In the case of SMS-EMOA, we adopted five hours as our maximum computation time (SMS-EMOA requires more than five hours when dealing with 4 or more objectives).

In order to assess performance, we adopted the *hypervolume indicator* (φ). [3] and the additive ϵ-indicator [4] because these two indicators are Pareto compliant and SMS-EMOA is based in φ and our MC-MOEA approach is based on the maximin fitness function, which can be considered as the binary ϵ-indicator of a solution with respect to a reference set defined by the remaining non-dominated solutions of the population. To compute φ, we used the following reference points: For the ZDT test problems, we used $y_{ref} = [1.1, 1.1]$. For DTLZ1, we used $y_{ref} = [y_1, \ldots, y_M] \mid y_i = 0.7$. For DTLZ(2-6), we used $y_{ref} = [y_1, \ldots, y_M] \mid y_i = 1.1$. For DTLZ7, we used $y_{ref} = [y_1, \ldots, y_M]$ $y_M = 6.1$ and $y_{i \neq M} = 1.1$.

5.2 Results

In Table 1, we present the results with respect to the hypervolume indicator and we can see that our MC-MOEAs obtained competitive results with respect to SMS-EMOA and APP-SMS-EMOA. One important thing is that our proposed MC-MOEAs presented a consistent behavior when we increased the number of objecives unlike NSGA-II. To validate the results in our experiments, we performed a statistical analysis using Wilcoxon's rank sum on our MC-MOEAs with respect to SMS-EMOA and APP-SMS-EMOA and, we obtained that only in the problems DTLZ3 (with 3, 5, 6, and 7 objectives) and DTLZ6 (with 4 and 5 objectives) the null hypothesis ("medians are equal") can be rejected at the 5% level. If we check these problems in Table 1, we can see that our MC-MOEAs obtained better results than SMS-EMOA and APP-SMS-EMOA. This means that in these problems our MC-MEOAs significantly outperformed both SMS-EMOA and APP-SMS-EMOA. Note, however, that we could not include the table with the results of the Wilcoxon's rank sum due to space limitations.

Table 3 shows the results with respect to the additive epsilon indicator. In this case, we only compared our MC-MOEAs with respect to APP-SMS-EMOA because, as noted in Table 1, NSGA-II did not have a consistent behavior when we increased the number of objectives and SMS-EMOA required a very large computational time, making the comparison unfair. The results show that our MC-MOEAs outperformed APP-SMS-EMOA in most cases. For these experiments, we also performed a statistical analysis and we obtained that in the test

[3] The hypervolume was originally proposed by Zitzler and Thiele in [14], and it's defined as the size of the space covered by the Pareto optimal solutions. *φ rewards convergence towards the Pareto front as well as the maximum spread of the solutions obtained*. The disadvantage of this indicator is its high computational cost (the running time for calculating φ is exponential in the number of objectives).

[4] Given two approximate sets, A and B, the ϵ-indicator measures the smallest amount, ϵ, that must be used to translate the set, A, so that every point in B is covered [7].

	nsgaii	sms-emoa	app-sms-emoa	mc-moea(v1)	mc-moea(v2)	mc-moea(v3)
ZDT1 (2)	0.867920 (0.000489)	**0.871433 (0.000093)**	0.867986 (0.000421)	0.862576 (0.001036)	0.864293 (0.000873)	0.864215 (0.000937)
ZDT2 (2)	0.534002 (0.000553)	**0.537417 (0.001047)**	0.533294 (0.002330)	0.520382 (0.006974)	0.525339 (0.002581)	0.526132 (0.001956)
ZDT3 (2)	**1.325397 (0.000708)**	1.320005 (0.024943)	1.305149 (0.027317)	1.307519 (0.025795)	1.312000 (0.024431)	1.306731 (0.030536)
ZDT4 (2)	**0.856773 (0.010887)**	0.804517 (0.050998)	0.782593 (0.069529)	0.818084 (0.062050)	0.829491 (0.057508)	0.840295 (0.035847)
ZDT6 (2)	0.481464 (0.003094)	**0.490964 (0.002359)**	0.485441 (0.002728)	0.467497 (0.006066)	0.468000 (0.005054)	0.468757 (0.007073)
ZDT6 (2)	0.481464 (0.003094)	**0.490964 (0.002359)**	0.485214 (0.002735)	0.467497 (0.006066)	0.468000 (0.005054)	0.468757 (0.007073)
DTLZ1 (3)	0.315526 (0.000569)	**0.319138 (0.000008)**	0.306568 (0.006481)	0.315955 (0.000714)	0.316679 (0.000791)	0.316622 (0.000703)
DTLZ2 (3)	0.721960 (0.010450)	**0.776854 (0.000021)**	0.761623 (0.003126)	0.734580 (0.005829)	0.738114 (0.004938)	0.738805 (0.005594)
DTLZ3 (3)	0.059242 (0.179471)	0.000000 (0.000000)	0.000000 (0.000000)	0.715957 (0.011959)	0.716155 (0.013048)	**0.718686 (0.010902)**
DTLZ4 (3)	0.722625 (0.014284)	**0.776861 (0.000026)**	0.763312 (0.002203)	0.736373 (0.005998)	0.742850 (0.004866)	0.744456 (0.004385)
DTLZ5 (3)	0.440252 (0.000208)	**0.441267 (0.000003)**	0.438059 (0.001219)	0.430387 (0.005275)	0.431899 (0.003225)	0.432158 (0.003319)
DTLZ6 (3)	0.406129 (0.024792)	**0.414851 (0.019473)**	0.350072 (0.101829)	0.390702 (0.030285)	0.400420 (0.019828)	0.392088 (0.018441)
DTLZ7 (3)	2.007096 (0.005727)	**2.037787 (0.089016)**	1.923641 (0.120586)	1.992563 (0.011031)	2.002309 (0.005891)	1.984739 (0.085183)
DTLZ1 (4)	0.199714 (0.072810)	0.227113 (0.019905)	0.225110 (0.004836)	0.232710 (0.000636)	**0.232832 (0.000753)**	0.232626 (0.000641)
DTLZ2 (4)	0.916315 (0.050375)	**1.074969 (0.000534)**	1.048196 (0.005676)	0.965540 (0.011771)	0.972949 (0.009525)	0.972032 (0.008293)
DTLZ3 (4)	0.000000 (0.000000)	0.000000 (0.000000)	0.000000 (0.000000)	0.940267 (0.027559)	**0.945389 (0.021329)**	0.937558 (0.028903)
DTLZ4 (4)	0.919121 (0.055369)	**1.076531 (0.000323)**	1.049698 (0.002584)	0.983616 (0.008471)	0.990356 (0.007794)	0.991357 (0.008939)
DTLZ5 (4)	0.427341 (0.002086)	**0.434905 (0.002811)**	0.411700 (0.012550)	0.246597 (0.021079)	0.252224 (0.025367)	0.262790 (0.019642)
DTLZ6 (4)	0.082789 (0.025430)	0.005002 (0.003624)	0.017953 (0.015585)	0.188079 (0.018994)	0.245121 (0.013218)	**0.245153 (0.013223)**
DTLZ7 (4)	0.673851 (0.019135)	**0.799224 (0.179936)**	0.516516 (0.139265)	0.697724 (0.011902)	0.718840 (0.012895)	0.715425 (0.012490)
DTLZ1 (5)	0.000000 (0.000000)	0.000000 (0.000000)	0.160563 (0.002329)	0.158865 (0.028230)	0.163938 (0.000865)	**0.164163 (0.000575)**
DTLZ2 (5)	0.607031 (0.362156)	0.810554 (0.081746)	**1.292202 (0.005203)**	1.129832 (0.016540)	1.137390 (0.016104)	1.138317 (0.014105)
DTLZ3 (5)	0.000000 (0.000000)	0.000000 (0.000000)	0.000000 (0.000000)	**1.127122 (0.038275)**	1.125465 (0.030709)	1.108675 (0.048260)
DTLZ4 (5)	0.552563 (0.394310)	0.619604 (0.094095)	**1.288936 (0.008553)**	1.163695 (0.015934)	1.173994 (0.015513)	1.173374 (0.016415)
DTLZ5 (5)	0.418359 (0.004244)	0.253338 (0.015809)	**0.419592 (0.012492)**	0.190355 (0.025803)	0.193947 (0.022278)	0.201230 (0.020934)
DTLZ6 (5)	0.000000 (0.000000)	0.000000 (0.000000)	0.002493 (0.003237)	0.071664 (0.017627)	**0.113595 (0.020441)**	0.113595 (0.020441)
DTLZ7 (5)	0.105390 (0.008146)	0.003702 (0.005092)	0.071404 (0.047761)	0.116535 (0.007517)	0.120316 (0.008210)	**0.120386 (0.008311)**
DTLZ1 (6)	0.000000 (0.000000)	-	**0.114030 (0.001116)**	0.096169 (0.041014)	0.103155 (0.032221)	0.102862 (0.033258)
DTLZ2 (6)	0.014439 (0.028545)	-	**1.497165 (0.012818)**	1.240839 (0.031766)	1.243283 (0.032054)	1.251246 (0.031293)
DTLZ3 (6)	0.000000 (0.000000)	-	0.000000 (0.000000)	**1.181907 (0.234392)**	1.118765 (0.380109)	1.009064 (0.459079)
DTLZ4 (6)	0.015083 (0.033135)	-	**1.511018 (0.005841)**	1.272827 (0.036356)	1.290147 (0.035050)	1.290147 (0.035050)
DTLZ5 (6)	0.402419 (0.022651)	-	**0.423127 (0.016780)**	0.171457 (0.020233)	0.172391 (0.022213)	0.171830 (0.018877)
DTLZ6 (6)	0.000000 (0.000000)	-	**0.000017 (0.000060)**	0.000000 (0.000000)	0.000000 (0.000000)	0.000000 (0.000000)
DTLZ7 (6)	0.010739 (0.002179)	-	0.002785 (0.004630)	0.009829 (0.001956)	0.010246 (0.002577)	**0.010804 (0.002256)**
DTLZ1 (7)	0.000000 (0.000000)	-	**0.080705 (0.001011)**	0.050365 (0.035001)	0.057601 (0.031938)	0.052974 (0.031998)
DTLZ2 (7)	0.002535 (0.007792)	-	**1.663548 (0.020716)**	1.266017 (0.083504)	1.307874 (0.073108)	1.313721 (0.050521)
DTLZ3 (7)	0.000000 (0.000000)	-	0.000000 (0.000000)	0.666609 (0.658681)	0.841286 (0.642597)	**0.901860 (0.597147)**
DTLZ4 (7)	0.000049 (0.000228)	-	**1.721242 (0.010973)**	1.290180 (0.092052)	1.322271 (0.085731)	1.322271 (0.085731)
DTLZ5 (7)	0.426098 (0.017081)	-	**0.435360 (0.024210)**	0.141655 (0.023779)	0.157740 (0.025355)	0.173623 (0.029061)
DTLZ6 (7)	**0.000000 (0.000000)**	-	0.000000 (0.000000)	0.000000 (0.000000)	0.000000 (0.000000)	0.000000 (0.000000)
DTLZ7 (7)	**0.000894 (0.000231)**	-	0.000000 (0.000000)	0.000416 (0.000230)	0.000372 (0.000232)	0.000360 (0.000220)
DTLZ1 (8)	0.000000 (0.000000)	-	**0.056576 (0.000741)**	0.027953 (0.021725)	0.024212 (0.024790)	0.025604 (0.022569)
DTLZ2 (8)	0.000000 (0.000000)	-	**1.811587 (0.035750)**	1.061160 (0.251401)	1.277504 (0.107501)	1.281810 (0.111406)
DTLZ3 (8)	0.000000 (0.000000)	-	0.000000 (0.000000)	**0.388426 (0.636684)**	0.372327 (0.571584)	0.226089 (0.460711)
DTLZ4 (8)	0.000221 (0.001189)	-	**1.938835 (0.014867)**	0.898659 (0.382592)	1.143849 (0.270982)	1.143849 (0.270982)
DTLZ5 (8)	0.446433 (0.028464)	-	**0.464482 (0.023309)**	0.136816 (0.037378)	0.155077 (0.028200)	0.152089 (0.030343)
DTLZ6 (8)	**0.000000 (0.000000)**	-	0.000000 (0.000000)	0.000000 (0.000000)	0.000000 (0.000000)	0.000000 (0.000000)
DTLZ7 (8)	**0.000085 (0.000026)**	-	0.000000 (0.000000)	0.000015 (0.000019)	0.000016 (0.000013)	0.000016 (0.000013)

Table 1. Results obtained with respect to the hypervolume indicator. The value in parentheses in the first column indicates the number of objectives. We show average values over 30 independent runs. The values in parentheses of the other columns correspond to the standard deviations.

problems in which APP-SMS-EMOA outperformed our MC-MOEAs, we can not reject the null hypothesis ("medians are equal") and, in many cases, when our MC-MOEAs outperformed APP-SMS-EMOA, the null hypothesis can be

Set of problems	Objectives	NSGA-II	SMS-EMOA	App-SMS-EMOA	MC-MOEAs
ZDT	2	$\lesssim 1s$	$5s - 10s$	$5s - 10s$	$\lesssim 1s$
DTLZ	3	$2s - 4s$	$4568s - 8468s$	$231s - 307s$	$3s - 9s$
DTLZ	4	$3s - 4s$	$14448s - 14650s$	$378s - 423s$	$5s - 12s$
DTLZ	5	$4s - 5s$	$15423s - 18000s$	$472s - 499s$	$9s - 14s$
DTLZ	6	$5s - 6s$	-	$531s - 584s$	$8s - 16s$
DTLZ	7	$5 - 6s$	-	$536s - 583s$	$9 - 18s$
DTLZ	8	$5s - 7s$	-	$525s - 583$	$9s - 16s$

Table 2. Running time required per run, s = seconds. All algorithms were implemented in the C programming language and they were executed on PCs with the same hardware and software platform.

rejected at the 5% level. An important advantage of our MC-MOEAs is that computing the maximin fitness function is an inexpensive process, since their complexities are linear with respect to the number of objectives. In Table 2, we can see that our MC-MOEAs require much less time than SMS-EMOA and even much less time than APP-SMS-EMOA. Thus, we argue that our MC-MOEAs can be a good alternative for dealing with many objective optimization problems.

6 Conclusions and Future Work

In this work, we have studied the maximin fitness function and its properties with the aim of identifying its advantages and disadvantages. Then, we proposed some mechanisms to improve it. Our study encompassed three selection operators (one of them was proposed in [10] and the other two were proposed here). These operators were incorporated into a MOEA that uses simulated binary crossover (SBX) and parameter-based mutation (PM), giving rise to the main proposal of this paper, which is called: *Maximin-Clustering Multi-Objective Evolutionary Algorithm (MC-MOEA)*. We compared our proposed MC-MOEA with respect to a Pareto-based MOEA (NSGA-II) and with respect to two hypervolume-based MOEAs (SMS-EMOA and APP-SMS-EMOA). Our results showed that our MC-MOEA outperformed NSGA-II in most cases and that it was competitive with respect to SMS-EMOA and APP-SMS-EMOA with respect to the hypervolume indicator, but at a much lower computational cost. Also, it was better than APP-SMS-EMOA in most cases with respect to additive epsilon indicator. Thus, we believe that our proposed selection operators can be a viable alternative for dealing (at an affordable computational cost) with many-objective optimization problems. As part of our future work, we plan to study the behavior of our selection operators if we allow that the clustering technique iterates for a longer time. We also plan to incorporate our selection operator into a different approach (e.g., particle swarm optimization) in order to assess the impact of the search engine in the results. Finally, we plan to compare our approach with respect to AGE, which is based on Maximin fitness [6], and with respect to MOEA/D, which is based on decomposition and is known to be very competitive [11].

References

1. Johannes Bader and Eckart Zitzler. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *Evolutionary Computation*, 19(1):45–76,

	mc-moea(v1)	app-sms-emoa	mc-moea(v2)	app-sms-emoa	mc-moea(v3)	app-sms-emoa
ZDT1 (2)	**0.002122 (0.001941)**	0.266333 (0.027070)	**0.001478 (0.000885)**	0.316856 (0.025420)	**0.001500 (0.001088)**	0.327322 (0.028306)
ZDT2 (2)	**0.001022 (0.001199)**	0.380433 (0.038094)	**0.000467 (0.000452)**	0.460300 (0.038666)	**0.000556 (0.000482)**	0.420622 (0.038305)
ZDT3 (2)	**0.002122 (0.002642)**	0.290033 (0.042676)	**0.001022 (0.001229)**	0.326000 (0.036908)	**0.001467 (0.002229)**	0.308822 (0.036630)
ZDT4 (2)	0.319067 (0.157283)	**0.310033 (0.165624)**	0.347045 (0.154540)	**0.291311 (0.182824)**	**0.308167 (0.145411)**	0.321345 (0.220199)
ZDT6 (2)	**0.002811 (0.008674)**	0.918600 (0.057853)	**0.001633 (0.003862)**	0.931322 (0.051950)	**0.003433 (0.007189)**	0.899189 (0.070228)
DTLZ1 (3)	**0.000004 (0.000024)**	0.003804 (0.000797)	**0.000004 (0.000024)**	0.006236 (0.000774)	**0.000000 (0.000000)**	0.006609 (0.000824)
DTLZ2 (3)	**0.000000 (0.000000)**	0.080467 (0.002483)	**0.000000 (0.000000)**	0.088733 (0.003025)	**0.000000 (0.000000)**	0.090404 (0.003393)
DTLZ3 (3)	0.000400 (0.000238)	**0.000391 (0.000119)**	**0.000373 (0.000116)**	0.001600 (0.000391)	**0.000365 (0.000091)**	0.001276 (0.000299)
DTLZ4 (3)	**0.000000 (0.000000)**	0.079884 (0.003326)	**0.000000 (0.000000)**	0.096458 (0.003444)	**0.000000 (0.000000)**	0.090964 (0.002840)
DTLZ5 (3)	**0.000000 (0.000000)**	0.047689 (0.002008)	**0.000000 (0.000000)**	0.064009 (0.002219)	**0.000000 (0.000000)**	0.063933 (0.002449)
DTLZ6 (3)	**0.182560 (0.110139)**	0.285391 (0.264605)	**0.161787 (0.084851)**	0.293862 (0.313067)	**0.139133 (0.071011)**	0.368169 (0.353202)
DTLZ7 (3)	**0.000000 (0.000000)**	0.052507 (0.003122)	**0.000004 (0.000024)**	0.064120 (0.004345)	**0.000009 (0.000033)**	0.060058 (0.003881)
DTLZ1 (4)	**0.000000 (0.000000)**	0.008422 (0.000779)	**0.000000 (0.000000)**	0.020658 (0.001978)	**0.000004 (0.000024)**	0.026813 (0.002857)
DTLZ2 (4)	**0.000000 (0.000000)**	0.102338 (0.005304)	**0.000000 (0.000000)**	0.107911 (0.004461)	**0.000000 (0.000000)**	0.109924 (0.004891)
DTLZ3 (4)	**0.000271 (0.000064)**	0.000307 (0.000085)	**0.000329 (0.000136)**	0.001991 (0.000666)	**0.000325 (0.000095)**	0.002360 (0.000945)
DTLZ4 (4)	**0.000000 (0.000000)**	0.110169 (0.004690)	**0.000000 (0.000000)**	0.121022 (0.004768)	**0.000000 (0.000000)**	0.124916 (0.005537)
DTLZ5 (4)	**0.000053 (0.000065)**	0.249071 (0.013026)	**0.000031 (0.000056)**	0.242676 (0.011803)	**0.000027 (0.000053)**	0.234964 (0.013292)
DTLZ6 (4)	0.170471 (0.041638)	**0.135711 (0.018356)**	0.211160 (0.046797)	**0.157236 (0.019663)**	0.210382 (0.046157)	**0.155605 (0.019630)**
DTLZ7 (4)	**0.000027 (0.000053)**	0.087022 (0.013190)	**0.000035 (0.000059)**	0.080956 (0.011480)	**0.000018 (0.000045)**	0.089262 (0.012833)
DTLZ1 (5)	**0.000000 (0.000000)**	0.089964 (0.008869)	**0.000000 (0.000000)**	0.076747 (0.014855)	**0.000000 (0.000000)**	0.070053 (0.012736)
DTLZ2 (5)	**0.000000 (0.000000)**	0.134591 (0.006910)	**0.000000 (0.000000)**	0.155960 (0.008473)	**0.000000 (0.000000)**	0.141640 (0.007720)
DTLZ3 (5)	**0.000124 (0.000059)**	0.000533 (0.000138)	**0.000124 (0.000059)**	0.000462 (0.000082)	**0.000124 (0.000048)**	0.000458 (0.000149)
DTLZ4 (5)	**0.000000 (0.000000)**	0.210578 (0.008651)	**0.000000 (0.000000)**	0.194676 (0.008037)	**0.000000 (0.000000)**	0.187938 (0.007237)
DTLZ5 (5)	**0.000031 (0.000056)**	0.164760 (0.007025)	**0.000027 (0.000063)**	0.176467 (0.007686)	**0.000027 (0.000072)**	0.182858 (0.008700)
DTLZ6 (5)	0.094347 (0.034605)	**0.042333 (0.009989)**	0.142182 (0.037791)	**0.041249 (0.009875)**	0.142182 (0.037791)	**0.041249 (0.009875)**
DTLZ7 (5)	**0.000000 (0.000000)**	0.099422 (0.023848)	**0.000000 (0.000000)**	0.095249 (0.024001)	**0.000004 (0.000024)**	0.083547 (0.022144)
DTLZ1 (6)	**0.000000 (0.000000)**	0.254507 (0.012576)	**0.000000 (0.000000)**	0.239484 (0.018511)	**0.000000 (0.000000)**	0.233298 (0.018348)
DTLZ2 (6)	**0.000000 (0.000000)**	0.218396 (0.017481)	**0.000000 (0.000000)**	0.222662 (0.018911)	**0.000000 (0.000000)**	0.207076 (0.016790)
DTLZ3 (6)	0.000178 (0.000173)	**0.000067 (0.000096)**	**0.000240 (0.000240)**	0.000604 (0.000315)	**0.000151 (0.000178)**	0.000973 (0.000239)
DTLZ4 (6)	**0.000000 (0.000000)**	0.354707 (0.010429)	**0.000000 (0.000000)**	0.337271 (0.008929)	**0.000000 (0.000000)**	0.337271 (0.008929)
DTLZ5 (6)	**0.000040 (0.000092)**	0.150840 (0.006913)	**0.000018 (0.000045)**	0.136347 (0.007308)	**0.000000 (0.000000)**	0.157738 (0.006627)
DTLZ6 (6)	**0.000000 (0.000000)**	0.990302 (0.005175)	**0.000000 (0.000000)**	0.987258 (0.006683)	**0.000000 (0.000000)**	0.987258 (0.006683)
DTLZ7 (6)	**0.000000 (0.000000)**	0.084836 (0.027049)	**0.000000 (0.000000)**	0.077058 (0.025112)	**0.000000 (0.000000)**	0.078431 (0.024091)
DTLZ1 (7)	**0.000000 (0.000000)**	0.552369 (0.027195)	**0.000000 (0.000000)**	0.507302 (0.035147)	**0.000000 (0.000000)**	0.582867 (0.031450)
DTLZ2 (7)	**0.000000 (0.000000)**	0.327400 (0.026559)	**0.000000 (0.000000)**	0.283538 (0.023866)	**0.000000 (0.000000)**	0.274733 (0.023531)
DTLZ3 (7)	**0.000169 (0.000179)**	0.004080 (0.003656)	**0.000227 (0.000176)**	0.003151 (0.001397)	**0.000253 (0.000177)**	0.003169 (0.001462)
DTLZ4 (7)	**0.000000 (0.000000)**	0.525089 (0.015005)	**0.000000 (0.000000)**	0.496098 (0.011994)	**0.000000 (0.000000)**	0.496098 (0.011994)
DTLZ5 (7)	**0.000009 (0.000033)**	0.135000 (0.006122)	**0.000035 (0.000059)**	0.144400 (0.006197)	**0.000013 (0.000040)**	0.133316 (0.005925)
DTLZ6 (7)	**0.000000 (0.000000)**	0.935022 (0.019142)	**0.000000 (0.000000)**	0.932467 (0.019074)	**0.000000 (0.000000)**	0.932467 (0.019074)
DTLZ7 (7)	**0.000000 (0.000000)**	0.042213 (0.019122)	**0.000000 (0.000000)**	0.038800 (0.017292)	**0.000000 (0.000000)**	0.040049 (0.017645)
DTLZ1 (8)	**0.000000 (0.000000)**	0.752924 (0.041588)	**0.000000 (0.000000)**	0.739569 (0.032785)	**0.000000 (0.000000)**	0.779698 (0.039500)
DTLZ2 (8)	**0.000000 (0.000000)**	0.480196 (0.024395)	**0.000000 (0.000000)**	0.370102 (0.019897)	**0.000000 (0.000000)**	0.368333 (0.019622)
DTLZ3 (8)	**0.000800 (0.000958)**	0.007791 (0.006625)	**0.000755 (0.000825)**	0.004436 (0.002156)	**0.000711 (0.000816)**	0.006440 (0.004454)
DTLZ4 (8)	**0.000000 (0.000000)**	0.778618 (0.018189)	**0.000000 (0.000000)**	0.691338 (0.021952)	**0.000000 (0.000000)**	0.691338 (0.021952)
DTLZ5 (8)	**0.000013 (0.000040)**	0.144769 (0.005696)	**0.000004 (0.000024)**	0.141418 (0.004752)	**0.000018 (0.000045)**	0.144693 (0.004958)
DTLZ6 (8)	**0.000000 (0.000000)**	0.831440 (0.040860)	**0.000000 (0.000000)**	0.814929 (0.043509)	**0.000000 (0.000000)**	0.814929 (0.043509)
DTLZ7 (8)	**0.000000 (0.000000)**	0.035400 (0.012263)	**0.000000 (0.000000)**	0.039831 (0.012084)	**0.000000 (0.000000)**	0.039831 (0.012084)

Table 3. Results obtained with respect to the additive epsilon indicator. The value in parentheses in the first column indicates the number of objectives. We show average values over 30 independent runs. The values in parentheses of the other columns correspond to the standard deviations.

2. Richard Balling. Pareto sets in decision-based design. *Journal of Engineering Valuation and Cost Analysis*, 3:189–198, 2000.
3. Richard Balling. The Maximin Fitness Function; Multiobjective City and Regional Planning. In Carlos M. Fonseca, Peter J. Fleming, Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele, editors, *Evolutionary Multi-Criterion Optimization. Second International Conference, EMO 2003*, pages 1–15, Faro, Portugal, April 2003. Springer. Lecture Notes in Computer Science. Volume 2632.
4. Richard Balling and Scott Wilson. The Maximin Fitness Function for Multiobjective Evolutionary Computation: Application to City Planning. In Lee Spector, Erik D. Goodman, Annie Wu, W.B. Langdon, Hans-Michael Voigt, Mitsuo Gen, Sandip Sen, Marco Dorigo, Shahram Pezeshk, Max H. Garzon, and Edmund Burke, editors, *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO'2001)*, pages 1079–1084, San Francisco, California, 2001. Morgan Kaufmann Publishers.
5. Nicola Beume, Boris Naujoks, and Michael Emmerich. SMS-EMOA: Multiobjective selection based on dominated hypervolume. *European Journal of Operational Research*, 181(3):1653–1669, 16 September 2007.
6. Karl Bringmann, Tobias Friedrich, Frank Neumann, and Markus Wagner. Approximation-guided evolutionary multi-objective optimization. In *Proceedings of the Twenty-Second international joint conference on Artificial Intelligence - Volume Volume Two*, IJCAI'11, pages 1198–1203. AAAI Press, 2011.
7. Carlos A. Coello Coello, Gary B. Lamont, and David A. Van Veldhuizen. *Evolutionary Algorithms for Solving Multi-Objective Problems*. Springer, New York, second edition, September 2007. ISBN 978-0-387-33254-3.
8. Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA–II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.
9. Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable Test Problems for Evolutionary Multiobjective Optimization. In Ajith Abraham, Lakhmi Jain, and Robert Goldberg, editors, *Evolutionary Multiobjective Optimization. Theoretical Advances and Applications*, pages 105–145. Springer, USA, 2005.
10. Adriana Menchaca-Mendez and Carlos 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)*, pages 3143–3150, Brisbane, Australia, June 10-15 2012. IEEE Press.
11. Qingfu Zhang and Hui Li. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, December 2007.
12. Eckart Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, November 1999.
13. Eckart Zitzler, Kalyanmoy Deb, and Lothar Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, Summer 2000.
14. Eckart Zitzler and Lothar Thiele. Multiobjective Optimization Using Evolutionary Algorithms—A Comparative Study. In A. E. Eiben, editor, *Parallel Problem Solving from Nature V*, pages 292–301, Amsterdam, September 1998. Springer-Verlag.