# The Role of $\epsilon$-dominance in Multi Objective Particle Swarm Optimization Methods

**Sanaz Mostaghim**
Electrical Engineering Department,
University of Paderborn,
Paderborn, Germany
mostaghim@date.upb.de

**Jürgen Teich**
Computer Science Department
Friedrich-Alexander-University,
Erlangen, Germany
teich@informatik.uni-erlangen.de

**Abstract- In this paper, the influence of $\epsilon$-dominance on Multi-objective Particle Swarm Optimization (MOPSO) methods is studied. The most important role of $\epsilon$-dominance is to bound the number of non-dominated solutions stored in the archive (archive size), which has influences on computational time, convergence and diversity of solutions. Here, $\epsilon$-dominance is compared with the existing clustering technique for fixing the archive size and the solutions are compared in terms of computational time, convergence and diversity. A new diversity metric is also suggested. The results show that the $\epsilon$-dominance method can find solutions much faster than the clustering technique with comparable and even in some cases better convergence and diversity.**

## 1 Introduction

Archiving is studied in many Multi-objective Optimization (MO) methods. In the context of Evolutionary MO methods, archiving is called elitism and is used in several methods like Rudolph's Elitist MOEA, Elitist NSGA-II, SPEA, PAES (see [2] for all) and SPEA2 [17]. In these methods, the *non-dominated* (best) solutions of each generation are kept in an external population, called archive. Therefore the archive must be updated in each generation. The time needed for updating the archive depends on the archive size, population size and the number of objectives and increases extremely when increasing the values of these three factors [10]. There are a few studies on data structures for storing the archive as a non-dominated set e.g., in [12, 6]. These data structures also take lots of time, when increasing the archive size. Indeed it is reasonable to fix the size of the archive to avoid the large number of comparisons during updating. There are several methods, like clustering, truncation in SPEA2 and crowding techniques to fix the archive size. These methods must also keep good diversity of solutions, which tends to make them the most expensive part in the updating procedure. Here, we propose to use the idea of $\epsilon$-dominance in [13, 9] to fix the size of the archive to a certain amount. This size depends on $\epsilon$. By increasing $\epsilon$, the archive size decreases. We use this method to obtain the approximate Pareto-front and compare the method with the existing MOPSO method which uses a clustering technique. The $\epsilon$-dominance method has influence on the convergence and diversity of solutions, while reducing the computational time. In some cases the computational time is much less than 100 times that of the clustering technique. The comparison is done in terms of computational time, convergence and diversity of solutions. There are several metrics for comparing convergence and diversity of solutions. Here we suggest a new idea for a diversity metric (Sigma method), which is inspired from [11]. In [11], we have proposed the Sigma method for finding the local guides in MOPSO. The idea of this method can be used to find a good diversity metric for different test functions, however, for some functions other diversity metrics are also suggested [8]. We also study using an initial archive instead of an empty one. This has more influences in MOPSO techniques than other MO methods. The empty archive is filled in the first generation by the non-dominated solutions in the initial population and these archive members will be the local guides for the particles in the population. But if they are not in well distributed positions, we will lose the diversity of solutions just after one generation, therefore there is the need to have an initial well distributed archive.

In this paper, the definitions of domination, and $\epsilon$-domination are studied in Section 2. In Section 3 the MOPSO method is briefly reviewed and in Section 4 the combination of MOPSO and $\epsilon$-dominance, the results on different test functions and comparison with clustering technique are studied. Finally we conclude the paper in Section 5.

## 2 Definitions

A multi-objective optimization problem is of the form

$$minimize \quad \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x})\} \tag{1}$$

subject to $\vec{x} \in S$, involving $m$ conflicting objective functions $f_i : \Re^n \rightarrow \Re$ that we want to minimize simultaneously. The *decision vectors* $\vec{x} = (x_1, x_2, \ldots, x_n)^T$ belong to the feasible region $S \subset \Re^n$. The feasible region is formed by constraint functions.
We denote the image of the feasible region by $Z \subset \Re^m$ and call it a feasible objective region. The elements of $Z$ are called objective vectors and they consist of objective (function) values $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \ldots, f_m(\vec{x}))$.

**Definition 1: Domination** A decision vector $\vec{x}_1 \in S$ is said to *dominate* a decision vector $\vec{x}_2 \in S$ (denoted $\vec{x}_1 \prec \vec{x}_2$) iff:

- The decision vector $\vec{x}_1$ is not worse than $\vec{x}_2$ in all objectives, i.e., $f_i(\vec{x}_1) \leq f_i(\vec{x}_2)\ \forall i = 1, \ldots, m$.

- The decision vector $\vec{x}_1$ is strictly better than $\vec{x}_2$ in at least one objective, or $f_i(\vec{x}_1) < f_i(\vec{x}_2)$ for at least one $i = 1, \ldots, m$.

**Definition 2: Weak Domination** A decision vector $\vec{x}_1$ *weakly dominates* $\vec{x}_2$ (denoted $\vec{x}_1 \preceq \vec{x}_2$) iff:

- The decision vector $\vec{x}_1$ is not worse than $\vec{x}_2$ in all objectives, i.e., $f_i(\vec{x}_1) \leq f_i(\vec{x}_2)\ \forall i = 1, \ldots, m$.

**Definition 3: Pareto Optimal Front** A decision vector $\vec{x}_1 \in S$ is called *Pareto-optimal* if there does not exist another $\vec{x}_2 \in S$ that dominates it. An objective vector is called Pareto-optimal if the corresponding decision vector is Pareto-optimal.

Let $F \subseteq \Re^m$ be a set of vectors. The Pareto Optimal Front $F^* \subseteq F$ contains all vectors $\vec{x}_1 \in F$, which are not dominated by any vector $\vec{x}_2 \in F$:

$$F^* := \{\vec{x}_1 \in F | \nexists\, \vec{x}_2 \in F : \vec{x}_2 \prec \vec{x}_1\} \tag{2}$$

**Definition 4: $\epsilon$-domination** A decision vector $\vec{x}_1 \in S$ is said to *$\epsilon$-dominate* a decision vector $\vec{x}_2 \in S$ for some $\epsilon > 0$ (denoted $\vec{x}_1 \prec_\epsilon \vec{x}_2$) iff:

- $f_i(\vec{x}_1)/(1+\epsilon) \leq f_i(\vec{x}_2)\ \forall i = 1, \ldots, m$.
- $f_i(\vec{x}_1)/(1+\epsilon) < f_i(\vec{x}_2)$ for at least one $i = 1, \ldots, m$.

Figure 1 shows the concept of $\epsilon$-domination. By considering this definition, the domination areas increase by increasing the objective values. For smaller values of objectives the dominating area is smaller than for larger objective values.

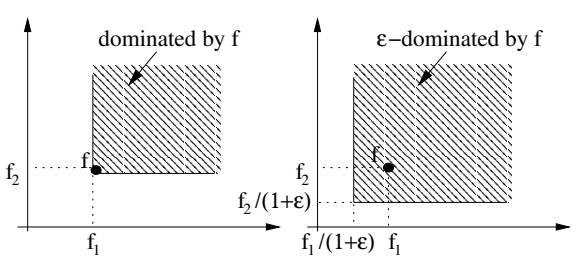


Figure 1: Domination and $\epsilon$-domination

**Definition 5: $\epsilon$-approximate Pareto Front** Let $F \subseteq \Re^m$ be a set of vectors and $\epsilon > 0$. The $\epsilon$-approximate Pareto Front $F_\epsilon \subseteq F$ contains all vectors $\vec{x}_1 \in F$, which are not $\epsilon$-dominated by any vector $\vec{x}_2 \in F$:

$$\forall \vec{x}_2 \in F : \exists\, \vec{x}_1 \text{ such that } \vec{x}_1 \prec_\epsilon \vec{x}_2 \tag{3}$$

We have to note that the set $F_\epsilon$ is not unique, but contains just certain amount of vectors, depending on the $\epsilon$ value. This has been studied in [13, 9]. For any finite $\epsilon$ and any set $F$ with objective vectors $1 \leq f_i \leq K$, $\forall i \in \{1, \ldots, m\}$, there exists a set $F_\epsilon$ containing:

$$|F_\epsilon| = O[(\frac{logK}{log(1+\epsilon)})^{m-1}] \tag{4}$$

Here, we consider that $\epsilon$ is the same for all objectives.

## 3 MOPSO Methods

Figure 2 shows the algorithm of a Multi-objective Optimization method, which we use here for Multi Objective Particle Swarm Optimization technique. MOPSO methods are studied in [1, 6, 11, 14]. In this algorithm $t$ denotes the generation index, $P_t$ the population, and $A_t$ the *archive* at generation $t$. In Step 2 the population $P_t$ is initialized, which contains the initial particles, their positions $\vec{x}_t^{\,i}$ and their initial velocities $\vec{v}_t^{\,i}$. The external archive $A_t$ is also initialized in this step, which is empty. The function $Evaluate$ in Step 3, evaluates the particles in the population $P_t$ and the function $Update(P_t, A_t)$ updates the archive and stores the non-dominated solutions among $P_t$ and $A_t$ in the archive.

BEGIN
Step 1: $t = 0$;
Step 2: Initialize population $P_t$ and archive $A_t$
Step 3: *Evaluate* $P_t$
Step 4: $A_{t+1} := Update(P_t, A_t)$
Step 5: $P_{t+1} := Generate(P_t, A_t)$
Step 6: $t = t + 1$
Step 7: Unless a $termination\ criterion$ is met, goto Step 3
END

Figure 2: Typical structure of an archive-based MOPSO.

Step 5 is the most critical Step in MOPSO techniques. In this step the velocity and position of each particle $i$ is updated as below:

$$\begin{aligned} v^i_{j,t+1} &= w v^i_{j,t} + c_1 R_1 (p^i_{j,t} - x^i_{j,t}) + c_2 R_2 (p^{i,g}_{j,t} - x^i_{j,t}) \\ x^i_{j,t+1} &= x^i_{j,t} + v^i_{j,t+1} \end{aligned} \tag{5}$$

where $j = 1, \ldots, n$, $w$ is the inertia weight of the particle, $c_1$ and $c_2$ are two positive constants, and $R_1$ and $R_2$ are random values in the range $[0, 1]$.

According to Equation 5, each particle has to change its position $\vec{x}^i_t$ towards the position of a local guide $\vec{p}_t^{\,i,g}$ which must be selected from the updated set of non-dominated solutions stored in the archive $A_{t+1}$. How to select the local guide from the archive has a great impact on convergence and diversity of the solutions and is studied in [1, 6, 7, 11]. In this equation, $\vec{p}^{\,i}$ is like a memory for the particle $i$ and keeps the non-dominated (best) position of the particle by comparing the new position $\vec{x}^i_{t+1}$ in the objective space with $\vec{p}^{\,i}_t$ ($\vec{p}^{\,i}_t$ is the last non-dominated (best) position of the particle $i$).

At the end of this step, a turbulence factor is added to the positions of the particles. This is done by adding a random value to the current position of each particle:

$$x^i_{j,t+1} = x^i_{j,t+1} + R_T\, x^i_{j,t+1} \tag{6}$$

Where $R_T \in [-1, 1]$ is a random value added to the updated position of each particle with a probability.

The steps of the MOPSO algorithm are iteratively repeated until a termination criterion is met such as a maximum number of generations or when there has been no change in the

set of non-dominated solutions found for a given number of generations. The output of the MOPSO method is the set of non-dominated solutions stored in the final archive.

# 4 Archiving

As it is explained in Section 3, an external archive is used to keep non-dominated solutions found in each generation. The archive members must be updated in each generation by the function *Update* (Step 4, Figure 2). The *Update* function compares whether members of the current population $P_t$ are non-dominated with respect to the members of the actual archive $A_t$ and how and which of such candidates should be considered for insertion into the archive and which should be removed. Thereby, an archive is called *domination-free* if no two points in the archive do dominate each other. Obviously, during execution of the function $Update$, dominated solutions must be deleted in order to keep the archive domination-free. Several data structures are proposed for storing the non-dominated solutions in the archive [6, 12] in order to reduce the computational time of the method. In this section, we are trying to focus on two properties of the archive and their influences on the results of the method: The size of archive and the initial archive members.

## 4.1 Archive Size and $\epsilon$-dominance

In most of multi-objective optimization (MO) methods the archive must contain a certain amount of solutions, while keeping a good diversity of solutions. In some of MO methods (e.g., [17]) the archive must have a fixed size and in the case that the number of non-dominated solutions is less than the fixed size, some particles of the population are selected at random to be inserted in the archive. In the case that the size of the set of non-dominated solutions becomes higher than the fixed size, truncation or clustering techniques are applied. In [16, 11], the archive size has an upper bound and as soon as the number of non-dominated solutions becomes higher than the archive size, truncation techniques are used. However, we have to note that by increasing the size of the archive the computational time increases. In [10], the computational time of the different test functions with different number of objectives and archive sizes, for different population sizes are discussed. By increasing the number of objectives and population size and archive size, the computational time of the method increases extremely.

Here, we propose to use the concept of $\epsilon$-domination instead of domination when updating the archive i.e., instead of comparing the particles using domination criterion, we compare them using the $\epsilon$-dominance criterion. Therefore, the size of the archive will have an upper bound of $O[(\frac{logK}{log(1+\epsilon)})^{m-1}]$, where $K$ is the upper bound of the objective values. It is obvious that the size of the archive depends on the $\epsilon$ value. Hence by using this $\epsilon$-dominance we can keep the size of the archive limited and we can reduce the computational time. Applying the $\epsilon$-dominance in MOPSO techniques also has influence on the convergence and diversity of the results that will be discussed later.

## 4.2 Initial Archive

In the MOPSO methods the initial archive is *empty*. So in the first generation the non-dominated solutions of the initial population are stored in the archive and the particles of the population should select their best local guide among these archive members. Selecting the first local guides from the archive has a great impact on the diversity of solutions in the next generations especially in methods explained in [6, 11]. Hence the diversity of solutions depends on the first non-dominated solutions. But if the initial archive is not empty and contains some well-distributed non-dominated solutions, the solutions converge faster than before, while keeping a good diversity. Figure 3 left shows the initial population and the non-dominated particles among them which are stored in the empty archive. In this figure, particles se-

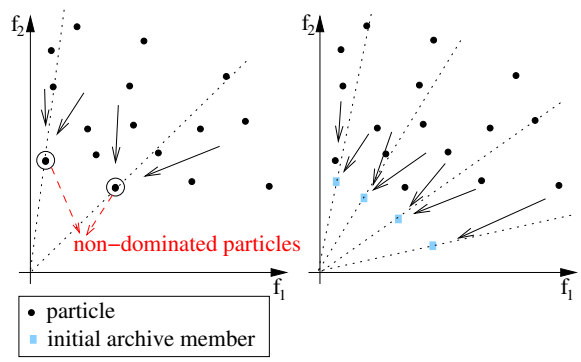


Figure 3: Influence of the initial archive

lect one of these archive members as the local guide by using Sigma method [11] and one can imagine that after one generation particles will move towards the left part of the space. In Figure 3 right, the initial archive is not empty, but it has some members which dominate all the particles in the population. This time in the next generation the particles will obtain a better diversity than in the left figure.

Now, the question is how to find a good initial archive. The initial archive can be found in different ways. The first possibility is to run the MOPSO with an empty archive for a large population and a few generations. The large population gives us a good diversity and a few generations (e.g., 5 generations) develops the population just to a little convergence. Another possibility is to use the results of a *short* MOEA (Multi-Objective Evolutionary Algorithm) method. Here, *short* means a MOEA with a few individuals and a few generations (e.g., 10 individuals and 10 generations). We know that MOEA can give us some good solutions with a very good diversity after a few generations. *Short* MOEAs has also been used in combination with other methods like subdivision methods [15].

# 5 $\epsilon$-dominance and MOPSO

In this section we apply the $\epsilon$-dominance in the MOPSO method and then compare it with MOPSO using the clustering technique. Both of these methods use the Sigma

method [11] for finding the best local guides. The clustering technique is explained in [16] and is also used in MOPSO in [11]. Here, we also use an initial archive for each test function. The initial archives are the results of a *short* MOPSO using the Sigma method. The *short* MOPSOs have a bigger population size than the usual MOPSO and are run for a few generations. In this section, we study the influence of $\epsilon$-dominance on the computational time, convergence and diversity of solutions and compare the convergence and diversity of solutions. For comparing the diversity of solutions, we also suggest a new diversity metric.

### 5.1 Diversity Metric

We can consider the position of each solution in 2- and 3-objective spaces by polar coordinates ($r$ and $\theta$) and spherical ($r$, $\theta$ and $\phi$) coordinates respectively. Inspired from these coordinates, we can formulate the diversity of solutions by a well distribution in terms of their angles $\theta$ for 2-objective spaces and $\theta$ and $\phi$ for 3-objective spaces. However, for higher dimensional objective spaces, we can not define a coordinate axes which gives us a simple distribution like in polar or spherical coordinates. Therefore, we suggest to use the concept of the Sigma method, which we have introduced in [11] for calculating the local guides in MOPSO. Here, we explain briefly the Sigma method and how one can use it to calculate the diversity of solutions.

**Sigma Method [11]** In this method, a value $\sigma_i$ is assigned to each solution with coordinates $(f_{1,i}, f_{2,i})$ so that all the solutions which are on the line $f_2 = af_1$ have the same value of $\sigma$. Therefore, $\sigma$ is defined as follows:

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2} \tag{7}$$

Figure 4 shows the values of $\sigma$ for different lines. Indeed $\sigma$ states the angle between the line $f_2 = \frac{f_{2,i}}{f_{1,i}} f_1$ and the axis $f_1$.

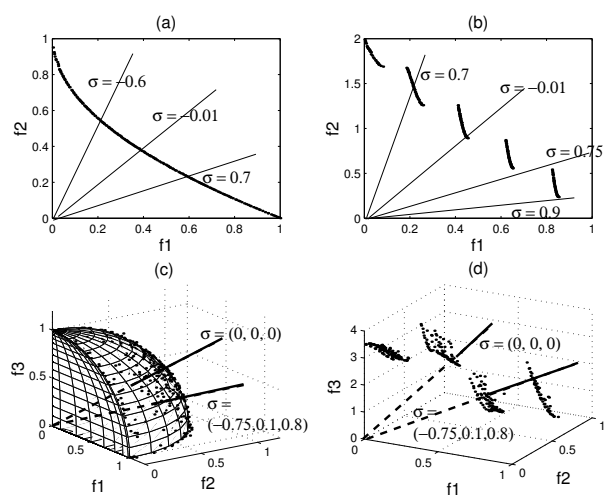


Figure 4: Sigma method for 2- and 3-objective spaces.

In the general case, $\sigma$ is a vector of $\binom{m}{2}$ elements, where $m$ is the dimension of the objective space. In this case, each element of $\vec{\sigma}$ is the combination of two coordinates in terms of the Equation (7). For example for three coordinates of $f_1$, $f_2$ and $f_3$, it is defined as follows:

$$\vec{\sigma} = \begin{pmatrix} f_1^2 - f_2^2 \\ f_2^2 - f_3^2 \\ f_3^2 - f_1^2 \end{pmatrix} / (f_1^2 + f_2^2 + f_3^2) \tag{8}$$

Different values of $\vec{\sigma}$ for different values of $f_1$, $f_2$ and $f_3$ are shown in Figure 4. In the general case, when a point has the same position in each dimension (e.g., $f_1 = f_2 = f_3$ in 3 dimensional space), $\vec{\sigma} = \vec{0}$.
We have to note that the objective functions must contain positive values, otherwise we have to transform them into the positive regions and when the objectives are not in the same ranges scaled sigma values [11] are used.

**Sigma Diversity Metric** Figure 5 shows the idea of using the Sigma method as a diversity metric for 2-objective spaces. As it is shown, $k+1$ lines with different sigma values are drawn from the origin. These lines are called *reference lines* and have the angle $\frac{i}{k}(\frac{\pi}{2})$ to the $f_1$ axis, where $i = 0, 1, \ldots, k$. We consider $k+1$ reference lines for computing the diversity of an archive with the size of $|A|$ ($|A| = k+1$). In the next step, the sigma value of each

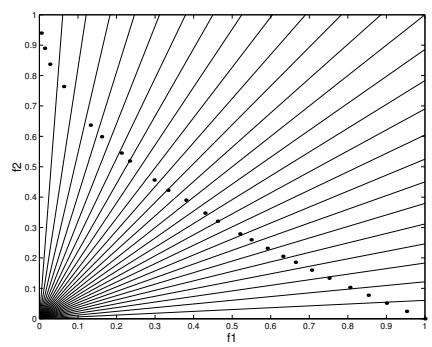


Figure 5: 2-objective Sigma Diversity Metric. Black points are the solutions of a 2-objective test function and the lines are reference lines.

reference line should be computed, which we call *reference sigma value*. In the case that all of the $|A|$ solutions are well distributed, they should be spread in many different regions.

For higher dimensional spaces, the reference lines are also defined by lines passing the origin. In order to find the angles between the lines and each coordinate axis, it is easier to find *reference points* located on the reference lines. Hence in an $m$ dimensional space, the coordinate of each reference point is a vector of $m$ elements. Algorithm 1 calculates the coordinates of the reference points. For example we obtain $(1, 0)$, $(1, tan(\frac{\pi}{6}))$, $(1, tan(\frac{\pi}{3}))$, for $m = 2$ and $k = 3$. Indeed in this Algorithm, the first coordinate of point $i$ is kept constant ($f_{1,i} = 1$) and the other coordinates are changing. However for obtaining the whole reference points, the algorithm must be repeated $m$ times, each time

one of the coordinates must be kept constant. In our example in the second ($m$th) run we keep the second coordinate constant and obtain $(0, 1)$, $(tan(\frac{\pi}{6}), 1)$, $(tan(\frac{\pi}{3}), 1)$. The

**Algorithm 1** Calculate coordinates of reference points

**Require:** $m,\ k$
**Ensure:** $(f_{1,i}, f_{2,i}, \ldots, f_{m,i}),\ \forall i = 1, 2, \ldots, (I-1)$
  $i = 1$
  **for** $j_1 = 0$ to $k-1$ **do**
    **for** $j_2 = 0$ to $k-1$ **do**
      ...
      **for** $j_{m-1} = 0$ to $k-1$ **do**
        $f_{1,i} = 1$
        $f_{2,i} = f_{1,i}\ tan(\pi\ j_1/2k)$
        $f_{3,i} = f_{1,i}\ tan(\pi\ j_2/2k)$
        ...
        $f_{m,i} = f_{1,i}\ tan(\pi\ j_{m-1}/2k)$
        $i = i + 1$
      **end for**
    **end for**
  **end for**
  I = i

number of reference points produced by Algorithm 1 depends on $k$ and $m$. In 2-objective spaces, the number of reference lines is equal to $k + 1$. In higher dimensional spaces $k$ is the number of regions which are separated by reference lines on the plane generated by only two of coordinate axes. For example in Figure 6, on the plane generated by $f_1$ and $f_3$ axes, there are four regions separated by reference lines. In higher dimensional spaces, the number of reference points made by Algorithm 1 is more than the required number of reference lines. Because by repeating the algorithm some points lie on the same reference line. In the previous example, the points $(1, tan(\frac{\pi}{6}))$ and $(tan(\frac{\pi}{3}), 1)$, and $(1, tan(\frac{\pi}{3}))$ and $(tan(\frac{\pi}{6}), 1)$ are on the same lines. Therefore, the number of reference lines can be calculated after finding the sigma value (vector) of each reference point. The points located on a reference line will have the same sigma vectors and the number of reference lines is the number of non-repeated reference sigma vectors. Table 1 shows

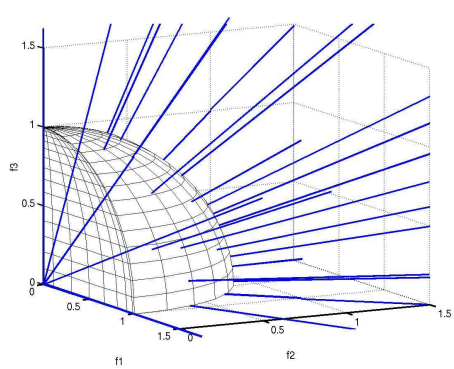


Figure 6: Reference lines in 3-objective space ($k = 4$).

the number of reference lines for different values of $k$ in 3-objective spaces. A binary *Flag* is also kept beside each reference sigma vector, which is 0 at the beginning. the Flag of each reference sigma vector can only turn to 1, when at least one solution has a sigma vector equal or with a distance [1] less than $d$ to it. A counter $C$ counts the reference lines with Flags equal to 1 and the diversity metric $D$ becomes:

Table 1: Number of reference lines obtained from different $k$ values for 3-objective spaces

| $k$ | number of ref. | $d$ |
|---|---|---|
| 4 | 25 | 0.15 |
| 6 | 67 | 0.1 |
| 8 | 133 | 0.1 |
| 10 | 223 | 0.1 |
| 12 | 337 | 0.1 |
| 14 | 475 | 0.05 |
| 16 | 637 | 0.05 |
| 18 | 823 | 0.05 |
| 20 | 1033 | 0.05 |

$$D = \frac{C}{\text{number of reference lines}} \tag{9}$$

The value of $d$ depends on the test function, however it should decrease when increasing the number of reference lines. Table 1 shows an example on choosing $d$ for 3-objective test functions.

The Sigma diversity metric is easy to implement and is very efficient in computing the diversity of solutions in high dimensional spaces. The 2-objective Sigma diversity metric seems to have some similarities to Entropy approach [5], Sparsity measurement [3] and [8], especially when measuring the diversity of very convex (or non-convex) objective fronts. But in comparison to them, it is very easy to calculate the diversity of solutions in high dimensional spaces using the sigma diversity metric. The Sigma diversity metric like the Sigma method can also be scaled for different ranges of the objective values. However, the objective values must contain just the positive values, and the negative values must be transfered to the positive part (i.e., upper right quadrant of a circle in two dimensions). This is possible without loss of generality

### 5.2 Test Functions

The test functions are 2- and 3-objective optimization problems selected from [16], [4] and are shown in Table 2.

In Table 2, $x_i \in [0, 1]$. For test functions $t_1$ and $t_2$, $n = 30$ and for test functions $t_3$ and $t_4$, $n = 3$.

#### 5.2.1 Parameter Settings

The tests are done for 120 particles in the population and 300 generations with the turbulence factor of 0.01 and inertia weight of 0.4. Initial archives are obtained by running a MOPSO with population size 100 and 200 generations for test functions $t_1$ and $t_2$ and population size 500 and 10 generations for test functions $t_3$ and $t_4$.

[1] Euclidian distance

Table 2: Test Functions: $t_i$

| $t_i$ | Function |
|---|---|
| 1 | $g(x_2,\ldots,x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$<br>$h(f_1,g) = 1 - \sqrt{f_1/g}$<br>$f_1(x_1) = x_1$<br>$f_2(\vec{x}) = g(x_2,\ldots,x_n).h(f_1,g)$ |
| 2 | $g(x_2,\ldots,x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$<br>$h(f_1,g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$<br>$f_1(x_1) = x_1$<br>$f_2(\vec{x}) = g(x_2,\ldots,x_n).h(f_1,g) + 1$ |
| 3 | $f_1(\vec{x}) = (1+x_3)\cos(x_1\pi/2)\cos(x_2\pi/2)$<br>$f_2(\vec{x}) = (1+x_3)\cos(x_1\pi/2)\sin(x_2\pi/2)$<br>$f_3(\vec{x}) = (1+x_3)\sin(x_1\pi/2)$ |
| 4 | $f_1(\vec{x}) = x_1$<br>$f_2(\vec{x}) = x_2$<br>$f_3(\vec{x}) = 3.5 - \sum i = 1^n 2x_i sin(n\pi x_i)$ |

### 5.2.2 Results

The MOPSO is run for different $\epsilon$ values using the $\epsilon$-dominance technique. Then it is run again by using the clustering technique. In the clustering technique, the maximum archive size is set to the archive size obtained by the $\epsilon$-dominance.

Tables 3 and 4 show the results of 2- and 3-objective test functions. In these tables, $A$ is the MOPSO using the $\epsilon$-dominance and $B$ is the MOPSO using the clustering technique, size is the archive size, $t_A$ and $t_B$ are the CPU times needed to run each MOPSO on a 500 MHz Ultra-SPARC-IIe SUN Workstation, $N_{AB}$ refers to the number of solutions in $B$ that are weakly dominated by $A$ and $D$ is the Sigma diversity metric values (in percent). All the values recorded in Tables 3 and 4 are average values from five different runs with different initial populations.

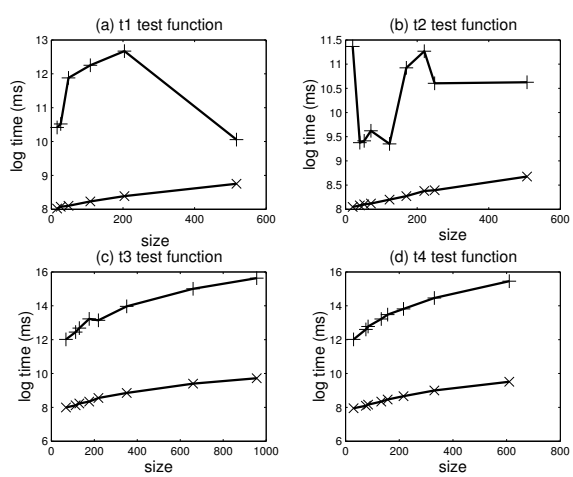


Figure 7: Influence of $\epsilon$ on CPU time (+ : Clustering, × : $\epsilon$-dominance).

Table 3: Results of 2-objective test functions (time in milliseconds). $A$: MOPSO using $\epsilon$-dominance and $B$: MOPSO using the clustering technique. ($*$ : the clustering can just find 616 solutions)

| test function $t1$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | size | $t_A$ | $t_B$ | $N_{AB}$ | $N_{BA}$ | $D_A$ | $D_B$ |
| 0.1 | 16 | 3041 | 33429 | 4 | 0 | 87 | 87 |
| 0.05 | 26 | 3312 | 37006 | 4 | 0 | 80 | 80 |
| 0.025 | 48 | 3305 | 144151 | 1 | 1 | 93 | 91 |
| 0.01 | 109 | 3754 | 204412 | 13 | 12 | 77 | 74 |
| 0.005 | 204 | 4401 | 318432 | 80 | 13 | 93 | 87 |
| 0.001 | 517 | 6331 | 23294 | 398 | 13 | 93 | 93 |
| 0.0001 | 941 | 8083 | 6096* | 828 | 7 | 97 | 93 |
| 0 | 616 | 6096 | - | - | - | - | - |
| test function $t2$ | | | | | | | |
| 0.025 | 20 | 3127 | 86329 | 0 | 9 | 50 | 45 |
| 0.01 | 40 | 3229 | 11807 | 3 | 5 | 30 | 28 |
| 0.0075 | 52 | 3300 | 12270 | 0 | 11 | 27 | 30 |
| 0.005 | 71 | 3358 | 15102 | 6 | 14 | 34 | 48 |
| 0.0025 | 123 | 3635 | 11509 | 80 | 2 | 43 | 53 |
| 0.001 | 170 | 3909 | 55498 | 33 | 54 | 51 | 60 |
| 0.0005 | 220 | 4353 | 78289 | 32 | 65 | 59 | 58 |
| 0.00025 | 249 | 4416 | 40306 | 179 | 7 | 61 | 61 |
| 0.0001 | 507 | 5868 | 41203 | 460 | 2 | 55 | 51 |
| 0 | 730 | 7560 | - | - | - | - | - |

Table 4: Results of 3-objective test functions (time in milliseconds). $A$: MOPSO using $\epsilon$-dominance and $B$: MOPSO using the clustering technique.

| test function $t3$ | | | | | | | |
|---|---|---|---|---|---|---|---|
| $\epsilon$ | size | $t_A$ | $t_B$ | $N_{AB}$ | $N_{BA}$ | $D_A$ | $D_B$ |
| 0.1 | 68 | 2968 | 164954 | 16 | 0 | 77 | 91 |
| 0.07 | 113 | 3368 | 254867 | 17 | 0 | 83 | 83 |
| 0.06 | 130 | 3740 | 321244 | 12 | 0 | 86 | 93 |
| 0.05 | 176 | 4188 | 555989 | 19 | 2 | 91 | 93 |
| 0.04 | 219 | 5191 | 510482 | 25 | 1 | 98 | 98 |
| 0.03 | 351 | 6979 | 1161747 | 34 | 8 | 70 | 83 |
| 0.02 | 660 | 12126 | 3277912 | 73 | 7 | 92 | 93 |
| 0.015 | 956 | 16694 | 6154627 | 135 | 11 | 98 | 97 |
| 0 | 10692 | 172782 | - | - | - | - | - |
| test function $t4$ | | | | | | | |
| 0.1 | 30 | 2839 | 165275 | 3 | 0 | 32 | 28 |
| 0.05 | 76 | 3275 | 298039 | 1 | 0 | 23 | 29 |
| 0.04 | 84 | 3505 | 353349 | 1 | 1 | 23 | 28 |
| 0.03 | 133 | 4194 | 552847 | 1 | 0 | 22 | 25 |
| 0.025 | 157 | 4735 | 715612 | 1 | 1 | 21 | 24 |
| 0.02 | 216 | 5788 | 1001681 | 7 | 1 | 24 | 27 |
| 0.015 | 331 | 8103 | 1907411 | 2 | 0 | 18 | 19 |
| 0.01 | 610 | 13574 | 5152398 | 10 | 1 | 19 | 21 |
| 0 | 21351 | 373641 | - | - | - | - | - |

**Influence on Computational time** Figure 7 shows the CPU times of the two methods in Tables 3 and 4 graphically, where size is the archive size and the CPU time is shown in logarithmic milliseconds values. For all test functions, the CPU time increases when increasing the archive

size. We have to note that when the limit of the archive size is bigger than the number of non-dominated solutions, clustering is not applied to the archive. This can be observed especially for both of the 2-objective test functions, the CPU time of the clustering technique decreases for large archive sizes. In both of 2- and 3-objective test functions the CPU time of the program when using the $\epsilon$-dominance is much less than when using the clustering techniques. The clustering technique takes in some cases more than 100 times the $\epsilon$-dominance to find the same number of solutions.

In Table 3, the $\epsilon$-dominance method finds 941 solutions for the test function $t_1$, when $\epsilon = 0.0001$. But if we apply the method using the clustering technique, we see that we never reach the number of 941 as the archive size in order to apply clustering on it, therefore it will take less time than the $\epsilon$-dominance method.

**Influence on convergence** In Tables 3 and 4, the factor $N_{AB}$ shows the number of solutions in set $B$ that are weakly dominated by the solutions in set $A$, i.e.:

$$N_{AB} \quad := \quad |\{b \in B | \exists a \in A : a \preceq b\}| \qquad (10)$$

By comparing $N_{AB}$ and $N_{BA}$, where A is the MOPSO using the $\epsilon$-dominance and B MOPSO using clustering technique, we can conclude that for the same archive sizes the $\epsilon$-dominance dominates more solutions of the results of clustering technique. This also depends on the archive size and the number of objectives. For test functions t1, t2 and t3, the values of $N_{AB}$ are much higher than $N_{BA}$, which we can conclude a better convergence. However, for the 3-objective test function t4, they are comparable.

**Influence on diversity** As it is explained, we have introduced a new diversity metric, which are demonstrated as the $D$ values in Tables 3 and 4. The $D$ value is shown in percent. Here, we study the results of 2- and 3-objective test functions separately:

**- 2-objective test functions:** In both of the 2-objective test functions ($t_1$ and $t_2$), we have used the same number of reference lines (Sigma values) as the archive size i.e., $k + 1 = |A|$. Therefore, it is clear for the test function $t_2$, which has discontinuities, the value of $D$ will never reach 100%. The values of $d$ used in measuring the diversities are as follows: for archive sizes less than 20, 0.1, between 20 and 50, 0.05, between 50 and 500, 0.01, and more than 500, 0.005. Comparing the diversity of the $\epsilon$-dominance method with the clustering method by using $D$ values, we conclude that for bigger archive sizes the $\epsilon$-dominance has bigger $D$ values, which means a better diversity. In some cases the clustering method obtains higher $D$ values than using $\epsilon$-dominance. However, the diversity of solutions is comparable.

**- 3-objective test functions:** In both of the 3-objective test functions, we can not achieve the best diversity of solutions. However, the clustering method gives us better diversity of solutions than the $\epsilon$-dominance method. One of the reasons may be the shape of the approximate Pareto-front. In Table 4, the number of reference lines is determined by the value of $k$ in Table 1. In our experiments, we have used

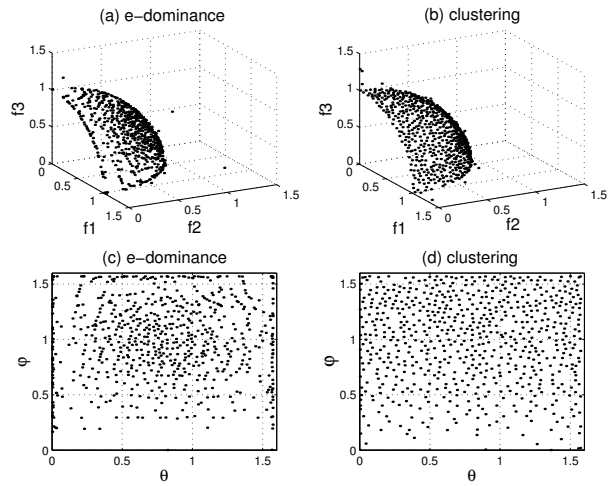


Figure 8: $t_3$ test function ($\epsilon = 0.02$). (a),(b) objective space and (c),(d) $\theta - \phi$ axis of the spherical coordinate.

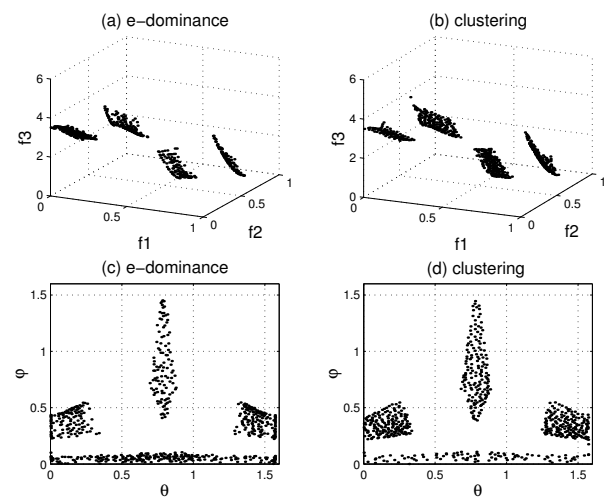


Figure 9: $t_4$ test function ($\epsilon = 0.01$). (a),(b) objective space and (c),(d) $\theta - \phi$ axis of the spherical coordinate.

the number of reference lines very close to the archive sizes. Figures 8 and 9 show the results of the $t_3$ and $t_4$ test functions and also the $\theta$-$\phi$ axis of the solutions (spherical coordinates), for having a better observation on the diversity of solutions. We can observe that the $\epsilon$-dominance method can not obtain some solutions, therefore the diversity of solutions of the $\epsilon$-dominance, especially for test function $t_3$, is not as good as the clustering method.

## 6 Conclusion and Future Work

In this paper, the influence of the $\epsilon$-dominance in comparison to the clustering techniques is studied. The $\epsilon$-dominance bounds the number of solutions in the archive and decreases the computational time. The computational time in some cases is much less than the method using the clustering technique. Using $\epsilon$-dominance has also influence on convergence and diversity of solutions. The obtained solutions have comparable convergence and diversity when compared to clustering technique and in some cases are better in con-

vergence and diversity, especially for 2-objective test functions.

The diversity of the solutions is compared with a new diversity metric called Sigma metric. According to this metric, the diversity of the solutions obtained by $\epsilon$-dominance is getting worse than the clustering technique for an increasing number of objectives. However, we have to consider that the results are just for the recorded number of generations and if we run the methods for a large number of generations we obtain a very good diversity and convergence of solutions.

The introduced diversity metric *Sigma diversity metric* is easy to implement and efficient for high dimensional spaces which makes it worthy in comparison to other diversity metrics. Dealing with continuous and positive objective values it will give us a very good measurement of diversity of solutions. In the case of negative objective values or when the objective functions have different ranges, scaled Sigma method should be used.

In this paper, we have also suggested to use an initial archive instead of an empty archive. This has influence on the diversity of the solutions.

In the future we would like to investigate and compare the $\epsilon$-dominance method for different number of generations for different test functions with higher number of objectives.

## Bibliography


[1] C. A. Coello Coello and M. S. Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *IEEE Proceedings World Congress on Computational Intelligence*, pages 1051–1056, 2003.

[2] K. Deb. *Multi-Objective Optimization using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[3] K. Deb, M. Mohan, and SV. Mishra. A fast multi-objective evolutionary algorithm for finding well-spread pareto-optimal solutions. In *KanGAL Report No. 2003002,Indian Institute Of Technology Kanpur*, 2002.

[4] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *IEEE Proceedings World Congress on Computational Intelligence*, 2002.

[5] A. Farhang-Mehr and S. Azarm. Diversity assessment of paretooptimal sets: An entropy approach. In *IEEE Proceedings World Congress on Computational Intelligence*, 2002.

[6] J. E. Fieldsend and S. Singh. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. In *The 2002 U.K. Workshop on Computational Intelligence*, pages 34–44, 2002.

[7] X. Hu, R. Eberhart, and Y. Shi. Particle swarm with extended memory for multiobjective optimization. In *IEEE Swarm Intelligence Symposium*, pages 193–198, 2003.

[8] V. Khare, X. Yao, and K. Deb. Performance scaling of multi-objective evolutionary algorithms. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 376–390, 2003.

[9] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler. Archiving with guaranteed convergence and diversity in multi-objective optimization. In *Genetic and Evolutionary Computation Conference (GECCO02)*, pages 439–447, 2002.

[10] S. Mostaghim and J. Teich. Quad-trees: A data structure for storing pareto-sets in multi-objective evolutionary algorithms with elitism. In *Evolutionary Computation Based Multi-Criteria Optimization: Theoretical Advances and Applications*, to appear, 2003.

[11] S. Mostaghim and J. Teich. Strategies for finding good local guides in multi-objective particle swarm optimization. In *IEEE Swarm Intelligence Symposium*, pages 26–33, 2003.

[12] S. Mostaghim, J. Teich, and A. Tyagi. Comparison of data structures for storing pareto-sets in MOEAs. In *IEEE Proceedings World Congress on Computational Intelligence*, pages 843–849, 2002.

[13] C. H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources (extended abstract). In *IEEE Symposium on Foundations of Computer Science*, 2000.

[14] K. E. Parsopoulos and M. N. Vrahatis. Recent approaches to global optimization problems through particle swarm optimization. In *Natural Computing, 1 (2-3)*, pages 235–306, Kluwer Academic Publishers, 2002.

[15] O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. Covering pareto sets by multilevel evolutionary subdivision techniques. In *Proceedings of Second International conference on Evolutionary Multi-Criterion Optimization*, pages 118–132, 2003.

[16] E. Zitzler. *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*. TIK-Schriftenreihe Nr. 30, Diss ETH No. 13398, Shaker Verlag, Germany, Swiss Federal Institute of Technology (ETH) Zurich, 1999.

[17] E. Zitzler, M. Laumanns, and L. Thiele. Spea2: Improving the strength pareto evolutionary algorithm. In *EUROGEN 2001, Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, 2001.