Strategies for Finding Good Local Guides in Multi-objective Particle Swarm Optimization (MOPSO)

Sanaz Mostaghim[1], Jürgen Teich[2]
[1] Electrical Engineering Department
Paderborn University, Paderborn, Germany
mostaghim@date.upb.de
[2] Computer Science Department
Friedrich-Alexander-University, Erlangen-Nuremberg, Germany
teich@informatik.uni-erlangen.de

Abstract - In multi-objective particle swarm optimization (MOPSO) methods, selecting the best *local guide* (the global best particle) for each particle of the population from a set of Pareto-optimal solutions has a great impact on the convergence and diversity of solutions, especially when optimizing problems with high number of objectives. This paper introduces the Sigma method as a new method for finding best local guides for each particle of the population. The Sigma method is implemented and is compared with another method, which uses the strategy of an existing MOPSO method for finding the local guides. These methods are examined for different test functions and the results are compared with the results of a multi-objective evolutionary algorithm (MOEA).

I. INTRODUCTION

During the past decades, stochastic iterative search methods have been investigated for solving optimization problems. Through cooperation and competition among the potential solutions, these techniques often can find optima very quickly when applied to complex optimization problems. The most commonly used population-based evolutionary computation techniques are motivated from the evolution of nature (i.e., evolutionary algorithms). Different from evolutionary computation techniques, a new evolutionary computation technique called Particle Swarm Optimization (PSO), is motivated from the simulation of social behavior. PSO was originally designed and developed by Eberhart and Kennedy [5]. A PSO consists of a population of particles, which on the contrary to evolutionary algorithms, survive up to the last generation. The particles search the variable space by moving with a special speed towards the best global particle (guide) by using their experience from the past generations.

Recently, investigators are paying more and more interest on PSO to solve multi-objective problems [1], [3], [4], [7]. Changing a PSO to a multi-objective PSO (MOPSO) requires a redefinition of what a guide is in order to obtain a front of optimal solutions. In MOPSO, the Pareto-optimal solutions should be used to determine the guide for each particle. But selecting the guide (the best local guide) from the set of Pareto-optimal solutions for each particle of the population [3] is very difficult yet an important problem for attaining convergence and diversity of solutions. In this paper, we propose a new method, called *Sigma* method for selecting the best local guide for each particle. The implementation results show that by using the Sigma method in a MOPSO, we can achieve a very good convergence and diversity of solutions. Also, we compare this method with an existing MOPSO which has the same structure as our implementation, but with using the idea of [3] for finding the best local guides. In this paper, another comparative study of a multi-objective evolutionary algorithm called SPEA2 [10] with the new Sigma method is also given. This paper has the following structure: We describe the structure of a PSO and a MOPSO in Section II. In Section III we investigate other MOPSO methods and their techniques on finding the best local guide for each particle, then we explain our suggestion in the remainder of the Section. In Section IV, the experiments and comparisons are described and a discussion concludes the paper in Section V.

A. *Multi-objective optimization*

A multi-objective optimization problem is of the form

$$minimize \quad \{f_1(\vec{x}), f_2(\vec{x}), \cdots, f_m(\vec{x})\} \qquad (1)$$

subject to $\vec{x} \in S$, involving $m(\geq 2)$ conflicting objective functions $f_i : \Re^n \rightarrow \Re$ that we want to minimize simultaneously. The *decision vectors* $\vec{x} = (x_1, x_2, \cdots, x_n)^T$ 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 $\vec{f}(\vec{x}) = (f_1(\vec{x}), f_2(\vec{x}), \cdots, f_m(\vec{x}))$.

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, or $f_i(\vec{x}_1) \leq f_i(\vec{x}_2)$ $\forall i = 1, \cdots, m$.
- 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 $i = 1, \cdots, m$.

and $\vec{x}_1$ *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, or $f_i(\vec{x}_1) \leq f_i(\vec{x}_2)$ $\forall i = 1, \cdots, m$.

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. Finally, an objective vector is called Pareto-optimal if the corresponding decision vector is Pareto-optimal.

II. PARTICLE SWARM OPTIMIZATION

A particle swarm optimization (PSO) method is a population based optimization technique and can be formulated as follows: A set of N particles may be considered as a population P_t in the generation t. Each particle i has a position defined by $\vec{x}^i = \{x_1^i, x_2^i, \cdots, x_n^i\}$ and a velocity defined by $\vec{v}^i = \{v_1^i, v_2^i, \cdots, v_n^i\}$ in the variable space S. In generation $t+1$, the velocity and position of each particle i is updated as below:

$$\begin{aligned} v_{j,t+1}^i &= w v_{j,t}^i + c_1 R_1 (p_{j,t}^i - x_{j,t}^i) + c_2 R_2 (p_{j,t}^{i,g} - x_{j,t}^i) \\ x_{j,t+1}^i &= x_{j,t}^i + v_{j,t}^i \end{aligned} \quad (2)$$

where $j = 1, \cdots, n$, 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 $[0, 1]$.
In Equation 2, $\vec{p}_t^{\,i,g}$ is the position of the global best particle in the population which guides the particles to move towards the optimum. $\vec{p}_t^i$ is the best position that particle i could find so far. Indeed, it is like a memory for the particle i and is updated in each generation. In a PSO, the performance of each particle is measured according to a pre-defined fitness function, which is related to the problem to be solved. The inertia weight w is employed to control the impact of the previous history of velocities on the current velocity, thus to influence the trade-off between global and local exploration abilities of the particles [8].

A. *Multi-objective particle swarm optimization*

The important part in multi-objective particle swarm optimization (MOPSO) is to determine the best global particle $\vec{p}_t^{\,i,g}$ for each particle i of the population. In single-objective PSO the global best particle is determined easily by selecting the particle which has the best position. Since in multi-objective optimization problems there is a set of Pareto-optimal solutions as the optimum solutions, each particle of the population should select one of the Pareto-optimals as its global best particle, which we call it the *best local guide*. Figure 1 shows a structure of a MOPSO with elitism, where t denotes the generation index, P_t the population, and A_t the *archive* at generation t. In this method, elitism is also considered, since not to loose the non-dominated solutions during generations. In Figure 1, the function *Evaluate*, evaluates the particles in the population P_t and the function $Update(P_t, A_t)$ 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 points must be deleted in order to keep the archive domination-free. Selecting the global best particle for each particle i is done in $FindGlobalBest(A_{t+1}, \vec{x}_t^{\,i})$ function. In this function each particle has to change its position $\vec{x}_t^i$ towards the position of a local guide which must be selected from the updated set of Pareto-optimal solutions stored in the archive A_{t+1}. This function will be discussed in the next section. In Step 5-c, $\vec{p}^i$ of the particle i is updated. $\vec{p}^i$ is like a memory for the particle i and keeps the non-dominated (best) position of the particle by comparing the new position $\vec{x}_{t+1}^{\,i}$ in the objective space with $\vec{p}_t^i$ ($\vec{p}_t^i$ is the last non-dominated (best) position of the particle i). The steps of an elitist MOPSO 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 an elitist MOPSO method is the set of non-dominated solutions stored in the final archive.

III. FINDING GOOD LOCAL GUIDES

As mentioned before, there exist already several MOPSO methods [4], [1], [3], [7]. In each of these methods (apart from the differences in the main algorithm), there is also a suggestion for finding the best local guides. In most of these methods, there is an inspiration of multi-objective evolutionary algorithms (MOEA). In this section, we discuss briefly some of these methods, their advantages and disadvantages, then we introduce a new method for finding the best local best guides.

Hu and Eberhart's MOPSO [4]: Hu and Eberhart present a MOPSO that uses a dynamic neighborhood strategy. In their method explained for two-objective optimization, the best local guide $\vec{p}^{\,i,g}$ for the particle i is found in the objective

```
MOPSO Algorithm
BEGIN
  Input: Optimization problem.
  Output: Non-dominated solutions in archive ($A$)

  Step 1: $t = 0$
  Step 2: Initialization:
    Initialize population $P_t$:
    For $i = 1$ to $N$,
      Initialize $\vec{x}_t^{\,i}$, $\vec{v}_t^{\,i} = \vec{0}$ and $\vec{p}_t^{\,i} = \vec{x}_t^{\,i}$
    End;
    Initialize the archive $A_t := \{\}$;
  Step 3: $Evaluate(P_t)$;
  Step 4: $A_{t+1} := Update(P_t, A_t)$;
  Step 5: $P_{t+1} := Generate(P_t, A_t)$:
    For $i = 1$ to $N$,
          a) $\vec{p}_t^{\,i,g} := FindGlobalBest(A_{t+1}, \vec{x}_t^{\,i})$;
          b) For $j = 1$ to $n$
               $v^i_{j,t+1} = wv^i_{j,t} + R_1(p^i_{j,t} - x^i_{j,t})+$
                      $R_2(p^{i,g}_{j,t} - x^i_{j,t})$;
               $x^i_{j,t+1} = x^i_{j,t} + v^i_{j,t}$;
             End;
          c) If $(\vec{x}^{\,i}_{t+1} \prec \vec{p}_t^{\,i})$
                  $\vec{p}^{\,i}_{t+1} = \vec{x}^{\,i}_{t+1}$;
            else
                  $\vec{p}^{\,i}_{t+1} = \vec{p}_t^{\,i}$;
            End;
     End;
  Step 6: Unless a termination criterion is met:
          $t = t + 1$ and goto Step 3.
END
```

Fig. 1. Multi-objective PSO algorithm.

space as follows: First the distance of particle i to other particles is calculated in terms of the first objective value which is called *fixed objective*. Then k local neighbors based on the calculated distances are found. The local optima among the neighbors in terms of the second objective value is the best local guide $\vec{p}^{\,i,g}$ for the particle i. In this method, selecting the *fixed objective* must be done by having a priori knowledge about the objective functions and one-dimensional optimization is used to deal with multiple objectives. Therefore, selecting the best local guides depends on just one of the objectives.

Coello and Lechuga's MOPSO [1]: Coello and Lechuga propose a MOPSO which has the same structure as in Figure 1. In this method the objective space is divided to hypercubes, before selecting the best local guide $\vec{p}^{\,i,g}$ for each particle i, like as if putting a grid on the objective space. In the next step, a fitness value is assigned to each hypercube depending on the number of elite particles that lie in it. The more the elite particles are in a hypercube the less is its fitness value. Then roulette-wheel selection is applied on the hypercubes and one of them is selected. At the end, $\vec{p}^{\,i,g}$ is a random particle selected from the selected hypercube. Therefore, the best local guide is selected by using the roulette-wheel selection method, which is a random selection. Indeed, it is possible that a particle doesn't select a suitable guide as its local guide.

Fieldsend and Singh's MOPSO [3]: Fieldsend and Singh present a MOPSO, which uses an unconstraint archive. In their method, a different data structure (called dominated tree) for storing the elite particles facilitates the choice of a best local guide for each particle of the population. By using this special archive, they address to find the best local guide, which considers all the objective. In their method, they store the non-dominated solutions in the archive called dominated tree. The dominated tree consists of a list of $L = \lceil |A|/m \rceil$ composite points ordered by a weak dominance relation, where $|A|$ is the archive size:

$$\tau = \{\vec{c}_L \preceq \cdots \preceq \vec{c}_2 \preceq \vec{c}_1\} \tag{3}$$

In the dominated tree, each composite point is a vector of m elements and each element is constructed as follows: The first composite point c_1 is constructed by finding the particle $y_m \in A$ with maximum first coordinates; this value forms the first coordinate of the composite point: $c_{1,1} = max\ (y_{m,1})$, for $y_m \in A$. The particle y_m, is now associated with c_1 and is not considered anymore. Likewise the second coordinate of c_1 is given by the maximum second coordinates of the remaining points in A: $c_{1,2} = max\ (y_{m,2})$, for $y_m \in A$. This procedure is repeated to construct subsequent composite points until all elements of A are associated in the tree. In general, the dth coordinate of ith composite point is given by:

$$c_{i,d} = max(y_{m,d}) \tag{4}$$

Figure 2 shows this method for a two-objective example, where $\vec{c}_1, \cdots, \vec{c}_4$ are composite points. In this method the selection of the best local guide for a particle in the population is based upon its closeness (in objective space) to a particle in the archive. The best local guide for a particle i is that archive member of the composite point c_j contributing the vertex which is less than or equal to the corresponding objective in i. This is also shown in Figure 2. In the case that a composite point has more than one vertex less or equal to particle i, one of the vertices that meets the condition is selected at random (for more details see [3]).

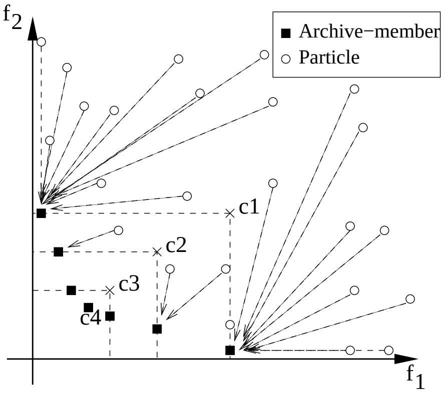

Fig. 2. Choosing the best local guide among the archive members for each particle in the population. Fieldsend and Singh‘s method [3].

The way that the local guides are selected in Fieldsend and Singh's MOPSO is said to be better than the method proposed by Coello and Lechuga [3]. However, they have both tested their algorithms for two-objective test functions. Considering higher dimensional objective spaces, Fieldsend and Singh's MOPSO tries just to guide the particles to special members of the archive, where the aim of PSO is to let the particles to fly towards the best solutions. This problem can also be seen in Figure 2 for a two-objective space. The particles which have both of their objective values more than c_1 must still select one of the archive members making the composite point c_1, where for most of them other guides coming from composite points c_2 or c_3 are more suitable.

A. *Sigma method*

In this Section we propose a new method called *Sigma method* for finding the best local guide for each particle. Before explaining the method in finding the best local guides, we discuss the basic idea of the Sigma method in its general form first. Later, we will explain how we can find the best local guide for each particle of the population in the objective space.

In our method, we assign a value σ_i to each point with coordinates $(f_{1,i}, f_{2,i})$ so that all the points which are on the line $f_2 = af_1$ have the same value of σ. So, we can define σ as follows:

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2} \tag{5}$$

According to Equation 5, all the points on the line $f_2 = af_1$ have the same σ values: $\sigma_i = (1 - a^2)/(1 + a^2)$. Figure 3 shows the values of σ for different lines. For the point with the coordinate $(f_{1,i}, f_{2,i})$, if $f_{1,i} = f_{2,i}$ then $\sigma = 0$. In the case that $f_{2,i} = 0$, $\sigma = 1$ and in the case $f_{1,i} = 0$, $\sigma = -1$. Therefore, when $a > 1$, σ is negative and when $a < 1$, σ has a positive value. Indeed σ states the angle between the line $f_2 = \frac{f_{2,i}}{f_{1,i}} f_1$ and the axis f_1.

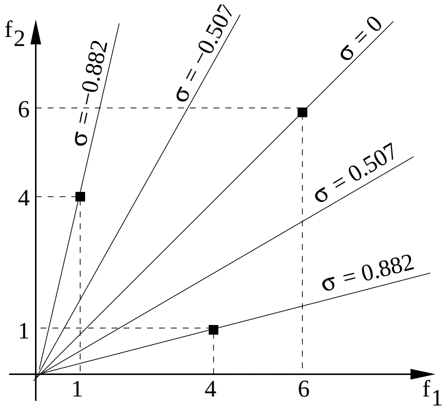

Fig. 3. Sigma method for a two-objective space.

In the general case, let σ be 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 5. For example for three coordinates of f_1, f_2 and f_3, it is defined as below:

$$\vec{\sigma} = \begin{pmatrix} f_1^2 - f_2^2 \\ f_2^2 - f_3^2 \\ f_3^2 - f_1^2 \end{pmatrix} / (f_1^2 + f_2^2 + f_3^2) \tag{6}$$

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), $\vec{\sigma} = \vec{0}$.

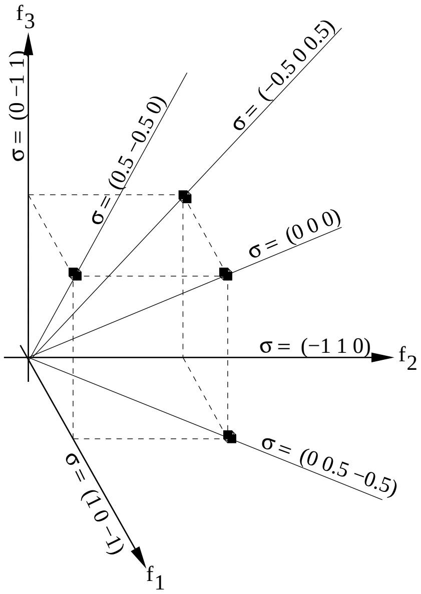

Fig. 4. Sigma method for a three-objective space.

A.1 *Finding the best local guides using the Sigma method*

Using the basic idea of Sigma method and by considering the objective space, finding the best local guide ($\vec{p}_t^{\,i,g}$) among the archive members for the particle i of population is as follows: In the first step, we assign the value σ_j to each particle j in the archive. In the second step, σ_i for particle i of the population is calculated. Then we calculate the distance between the σ_i and σ_j, $\forall j = 1, \cdots, |A|$. Finally, the particle k in the archive which its σ_k has the minimum distance to σ_i is selected as the best local guide for the particle i. Therefore, particle $\vec{p}_t^{\,i,g} = x^k$ is the best local guide for particle i. In other words, each particle that has a *closer* sigma value to the sigma value of the archive member, must select that archive member as the best local guide. In the case of two dimensional objective space, *closer* means the difference between the sigma values and in the case of m-dimensional objective space, it means the m-euclidian distance between the sigma values. Figure 5 shows how we can find the best local guide among the archive members for each particle of the population for a two-dimensional objective space.

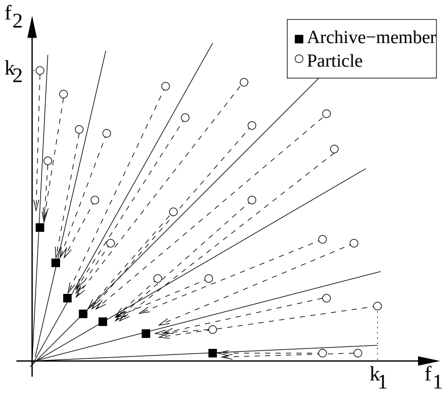

Fig. 5. Finding the best local guide for each particle of the population using the Sigma method.

The reason for selecting particle k from the archive members as the best local guide is that σ_k has the closest distance to σ_i among the archive members. When two σ values are close to each other, it means that the two particles are on two lines (e.g., $f_2 = af_1$ for two-dimensional space) which are close to each other (there is just a small angle between them). When comparing our approach to the method proposed by Fieldsend and Singh [3] (see Figure 2), the Sigma method lets the particles to fly directly towards the Pareto-optimal front, where [3] blocks the particles and guides them towards those archive members which are in the special positions in the objective space.

The algorithm of the Sigma method is shown in Figure 6. There, the function *Sigma* calculates the value of $\vec{\sigma}$ and *calcdist* computes the euclidian distance between its inputs. In this algorithm, $\vec{y}_j$ denotes the objective value of the jth element of the archive A.

```
FindGlobalBest Algorithm
BEGIN
  Input: $A$, $\vec{x}^i$
  Output: $\vec{p}^{\,i,g}$

  Step 1: Calculate the $\sigma$ for the members of $A$:
    For $j = 1$ to $|A|$,
      $\sigma_j = Sigma(\vec{y}^j)$
    End;
  Step 2: Calculate $\sigma_i$ for the particle $i$:
    $\sigma_i = Sigma(\vec{f}(\vec{x}^i))$;
    $dist = calcdist(\sigma_1, \sigma_i)$;
    For $j = 2$ to $|A|$,
      $tempdist = calcdist(\sigma_j, \sigma_i)$;
      If $tempdist \leq dist$,
        $dist = tempdist$;
        $g = j$;
      End;
    End;
END
```

Fig. 6. The algorithm of the Sigma method for finding the best local guide $\vec{p}^{\,i,g}$ for the particle i of the population.

IV. EXPERIMENTS

In this section first we will explain how we have implemented the Sigma method, then we present experiments based on some test functions. Finally, we present comparative results with respect to other existing methods.

A. *Sigma method in MOPSO*

The Sigma method is implemented as part of the MOPSO algorithm as described in Figures 1 and 6. In the case that the objectives are not in the same range, σ may be calculated as below for a two-objective optimization problem:

$$\sigma = \frac{(K_2 f_1)^2 - (K_1 f_2)^2}{(K_2 f_1)^2 + (K_1 f_2)^2} \tag{7}$$

where K_1, K_2 are the maximum values of the first and second objective values of the particles in the population, respectively (Figure 5). In our implementation, we have also added a turbulence factor [3] to the updated position of each particle in the population. This parameter is like the mutation operator in evolutionary algorithms and is done by adding a random value to the current position of each particle. We have implemented the turbulence factor as below:

$$x^i_{j,t} \quad = \quad x^i_{j,t} + R_T \; x^i_{j,t} \tag{8}$$

Where R_T is a random value in $[0, 1]$ which is added to the updated position of each particle with a probability.

B. *Test functions*

We have used different test functions for testing and comparing the Sigma method with other methods. These test functions are two- and three-objective optimization problems selected from [9], [2] and are shown in Table I.

TABLE I
TEST FUNCTIONS

Test	Function
1	$g(x_2, \cdots, x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ $h(f_1, g) = 1 - \sqrt{f_1/g}$ $f_1(x_1) = x_1$ $f_2(\vec{x}) = g(x_2, \cdots, x_n).h(f_1, g)$
2	$g(x_2, \cdots, x_n) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ $h(f_1, g) = 1 - \sqrt{f_1/g} - (f_1/g)\sin(10\pi f_1)$ $f_1(x_1) = x_1$ $f_2(\vec{x}) = g(x_2, \cdots, x_n).h(f_1, g) + 1$
3	$g(x_2, \cdots, x_n) = 1 + 10(n-1)(\sum_{i=2}^{n} x_i - 10\cos(4\pi x_i))$ $h(f_1, g) = 1 - \sqrt{f_1/g}$ $f_1(x_1) = x_1$ $f_2(\vec{x}) = g(x_2, \cdots, x_n).h(f_1, g)$
4	$f_1(\vec{x}) = (1 + x_2)\cos(x_1\pi/2)\cos(x_2\pi/2)$ $f_2(\vec{x}) = (1 + x_2)\cos(x_1\pi/2)\sin(x_2\pi/2)$ $f_3(\vec{x}) = (1 + x_2)\sin(x_1\pi/2)$

In Table I, $x_i \in [0, 1]$ for the test functions $1, 2$ and 4, and $x_1 \in [0, 1]$ and $x_i \in [-5, 5], i \neq 1$ for the test function 3. In the two-objective test functions, $n = 30$ and in the three-objective test function, $n = 3$.

C. *Parameter settings*

In our implementation, we use $w = 0.4$ and R_1 and R_2 as random values in the range $[0, 1]$. The probability of adding the turbulence factor is set to be 0.01 and 0.05 in different test functions. Population size is chosen $N = 300$ and is run for 100 and 500 generations. The size of the archive is set to 50 for the two-objective test functions and to 100 for the three-objective test function. In the case that the archive size increases the maximum defined size, clustering is applied on the elite particles in the archive [9]. The initial population consists of uniformly distributed particles in the variable space.

D. *Experimental results and comparison*

In this section we present results of using the Sigma method, then we compare this algorithm with two other methods. The first method, which we call D-tree method has the same structure as Figure 1. In this method, the function *FindGlobalBest* is implemented as introduced by Fieldsend and Singh [3]. The second method is MOGA which is the implementation of the Strength Pareto Evolutionary Algorithm (SPEA2) method [10].

Figures 7 and 8(a)-(c) show the results of the Sigma method, the D-tree method and MOGA applied on the test functions 1 and 2. Both of these test functions are convex functions, where the first one is continuous and the second one has discontinuities in the Pareto-optimal front. In these cases, the population size is set to $N = 300$, the number of generations to 100, archive size to 50. For the MOPSO methods, the turbulence probability is 0.01 and for the MOGA method the probability of mutation is 0.01 whereas the crossover probability is set to 0.8. Figures 7 and 8(d) show the results at the same time of comparison. As it is shown in these figures, the Sigma method has a better diversity and convergence than the D-tree method and MOGA. Convergence of the solutions can be tested by a quantitative comparison when using the $\mathcal{C}$ metric [9]. Although because of the differences in the diversity of the obtained results the $\mathcal{C}$ metric is not a proper quantitative metric for comparing [6], we try to use this metric to make Figures 7 and 8(d) more clear. For test function 1, $\mathcal{C}$(MOGA, Sigma)$= 0$ and $\mathcal{C}$(Sigma, MOGA)$= 0.26$ and it means that non of the solutions of the MOGA can weakly dominate the solutions of the Sigma method. For test function 2, $\mathcal{C}$(MOGA, Sigma)$= 0.24$ and $\mathcal{C}$(Sigma, MOGA)$= 0.1$. Here, we conclude that the results are

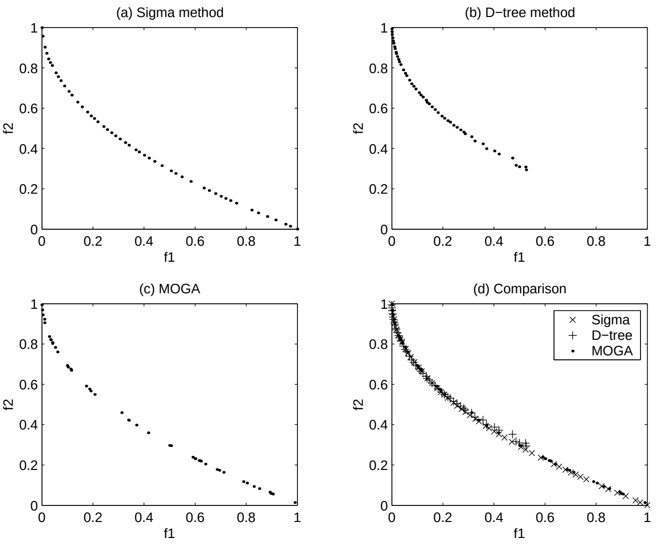

Fig. 7. Comparison of the Sigma method (a), the D-tree method (b) and MOGA (c) applied on test function 1.

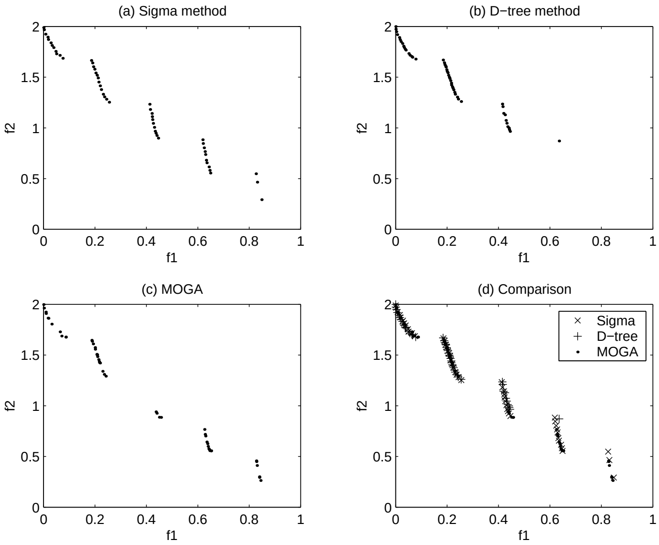

Fig. 8. Comparison of the Sigma method (a), D-tree method (b) and MOGA (c) applied on test function 2.

incomparable in terms of convergence, however we can observe that the result of the Sigma method has a better diversity than MOGA.

Figure 9 shows the results on test function 3. This test function has 21^9 local optima and finding the global Pareto-front is very difficult for each of the three methods. In this case, we have increased the turbulence probability to 0.05 and the number of generations to 500. As it is shown in Figure 9(d), the Sigma method can find better solutions than MOGA and D-tree method. Figures 10(a)-(c) show the results of the methods applied on test function 4. Here, we have chosen population size of 300 and 500 as the number of generations. For obtaining a better result from MOGA, we have also increased the Mutation probability to 0.1. As we can observe, MOGA can find solutions with a very good convergence as compared to both of the MOPSO methods. Comparing the Sigma method with the D-tree method, the Sigma method can find more solutions on the front. Considering diversity, the solutions of the MOPSO methods have better diversity than MOGA. In test function 4, when $f_1(x^j) + f_2(x^j) + f_3(x^j) = 1$ it means that particle j is on the Pareto-optimal front. So we can calculate err which can be defined among the archive members as follows:

$$err \quad = \quad 1 - (f_1(x^j) + f_2(x^j) + f_3(x^j)), \ \forall j = 1, \cdots, |A|$$

Hence we can calculate the histogram which is the amount of points which have a certain err. The points which have $err = 0$ are on the Pareto-optimal front. Figures 11(a)-(c) show the histograms of err applied on the results of Sigma method, D-tree method and MOGA.

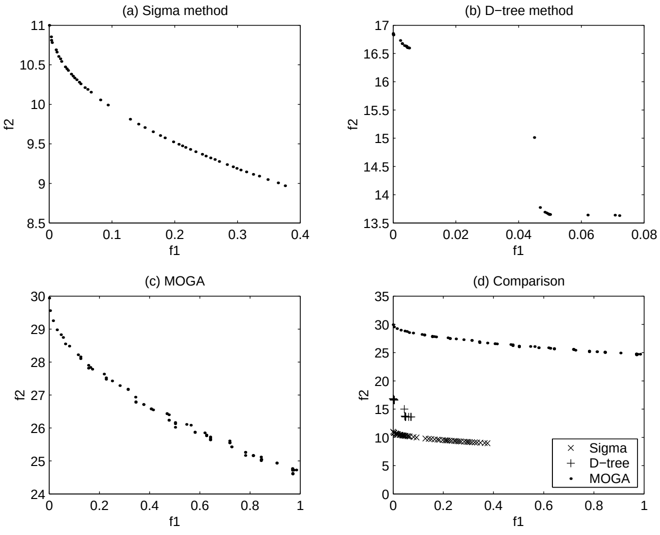

Fig. 9. Comparison of the Sigma method (a), D-tree method (b) and MOGA (c) applied on test function 3.

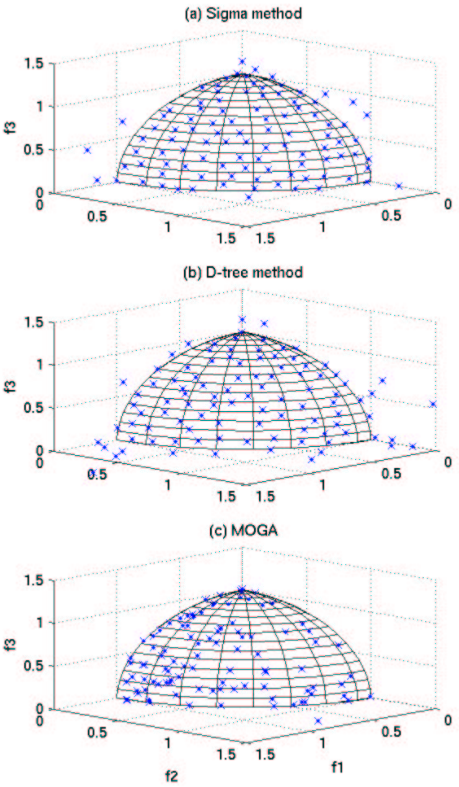

Fig. 10. Comparison of the Sigma method (a), the D-tree method (b) and MOGA (c) applied on test function 4.

The Sigma method and MOGA can find solutions with $err = 0$, where the D-tree method can not find. If we set a comparison line on $err = 0.5$, MOGA finds more solutions than the Sigma method and the D-tree method with $err < 0.5$. Comparing the Sigma method and the D-tree method, the Sigma method can find more solutions than the D-tree method with $err < 0.5$. However, we can observe that the sigma method can find solutions with a very good diversity and better than MOGA.

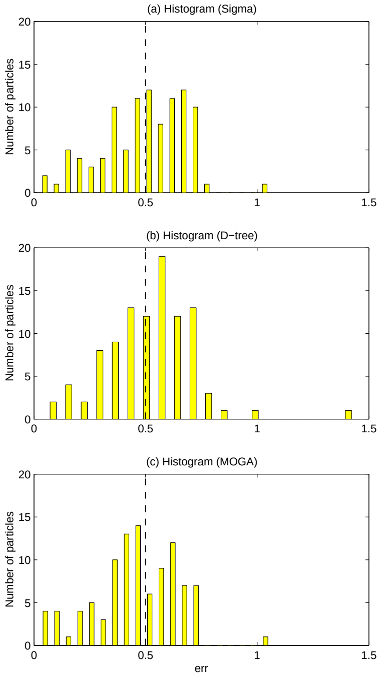

Fig. 11. Histograms of err for the results in Figures 10(a)-(c).

V. CONCLUSION AND FUTURE WORK

In this paper, we propose a new method called Sigma method in MOPSO for finding the best local guides in order to converge fast towards a Pareto-optimal front of high diversity. As our experiments has shown, this method can find solutions with very good diversity and convergence. However, we have to say that for getting good results of a MOPSO we had to increase the number of particles in the population. Because for finding the best local guide which is the most important part in PSO (for searching the variable space), we need enough distributed solutions in the objective space. We have also added the turbulence parameter to do some abrupt changes in the population. Comparing the Sigma method with the proposed method for finding the best local guides by Fieldsend and Singh [3], the Sigma method can find solutions with better convergence and diversity for two- and three-objective test functions. Compared to SPEA2 which is an evolutionary algorithm, the Sigma method can find solutions with better convergence and diversity for two-objective test functions. However, for three-objective test functions, the Sigma method can not find solutions with better convergence than SPEA2 but with a better diversity. These results are all for the recorded number of generations and population size, however all of these methods can find solutions with good convergence and diversity after a large amount of generations. We have to note that since the Sigma method can only work on the positive values of the objective space, all the test functions must have positive objective values. This must be done by having an approximate priori knowledge about the test functions.

In the future we would like to investigate and compare different clustering methods when using the Sigma method. As the Sigma method can give us a very good diversity of solutions, we would also like to investigate the influence of the parameters, especially for higher dimensional objective spaces to achieve a better convergence.

References

[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 (CEC2002)*, pages 1051–1056, 2002.

[2] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler. Scalable multi-objective optimization test problems. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC2002)*, 2002.

[3] 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, U.K., 2002.

[4] X. Hu and R. Eberhart. Multiobjective optimization using dynamic neighborhood particle swarm optimization. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC2002)*, pages 1677–1681, 2002.

[5] James Kennedy and Russell C. Eberhart. *Swarm Intelligence*. Morgan Kaufmann, 2001.

[6] J. Knowles and D. Corne. On metrics for comparing non-dominated sets. In *IEEE Proceedings, World Congress on Computational Intelligence (CEC2002)*, pages 711–716, 2002.

[7] K. E. Parsopoulos and M. N. C. Vrahatis. Particle swarm optimization method in multiobjective problems. In *of the 2002 ACM Symposium on Applied Computing (SAC)*, pages 603–607, 2002.

[8] Y. Shi and R. C. Eberhart. *Parameter Selection in Particle Swarm Optimization*. Evolutionary Programming, 591-600, 1998.

[9] 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.

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