2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

A New Strategy for Finding Good Local Guides in MOPSO

Man-Fai Leung, Sin-Chun Ng, *Senior Member*, *IEEE*, Chi-Chung Cheung, *Senior Member*, *IEEE*, and Andrew K Lui, *Member*, *IEEE*

***Abstract*—This paper presents a new algorithm that extends Particle Swarm Optimization (PSO) to deal with multi-objective problems. It makes two main contributions. The first is that the square root distance (SRD) computation among particles and leaders is proposed to be the criterion of the local best selection. This new criterion can make all swarms explore the whole Pareto-front more uniformly. The second contribution is the procedure to update the archive members. When the external archive is full and a new member is to be added, an existing archive member with the smallest SRD value among its neighbors will be deleted. With this arrangement, the non-dominated solutions can be well distributed. Through the performance investigation, our proposed algorithm performed better than two well-known multi-objective PSO algorithms, MOPSO-σ and MOPSO-CD, in terms of different standard measures.**

I. Introduction

Particle Swarm Optimization (PSO) was presented by Kennedy and Eberhart [1] in 1995, inspired by the behavior of swarms. Like bird flocking, there are a number of particles moving in a given search space. Each member moves in the space according to its previous best position and a global best leader's position of the swarm. PSO is an efficient algorithm for solving single objective problems and it motivates researchers to extend PSO to solve multi-objective problems (MOPs) [16].

MOPs consist of two or more objectives that need to be optimized simultaneously, though sometimes they may be in conflict. In MOPs, a set of non-dominated solutions (called Pareto-optimal solutions [11]) is generated instead of one global solution. A set of "good" Pareto-optimal solutions is obtained if it can:

1. Maximize the number of Pareto-optimal solutions;
2. Minimize the distance between the Pareto-optimal solutions and the true solutions, and
3. Maximize the distribution of the Pareto-optimal solutions.

Different research contributions have been made to deal with multi-objective problems after the first release of PSO [12 – 14]. Results showed that PSO is competitive with evolutionary algorithms (EAs) on multi-objective problems, though PSO is relatively younger than many EAs. Thus it is believed that there should still be room for improvement. Different research works, which have recently been carried out in this field, involve Multi-objective PSO algorithms (MOPSO).

A Pareto-optimal set, which will be generated when optimizing a multi-objective problem, usually consists of a number of non-dominated solutions that are found in a feasible region. A decision variable $\vec{x}$ is said to be a non-dominated solution when there are no other solutions in the feasible region which dominates it.

In [4], Mostaghim and Teich proposed a MOPSO algorithm with sigma method (i.e., MOPSO-σ) for finding local guides. Their performance investigation showed that the proposed method can find solutions with good convergence and diversity. However, the population size of search particles should be sufficiently large to obtain good results. Moreover, MOPSO-σ may sometimes fail to generate solutions with larger spread due to the premature convergence of some search swarms (i.e., the solution obtained is not good enough). [5] proposed an algorithm called MOPSO-CD that adopts the crowding distance mechanism for finding the global best guide and deleting solutions when the external archive is full. The crowding distance computation is used to promote diversity of solutions. Their performance investigation showed that their algorithm can generate a set of well-distributed solutions and it performs well in converging to the true Pareto-front (a full set of non-dominated solutions is called Pareto-front).

Recent investigations in [17, 18] showed that Sigma and crowding distance methods for leader selections in MOPSO are competitive. Both of them can produce very good results. So, it should be noted that the leader selection on local/global guide(s) plays a key role in MOPSO algorithms. Different leader selections may result in different trajectories of search swarm during their flights, and hence affect the quality, quantity, and distribution of the final solutions. Another key factor is the adoption of the external archive. The external archive is used to maintain a set of Pareto-optimal solutions. Although there are some MOPSO proposals [2, 3] that adopt unlimited archive size, they are not very popular because the number of non-dominated solutions can grow very fast and hence increase the computation cost significantly when updating the archive. Thus, spending a lot of time to find a huge number of solutions is not cost effective. To maintain a fixed size of the external archive, the way to remove existing members when the archive is full is very important. To

Man-Fai Leung, Sin-Chun Ng and Andrew Lui are with the School of Science and Technology, The Open University of Hong Kong, Hong Kong, China. (Their e-mails are : mfleung@ouhk.edu.hk, scng@ouhk.edu.hk and alui@ouhk.edu.hk respectively).

Chi-Chung Cheung is with the Department of Electronic and Information Engineering, Hong Kong Polytechnic University, Hong Kong, China (e-mail: encccl@polyu.edu.hk).

control the size of the external archive, MOPSO-σ adopts clustering-based size control [19] and MOPSO-CD adopts crowding-distance based size control. However, their controls are highly dependent on the optimization problem that they are applied to. In this paper, a better leader selection and a better archive size control algorithm are proposed to improve the performance of our MOPSO algorithm.

The remainder of this paper is as follows. Section II describes two popular MOPSO algorithms, MOPSO-σ and MOPSO-CD. Section III presents our proposed algorithm — MOPSO-SRD — with examples. Section IV shows the performance comparisons of MOPSO-σ, MOPSO-CD and MOPSO-SRD in different optimization problems. Section V presents our conclusions.

II. The Popular Existing MOPSO Algorithms

There are two most popular MOPSO algorithms, MOPSO-σ and MOPSO-CD. This section describes them and their limitations.

A. Multi-objective Particle Swarm Optimization Using Sigma Method (MOPSO-σ)

Sigma method is used to find a local guide for each particle [4]. To group particles into different archive members, the sigma value of each particle (denoted as σ) is calculated. The sigma value of a particle is defined as the slope of the line connecting the fitness value of the particle and the origin of the objective space. For 2-objective problems, σ is defined as:

$$\sigma = (f_1^2(x) - f_2^2(x)) / (f_1^2(x) + f_2^2(x)) \quad (1)$$

For 3-objective problems, σ is a vector and it is defined as:

$$\vec{\sigma} = \begin{pmatrix} f_1^2(x) - f_2^2(x) \\ f_2^2(x) - f_3^2(x) \\ f_3^2(x) - f_1^2(x) \end{pmatrix} / \left(f_1^2(x) + f_2^2(x) + f_3^2(x) \right) \quad (2)$$

In a two-objective problem, when the sigma values of two particles are equal, the two particles must lie on the same line. Based on this method, each particle in the swarm selects the leader particle from the archive by finding an archive member with the closest sigma value (i.e., the difference between the sigma values of the particle and the archive member is the smallest among all archive members). By using this method, all search particles can move directly towards the Pareto-front through their corresponding leaders, and it is hoped that they can obtain solutions with good convergence. Note that an external archive with clustering-based size control is used to store non-dominated solutions during the search process, and its size is fixed. Moreover, to retain the vitality of the swarm, MOPSO-σ has introduced a turbulence factor [2] to the MOPSO. This factor adds a random number to the position of each particle, which acts like a mutation operator in EAs. The performance investigation in [4] showed that the overall performance of MOPSO-σ is good in different multi-objective optimization problems. However, some studies reported that this method may sometimes cause premature convergence and hence not be able to generate a better spread to cover the Pareto-front [16, 22].

B. Multi-objective Particle Swarm Optimization using the Crowding Distance (MOPSO-CD)

Raquel and Naval adopted the crowding distance calculation mechanism, which is from the Non-dominated Sorting Genetic Algorithm II (NGSA-II) [15] into the PSO algorithm [5]. The crowding distance of a particular solution is calculated to estimate the density of the surrounding solutions. The calculated values of the archive solutions are sorted in descending order so that a global best guide can be selected randomly from archive members in a specified top portion (e.g., 10%) for each particle. Note that the boundary archive members are always set to an infinite value to ensure that they are always in the top portion (i.e., they are always available to be selected). The crowding distance computation is also used to remove solutions (i.e., archive members) when the archive is full. Finally, a mutation operator is used to enrich the search ability of the proposed algorithm.

MOPSO-CD is very popular to search for solutions to multi-objective problems. However, since each particle is associated with its own global guide solely selected from the top 10% less crowded area of the archive, it is too restrictive for those particles far away from the less crowded area and could possibly perturb their original flight [21].

III. MOPSO-SRD

To address the limitations of these two popular MOPSO algorithms, a new algorithm called MOPSO using the Square Root Distance (MOPSO-SRD) is proposed. This algorithm makes two main contributions by adding (a) a new leader selection algorithm and (b) a new control mechanism for the external archive.

A. Leader Selection Algorithm

In the original PSO algorithm, the swarm can converge quickly because the whole search swarm is guided by one global leader. As mentioned in Section I, PSO cannot be applied to solve multi-objective problems directly, and thus a leader selection algorithm is required for each particle to find its own leader. In MOPSO-SRD, each particle can freely choose its own leader, but not a single global best particle, by using the square root distance calculation. The square root distance calculation of two points $\vec{x}_1$ and $\vec{x}_2$ is shown as:

$$SRD(\vec{x}_1, \vec{x}_2) = \sum_{n=1}^{m} \sqrt{\left| f_n(\vec{x}_1) - f_n(\vec{x}_2) \right|} \quad (3)$$

In each generation, for each particle in the search swarm, the square root distance between the particle and all archive members is calculated and the archive member with the shortest square root distance is chosen as the leader of that particle, i.e.,

$$\text{Min}\left(\sum_{n=1}^{m} \sqrt{\left| f_n(\vec{y}_1) - f_n(\vec{x}) \right|}, \ldots, \sum_{n=1}^{m} \sqrt{\left| f_n(\vec{y}_k) - f_n(\vec{x}) \right|} \right) \quad (4)$$

where m is the number of objectives, $\{\vec{y}_1, \vec{y}_2, \cdots, \vec{y}_k\}$ is the archive set with k members, and $P = \vec{x}$ where P is the particle. Fig. 1 shows a small example to illustrate the operations of

the leader selection. There is a particle D to select a leader from an external archive which has three archive members (A, B and C). The particle D chooses archive member A as its leader because the square root distance of AD (*SRD*(D, A) = 2 units) is the smallest one among all others (*SRD*(D, B) = 3.46 units and *SRD*(D, C) = 2.73 units).

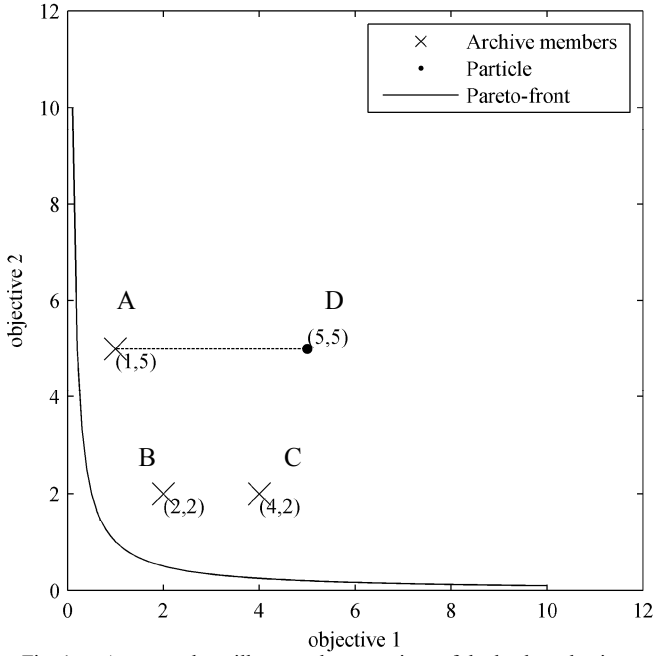

Fig. 1. An example to illustrate the operations of the leader selection.

As mentioned above, MOPSO-σ may not be able to generate a better spread to cover the Pareto-front because of the premature convergence. Compared with MOPSO-σ, a particle can be attracted by all possible leaders and thus all particles can fairly explore the whole Pareto-front. Fig. 2 shows 100 particles that are randomly generated in a two objective space, and they are grouped by using the SRD computation. Fig. 3 shows the same particles but they are grouped by using the sigma method. The crosses in these two figures are archive members and the dots are searching particles. It is clearly found that the particles grouped by using the SRD computation are more evenly distributed in relation to the Pareto-front than the sigma method. However, in Fig. 3, most of particles are guided by the center of the archive set and only a few particles are guided by archive members near the boundary of the whole archive set. Compared with MOPSO-CD, MOPSO-SRD does not have any restrictions on finding a local guide and thus the limitation found in MOPSO-CD does not exist in MOPSO-SRD.

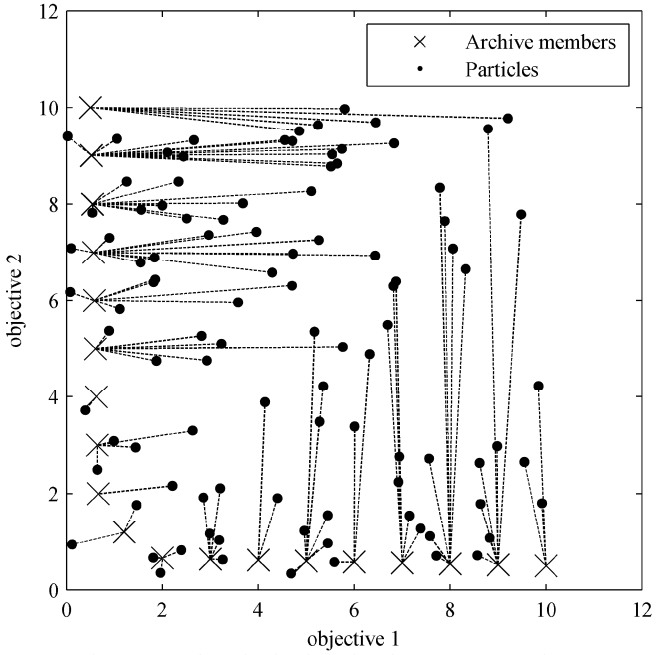

Fig. 2. Leader selection by using the SRD computation.

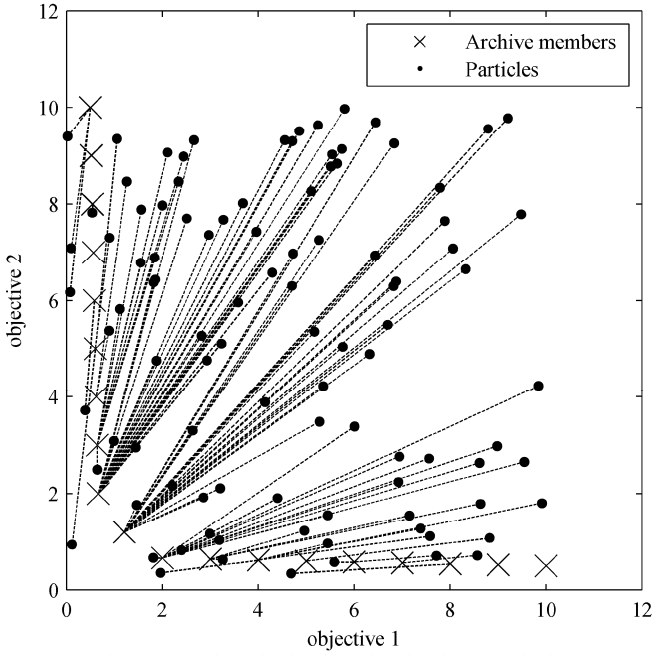

Fig. 3. Leader selection by using the sigma method.

B. Archive Control Algorithm

The two popular MOPSO algorithms, MOPSO-σ and MOPSO-CD, use an archive to store the set of non-dominated solutions. The archive adopts the concept of an archive controller in [6]. When a new archive member is found and there are no archive members in the archive that can dominate this new member, it is added into the archive. When the archive is full, a procedure is required to remove an archive member from the archive. MOPSO-σ uses a clustering-based size control to remove archive members when the archive is full. MOPSO-CD selects an archive member with the shortest crowding distance among all to remove when the archive is full. MOPSO-SRD considers the sum of SRD of an archive

member among its two neighbors. The calculation of this sum (called Neighbor Factor (NF) in this paper) is shown below:

$$NF(\vec{x}_i) = SRD(\vec{x}_{i-1}, \vec{x}_i) + SRD(\vec{x}_{i+1}, \vec{x}_i) \quad (5)$$

for the i^{th} archive member with n non-dominated solutions and 1<i<n. An archive member with the smallest NF value will be removed when the archive is full. Note that no boundary members will be considered for removal because they need to remain in the archive to maintain a well-distributed Pareto front. The reason to consider NF as a factor in the removal procedure is that it makes the remaining non-dominated solutions more evenly distributed and closer to the true Pareto-front.

Fig. 4 illustrates when the archive members are almost equally close to the Pareto-front, the member with the smallest value of NF will be removed. This measure can preserve diversity. Table I shows the NF value of each archive member (excluding the boundary member A and H). In this scenario, archive member C has the smallest NF value. Thus it is selected to be removed. Fig. 5 shows another scenario. Compared with Fig.4, it has an additional archive member I which is far away from the Pareto-front and it will significantly affect the decision to remove a selected member. This time archive member C will not be removed. Instead, the archive member I will be removed.

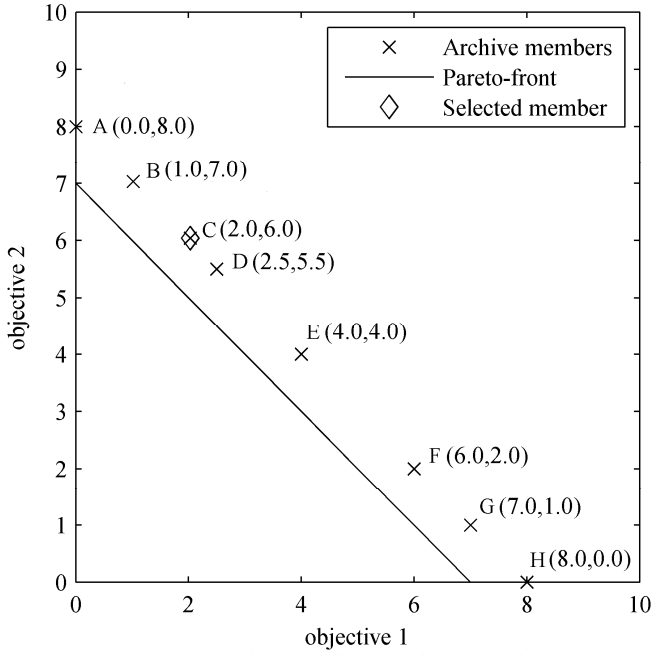

Fig. 4. Removal of an archive member (Scenario 1)

Table I. NF values of archive members in Fig. 4.

Archive members	B	C	D	E	F	G
Neighbor Factor (NF)	4	***3.41***	3.86	5.28	4.83	4

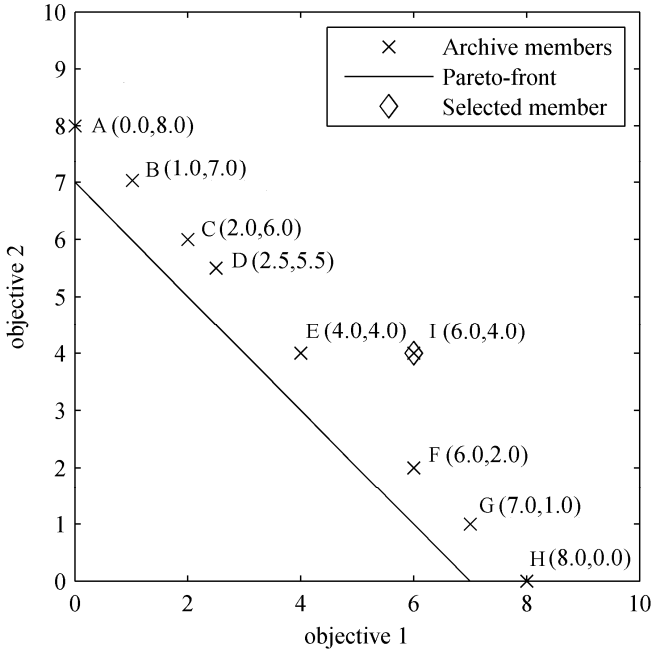

Fig. 5. Removal of an archive member (Scenario 2)

Table II. NF values of archive members in Fig. 5.

Archive members	B	C	D	E	I	F	G
Neighbor Factor (NF)	4	3.41	3.86	3.86	***2.83***	3.41	4

C. The Main MOPSO-SRD Algorithm

Fig. 6 shows the MOPSO-SRD algorithm. At the beginning, the positions, speeds and past best locations of all particles are initialized. Then each particle is evaluated based on the objective functions (fitness functions). The evaluated fitness values of particles are compared with each other and the one that is not dominated (i.e., a non-dominated solution) by others will be added into the archive. The above procedure will be repeated until the end of iterations. The new leader selection is applied in each iteration and each particle performs its flight (updates its velocity and new position) at the end of each iteration. Mutation is applied to enhance the exploratory ability of the algorithm. After moving to the new position, evaluation is carried out to re-calculate the fitness value of each particle. Then the archive will be updated, as well as the past best position of the swarm. If the archive is full, the proposed removal procedure will be applied to maintain the size of the archive.

```
Begin
1. Initialize Archive A={};
2. For each particle j,
     Initialize each particle's position randomly P[j]
     Initialize the speed of each particle to zero V[j]=0
     Initialize the past best of each particle Pb[j]=P[j]
     Evaluate particles P[j]
   End for
3. Update archive A
4. For i = 1 to the specified number of iterations
     For each particle j,
       SRD_Computation(A,j), which returns local best leader Lb
       Update each particle's new velocity V[j]:
         V[j]=w*V[j]+r1*c1*(Pb[j]-P[j])+r2*c2*(Lb-P[j])
         where
           w is an inertia weight,
           r1 and r2 are random numbers between 0 and 1,
           c1 is local weight and it is a constant,
           c2 is global weight and it is a constant,
           Pb is the particle with the past best value, and
           Lb is the particle with the local best value.
       Update new position of particles: P[j]=P[j]+V[j]
     End for
     Mutation
     Evaluate particles
     Update archive A
       Insert non-dominated solutions
       If A is full
         NF_Computation(A), which returns an archive member
         Remove the selected archive member
       End if
     Find the personal best position of each particle
     Increase iteration i by 1
  End for
End
```

Fig. 6. The MOPSO-SRD algorithm

IV. Performance Comparisons

This section describes the performance comparisons among MOPSO-σ, MOPSO-CD and MOPSO-SRD. Four performance measures and five multi-objective optimization problems were used to compare their performance. For each optimization problem, 50 independent runs were carried out. All the runs were performed under the same environment (Matlab) on Intel Core i3-3217U 1.8GHz CPU with 4GB DDR3 RAM.

A. Performance Metrics

Spacing (S): This performance metric proposed by Schott [7], is used to measure the distance variance of neighboring non-dominated solutions. The metric is defined as:

$$S = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(\theta - d_i)^2} \tag{6}$$

$$where\ d_i = min_j \sum_{k=1}^{m} | f_k^i(\vec{x}) - f_k^j(\vec{x}) |\ \ i, j = 1, 2, \ldots, n$$

θ is the mean of all d_is, m is the number of objectives, and n is the number of found non-dominated solutions. This performance metric is good if S is small. A value of zero means that all found non-dominated solutions are spaced equidistantly.

Error Ratio (ER): ER was proposed by Van Veldhuizen [8] and is used to measure the error percentage of found non-dominated solutions. The metric is defined as:

$$ER = \frac{1}{n}\sum_{i=1}^{n} e_i \tag{7}$$

If the non-dominated solution i belongs to the true Pareto-front, $e_i = 0$; otherwise, $e_i = 1$. This performance metric is good if ER is small. A value of zero means that all found non-dominated solutions belong to the true Pareto-front. Note that this metric requires prior knowledge about the true Pareto-front of the test case.

Generational distance (GD): GD was also proposed by Van Veldhuizen [8], and is used to measure the difference between found non-dominated solutions and the true Pareto-front. The metric is defined as:

$$GD = \frac{1}{n}\sqrt{\sum_{i=1}^{n}\alpha_i^2} \tag{8}$$

where α_i is the Euclidean distance between the non-dominated solution i and the nearest member of the true Pareto set. This performance metric is good if GD is small. A value of zero means that all the found non-dominated solutions lie on the true Pareto front. Like ER, this metric requires prior knowledge about the true Pareto-front of the test case.

Searching Time (ST): It is a common performance metric in the performance comparisons of optimization algorithms. The ST is the total time taken for the computer to execute in an experiment. The number of generations in each test function will be described later in this section. The performance of an algorithm is good if its searching time is short.

B. Test Functions

Kursawe test function: This consists of two objective functions with three variables [9]. Its true Pareto front is non-continuous. It is defined as:

$$\text{minimize}\begin{cases} f_1(x) = \sum_{i=1}^{2}\left(-10\exp\left(-0.2\sqrt{x_i^2 + x_{i+1}^2}\right)\right) \\ f_2(x) = \sum_{i=1}^{3}\left(|x_i|^{0.8} + 5\sin(x_i)^3\right) \end{cases} \tag{9}$$

where $-5 \le x_1, x_2, x_3 \le 5$.

ZDT1: This consists of two objective functions with 30 variables [10]. Its true Pareto front is convex and continuous. It is defined as:

$$\text{minimize}\begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h\left(f_1(x), g(x)\right) \end{cases} \tag{10}$$

$$\text{where}\begin{pmatrix} x = (x_1, \ldots, x_E) \\ g(x_2, \ldots, x_E) = 1 + 9\left(\sum_{i=2}^{E} x_i / (E-1)\right) \\ h(f_1, g) = 1 - \sqrt{f_1 / g} \\ E = 30 \text{ and } 0 \le x_i \le 1 \end{pmatrix}$$

ZDT2: This consists of two objective functions with 30 variables [10]. Its true Pareto front is concave and continuous. It is defined as:

$$\text{minimize}\begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h\left(f_1(x), g(x)\right) \end{cases} \quad (11)$$

$$\text{where}\begin{pmatrix} x = (x_1, \ldots, x_E) \\ g(x_2, \ldots, x_E) = 1 + 9\left(\sum_{i=2}^{E} x_i / (E-1)\right) \\ h(f_1, g) = 1 - (f_1 / g)^2 \\ E = 30 \text{ and } 0 \le x_i \le 1 \end{pmatrix}$$

ZDT3: This consists of two objective functions with 30 variables [10]. Its true Pareto front is non-continuous. It is defined as:

$$\text{minimize}\begin{cases} f_1(x) = x_1 \\ f_2(x) = g(x)h\left(f_1(x), g(x)\right) \end{cases} \quad (12)$$

$$\text{where}\begin{pmatrix} x = (x_1, \ldots, x_E) \\ g(x_2, \ldots, x_E) = 1 + 9\left(\sum_{i=2}^{E} x_i / (E-1)\right) \\ h(f_1, g) = 1 - \sqrt{f_1 / g} - (f_1 / g)\sin(10\pi f_1) \\ E = 30 \text{ and } 0 \le x_i \le 1 \end{pmatrix}$$

Viennet test function: This consists of three objective functions with two variables [20]. Its true Pareto front is a three dimensional convoluted line. It is defined as:

$$\text{minimize}\begin{cases} f_1(x,y) = 0.5(x^2+y^2) + \sin(x^2+y^2) \\ f_2(x,y) = (3x-2y+4)^2/8 + (x-y+1)^2/27 + 15 \\ f_3(x,y) = 1/(x^2+y^2+1) - 1.1\exp(-x^2-y^2) \end{cases} \quad (13)$$

where $-3 \le x, y \le 3$.

Table III. Parameter values for each algorithm with respect to each test function.

Algorithm	MOPSO-σ	MOPSO-CD	MOPSO-SRD
Kursawe			
inertia rate(w)	0.7	0.7	0.7
local weight (c1)	1.2	1.2	1.2
global weight (c2)	1.3	1.3	1.3
mutation rate	0.05	0.05	0.05
ZDT1			
inertia rate(w)	1.5	1.5	1.5
local weight (c1)	1.5	1.5	1.5
global weight (c2)	1.5	1.5	1.5
mutation rate	0.03	0.03	0.03
ZDT2			
inertia rate(w)	1.5	1.5	1.5
local weight (c1)	1.5	1.5	1.5
global weight (c2)	1.5	1.5	1.5
mutation rate	0.03	0.03	0.03
ZDT3			
inertia rate(w)	0.55	0.55	0.55
local weight (c1)	2	2	2
global weight (c2)	2	2	2
mutation rate	0.03	0.03	0.03
Viennet			
inertia rate(w)	1.5	1.5	1.5
local weight (c1)	1.5	1.5	1.5
global weight (c2)	1.5	1.5	1.5
mutation rate	0.03	0.03	0.03

The MOPSO-σ, MOPSO-CD and MOPSO-SRD algorithms were implemented in our simulation program. Since the parameters in the clustering method proposed by [4] are quite difficult to follow, for simplicity, an alternative proposed by [22] was implemented for the archive control algorithm for MOPSO-σ. In all experiments, the number of generations and population size were set to 100 for Kursawe test function. For ZDT1, ZDT2 and ZDT3, the number of generations and population size were set to 150 and 100 respectively. For Viennet test function, the number of generations and population size were set to 50 and 100 respectively. Table III lists out the parameter values (the inertia rate, the local weight, the global weight and the mutation rate) for each algorithm with respect to each test function. Figs 7 to 11 show some Pareto-fronts produced by these three algorithms for different optimization problems. It can be found that the Pareto-front produced by MOPSO-SRD is generally more evenly distributed than MOPSO-σ and MOPSO-CD. Tables IV to VIII show the performance comparisons of different MOPSO algorithms in different optimization problems. A number highlighted in bold is the winner in such corresponding performance metric. For example, the error rate (*ER*) of MOPSO-SRD is 0.0342, which is smaller (better) than MOPSO-σ (0.0598) and MOPSO-CD (0.269) in Table IV. Table IX summarizes the statistics of winners of different performance metrics in different optimization problems. For a particular metric with respect to a test function, the algorithm which outperforms other algorithms (i.e., the winner) is counted. For example, MOPSO-σ performed better than MOPSO-CD and MOPSO-SRD in ZDT1 test function in terms of the searching time (ST), so MOPSO-σ is counted once in the column of ST in Table IX. Note that the comparison focuses on the average value only, not the standard deviation. It can be found that MOPSO-CD is good in terms of searching time but MOPSO-SRD is good in the rest of the performance metrics (totally three performance metrics). Overall, the performance of MOPSO-σ was the worst and MOPSO-SRD generally outperformed MOPSO-σ and MOPSO-CD.

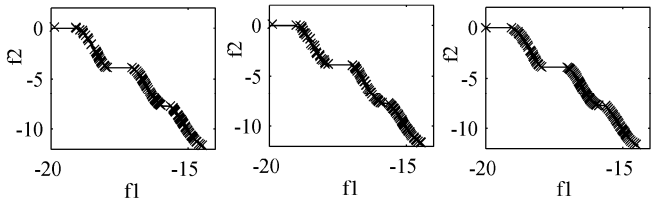

Fig. 1. Pareto-fronts produced by MOPSO-σ (left), MOPSO-CD (middle) and MOPSO-SRD (Right) for Kursawe test function.

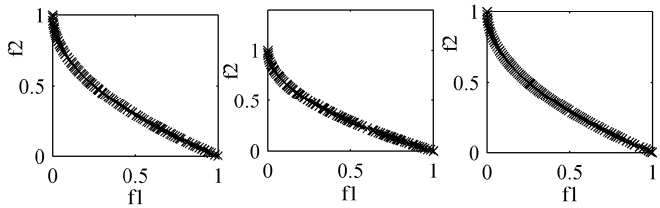

Fig. 2. Pareto-fronts produced by MOPSO-σ (left), MOPSO-CD (middle) and MOPSO-SRD (Right) for ZDT1 test function.

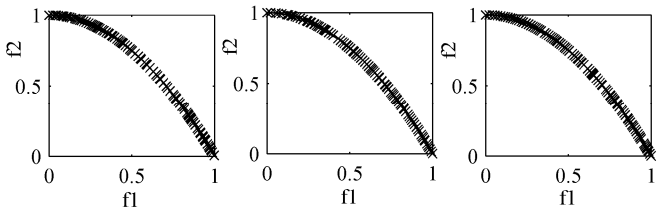

Fig. 3. Pareto-fronts produced by MOPSO-σ (left), MOPSO-CD (middle) and MOPSO-SRD (Right) for ZDT2 test function.

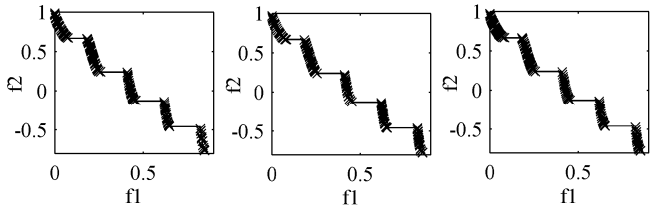

Fig. 4. Pareto-fronts produced by MOPSO-σ (left), MOPSO-CD (middle) and MOPSO-SRD (Right) for ZDT3 test function.

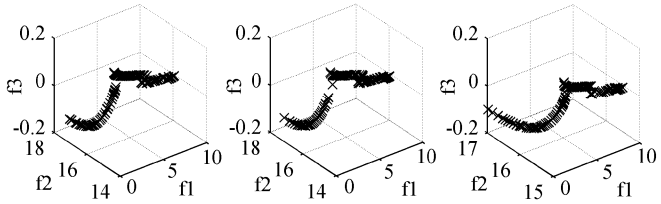

Fig. 5. Pareto-fronts produced by MOPSO-σ (left), MOPSO-CD (middle) and MOPSO-SRD (Right) for Viennet test function.

Table IV. Results for the Kursawe test function.

Algorithm	MOPSO-σ	MOPSO-CD	MOPSO-SRD
		S	
Average	0.137	**0.123**	0.125
Std. Dev.	0.0102	0.0137	0.0052
		ER	
Average	0.0598	0.269	**0.0342**
Std. Dev.	0.02788	0.04071	0.01751
		GD	
Average	2.24E-03	6.27E-03	**1.38E-03**
Std. Dev.	0.0008462	0.001410	0.0002361
		ST	
Average	17.6 seconds	**7.55 seconds**	14.2 seconds
Std. Dev.	0.8457 seconds	0.8737 seconds	0.9042 seconds

Table V. Results for the ZDT1 test function.

Algorithm	MOPSO-σ	MOPSO-CD	MOPSO-SRD
		S	
Average	8.28E-03	9.50E-03	**4.13E-03**
Std. Dev.	0.0008000	0.0006542	0.0007683
		ER	
Average	0.0800	0.0606	**0.0328**
Std. Dev.	0.03130	0.02478	0.02071
		GD	
Average	7.09E-05	4.99E-05	**3.27E-05**
Std. Dev.	3.237E-05	2.482E-05	2.740E-05
		ST	
Average	**18.00 seconds**	21.3 seconds	21.2 seconds
Std. Dev.	0.8084 seconds	0.9598 seconds	0.8322 seconds

Table VI. Results for the ZDT2 test function.

Algorithm	MOPSO-σ	MOPSO-CD	MOPSO-SRD
		S	
Average	1.45E-02	**9.68E-03**	9.76E-03
Std. Dev.	0.01191	0.0005846	0.003096
		ER	
Average	0.02	0.018	**0.017**
Std. Dev.	0.01591	0.01587	0.01266
		GD	
Average	6.56E-05	**2.20E-05**	5.28E-05
Std. Dev.	5.533E-05	2.488E-05	4.524E-05
		ST	
Average	4.50 seconds	19.6 seconds	**4.35 seconds**
Std. Dev.	0.9834 seconds	2.0264 seconds	0.612373137

Table VII. Results for the ZDT3 test function.

Algorithm	MOPSO-σ	MOPSO-CD	MOPSO-SRD
		S	
Average	3.24E-02	2.83E-02	**2.82E-02**
Std. Dev.	0.002499	0.0008293	0.001162
		ER	
Average	0.197	0.264	**0.165**
Std. Dev.	0.06390	0.08463	0.05950
		GD	
Average	9.64E-05	8.38E-05	**4.97E-05**
Std. Dev.	4.103E-05	3.040E-05	1.948E-05
		ST	
Average	20.8 seconds	**12.5 seconds**	16.4 seconds
Std. Dev.	1.3886 seconds	1.0221 seconds	0.7559 seconds

Table VIII. Results for Viennet test function.

Algorithm	MOPSO-σ	MOPSO-CD	MOPSO-SRD
		S	
Average	6.29E-02	7.02E-02	**6.26E-02**
Std. Dev.	0.009477	0.02628	0.01147
		ER	
Average	0.150	0.175	**0.101**
Std. Dev.	0.03605	0.04515	0.03442
		GD	
Average	8.83E-04	1.02E-03	**5.59E-04**
Std. Dev.	0.0005250	0.0006130	0.0003132
		ST	
Average	6.91 seconds	**5.84 seconds**	7.53 seconds
Std. Dev.	0.4925 seconds	0.3529 seconds	0.6109 seconds

Table IX. The summary of their overall performance.

Algorithm / Metric	No. of winners		
	MOPSO-σ	MOPSO-CD	MOPSO-SRD
S	0	2	**3**
ER	0	0	**5**
GD	0	1	**4**
ST	1	**3**	1
Total	1	6	**13**

V. CONCLUSION

This paper has presented a new MOPSO algorithm called MOPSO using the square root distance (MOPSO-SRD) to solve multi-objective optimization problems. It makes two main contributions: (a) a new leader selection algorithm is proposed by using the square root distance (SRD) computation and (b) a new archive control algorithm to remove an archive member if the archive is full. The performance investigation showed MOPSO-SRD generally

outperforms two popular MOPSO algorithms, MOPSO-σ and MOPSO-CD, in terms of different performance metrics in different multi-objective optimization problems. Future work will focus on optimizing more multi-objective optimization problems with higher dimensions.

REFERENCES

[1] Eberhart R, Kennedy J., "A new optimizer using particle swarm theory", *Proceedings of the Sixth International Symposium on. IEEE*, 1995.

[2] Fieldsend J E, Uk E Q, Singh S., "A Multi-Objective Algorithm based upon Particle Swarm Optimisation, an Efficient Data Structure and Turbulence", *Proceedings of the 2002 U. K. Workshop on Computational Intelligence*, pp. 37 – 44, 2002.

[3] Alvarez-Benitez J E, Everson R M, Fieldsend J E, "A MOPSO algorithm based exclusively on pareto dominance concepts", *Evolutionary Multi-Criterion Optimization, Springer Berlin Heidelberg*, 2005.

[4] Mostaghim S, Teich J, "Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO)", *Proceedings of Swarm Intelligence Symposium*, pp., 26-33, 2003.

[5] Raquel C R, Naval Jr P C, "An effective use of crowding distance in multiobjective particle swarm optimization", *Proceedings of the 2005 conference on Genetic and evolutionary computation*, 2005.

[6] Coello C A C, Pulido G T, Lechuga M S, "Handling multiple objectives with particle swarm optimization",. *Evolutionary Computation, IEEE Transactions on*, 2004, 8(3): 256-279, 2004.

[7] Schott J R, "Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization", *AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH*, 1995.

[8] Van Veldhuizen D A, "Multiobjective evolutionary algorithms: classifications, analyses, and new innovations", *AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH SCHOOL OF ENGINEERING*, 1999.

[9] Kursawe F, "A variant of evolution strategies for vector optimization", *Parallel Problem Solving from Nature, Springer Berlin Heidelberg*,193-197, 1991

[10] Zitzler E, Deb K, Thiele L, "Comparison of multiobjective evolutionary algorithms: Empirical results", *Evolutionary computation*, 2000.

[11] Deb K, "Multi-objective optimization", *Multi-objective optimization using evolutionary algorithms*, 2001: 13-46.

[12] Moore J, Chapman R, "Application of particle swarm to multiobjective optimization ", *Department of Computer Science and Software Engineering, Auburn University*, 1999.

[13] Parsopoulos K E, Vrahatis M N, "Particle swarm optimization method in multiobjective problems", *Proceedings of the 2002 ACM symposium on Applied computing*, 603-607, 2002.

[14] Coello Coello C A, Lechuga M S, "MOPSO: A proposal for multiple objective particle swarm optimization" *Proceedings of Evolutionary Computation*, 1051-1056, 2002.

[15] Deb K, Pratap A, Agarwal S, et al., "A fast and elitist multiobjective genetic algorithm: NSGA-II", *Evolutionary Computation, IEEE Transactions on*, 6(2): 182-197, 2002.

[16] Reyes-Sierra M, Coello CAC. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. International Journal of Computational Intelligence Research, 2(3): 287-308, 2006.

[17] Durillo J J, García-Nieto J, Nebro A J, et al., "Multi-objective particle swarm optimizers: An experimental comparison", *Evolutionary Multi-Criterion Optimization. Springer Berlin Heidelberg*, 495-509, 2009.

[18] Castro O R, Britto A, Pozo A, "A comparison of methods for leader selection in many-objective problems" *Evolutionary Computation (CEC), IEEE Congress on*, 1-8, 2012.

[19] Zitzler E, "Evolutionary algorithms for multiobjective optimization: Methods and applications", *Ithaca: Shaker*, 1999.

[20] Viennet R, Fonteix C, Marc I, "Multicriteria optimization using a genetic algorithm for determining a Pareto set", *International Journal of Systems Science*, 27(2): 255-260, 1996.

[21] F. T. S. Chan and M. K. Tiwari, "Swarm Intelligence: Focus on Ant and Particle Swarm Optimization", pp. 532, Itech Education and Publishing, Australia, December, 2007.

[22] Padhye, Nikhil, Juergen Branke, and Sanaz Mostaghim. "Empirical comparison of mopso methods-guide selection and diversity preservation." *Evolutionary Computation, 2009. CEC'09. IEEE Congress on. IEEE*, 2009.