

# Reference Point-Based Particle Swarm Optimization Using a Steady-State Approach

Richard Allmendinger<sup>1</sup>, Xiaodong Li<sup>2</sup>, and Jürgen Branke<sup>1</sup>

<sup>1</sup> University of Karlsruhe, Institute AIFB, Karlsruhe, Germany

<sup>2</sup> RMIT University, School of Computer Science and Information Technology,  
Melbourne, Australia

**Abstract.** Conventional multi-objective Particle Swarm Optimization (PSO) algorithms aim to find a representative set of Pareto-optimal solutions from which the user may choose preferred solutions. For this purpose, most multi-objective PSO algorithms employ computationally expensive comparison procedures such as non-dominated sorting. We propose a PSO algorithm, Reference point-based PSO using a Steady-State approach (RPSO-SS), that finds a preferred set of solutions near user-provided reference points, instead of the entire set of Pareto-optimal solutions. RPSO-SS uses simple replacement strategies within a steady-state environment. The efficacy of RPSO-SS in finding desired regions of solutions is illustrated using some well-known two and three-objective test problems.

## 1 Introduction

The use of Particle Swarm Optimization (PSO) in multi-objective problem solving has become more popular in recent years [1,2,3]. The aim of such multi-objective optimization techniques is to obtain a set of trade-off solutions with respect to multiple conflicting objectives. However, using Evolutionary Multi-Objective Optimization (EMOO) approaches to find the entire set of optimal solutions on problems that have a large number of objectives may lead to a much higher computational cost [4,5]. This is especially true for the widely used NSGA-II [6], which uses a non-dominated sorting procedure that has a complexity of  $O(N^2)$  ( $N$  is the population size). Since a decision maker (DM) is perhaps only interested in few (rather than all) Pareto-optimal solutions, much computational cost can be saved by focusing search on the preferred region of the Pareto-front. For this purpose, preference-based approaches such as the goal attainment method [7], the  $\varepsilon$ -constraint method [8] or the reference point-based approach [9] are drawn upon. The disadvantage of using these classical problem solving methods is that they are designed to find in each run only one single optimal solution instead of several. Given that the DM is interested in more than one solution to choose from, the classical approaches must be enhanced.

A solution to this problem would be to use population-based approaches such as multi-objective Evolutionary Algorithms (EAs) and PSO algorithms such as the Non-dominated Sorting PSO (NSPSO) [2]. In NSPSO, each particle needs

to choose a leader, which is done by finding the first non-dominated front, using the non-dominated sorting procedure, and choosing a particle randomly from the least crowded area of the first non-dominated front. This approach can be computationally expensive, especially for a large number of objectives, which is not necessarily the result of a high number of fitness evaluations but the result of the computational overhead added by other operations required by the optimization algorithm, such as the non-dominated sorting procedure or niching calculations. In this study, we use a steady-state approach that employs a simple replacement strategy, which avoids most of the computational overload caused by operations other than fitness evaluations. This method has been shown to be beneficial for EAs [10]. Although many multi-objective EAs exist in the literature, very few make use of preference-based search [11]. We propose a multi-objective PSO algorithm that employs a preference-based method in order to find a set of preferred Pareto-optimal solutions near user-provided reference points. Our approach, which we call Reference point based PSO using a Steady-State approach (RPSO-SS), achieves this aim in an effective and simple manner.

In Section 2 we introduce general background, including preference-based approaches, PSO and the steady-state approach. Section 3 describes the proposed algorithm, followed by numerical results and analysis in Section 4. Section 5 provides the concluding remarks.

## 2 Background

In this section we describe the main components used for the proposed algorithm.

### 2.1 Particle Swarm Optimization

PSO was developed by Kennedy and Eberhart [12], inspired by social behavior of bird flocking. PSO algorithms are simple and effective population-based stochastic techniques, which have their origin in the single objective optimization, but recently gained more popularity in the field of MOO. While several variants of multi-objective PSO have been developed [2,3,13,14], the basic algorithm is the same. A population of particles is initialized in an  $n$ -dimensional search space in which each particle  $\mathbf{x}_i = (x_{i,1}, \dots, x_{i,n})$  represents a (possible) solution. At any time, each particle is aware of its current position, its own personal best position, its current velocity and the single global (or local) best position. The global (or local) best position is represented as  $\mathbf{p}_g = (p_{g,1}, \dots, p_{g,n})$ , and symbolizes either the best position of all particles in the population (global best) or the best position within a local neighborhood (local best) of the particle. The personal best position represents the best position found by a particle so far and is denoted as  $\mathbf{p}_i = (p_{i,1}, \dots, p_{i,n})$ . The velocity  $\mathbf{v}_i = (v_{i,1}, \dots, v_{i,n})$  gives the position change of a particle. Equation (1) is used to calculate the new velocity for each particle. To update the new position of each particle, Equation (2) is used:

$$v_{i,j} \leftarrow \chi \cdot (v_{i,j} + r_1 \cdot (p_{i,j} - x_{i,j}) + r_2 \cdot (p_{g,j} - x_{i,j})) \quad (1)$$

$$x_{i,j} \leftarrow x_{i,j} + v_{i,j} \quad (2)$$

where  $j = 1, \dots, n$  and  $i = 1, \dots, N$ , with  $N$  as population size.  $r_1$  and  $r_2$  are two random numbers, chosen from the interval  $[0, \frac{\varphi}{2}]$ .  $\varphi$  is a positive constant and equal to 4.1 [1].  $\chi$  is a constriction factor; used to control and restrict the velocity's magnitude. This, suggested by Clerc and Kennedy, PSO variant is called *Constriction Type 1'* and is often used with  $\chi = 0.7298$ , calculated according to  $\chi = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$  [1]. To reduce the likelihood of particles leaving the search space, a particle's velocity can be constricted by setting the maximum velocity  $V_{max}$  to the upper and lower bounds of the decision variable ranges.

## 2.2 Reference Point-Based Approach

Reference point-based approaches belong to the family of preference-based approaches, which consider user information in order to concentrate the search on a preferred region of the search space [8]. One of the first steps of using higher-level information to navigate the search was proposed by Kuhn and Tucker [15]. Wierzbicki was the first who used user information in the form of reference points [9]. His approach aimed to find optimal solutions near a user-provided reference point  $\bar{\mathbf{z}} = (\bar{z}_1, \dots, \bar{z}_m)$  by solving an achievement scalarizing function  $s(f(\mathbf{x}))$ :

$$\text{minimize } s(f(\mathbf{x})) = \max_{i=1, \dots, m} [w_i \cdot (f_i(\mathbf{x}) - \bar{z}_i)] \quad (3)$$

where a user-specified positive weighting vector  $\mathbf{w} = (w_1, \dots, w_m)$  is used to scalarize the individual objectives  $f_i$ ,  $i = 1, \dots, m$ . The drawback of these proposals is that they form a single optimization problem, which matches only the DM's specific preferences. This leads to a minimization problem that has to be repeated with a new reference point and/or weighting vector if the user is dissatisfied with the obtained solution. This issue can be easily overcome by using the notion of dominance as usually used in an EMOO algorithm.

## 2.3 Steady-State Approach

In a steady-state EA an offspring is generated one at a time. A replacement strategy is often employed to compare the offspring with its parents. The offspring only replaces a weaker parent. Note that this procedure results in a population size that is constant during the entire run of the algorithm. There is no notion of generation.

Mumford introduced a Simple EA for Multi-objective Optimization (SEAMO) that uses the steady-state approach. SEAMO was at first designed for the NP-hard, 0-1 multiple knapsack problem, but modified for continuous functions [10,16]. SEAMO uses a steady-state population and applies a simple elitist replacement strategy that relies on comparisons between parents and their generated offspring. More precisely, an offspring for each population member is sequentially produced by applying crossover and mutation with a randomly selected second parent. If the offspring dominates one of the parents, then the

offspring replaces the parent. In order to encourage diversity in the population, *duplicate* individuals are eliminated in SEAMO. First, the offspring is compared with every individual in the population. If the offspring is a *duplicate* to another individual in the population then it does not replace any other individual. Two individuals,  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , are deemed to be *duplicates* if  $x_{1,j}-\epsilon \leq x_{2,j} \leq x_{1,j}+\epsilon$  applies to all objective function values  $x_{1,j}$  and  $x_{2,j}$  of  $\mathbf{x}_1$  and  $\mathbf{x}_2$ , where  $\epsilon$  is an error term.

### 3 RPSO-SS

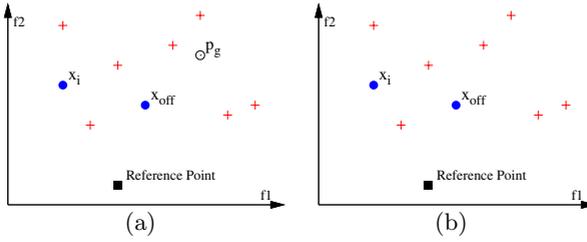
The algorithm proposed here is a multi-objective PSO variant that is able to locate Pareto-optimal solutions closest to user-provided reference points. This is achieved without employing the non-dominated sorting procedure, instead we use simple replacement strategies in a steady-state environment. This section presents the basic Reference point based PSO using a Steady-State approach (RPSO-SS) (see Section 3.1) and another variant of RPSO-SS, which uses an extended replacement strategy and a sampling method to choose a leader  $\mathbf{p}_g$  (see Section 3.2). We first outline the proposed steady-state approach in PSO, which slightly differs from the steady-state approach for EAs. In RPSO-SS, instead of using two parents, the new position of the current particle  $\mathbf{x}_i$  is generated based on a randomly selected  $\mathbf{p}_g$ ; we select  $\mathbf{p}_g$  randomly in order to maintain diversity. The new position is analogous to an offspring in conventional EAs. The replacement strategy applied to the three particles ( $\mathbf{x}_i$ ,  $\mathbf{p}_g$  and the offspring) is based on pairwise comparisons and will keep only two particles in the population so that the population size remains constant. This replacement strategy will provide the necessary selection pressure to drive the population moving towards the Pareto-optimal front. The same process is repeated for the entire population.

#### 3.1 Basic RPSO-SS

**Initialization.** The DM is asked to provide his or her preferences. This includes one or more reference points and the desired spread of solutions, specified by the error term  $\epsilon$ . The swarm is initialized with a population size that is constant over the entire run of the algorithm. The velocity  $\mathbf{v}_i$  is set randomly to be within the variable ranges. The population is divided into equal sized clusters, each focused on one reference point. The following main loop is applied for each cluster separately.

**Main Loop.** Do the following for each particle  $\mathbf{x}_i$  in each cluster until a stopping criterion is met:

1. Choose a particle randomly from the current cluster as the  $\mathbf{p}_g$ .
2. Produce an offspring based on  $\mathbf{x}_i$ , (the personal best position)  $\mathbf{p}_i$  and  $\mathbf{p}_g$  by applying Equations (1) and (2).
3. (a) If the offspring dominates  $\mathbf{x}_i$ , then it replaces  $\mathbf{x}_i$ ,  
 (b) else if the offspring dominates  $\mathbf{p}_g$ , then it replaces  $\mathbf{p}_g$  (see Fig. 1),



**Fig. 1.** An example of main loop 3.(b): a) An offspring  $x_{off}$  is first generated. b) Since  $x_{off}$  dominates particle  $p_g$  (and not  $x_i$ ),  $x_{off}$  replaces particle  $p_g$ . Note that the population size remains constant during the entire procedure.

- (c) else if the offspring is non-dominated to  $x_i$  and to  $p_g$ , then the two particles closest to the reference point are kept, and the distant particle is deleted.

Our experience shows that the steady-state approach works well on locating the entire front without using non-dominated sorting. However, when reference points are considered, the population tends to converge to a single non-dominated solution that is closest to the reference point. To prevent this, an error term  $\epsilon$  is introduced to control the spread of the solutions. This can be done by eliminating duplicate particles using the same procedure as used in SEAMO (see Section 2.3). Using a larger error term will lead to a more scattered distribution of particles, whereas a smaller value will result in a more concentrated search.

Moreover, to further involve the user in the search process we ask the DM to reset a reference point that is dominated by an offspring that is kept in the population. The reference point can also be reset automatically. We apply following formula:  $\bar{z}_i = \bar{z}_{i, current} - |\bar{z}_{i, original} \cdot step\_width|$ , in which  $i = 1, \dots, m$ .  $\bar{z}_i$  is the new component value,  $\bar{z}_{i, current}$  is the current component value and  $\bar{z}_{i, original}$  the original aspiration level of objective  $i$  of reference point  $\bar{z}$ ;  $step\_width$  specifies how far the reference point will be reset with respect to the original provided reference point  $\bar{z}_{i, original}$ . Note that this procedure leads to a new reference point that always dominates the old one.

### 3.2 An Extended Replacement Strategy with Sampling

The replacement strategy used in the basic RPSO-SS won't keep an offspring in the population that has a larger Euclidean distance than  $x_i$  and  $p_g$  to the reference point (see Section 3.1, Step 3.(c) of the main loop). Perhaps it would make more sense if such an offspring were allowed to replace another particle  $x_r$ . Replacement strategy with sampling extends the replacement strategy of the basic RPSO-SS and provides the offspring with the opportunity to replace particles other than  $x_i$  or  $p_g$ . Here, a randomly selected particle  $x_r$  will be replaced if it is either dominated by the offspring, or if it is non-dominated to the offspring and has a larger Euclidean distance than the offspring to the reference point. If

no such  $\mathbf{x}_r$  can be found in the population, then the offspring won't replace any particle.

Instead of using a single randomly selected particle as a  $\mathbf{p}_g$ , we can sample several particles randomly. Among these particles, the one with the shortest Euclidean distance to the reference point is chosen as the  $\mathbf{p}_g$ . In the special case where the sample size is equal to one, the single particle is always chosen as  $\mathbf{p}_g$ . Obviously, a larger sample size would increase the chance of finding a better  $\mathbf{p}_g$ , at the cost of higher computational effort. In our study here, we investigate the performance of two different sample sizes: size = 1 and size = 5.

## 4 Experiments

For validation purposes, the basic RPSO-SS and the extended replacement strategy with sample size = 1 (denoted by RS-S1) and size = 5 (denoted by RS-S5) were examined on some well-known test problems. ZDT1, ZDT2, ZDT3 and ZDT4 [17] were chosen to investigate the performance on two-objective problems. For the three-objective problems, DTLZ2 and DTLZ6 [18] were chosen. For the ZDT functions, we used a population size of 200 and terminated the simulation by 600 iterations. The DTLZ functions were run for 800 iterations using a population size of 400. For all experiments the reference points can be reset with a *step\_width* = 0.25 if they are in the feasible space.

To measure the performance we computed the normalized hypervolume (NH) of the last iteration for RPSO-SS, RS-S1 and RS-S5. NH is obtained by evaluating the hypervolume [19] of a normalized set of non-dominated solutions. In order to normalize a solution set an *ideal* and a *nadir point* are required. The nadir point and the ideal point are defined by the maximal and minimal objective function values, respectively, of the union of all non-dominated sets obtained across all runs of all comparing variants (RPSO-SS, RS-S1 and RS-S5). Our approaches were run for 50 separate runs. In order to provide fair comparisons of the NH value, the same nadir and ideal point are used across the solution sets obtained for a particular problem. The aim is to maximize the NH. The advantage of the NH metric is that the diversity and the closeness of the found solutions can be expressed by just one numerical value, and that a known Pareto-optimal set is not required.

### 4.1 Results and Discussion

The procedure for obtaining preferred Pareto-optimal solutions closest to the reference points has a complexity of  $O(N)$ . Found solutions using this procedure for ZDT1 are shown in Fig. 2 (indicated with  $\epsilon = 0.0x_i$ ) where two reference points ((0.2, 0.4) and (0.6, 0.5)) are chosen. If we adopt the procedure for checking duplicates using the  $\epsilon$ -parameter (see Section 3.1) we will be able to control the spread of the solutions. However, this procedure has a complexity of  $O(N^2)$ . Fig. 2 also shows the spreads of solutions using three different  $\epsilon$  values of  $0.005x_i$ ,  $0.01x_i$  and  $0.03x_i$ . Solutions with  $\epsilon = 0.0x_i$  are shown on the true Pareto-optimal

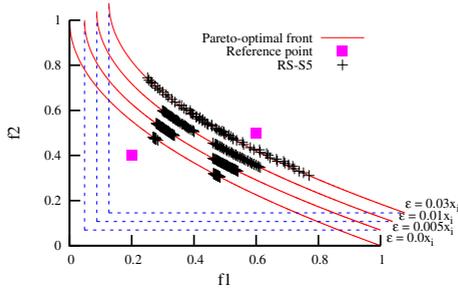


Fig. 2. Effect of different  $\epsilon$  values on ZDT1

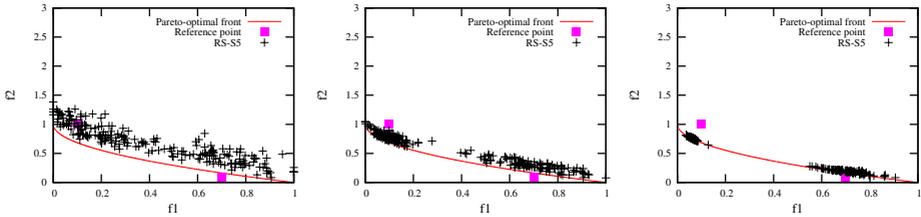


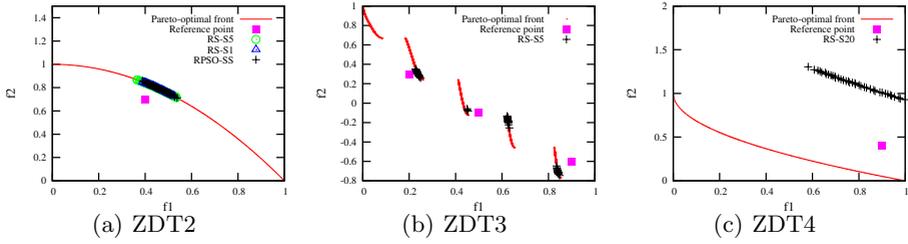
Fig. 3. Snapshots of RS-S5 on ZDT1 for iteration 10, 25 and 350 (from left to right)

front. Solutions with other  $\epsilon$  values are shown with an offset to the true Pareto-optimal front. For all following experiments we set the  $\epsilon$  at  $0.005x_i$  for every component of the objective variable  $\mathbf{x}$ . Fig. 3 shows some snapshots of a single run of RS-S5 on ZDT1. It can be seen that after just a few iterations two clusters have been formed around the provided reference points ((0.1, 1.0) and (0.7, 0.1)).

Next we consider ZDT2, which has a smooth concave front. As shown in Fig. 4a, one reference point (0.4, 0.7) is chosen. It can be seen that the Pareto-optimal fronts of all compared variants have a similar distribution of non-dominated solutions. The slightly larger spread of RS-S5 was observed for all investigated problems.

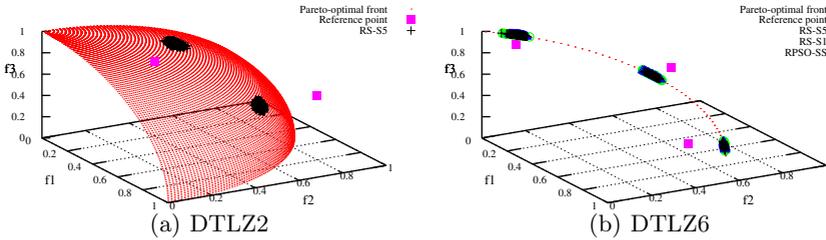
On ZDT3, which has a discontinuous Pareto-front, none of the strategies had any difficulties finding efficient preferred solutions. As shown in Fig. 4b, three reference points are chosen ((0.2, 0.3), (0.5, -0.1) and (0.9, -0.6)). It can also be seen that for the reference point, which is lying between two disconnected sections, efficient solutions on both sections have been found.

For the highly multi-modal problem ZDT4, we chose one reference point (0.9, 0.4). It seems to be a challenging problem to existing multi-objective PSO algorithms, including NSPSO, which tends to get stuck on one of the many local Pareto-fronts. None of our variants performed well on this problem. However, when sample size is increased to 20 (RS-S20), the variant was able to find the 2nd best front with most particles lying close to the reference point (see Fig. 4c). This reveals that a larger sample size significantly affects the outcome.



**Fig. 4.** Non-dominated solutions found on ZDT2, ZDT3 and ZDT4. On ZDT3 and ZDT4 only the results obtained by RS-S5 and RS-S20 are shown respectively.

Test problem DTLZ2 and DTLZ6 have a non-convex frontier and a Pareto-optimal front that is a curve, respectively. We use two reference points  $((0.2, 0.4, 0.7)$  and  $(0.8, 0.8, 0.6))$  for test function DTLZ2 (see Fig. 5a), and three  $((0.1, 0.1, 0.9)$ ,  $(0.55, 0.55, 0.8)$  and  $(0.6, 0.6, 0.1))$  for DTLZ6 (see Fig. 5b). Two and three well-separated sets of solutions without outliers are discovered for DTLZ2 and DTLZ6, respectively. From Fig. 5b it is apparent what was already indicated in Fig. 4a for test problem ZDT2, there is a larger spread of solutions found by RS-S5. The distribution of solutions found by RPSO-SS and RS-S1 are very similar.



**Fig. 5.** a) Non-dominated solutions found by RS-S5 on DTLZ2. b) Non-dominated solutions found by RPSO-SS, RS-S1 and RS-S5 on DTLZ6.

To confirm the visual results we calculated the NH and reported the mean and the standard deviation among 50 runs for all test problems, except ZDT4. In addition, we calculated the NH for the ZDT and DTLZ test problems using a population size of 100 and 200, respectively. Note that the same ideal and nadir point are used across the solution sets obtained for both of the population sizes on a particular problem. From the summarized results of Table 1 it can be seen that in terms of the NH, RS-S5 has the largest NH values for all test problems, whereas RPSO-SS and RS-S1 have similar values. This conforms the spread distributions observed in Fig. 4a for test problem ZDT2 and in Fig. 5b for DTLZ6. Furthermore, although the NH values are very similar for a particular problem with respect to the population sizes, there is a strong tendency that the NH increases with the population size. We attribute the superior result of RS-S5

**Table 1.** Mean and variance values of the normalized hypervolume for population size of 100 and 200 on ZDT test problems, and 200 and 400 for DTLZ test problems (bold indicates the best results obtained on a problem for a particular population size)

Population Size	Problem	RPSO-SS		RS-S1		RS-S5	
		Mean	STD	Mean	STD	Mean	STD
100	ZDT1	0.2818	0.0042	0.2821	0.0128	<b>0.3178</b>	0.0683
	ZDT2	0.4187	0.1530	0.4035	0.1248	<b>0.5228</b>	0.1750
	ZDT3	0.5585	0.0063	0.5576	0.0015	<b>0.5635</b>	0.0174
200	ZDT1	0.3133	0.0131	0.3074	0.0036	<b>0.3321</b>	0.0356
	ZDT2	0.3921	0.0456	0.3843	0.0213	<b>0.4630</b>	0.0582
	ZDT3	0.5688	0.0176	0.5643	0.0010	<b>0.5727</b>	0.0140
200	DTLZ2	0.3550	0.0225	0.3593	0.0362	<b>0.4311</b>	0.0566
	DTLZ6	0.2701	0.0087	0.2667	0.0015	<b>0.2901</b>	0.0582
	DTLZ7	0.2184	0.0353	0.2098	0.0273	<b>0.2537</b>	0.0882
400	DTLZ2	0.3604	0.0303	0.3488	0.0217	<b>0.4305</b>	0.0740
	DTLZ6	0.2777	0.0130	0.2730	0.0011	<b>0.2999</b>	0.0688
	DTLZ7	0.2222	0.0541	0.2132	0.0344	<b>0.2929</b>	0.1012

to the fact that the sampling method provides a more reliable way of choosing a leader. This results in a greater selection pressure, whereas that leads to a better convergence towards the reference point.

## 5 Conclusions

We have proposed a multi-objective PSO algorithm that finds a set of Pareto-optimal solutions near user-provided reference points, using a steady-state approach. We have suggested two different variants, the basic RPSO-SS and a variant that uses an extended replacement strategy and a sampling method to choose a leader  $p_g$ . Both variants manage to find preferred solutions without using the non-dominated sorting procedure or niching calculations, with the variant using a sample size greater than one having the best convergence. In order to avoid duplicates in the population, but above all to control the extent of the spread of obtained solutions, a parameter  $\epsilon$  is incorporated. This parameter specifies how close two population members can be located in the objective space without being deemed to be duplicates. This parameter can be used in conjunction with a method to automatically reset the reference point (or manually by the decision maker), which may further improve the performance of RPSO-SS. Our results on some well-known test problems demonstrated that the proposed approach is able to discover preferred areas of the Pareto-front. Work is in progress on improving the performance on multi-modal functions, such as ZDT4. The development of more efficient replacement and leader selection strategies is another aim that we are pursuing.

## References

1. Clerc, M., Kennedy, J.: The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation* 6(1), 58–73 (2002)

2. Li, X.: A non-dominated sorting particle swarm optimizer for multiobjective optimization. In: Proceedings of Genetic and Evolutionary Computation Conference 2003 (GECCO 2003), pp. 37–48 (2003)
3. Li, X.: Better spread and convergence: Particle swarm multiobjective optimization using the maximin fitness function. In: Deb, K., et al. (eds.) GECCO 2004. LNCS, vol. 3103, pp. 117–128. Springer, Heidelberg (2004)
4. Deb, K., Chaudhurid, S., Miettinen, K.: Estimating nadir objective vector quickly using evolutionary approaches. Technical Report No. 2005009, KanGAL, Kanpur India (2005)
5. Corne, D.W., Knowles, J.D.: Techniques for highly multiobjective optimisation: Some nondominated points are better than others. In: Proceedings of Genetic and Evolutionary Computation Conference 2007 (GECCO 2007), pp. 773–780 (2007)
6. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6(2), 182–197 (2002)
7. Gembicki, F.W.: Vector optimization for control with performance and parameter sensitivity indices. PhD thesis, Case Western Reserve University, Ohio, USA (1974)
8. Chankong, V., Haimes, Y.Y.: *Multiobjective Decision Making Theory and Methodology*. Elsevier Science Publishing Co., Amsterdam (1983)
9. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimisation. *Multiple Criteria Decision Making Theory and Applications*, 468–486 (1980)
10. Mumford-Valenzuela, C.L.: A simple approach to evolutionary multiobjective optimization. *Evolutionary Computation Based Multi-Criteria Optimization: Theoretical Advances and Applications*, 55–79 (2004)
11. Coello Coello, C.A.: Handling preferences in evolutionary multiobjective optimization: A survey. In: Proceedings of the 2000 Congress on Evolutionary Computation (CEC 2000), pp. 30–37 (2000)
12. Eberhart, R., Kennedy, J.: A new optimizer using particle swarm theory. In: Proceedings of the Sixth International Symposium on Micro Machine and Human Science (MHS 1995), pp. 39–43 (1995)
13. Parsopoulos, K.E., Vrahatis, M.N.: Particle swarm optimization method in multiobjective problems. In: Proceedings of the 2002 ACM Symposium on Applied Computing (SAC 2002), pp. 603–607 (2002)
14. Sierra, M.R., Coello Coello, C.A.: Improving PSO-based multi-objective optimization using crowding, mutation and  $\epsilon$ -dominance. In: Proceedings of the Third Evolutionary Multi-Criterion Optimization Conference (EMO 2005), pp. 505–519 (2005)
15. Kuhn, H.W., Tucker, A.W.: Nonlinear programming. In: Proceedings of Second Berkeley Symposium on Mathematical Statistics and Probability, pp. 481–492 (1950)
16. Mumford-Valenzuela, C.L.: Simple population replacement strategies for a steady-state multi-objective evolutionary algorithm. In: Proceedings of Genetic and Evolutionary Computation Conference 2004 (GECCO 2004), pp. 1389–1400 (2004)
17. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation* 8(2), 173–195 (2002)
18. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. Technical Report No. 112, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), Zurich, Switzerland (2001)
19. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach. *IEEE Transactions on Evolutionary Computation* 3(4), 257–271 (1999)