Choosing Leaders for Multi-objective PSO Algorithms Using Differential Evolution

W.R.M.U.K. Wickramasinghe and X. Li

School of Computer Science and Information Technology
RMIT University, Melbourne, VIC 3001, Australia
{uwickram,xiaodong}@cs.rmit.edu.au
http://www.cs.rmit.edu.au/

Abstract. The fast convergence of particle swarm algorithms can become a downside in multi-objective optimization problems when there are many local optimal fronts. In such a situation a multi-objective particle swarm algorithm may get stuck to a local Pareto optimal front. In this paper we propose a new approach in selecting leaders for the particles to follow, which in-turn will guide the algorithm towards the Pareto optimal front. The proposed algorithm uses a Differential Evolution operator to create the leaders. These leaders can successfully guide the other particles towards the Pareto optimal front for various types of test problems. This simple yet robust algorithm is effective compared with existing multi-objective particle swarm algorithms.

Keywords: Hybrid, Particle Swarm, Differential Evolution, Multi-objective optimization, Multi-modal problems.

1 Introduction

Particle Swarm Optimization (PSO) has been studied to be an effective Evolutionary Multi-Objective (EMO) algorithm [1]. The main advantage of PSO is the fast convergence [2]. PSO algorithms operate by the notion of *following a leader* to scan the search-space. This movement of following certain particles of a population can become a disadvantage in problems where there are many local optimal fronts. The multi-objective PSO algorithms can prematurely converge onto a local front rather than finding the optimal global front [3, 4]. In our research we developed a mechanism to avoid this downside of multi-objective PSO algorithms. We propose a hybrid PSO algorithm which uses a Differential Evolution (DE) [5] operator to *generate* leaders. This mechanism of generating particles offers the ability to obtain a diverse range of leaders some of which maybe outside a local optimal front. This becomes an advantage to the multi-objective PSO algorithm because now particles are attracted towards these leaders increasing the likelihood of moving the population out of a local optimal front. The proposed hybrid algorithm is simple, robust and efficient in solving a variety of multi-objective problems including problems with many local optimal fronts.

X. Li et al. (Eds.): SEAL 2008, LNCS 5361, pp. 249–258, 2008.

This paper is organized as follows. Section 2 briefly describes the background material including the formal definitions of the PSO and DE variants used in this research. Section 3 presents related work carried out in the field. Section 4 presents our algorithm. The experiments used to evaluate the algorithm will be provided in section 5. Finally, in section 6, we present our conclusions and avenues for future research.

2 Background

We will first present the background material used in this research.

2.1 Particle Swarm Optimization

PSO is a nature-inspired Evolutionary Algorithm (EA) which mimics the behaviour of swarming bees, flocking birds or schooling fish [2]. These behaviours are modelled as rules governing the movement of particles in the search-space. Each particle moves in the search-space by adjusting its position and velocity, which are influenced by its interaction with other particles in its neighbourhood. The i^{th} particle's velocity and position at time t are updated to time $t+1$ according to the following two equations respectively:

$$\mathbf{v}_i(t+1) = \chi(\mathbf{v}_i(t) + \phi_1(\mathbf{p}_i - \mathbf{x}_i(t)) + \phi_2(\mathbf{p}_g - \mathbf{x}_i(t))) \quad (1)$$

$$\mathbf{x}_i(t+1) = \mathbf{x}_i(t) + \mathbf{v}_i(t) \quad (2)$$

This version of PSO is known as the *Constriction Type 1"* as defined in [6]. Here, $\mathbf{v}_i$ is the velocity of the i^{th} particle and $\mathbf{x}_i$ is its position. The variables ϕ_1 and ϕ_2 are random numbers generated uniformly between $[0, \frac{\varphi}{2}]$. Here, φ is a constant equal to 4.1 [6]. $\mathbf{p}_i$ is the best position found by the particle (also known as *personal best*); $\mathbf{p}_g$ is the best position found in the particle's neighbourhood (also known as *global best* or *leader*). χ is called the constriction factor, and is used to prevent a particle from exploring too far in the search-space. We used $\chi = 0.7298$, which was calculated from $\frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}|}$ [6]. Essentially each particle moves towards somewhere between its personal best and the global best.

2.2 Differential Evolution

DE differs from other EAs in their mechanism of generating offspring. In EAs, an individual plays the role of a parent to generate an offspring. This is similar in PSO where a particle updates its velocity to move to another position, which mimics the creation of an offspring, if the original particle was considered as a parent. In DE, an offspring is generated by using vector differences among individuals in the population [5].

There are many schemes of generating individuals in DE. Here, we present the simplest and most popular scheme, the **DE/rand/1/bin**, which is used in this

research. For an individual vector $\mathbf{x}_i$ a trial vector $\mathbf{u}_i$ is generated using three other individuals $\mathbf{x}_{r1}, \mathbf{x}_{r2}, \mathbf{x}_{r3}$ from the population such that $i \neq r1 \neq r2 \neq r3$. The decision variable j of an individual is generated using equation (3).

$$\mathbf{u}_i = u_{j,i} = \begin{cases} x_{j,r1} + F(x_{j,r2} - x_{j,r3}) & if \quad (rand_j < CR \quad or \quad j = j_{rand}) \\ x_{j,i} & otherwise \end{cases} \tag{3}$$

here, $j_{rand} \in [1, D]$ where D is the number of dimensions in the search-space. $F \in [0, 1]$ is called the scaling factor. CR is the crossover ratio and $rand_j$ is randomly generated uniformly between $[0, 1]$. The traditional values used for DE/rand/1/bin are $CR = 0.9$ and $F = 0.5$.

In a traditional multi-objective DE algorithm if $\mathbf{u}_i$ dominates $\mathbf{x}_i$ then $\mathbf{x}_i$ is replaced by $\mathbf{u}_i$, if not $\mathbf{u}_i$ is discarded [7]. In the proposed algorithm the trial vector $\mathbf{u}_i$ is chosen as the leader ($\mathbf{p}_g$) for the particle $\mathbf{x}_i$. The particle will update its velocity and positions according to equations (1) and (2) using this $\mathbf{p}_g$.

We observed that the traditional values for CR and F are unsuitable for multi-objective optimization problems from our initial experiments. A higher CR value allows less frequent crossover operations. This, contributes in placing leaders far from the vicinity of other particles in the search-space. These leaders could be placed in worst positions than the current particles. To overcome this issue we used $CR = 0.2$ and $F = 0.4$, which locates the leaders near the vicinity of other particles. A lower CR value is also present as desirable in other examples of multi-objective optimization algorithms using DE in the literature [7, 8, 9, 10].

3 Related Work

There are several examples of multi-objective PSO algorithms where leaders are chosen by the notion of *dominance* in the literature. In DNPSO [11], particles choose a leader from their neighbours. Here, a particle will have a set of dynamically changing neighbours and it will choose the best particle among them as a leader. NSPSO [3] adopts the non-dominated sorting mechanism proposed in NSGA-II [12]. In NSPSO, leaders are chosen from the first non-dominated front of the current population. Here, each particle chooses a leader randomly from the top 10% of least crowded particles of the first non-dominated front. In a later study maximinPSO [4] was proposed, which used the maximin fitness function [13] to derive dominance. For every iteration a set of particles was extracted using the maximin fitness. From this set the top 10% of particles with the least maximin value was chosen as candidates to be leaders. Each particle will create a leader by selecting values from all dimensions from this set of particles. In maximinPSO, a particle's leader is not present in the population, but created from several candidates for leaders. The difference from the approach of creating leaders in maximinPSO and the proposed algorithm is that leaders can be positioned outside the first non-dominated front.

In other studies of multi-objective PSO algorithms, leaders were selected from an external archive. An elite archive is used in [14] to store the non-dominated individuals from every iteration. The archive has a special data structure called

a *dominance tree* which is updated with the non-dominated particles found in each run. The algorithms proposed in [15] and [16] both use an archive to store particles which are candidates for leaders. In [15], the non-dominant particles are added to the archive in each iteration. This archive is truncated to a fixed size according to the crowding distance of the particles. This ensures that the least crowded particles will be candidates for leaders. The *Sigma method* is used in [16] to obtain suitable particles as leaders in each iteration. These leaders are then stored in the archive in each iteration and truncated so that the particles with smallest sigma values are retained. Recently, in [17] a methodology of choosing a random particle from the population as a leader was proposed. A particle will adjust its velocity and position and then according to some replacement rules it will decide whether to replace this leader or another particle if it is in a dominant position. In all of these multi-objective PSO algorithms the leaders are extracted by members of the population. In contrast, in our proposed hybrid algorithm leaders are not necessarily particles of the current population.

It is useful to notice that there have been several hybrid single-objective DE and PSO algorithms proposed in the literature. A study in [18] proposes a PSO algorithm, where from time to time the DE operator is used to create an individual which will replace a particle than using the PSO rules to move it. DEPSO [19] uses the DE operator to updating the personal best position of particles. However, this updating procedure is done once every couple of iterations. A similar approach to [18] was proposed in [20] where the particles' movement is determined partly by a DE operator and partly by PSO update rules. In a recent study [21] the PSO update rules were modified by using a DE operator. Here, every particle underwent this modified PSO rules to move in the search-space. In all these algorithms the movement of the particle was influenced by a DE operator. In our proposed hybrid algorithm the particles move following the PSO rules, while only a leader is created using a DE operator. This mechanism ensures a faster convergence because of the PSO update rules, while retaining a multi-objective PSO's ability to move outside a local optimal front because of a diverse range of leaders generated by DE.

4 The MDEPSO Algorithm

The proposed MDEPSO algorithm has the following steps:

- **Step 1: Initialize the particles**
  A population of size N is first initialized. Here, a particle's decision variables are initialized from equation (4).

$$rand(0.0, 1.0) * (UB - LB) + LB \tag{4}$$

  here, $rand(0.0, 1.0)$ represents a random number generated uniformly between $[0.0, 1.0]$. LB and UB are the lower-bounds and upper-bounds respectively of the decision variables of a multi-objective problem instance. The velocity is initialized to a random value in the interval $[0, UB - LB]$. The

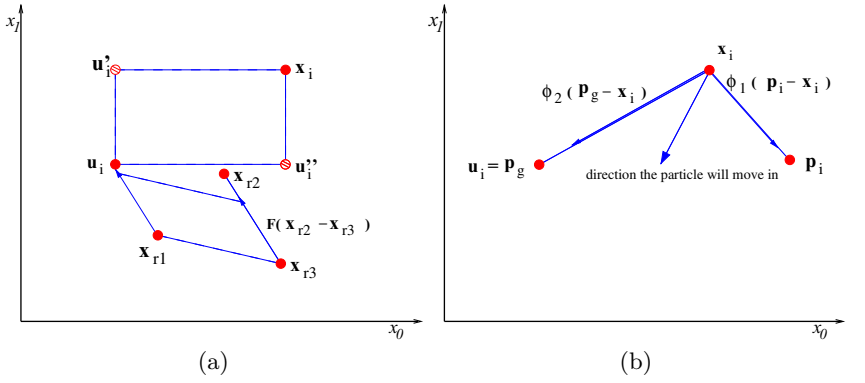


Fig. 1. (a) Creating a leader vector using the DE operator (b) Applying the PSO rules to move the particle

personal best of an individual is set to the current values of the decision variables. Half of the populations' direction is reversed by setting the velocity to negative according to a coin toss to better explore the search-space. The particles are then evaluated with the objective functions and fitness is assigned. The fitness values are used to determine the dominance.

– **Step 2: Generate leaders and move the particles**
Each particle $\mathbf{x}_i$ will choose three other particles $\mathbf{x}_{r1}, \mathbf{x}_{r2}, \mathbf{x}_{r3}$ from the population such that $i \neq r1 \neq r2 \neq r3$. Then a leader vector ($\mathbf{u}_i = \mathbf{p}_g$) is derived from the DE operator in equation (3). Each particle will move towards their generated leader vectors (figure 1) updating their velocities and positions according to the PSO rules in equations (1) and (2).

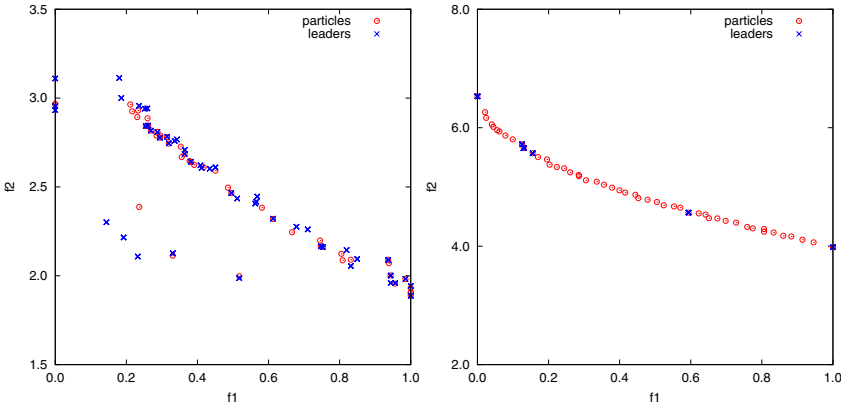
– **Step 3: Update the particles' personal bests**
The particles are evaluated according to the objective functions and fitness is assigned. Next, the particles' personal bests are updated according to their current positions and best positions found so far.

– **Step 4: Obtain the particles to move to the next iteration**
The population of N particles at the beginning of the iteration is combined with the N number of particles that have changed their positions to create a population of size $2N$. The non-dominated sorting process [12] is applied to this $2N$ population to obtain N particles which are carried over to the next iteration.

The steps 2 to 4 are repeated until the maximum number of iterations is reached.

Advantage of Generating Leaders Using DE

Figure 2 shows the generated leaders for the two-objective ZDT4 [22] test problem within an iteration. The DE operator takes the differences between vectors



(a) MDEPSO (b) NSPSO

Fig. 2. Positions of leaders for ZDT4 within an iteration for MDEPSO and NSPSO

to generate a leader. This becomes an advantage because there is a chance for particles not in the first non-dominated front to be selected in the process of creating a leader. The vector differences between a particle in the first non-dominated front and one in a different front can be large, that the generated leader will be in a position outside the first non-dominated front. These leaders can now successfully attract other particles to move from the current local optimal front towards the global optimal front. In other multi-objective PSO algorithms particles would rarely move outside a local optimal front because all leaders are trapped to the same front. This feature of generating leaders outside of the current local front, gives MDEPSO the ability to escape a local front.

Selecting leaders from the first non-dominated front can also be a disadvantage for multi-objective PSO algorithms mainly because many particles could follow the same leader. This restrains the PSO algorithm from searching the search-space effectively.

5 Experiments

We used the following problems; two-objective ZDT [22], three-objective DTLZ [23], and two-objective WFG [24] test suites. These test problem suites contain many varieties of multi-objective problems including multi-modal problems (many local optimal fronts with one global optimal front). We present the Normalized Hyper-Volume metric [25], which provides the convergence and spread of solutions along the solution front for comparing the effectiveness of MDEPSO against NSPSO [3], maximinPSO [4], OMOPSO [15] and NSGA-II [12] on the same test problems.

For each problem a constant population of 200 individuals were used. All the algorithms were executed for a maximum of 750 iterations on all test problems. For simpler problems like ZDT1-ZDT3, MDEPSO found the Pareto optimal

Table 1. Normalized Hyper-Volume for MDEPSO, NSPSO, maximinPSO, OMOPSO and NSGA-II on ZDT, DTLZ and WFG test problem suites

	MDEPSO	**NSPSO**	**maximinPSO**	**OMOPSO**	**NSGA-II**
ZDT4	*0.0000* ± *0.0000*	*0.0000* ± *0.0000*	*0.0000* ± *0.0000*	0.0022 ± 0.0114	**0.6636** ± **0.0001**
ZDT6	**0.4033** ± **0.0001**	**0.4033** ± **0.0001**	0.4027 ± 0.0002	0.4014 ± 0.0001	0.4025 ± 0.0002
WFG3	**0.4428** ± **0.00001**	0.4426 ± 0.0001	0.4413 ± 0.0006	0.4422 ± 0.0000	0.4428 ± 0.0001
WFG4	0.2175 ± 0.0014	0.2161 ± 0.0008	0.2163 ± 0.0016	0.2151 ± 0.0019	**0.2205** ± **0.0001**
WFG5	**0.1995** ± **0.0035**	0.1974 ± 0.0002	0.1975 ± 0.0001	0.1963 ± 0.0001	0.1973 ± 0.0001
DTLZ1	0.7749 ± 0.0001	*0.0000* ± *0.0000*	*0.0000* ± *0.0000*	*0.0000* ± *0.0000*	**0.7852** ± **0.0018**
DTLZ3	*0.0000* ± *0.0000*	*0.0000* ± *0.0000*	*0.0000* ± *0.0000*	*0.0000* ± *0.0000*	**0.4084** ± **0.0029**
DTLZ6	**0.0959** ± **0.0000**	0.0909 ± 0.0214	0.0956 ± 0.0001	0.0949 ± 0.0000	0.0957 ± 0.0000
DTLZ7	**0.3094** ± **0.0013**	0.1818 ± 0.1381	0.2827 ± 0.0553	0.2758 ± 0.0030	0.3045 ± 0.0013

fronts in under 50 iteration, but for consistency the maximum number of iterations were fixed. Each algorithm was executed for 50 runs on each test problem and the results were averaged. The parameter values for MDEPSO, NSPSO and maximinPSO were the standard values presented in Constriction Type 1" PSO. $CR = 0.2$ and $F = 0.4$ was used for the DE operator in MDEPSO. An archive size of 100 and mutation probability of 0.5 was used in OMOPSO. In NSGA-II, a crossover probability of 0.9 on a SBX crossover operator was used. The mutation probability was set as $1/D$, where D is the number of dimensions of the problem instance.

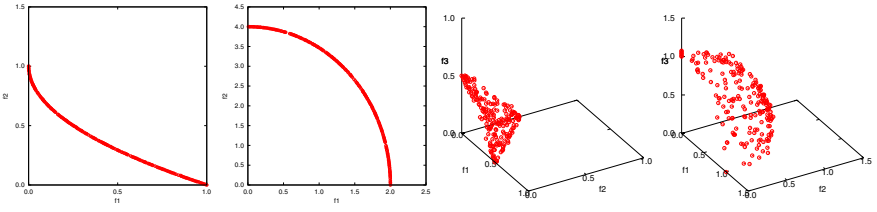
Table 1 shows the normalized hyper-volume metric of all the algorithms on the test problems used in this study. Here, we have only illustrated some of the test problems due to the limitation of space. MDEPSO on average outperforms all the other multi-objective PSO algorithms. However, for ZDT4 and DTLZ3, MDEPSO was unable to converge to the global Pareto front. We introduced a mutation mechanism for MDEPSO to overcome this.

The MDEPSO algorithm's Step 3 is updated with a mutation operator. A particle is mutated with a given probability using a PSO mutation rule. Choose a particle for mutation if probability is less than some δ. A particle chosen for mutation will update each decision variable j if the probability is less than δ. The decision variable is *mutated* according to equation (5). We used $\delta = 0.1$ as the mutation probability. Each particle will update its personal best after the mutation is applied.

$$x_{j,i} = (x_{j,i} + rand(LB, UB))/2 \tag{5}$$

Table 2. Normalized Hyper-Volume for MDEPSO, NSPSO and maximinPSO after the mutation operator was added compared with OMOPSO and NSGA-II on ZDT4, WFG4, DTLZ1 and DTLZ3

	MDEPSO	**NSPSO**	**maximinPSO**	**OMOPSO**	**NSGA-II**
ZDT4	0.6628 ± 0.0005	0.1331 ± 0.0053	0.2448 ± 0.0514	0.0022 ± 0.0114	**0.6636 ± 0.0001**
WFG4	0.2196 ± 0.0005	0.2187 ± 0.0008	0.2185 ± 0.0015	0.2151 ± 0.0019	**0.2205 ± 0.0001**
DTLZ1	0.7754 ± 0.0001	0.0840 ± 0.0059	0.6925 ± 0.0008	*0.0000* ± *0.0000*	**0.7852 ± 0.0018**
DTLZ3	0.3628 ± 0.0053	*0.0000* ± *0.0000*	0.0586 ± 0.0481	*0.0000* ± *0.0000*	**0.4084 ± 0.0029**



(a) ZDT4 (b) WFG4 (c) DTLZ1 (d) DTLZ3

Fig. 3. Solution fronts obtained by MDEPSO on ZDT4, WFG4, DTLZ1 and DTLZ3

We employed this same mutation rule on NSPSO and maximinPSO also. OMOPSO already has a mutation rule in its functionality. After 50 runs of the modified MDEPSO we observed that for all of the test problems MDEPSO was able to locate the global optimal front. Table 2 shows the normalized hyper-volume values after the mutation operator was added. We have listed only values for the multi-modal problems because for the other problems the normalized hyper-volume values were similar.

Comparing the results from Table 1 and 2 it is clear that MDEPSO outperforms the other multi-objective PSO algorithms in the test problems, especially for the multi-modal problems. One of the main objectives of this research was to find a method for multi-objective PSO algorithms to better handle problems with many local fronts. This was possible with the selection method of leaders in MDEPSO with mutation. On average MDEPSO gives a higher hyper-volume value, which is slightly better than NSGA-II in locating the Pareto optimal fronts on a range of different test problems. However, the crux of the research was not to outperform NSGA-II. We have included the results of NSGA-II to compare the performance between the other PSO algorithms and MDEPSO. MDEPSO is capable of locating the global Pareto front of all the test problems where other PSO algorithms were unable to.

Figure 3 shows the final non-dominated solution sets obtained by MDEPSO with mutation at the end of 750 iterations for the multi-modal problems.

6 Conclusion and Future Work

The proposed hybrid MDEPSO algorithm is effective in locating the Pareto optimal front in difficult multi-modal multi-objective problems than existing PSO algorithms. The unique feature of generating and selecting leaders provides a greater range of directions and positions particles can move to, including those positions outside a local optimal front. MDEPSO is a simple and robust EMO algorithm which has a comparable performance to NSGA-II.

It will be interesting to study if other combination mechanisms can be used to generate leaders other than DE. We also would like to investigate if this hybrid algorithm is capable of solving rotated multi-objective problems, where many state-of-art EMO algorithms are unsuccessful.

References

1. Reyes-Sierra, M., Coello Coello, C.A.: Multi-Objective Particle Swarm Optimizers: A Survey of the State-of-the-Art. International Journal of Computational Intelligence Research 2, 287–308 (2006)
2. Kennedy, J., Eberhart, R.C.: Swarm intelligence. Morgan Kaufmann Publishers Inc., San Francisco (2001)
3. Li, X.: A non-dominated sorting particle swarm optimizer for multiobjective optimization. In: Cantú-Paz, E., Foster, J.A., Deb, K., Davis, L., Roy, R., O'Reilly, U.-M., Beyer, H.-G., Kendall, G., Wilson, S.W., Harman, M., Wegener, J., Dasgupta, D., Potter, M.A., Schultz, A., Dowsland, K.A., Jonoska, N., Miller, J., Standish, R.K. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 37–48. Springer, Heidelberg (2003)
4. 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. 3102, pp. 117–128. Springer, Heidelberg (2004)
5. Price, K., Storn, R.M., Lampinen, J.A.: Differential Evolution: A Practical Approach to Global Optimization. Natural Computing Series. Springer, Secaucus (2005)
6. Clerc, M., Kennedy, J.: The particle swarm - explosion, stability, and convergence in a multidimensional complex space. IEEE Transactions on Evolutionary Computation 6, 58–73 (2002)
7. Abbass, H., Sarker, R., Newton, C.: PDE: a pareto-frontier differential evolution approach for multi-objective optimization problems. IEEE Congress on Evolutionary Computation (CEC) 2, 971–978 (2001)
8. Huang, V.L., Suganthan, P.N., Qin, A.K., Baskar, S.: Multiobjective differential evolution with external archive and harmonic distance-based diversity measure. In: School of Electrical and Electronic Engineering Nanyang, Technological University Technical Report (2005)
9. Kukkonen, S., Lampinen, J.: GDE3: the third evolution step of generalized differential evolution. In: IEEE Congress on Evolutionary Computation (CEC), pp. 443–450 (2005)
10. Robic, T., Filipic, B.: Demo: Differential evolution for multiobjective optimization. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 520–533. Springer, Heidelberg (2005)

11. Hu, X., Eberhart, R., Shi, Y.: Particle swarm with extended memory for multiobjective optimization. In: IEEE Swarm Intelligence Symposium (SIS), pp. 193–197 (2003)
12. Deb, K., Agrawal, S., Pratab, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6, 182–197 (2002)
13. Balling, R.: The maximin fitness function; multi-objective city and regional planning. In: Fonseca, C.M., Fleming, P.J., Zitzler, E., Deb, K., Thiele, L. (eds.) EMO 2003. LNCS, vol. 2632, pp. 1–15. Springer, Heidelberg (2003)
14. Fieldsend, J., Everson, R., Singh, S.: Using unconstrained elite archives for multiobjective optimization. IEEE Transactions on Evolutionary Computation 7, 305–323 (2003)
15. Reyes-Sierra, M., Coello Coello, C.A.: Improving pso-based multi-objective optimization using crowding, mutation and epsilon-dominance. In: Coello Coello, C.A., Hernández Aguirre, A., Zitzler, E. (eds.) EMO 2005. LNCS, vol. 3410, pp. 505–519. Springer, Heidelberg (2005)
16. Mostaghim, S., Teich, J.: Strategies for finding good local guides in multi-objective particle swarm optimization. In: IEEE Swarm Intelligence Symposium (SIS), pp. 26–33 (2003)
17. Allmendinger, R.: Reference point-based particle swarm optimization using a steady-state approach: Master's thesis (2008)
18. Hendtlass, T.: A combined swarm differential evolution algorithm for optimization problems. In: Monostori, L., Váncza, J., Ali, M. (eds.) IEA/AIE 2001. LNCS, vol. 2070, pp. 11–18. Springer, Heidelberg (2001)
19. Zhang, W.J., Xie, X.F.: DEPSO: Hybrid particle swarm with differential evolution operator. In: IEEE International Conference on Machine Learning and Cybernetics (ICMLC), pp. 3816–3821 (2003)
20. Hao, Z.F., Guo, G.H., Huang, H.: A particle swarm optimization algorithm with differential evolution. IEEE International Conference on Machine Learning and Cybernetics (ICMLC) 2, 1031–1035 (2007)
21. Xu, X., Li, Y., Fang, S., Wu, Y., Wang, F.: A novel differential evolution scheme combined with particle swarm intelligence. In: IEEE Congress on Evolutionary Computation, CEC (2008)
22. Zitzler, E., Deb, K., Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation 8, 173–195 (2000)
23. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable test problems for evolutionary multi-objective optimization. In: Evolutionary Multiobjective Optimization (EMO): Theoretical Advances and Applications, pp. 105–145. Springer, Heidelberg (2005)
24. Huband, S., Hingston, P., Barone, L., While, R.L.: A review of multiobjective test problems and a scalable test problem toolkit. IEEE Transactions on Evolutionary Computation 10, 477–506 (2006)
25. Zitzler, E., Thiele, L.: Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Transactions on Evolutionary Computation 3, 257–271 (1999)