Better Spread and Convergence: Particle Swarm Multiobjective Optimization Using the Maximin Fitness Function

Xiaodong Li

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

Abstract. Maximin strategy has its origin in game theory, but it can be adopted for effective multiobjective optimization. This paper proposes a particle swarm multiobjective optimiser, *maximinPSO*, which uses a fitness function derived from the maximin strategy to determine Pareto-domination. The maximin fitness function has some very desirable properties with regard to multiobjective optimization. One advantage is that no additional clustering or niching technique is needed, since the maximin fitness of a solution can tell us not only if a solution is dominated or not (with respect to the rest of the population), but also if it is clustered with other solutions, i.e., diversity information. This paper demonstrates that on the ZDT test function series, *maximinPSO* produces an almost perfect convergence and spread of solutions towards and along the Pareto-optimal front respectively, outperforming one of the state-of-art multiobjective EA algorithms, NSGA II, in all the performance measures used.

1 Introduction

Particle Swarm Optimization (PSO) has become increasingly popular as an efficient optimization method for single objective optimization, and more recently it has shown promising results for solving multiobjective optimization problems [3,4,5,6,7]. PSO is an optimization technique inspired by studies of the social behaviour of insects and animals [1][2]. The social behaviour is modelled in a PSO to guide a population of particles (or potential solutions) moving towards the most promising region of the search space. In PSO, each particle represents a candidate solution, $\mathbf{x_i} = (x_{i1}, x_{i2}, \ldots, x_{iD})$. D is the dimension of the search space. The i-th particle of the swarm population knows: **a**) its personal best position $\mathbf{p_i} = (p_{i1}, p_{i2}, \ldots, p_{iD})$, i.e., the best position this particle has visited so far that yields the highest fitness value; and **b**) the global best position, $\mathbf{p_g} = (p_{g1}, p_{g2}, \ldots, p_{gD})$, i.e., the position of the best particle that gives the best fitness value in the entire population; and **c**) its current velocity, $\mathbf{v_i} = (v_{i1}, v_{i2}, \ldots, v_{iD})$, which represents its position change. The following equation (1) uses the above information to calculate the new updated velocity

K. Deb et al. (Eds.): GECCO 2004, LNCS 3102, pp. 117–128, 2004.

for each particle in the next iteration step. Equation (2) updates each particle's position in the search space.

$$v_{id} = wv_{id} + c_1 r_1(p_{id} - x_{id}) + c_2 r_2(p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \ , \quad (2)$$

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$; N is the size of the swarm population; w is the inertia weight, which is often used as a parameter to control exploration/exploitation in the search space; c_1 and c_2 are two coefficients (positive constants); r_1 and r_2 are two random numbers within the range [0, 1]. There is also a V_{MAX}, which sets the upper and lower bound for velocity values.

Recently Balling in [8] proposed a very interesting multi-objective optimization technique based on fitness derived from using the maximin strategy [9]. In sharp contrast to almost all other existing multi-objective algorithms, Balling demonstrated that by using the maximin fitness, there is no need to use any additional niching technique, since using the maximin fitness by itself penalizes clustering of solutions.

In this paper, *maximinPSO*, a PSO model using the maximin fitness is proposed for multi-objective optimization. *maximinPSO* adopts a similar approach as NSPSO[7], except that it uses the maximin fitness to rank individuals in the population (rather than the non-dominated sorting procedure), and there is no niching method used. The paper is organized as follows: Section 2 first introduces the concept of dominance, from which the maximin fitness function is derived. Section 3 defines the maximin fitness function and describes its key properties in relation to the proposed *maximinPSO* algorithm. Section 4 introduces the *maximinPSO* algorithm formally. Section 5 presents test functions, performance measures, as well as the results and analysis of experiments carried out with the *maximinPSO* over the test functions. Finally Section 6 concludes the paper.

2 Multiobjective Optimization and the Notion of Dominance

Assuming minimization, multi-objective optimization strives to simultaneously minimize m objectives:

$$\textbf{Minimize } \mathbf{y} = f(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), \ldots, f_m(\mathbf{x})) \quad (3)$$

where $\mathbf{x}$ is a n-dimensional decision variable vector, $\mathbf{x} = (x_1, \ldots, x_n) \in X$, and $\mathbf{y} = (y_1, \ldots, y_n) \in Y$. X is the decision variable space, whereas Y is the objective space. Each objective depends on the decision vector $\mathbf{x}$. A decision vector $\mathbf{u} \in X$ is said to strictly dominate another decision vector $\mathbf{v} \in X$ (denoted by $\mathbf{u} \prec \mathbf{v}$) if and only if

$$\forall i \in \{1, \ldots, m\} : f_i(\mathbf{u}) \leq f_i(\mathbf{v}) \quad \text{and} \quad \exists j \in \{1, \ldots, m\} : f_j(\mathbf{u}) < f_j(\mathbf{v}) \tag{4}$$

$\mathbf{u}$ weakly dominates $\mathbf{v}$ (denoted by $\mathbf{u} \preceq \mathbf{v}$) if and only if

$$\forall i \in \{1, \ldots, m\} : f_i(\mathbf{u}) \leq f_i(\mathbf{v}) \tag{5}$$

A decision vector $\mathbf{x} \in X$ is said to be Pareto-optimal with respect to X if and only if there is no other decision vector in X that dominates $\mathbf{x}$.

The set of all Pareto-optimal solutions in the decision variable space is called the *Pareto-optimal* set. The corresponding set of objective vectors is called the *Pareto-optimal front*. In this paper, for clarity, we denote the Pareto-optimal front as P^*, and the set of non-dominated solutions found as Q.

3 The Maximin Fitness Function

Maximin strategy has its origin in game theory [9]. Rawls in [10] used a nice example of the maximin strategy to illustrate his theory on principles of justice. Balling was the first to propose the use of the maximin fitness function for multiobjective optimization [8]. The maximin fitness for a decision vector $\mathbf{u}$ can be calculated through the following steps. First the **min** function is called to obtain the minimal value from set $\{f_i(\mathbf{u}) - f_i(\mathbf{v}) \mid \forall i \in \{1, \ldots, m\}$:

$$\mathbf{min}_{i=1,\ldots,m}\{f_i(\mathbf{u}) - f_i(\mathbf{v})\} \tag{6}$$

Then the **max** function is applied over the set of minimal values of all possible pairs of $\mathbf{u}$ and another decision vector (other than $\mathbf{u}$) in the population:

$$\mathbf{max}_{j=1,\ldots,N;u\neq v}\{\mathbf{min}_{i=1,\ldots,m}\{f_i(\mathbf{u}) - f_i(\mathbf{v})\}\} \tag{7}$$

In equation (7) two loops of comparison take place, with the **min** first stepping through all the objectives from 1 to m, and then the **max** looping through all candidate solutions in the population from 1 to N, except $\mathbf{u}$. To obtain all non-dominated solutions, another loop will be required to check each solution in the population, from 1 to N. As a result, the overall complexity is $O(mN^2)$.

The maximin fitness value for the decision vector $\mathbf{u}$ is defined as [8]:

$$f_{maximin} = \mathbf{max}_{j=1,\ldots,N;u\neq v}\{\mathbf{min}_{i=1,\ldots,m}\{f_i(\mathbf{u}) - f_i(\mathbf{v})\}\} \tag{8}$$

Given equation (8), it is obvious that for any solution (i.e., a decision vector) to be a non-dominated solution with respect to the current population, its maximin fitness value must be less than zero. Any solution with a maximin fitness equal to zero is a weakly-dominated solution. Any solution with a maximin fitness value greater than zero is a dominated solution.

One unique property that makes the maximin fitness function so appealing to multiobjective optimization is that the maximin fitness value can be used to reward diversity and penalize clustering of non-dominated solutions, therefore no additional diversity maintaining mechanism such as a niching technique is necessary. This can be illustrated by the two examples as shown in Fig. 1 [8]. Fig. 1 a) shows the maximin fitness values calculated for solution **A, B** and **C** respectively. Since the three solutions are non-dominated with each other, the maximin fitness values are negative (written in parentheses). Fig. 1 a) also shows that the maximin fitness is the same for the three equally-spaced solutions, however, Fig. 1 b) shows that the two closely-spaced solutions **B** and **C** are penalized by having a higher fitness than **A**. With the assumption of minimization, the smaller the fitness value is, the better the solution, so **A**(-1.5) is rewarded by getting an even smaller fitness than **A**(-1) in Fig. 1 a), whereas **B**(-0.5) and **C**(-0.5) are penalized by getting a higher fitness than **B**(-1) and **C**(-1) in Fig. 1 a). In the case when **B** and **C** are completely overlapped, the maximin fitness will be zero for both **B** and **C**.

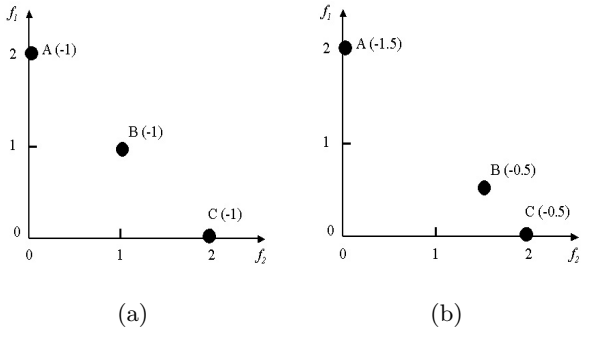


Fig. 1. a) Three non-dominated solutions with an equal fitness value; b) three non-dominated solutions with higher fitness values assigned to solutions that are close to each other.

Balling [8] also showed that maximin fitness favors the middle of a convex front, and the two extreme solutions of a concave front (as shown in Fig. 2). Fig. 2 might give you the impression that using maximin fitness will result in more solutions clustering in the middle of a convex front or two extreme ends of a concave front. This is not necessarily true when there is a sufficiently large number of solutions along the front, because maximin fitness works against clustering of the solutions. In fact, maximin fitness works against clustering of solutions regardless of if the front is convex or concave, as long as there are sufficient numbers of solutions along the front.

For a more detailed description of the properties of the maximin fitness function, the reader can be referred to [8].

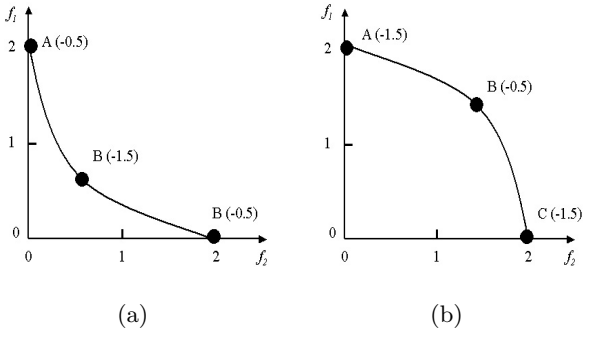


Fig. 2. a) The maximin fitness favors the middle of a convex front, whereas in b) the maximin fitness favors the two extreme solutions of a concave front.

Choosing $\mathbf{p_g}$ for each particle. Our objective is to propel particles in the population towards the current non-dominated front Q as well as the less crowded areas along Q. For each particle, we have decided to choose randomly its p_{gd} for each dimension $d = 1, \ldots, D$ of the particle, from a pool of non-dominated particles with the smallest maximin fitness values (they should be negative too). Fig. 3 illustrates how this works. Note that this method allows the $\mathbf{p_g}$ of a particle to be composed of different p_{gd} from different non-dominated particles. This will have the effect of emphasizing the less crowded areas as a whole on the best known non-dominated front over iterations.

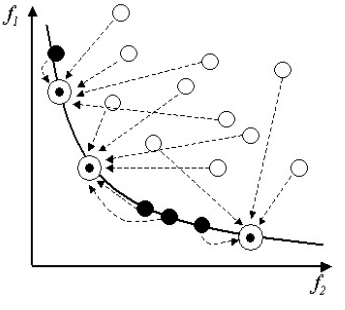


Fig. 3. A few "less crowed" non-dominated particles are used as a pool for choosing at random a p_{gd} (for each dimension $d = 1, \ldots, D$) for a particle in the population.

4 The *maximinPSO* Algorithm

This paper proposes a PSO, *maximinPSO*, which makes use of the maximin fitnesses (according to equation (8)) of individuals in a swarm to facilitate dominance comparison and diversity maintenance for the purpose of multiobjective

optimization. *maximinPSO* extends the basic PSO in a similar way as NSPSO [7]. Both *maximinPSO* and NSPSO combine particles and their offspring into a temporary population, and categorize the temporary population into non-dominated and dominated groups. The main difference between the two is that *maximinPSO* uses maximin fitness to identify the non-dominated particles in the swarm, whereas NSPSO uses the concept of non-dominated sorting [13]. Furthermore, *maximinPSO* does not use any additional clustering or niching technique for diversity maintenance, but NSPSO does use niching techniques as those in NSGA II. Hence *maximinPSO* will save computation that would normally be spent on niching. Having said so, *maximinPSO* still requires $O(mN^2)$ complexity to calculate the maximin fitnesses for the whole population.

The proposed *maximinPSO* can be summarized in the following steps:

1. An initial population of N particles is generated at random, and stored in $PSOList$; For each particle on $PSOList$, $\mathbf{p_i}$ is set to $\mathbf{x_i}$ by default; $\mathbf{v_i}$ is set to be within the variable ranges, with a probability of 0.5 being positive or negative; V_{MAX} is set to the bounds of the variable ranges.
2. Calculate the maximin fitness for all particles in the initial population; get the non-dominated solutions based on the maximin fitness values; sort the non-dominated solutions according to their maximin fitnesses (in ascending order), and store them in $nonDomPSOList$.
3. Iterate through the whole swarm population $PSOList$. For the i-th particle (from 1 to $PSOList$'s current size), do the following:
   a) Choose randomly a p_{gd} for each dimension $d = 1, \ldots, D$ of the i-th particle, from the top few particles (this number can be user-specified) of the $nonDomPSOList$, as the new $\mathbf{p_g}$ for the i-th particle.
   b) Produce an offspring for the i-th particle based on its $\mathbf{p_i}$ and $\mathbf{p_g}$.
   c) Store both the i-th particle and its offspring in a list $nextPopList$. Note that $nextPopList$ should be twice the size of $PSOList$, since it now contains both the parent and its offspring.
4. Calculate maximin fitness for all particles on $nextPopList$; get non-dominated solutions; copy the non-dominated solutions from $nextPSOList$ to the new $PSOList$ for the next iteration; if $PSOList$ contains less than N non-dominated particles, then randomly add weakly-dominated or dominated solutions from $nextPopList$, until $PSOList$ is filled up by N particles; if $PSOList$ contains more than N non-dominated particles, then no other particles are added to $PSOList$. Note that in such a case the size of $PSOList$ will be likely to grow over iterations, exceeding N non-dominated solutions towards the end of a run.
5. Go back to Step 3, until a termination criterion is met. If terminated, then calculate performance measures in the final iteration.

5 Experiments

5.1 Test Functions

In order to assess a multiobjective algorithm's ability to converge to the true Pareto-optimal front as well as to find diverse Pareto-optimal solutions, Zitzler

et al. [11] proposed six test functions with different problem features that may cause difficulties, such as convexity and non-convexity of P^*, disconnectedness of P^*, multiple local fronts, and non-uniformity of the search space. We choose five of the six functions, ZDT1, ZDT2, ZDT3, ZDT4, and ZDT6. All these test functions have two objectives, but have no constraints. The reader can refer to [11] for detailed definitions of these functions.

5.2 Performance Measures

We use the following performance metrics introduced by Zitzler *e*t al. [11]:

$$\mathcal{M}_1^*(Y') := \frac{1}{|Y'|} \sum_{p' \in Y'} \mathbf{min}\{||\mathbf{p}' - \bar{\mathbf{p}}||^*; \bar{\mathbf{p}} \in \bar{\mathbf{Y}}\} \tag{9}$$

$$\mathcal{M}_2^*(Y') := \frac{1}{|Y' - 1|} \sum_{p' \in Y'} |\{q' \in Y'; ||p' - q'||^* > \sigma^*\}| \tag{10}$$

$$\mathcal{M}_3^*(Y') := \sqrt{\sum_{i=1}^{n} \mathbf{max}\{||\mathbf{p}'_\mathbf{i} - \mathbf{q}'_\mathbf{i}||^*; \mathbf{p}', \mathbf{q}' \in \mathbf{Y}'\}} \tag{11}$$

where Y' denotes a set of objective vectors corresponding to the non-dominated solutions found, and $\bar{Y}$ the Pareto-optimal front. Note that a niche neighbourhood size, $\sigma^* > 0$, is used in equation (10) to calculate the distribution of the non-dominated solutions. $\mathcal{M}_1^*(Y')$ gives the average distance from Y' to $\bar{Y}$. $\mathcal{M}_2^*(Y')$ describes the goodness of distribution of solutions in Y'. $\mathcal{M}_2^*(Y')$ should give a value between $[0, |Y'|]$ as it estimates the number of niches in Y' based on σ^*. The higher the value, the better the distribution according to σ^*. $\mathcal{M}_3^*(Y')$ measures the extent of Y'. In addition to the above metrics, we also count the number of function evaluations used for a single run.

5.3 Results

The *maximinPSO* was given an initial population size of 200. A simulation run was terminated only when more than 2000 non-dominated solutions were found in an iteration step. Since the *PSOList* grows in size over time, by the time *PSOList* contains 2000 (or greater) non-dominated solutions, the population generally has already converged and stabilized. c_1 and c_2 were set to 2.0. w was gradually decreased from 1.0 to 0.4. V_{MAX} was set to the bounds of decision variables. This is a set of common parameter values used in a simple PSO model [2]. At each iteration step, for every particle, its p_{gd} is chosen at random (for each dimension $d = 1, \ldots, D$) from the top 20% of *nonDomPSOList* to construct the particle's new $\mathbf{p_g}$ (Step 3 of the *maximinPSO* algorithm). At the final iteration step, the $\mathcal{M}_1^*$, $\mathcal{M}_2^*$ and $\mathcal{M}_3^*$ were calculated according to equation (9), (10) and

(11). The number of evaluations taken for the run was also recorded. A set of $|P^*| = 500$ uniformly distributed Pareto-optimal solutions was used to calculate $\mathcal{M}_1^*$. For $\mathcal{M}_2^*$, the niche neighbourhood size σ^* was set to 0.01.

The results of *maximinPSO* were compared with that of the real-coded NSGA II [12]. As in the *maximinPSO*, an initial population of 200 was used. NSGA II was run for 100 generations so it takes 20000 evaluations for each run, as NSGA II uses a constant population size. In contrast, *maximinPSO* uses a population size that grows over time, which allows more non-dominated solutions to be discovered with relatively fewer evaluations. For NSGA II, we used the same parameter values as suggested by Deb et al. [13]. A crossover probability of 0.9 and a mutation probability of $1/n$ (n is the number of real-variables) were used. The SBX and real-parameter mutation operators, η_c and η_m were set to 20 respectively.

Both *maximinPSO* and the real-coded NSGA II were run 30 times. The results were averaged and summarized in Table 1.

Table 1. $\mathcal{M}_1^*$, $\mathcal{M}_2^*$, $\mathcal{M}_3^*$, and the number of evaluations (averaged over 30 runs).

Metric	Algorithm	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
$\mathcal{M}_1^*$	*maximinPSO*	7.74E-04 ±1.72E-05	1.01E-02 ±5.13E-02	3.44E-03 ±1.09E-04	7.68E-04 ±1.50E-05	1.84E-03 ±6.06E-04
	real-coded NSGA II	1.14E-03 ±5.56E-05	8.26E-04 ±3.44E-05	4.90E-03 ±1.45E-04	5.77E-02 ±1.09E-01	1.04E-01 ±1.02E-02
$\mathcal{M}_2^*$	*maximinPSO*	2.65E+03 ±3.89E+02	2.51E+03 ±8.12E+02	2.15E+03 ±1.25E+02	2.59E+03 ±3.23E+02	2.35E+03 ±3.22E+02
	real-coded NSGA II	1.96E+02 ±6.62E-02	1.96E+02 ±5.90E-02	1.97E+02 ±1.60E-01	1.69E+02 ±5.50E+01	1.95E+02 ±2.74E-01
$\mathcal{M}_3^*$	*maximinPSO*	1.40E+00 ±9.75E-03	1.31E+00 ±3.58E-01	1.96E+00 ±5.26E-03	1.40E+00 ±7.13E-03	1.17E+00 ±3.22E-05
	real-coded NSGA II	1.41E+00 ±3.60E-03	1.41E+00 ±3.69E-03	1.93E+00 ±9.43E-02	1.29E+00 ±2.33E-01	1.13E+00 ±2.67E-02
Num. of Evals.	*maximinPSO*	5.56E+03 ±5.45E+02	5.65E+03 ±1.43E+03	1.13E+04 ±1.17E+03	5.26E+03 ±5.02E+02	5.30E+03 ±5.45E+02
	real-coded NSGA II	2.00E+04 ±0.00E+00	2.00E+04 ±0.00E+00	2.00E+04 ±0.00E+00	2.00E+04 ±0.00E+00	2.00E+04 ±0.00E+00

5.4 Discussion

From Table 1, comparing with the real-coded NSGA II, *maximinPSO* consistently performed better on all test functions, except ZDT2. *maximinPSO* outperformed NSGA II on all performance measures for ZDT3, ZDT4, and ZDT6. More specifically, *maximinPSO* consistently converged better ($\mathcal{M}_1^*$), distributed solutions better along the non-dominated front ($\mathcal{M}_2^*$), had a better coverage ($\mathcal{M}_3^*$), and finally used fewer evaluations. For ZDT1, *maximinPSO* outperformed NSGA II on all metrics except $\mathcal{M}_3^*$ (but just slightly). *maximinPSO* has no difficulty in handling convex (ZDT1) and disconnected Pareto-fronts (ZDT3).

Handling non-convexity of Pareto-front. For ZDT2, we examined all 30 *maximinPSO* runs, and identified 3 very poor runs, which skewed the results for ZDT2 in the Table 1. Out of these 3 very poor runs, in two of which *maximinPSO* converged to a single end point of the Pareto-front, and in the 3rd run, *maximinPSO* only converged to a local front, but the spread of the solutions is still good. If the 3 poor runs were taken out, the remaining 27 runs are in fact equally good or better than NSGA II. This problem of converging to a single solution could be attributed to the fact that there is a higher probability that *maximinPSO* discovered non-dominated solutions on the end points of a concave Pareto-front too early. These non-dominated solutions around the end points subsequently attracted all other particles rather quickly before they even had a chance of going to other parts of the Pareto-front. To combat this problem, we increased the initial population size from 200 to 400 in order to encourage more particles to reach other parts of the Pareto-front. As expected, subsequently this problem was eliminated. The results of *maximinPSO* on ZDT2 using a population size of 400 are provided in Table 2 (averaged over 30 runs). Once again Table 2 shows that *maximinPSO* outperformed NSGA II on all performance metrics. It is also interesting to note that the number of evaluations used to get 2000 or more non-dominated solutions is not much greater than the *maximinPSO* with an initial population size of 200 (the last row in Table 1).

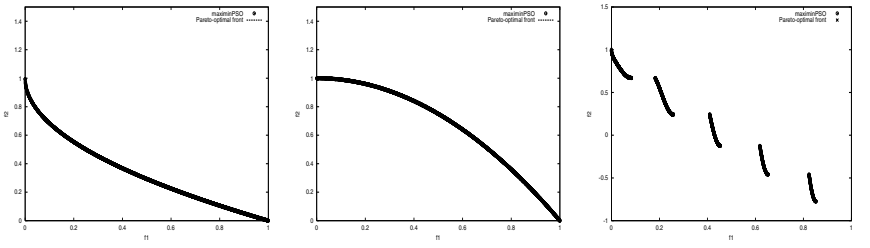
Table 2. Improved results of *maximinPSO* on ZDT2.

Algorithm	$\mathcal{M}^*1$	$\mathcal{M}^*2$	$\mathcal{M}^*3$	Num. of Evals.
maximinPSO	7.87E-04 ±9.70E-06	2.72E+03 ±4.55E+02	1.41E+00 ±1.46E-03	6.86E+03 ±6.58E+02
real-coded NSGA II	8.26E-04 ±3.44E-05	1.96E+02 ±5.90E-02	1.41E+00 ±3.69E-03	2.00E+04 ±0.00E+00

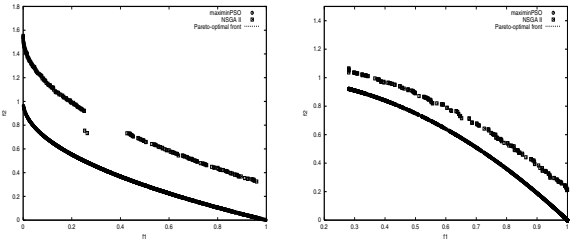
Fig. 4 shows the non-dominated solutions found for ZDT1, ZDT2, ZDT3, ZDT4 and ZDT6 in the final iteration step of a simulation run of *maximinPSO*. We achieved almost a perfect convergence, spread and coverage towards P^*.

Handling multiple local fronts. For ZDT4, where there is a large number of local fronts, *maximinPSO* was able to converge to P^* consistently, 30 out of 30 runs, while still maintaining a good spread and coverage of P^*. In contrast, all 30 NSGA II runs converged to a local front (see Fig. 4 d). These NSGA II runs also produced fewer non-dominated solutions, a poorer spread, and required a larger number of evaluations. Fig. 5 shows a few snapshots of a single *maximinPSO* run on ZDT4.

Handling non-uniform density of solutions. *maximinPSO* had no difficulty with ZDT6, where there is a non-uniform distribution of solutions (in decision variable space) corresponding to the Pareto-optimal front. *maximinPSO* was able to find a set of non-dominated solutions that corresponds to a set of smoothly distributed objective vectors in the objective space. In comparison, all



(a) ZDT1 (b) ZDT2 (c) ZDT3



(d) ZDT4 (e) ZDT6

Fig. 4. Non-dominated solutions found by *maximinPSO*. On ZTD4 and ZDT6, the results are also compared with solutions found by the real-coded NSGA II.

30 NSGA II runs only managed to converge to a local front, with a slightly worse spread (Fig. 4 e).

Making use of the discovered non-dominated solutions. One unique feature of *maximinPSO* is its ability to make use of the non-dominated solutions found so far to allow further improvement of the spread of solutions in future iteration steps. Since maximin fitnesses discourage clustering, those solutions appearing near the gaps in the current step would have lower maximin fitness values (see Fig. 6 step 8), hence more likely to be chosen to construct a new $\mathbf{p_g}$. As a result, other particles in the swarm will be more likely to be attracted to fill these gaps. Fig. 6 shows that during a run on ZDT6, *maximinPSO* was able to fill the gaps appearing on the distribution curve of the non-dominated solutions found over a number of iteration steps.

The importance of a larger population size. Population size plays a critical role in the performance of *maximinPSO*. Small population sizes do not work well when using *maximinPSO*. Especially for a problem with a concave front, the end points could become too dominated early on, hence attracting the rest of population to a single point. To avoid such problems, a reasonably large

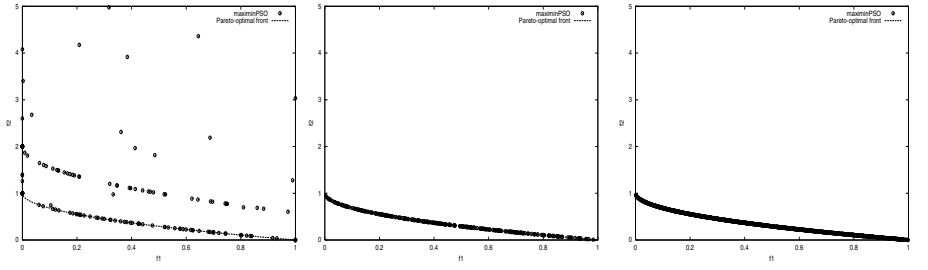


Fig. 5. Snapshots of a *maximinPSO* run on ZDT4, showing all particles at step 8, 10 and 13 (from left to right).

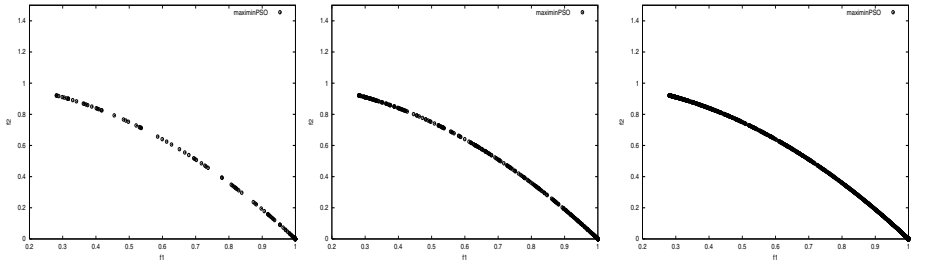


Fig. 6. On ZDT6, gaps among the distribution of solutions were filled over iteration step 8, 10 and 12 (from left to right).

population size is necessary to allow better sampling of the search space, thereby preventing certain individuals from becoming too dominated at the early stage of a run. As shown in the results on ZDT2 (Table 2), an initial population of 400 were needed to obtain a good convergence and spread of solutions consistently.

It may appear that using a larger population size would increase the amount of function evaluations, but for *maximinPSO*, at each iteration step we only evaluate the offspring produced. Parents are not evaluated again before being stored in *nextPopList* for the next iteration step (see step 3 of the *maximinPSO* algorithm). *maximinPSO* generally converges very quickly, most of the time less than 20 iteration steps for all runs reported in Table 1, which is also why it used fewer function evaluations than the NSGA II counterpart. However, in order to determine if each particle is non-dominated with respect to the rest of population, the maximin fitnesses have to be calculated for all particles in *nextPopList*.

6 Conclusion

This paper has described a PSO, *maximinPSO*, using the maximin fitness function for multiobjective optimization. Our experiments on the ZDT test function series have shown that *maximinPSO* is able to produce a convergence and

spread of solutions on the Pareto-optimal front almost perfectly, outperforming the real-coded NSGA II in all performance measures used.

Furthermore, *maximinPSO* is more computationally efficient than NSGA II, since it does not require any additional niching technique. *maximinPSO* was able to find more non-dominated solutions with fewer numbers of function evaluations as shown in our results. When there are sufficiently large numbers of individuals in the population, *maximinPSO* will become less sensitive to the shape of a Pareto-optimal front, whether it is convex or non-convex. *maximinPSO* also has no difficulty handling multiple local fronts and fronts with non-uniform solutions. In future, it would be interesting to see how *maximinPSO* performs on problems with constraints and those with more than two objectives.

References

1. Kennedy, J. and Eberhart, R.: Particle Swarm Optimization. In Proceedings of the Fourth IEEE International Conference on Neural Networks, Perth, Australia. IEEE Service Center(1995) 1942-1948.
2. Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm intelligence*, San Francisco: Morgan Kaufmann Publishers, 2001.
3. Hu, X. and Eberhart, R.: Multiobjective Optimization Using Dynamic Neighbourhood Particle Swarm Optimization. In Proceedings of the IEEE World Congress on Computational Intelligence, Hawaii, May 12-17, 2002. IEEE Press (2002).
4. Parsopoulos, K.E. and Vrahatis, M.N.: Particle Swarm Optimization Method in Multiobjective Problems, in Proceedings of the 2002 ACM Symposium on Applied Computing (SAC'2002) (2002) 603-607.
5. Fieldsend, E. and 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, Birmingham, UK(2002) 37-44.
6. Coello, C.A.C. and Lechuga, M.S.: MOPSO: A Proposal for Multiple Objective Particle Swarm Optimization, in Proceedings of Congress on Evolutionary Computation (CEC'2002), Vol. 2, IEEE Press (2002) 1051-1056.
7. Li, X.: A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization, in Erick Cantú-Paz et al. (editors), Genetic and Evolutionary Computation - GECCO 2003. Proceedings, Part I, Springer, Lecture Notes in Computer Science Vol. 2723, (2003) 37-48,.
8. Balling, R.: The Maximin Fitness Function; Multiobjective City and Regional Planning. In Proceedings of EMO 2003, (2003) 1-15.
9. Luce, R. D. and Raiffa, H.: Games and Decisions - Introduction and Critical Survey. John Wiley & Sons, Inc, New York (1957).
10. Rawls, J.: A Theory of Justice. Oxford University Press, London (1971).
11. Zitzler, E., Deb, K. and Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, **8**(2):173-195, April (2000).
12. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Chichester, UK (2001).
13. Deb, K., Agrawal,S., Pratap, A. and Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation 6(2): 182-197 (2002).