A Non-dominated Sorting Particle Swarm Optimizer for Multiobjective Optimization

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. This paper introduces a modified PSO, Non-dominated Sorting Particle Swarm Optimizer (NSPSO), for better multiobjective optimization. NSPSO extends the basic form of PSO by making a better use of particles' personal bests and offspring for more effective non-domination comparisons. Instead of a single comparison between a particle's personal best and its offspring, NSPSO compares all particles' personal bests and their offspring in the entire population. This proves to be effective in providing an appropriate selection pressure to propel the swarm population towards the Pareto-optimal front. By using the non-dominated sorting concept and two parameter-free niching methods, NSPSO and its variants have shown remarkable performance against a set of well-known difficult test functions (ZDT series). Our results and comparison with NSGA II show that NSPSO is highly competitive with existing evolutionary and PSO multiobjective algorithms.

1 Introduction

Multiobjective optimization problems represent an important class of real-world problems. Typically such problems involve trade-offs. For example, a car manufacturer may wish to maximize its profit, but meanwhile also to minimize its production cost. These objectives are typically conflicting to each other. A higher profit would increase the production cost. There is no single optimal solution. Often the manufacturer needs to consider many possible "trade-off" solutions before choosing the one that suits its need. The curve or surface (for more than 2 objectives) describing the optimal trade-off solutions between objectives is known as the Pareto front. One of the major goals in multiobjective optimization is to find a set of well distributed optimal solutions along the Pareto front.

In recent years, population-based optimization methods such as Evolutionary Algorithms have become increasingly popular for solving multiobjective optimization problems [1][2]. EA's success is due to its generic ability to handle large complex real-world problems. Since EAs maintain a population of solutions, this allows exploration of different parts of the Pareto front simultaneously. However not until recently, another population-based optimization

technique Particle Swarm Optimization (PSO) has been only applied to single objective optimization tasks. PSO technique is inspired by studies of social behavior of insects and animals [3]. The social behavior is modeled in a PSO to guide a population of particles (so-called swarm) moving towards the most promising area of the search space. In PSO, Each particle represents a candidate solution, $X_i = (x_{i1}, x_{i2}, \ldots, x_{id})$. d is the dimension of the search space. The i-th particle of the swarm population knows: a) its *personal best position* $P_i = (p_{i1}, p_{i2}, \ldots, p_{id})$, i.e., the best position this particle has visited so far that yields the highest fitness value; and b) the *global best position*, $P_g = (p_{g1}, p_{g2}, \ldots, p_{gd})$, i.e., the position of the best particle that gives the best fitness value in the entire population; and c) its current *velocity*, $V_i = (v_{i1}, v_{i2}, \ldots, v_{id})$, which represents its position change; The following equation (1) uses the above information to calculate the new updated velocity for each particle in the next iteration step. Equation (2) updates the each particle's position in the search space.

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

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

where $d = 1, 2, \ldots, D$; $i = 1, 2, \ldots, N$; N is the size of the swarm population; χ is a constriction factor which controls and constricts the velocity's magnitude; 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.

PSO has proved to be an efficient optimization method for single objective optimization, and more recently has also shown promising results for solving multiobjective optimization problems [4][5] [6][7]. What is in common among these works is the use of a basic form of PSO first introduced by Kennedy and Eberhart [3]. However the basic form of PSO has some serious limitations in particular when dealing with multiobjective optimization problems. In PSO, a particle is modified only through its personal best and global best to produce its offspring. At each iteration step, if the fitness of the offspring is better than the parent's personal best, then the personal best is updated with this offspring, however, there is no sharing of information with other particles in the population, except that each particle can access the global best. For multiobjective optimization, we argue that such sharing of information among all the individuals in a population is crucial in order to introduce the necessary selection pressure to propel the population moving towards the true Pareto-optimal front.

This paper introduces a modified PSO, Non-dominated Sorting Particle Swarm Optimizer (NSPSO), which is able to increase such "sharing" among all particles in a swarm population especially concerning how to allow the population as a whole to progress towards the true Pareto-optimal front. For clarity, in this paper, we use P^* to denote the true Pareto-optimal front, and Q the found non-dominated solution set.

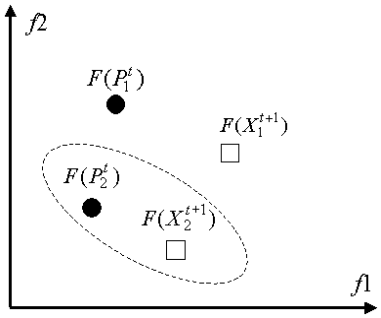

Fig. 1. Dominance relationships among 4 particles, including the personal best P_1^t of a particle X_1^t, and its potential offspring X_1^{t+1}, plus P_2^t and X_2^{t+1} for a second particle X_2^t, assuming minimization of f_1 and f_2.

2 Modifying PSO for better dominance comparison

One problem that can be identified with the basic form PSO is that dominance comparisons are not fully utilized in the process of updating the personal best P_i of each particle. This can be illustrated via the following example shown in Fig. 1. Note that $F(\cdot)$ denotes the evaluation of a particle in the objective space.

Fig.1 shows that the personal best P_1^t is mutually non-dominating with X_1^{t+1}, and P_2^t is non-dominating with X_2^{t+1}, however, X_1^{t+1} is dominated by X_2^{t+1} and P_2^t , and furthermore, P_1^t is also dominated by P_2^t. In a standard PSO, the i-th particle only has a single P_i^t, which is used to compare with its potential offspring X_i^{t+1} at time step $t+1$. If P_i^t is non-dominating with its potential offspring such as the situation shown in Fig. 1, then P_i^t will remain the same. The consequence of this kind of comparison is that the useful non-domination relationships among all the four particles will not be captured. Fig. 1 shows that if we allow all 4 particles to be compared, then we would have found P_2^t and X_2^{t+1} to be the better two to retain. This illustration shows that in a standard PSO, valuable non-domination comparisons are not effectively used. In other words, as a result of having only a single comparison between a particle's P_i^t and X_i^{t+1}, there is only a very weak selection pressure with respect to the non-dominated front that exists in the current population.

This would naturally lead to the next two questions - what if we allow all the personal bests of all particles and as well as these particles' offspring to be compared for non-domination relationships? How are we going to choose the global best for each particle in order to propel the population to move towards P^*, while also maintaining a diverse set of solutions?

3 Non-dominated Sorting PSO

Two major goals in multiobjective optimization are to obtain a set of non-dominated solutions as closely as possible to the true Pareto front P^*, and to

maintain a well-distributed solution set along the Pareto front. To achieve this using PSO, a Non-dominated Sorting Particle Swarm Optimizer (NSPSO) is proposed. In NSPSO, we adopt the non-dominated sorting concept used in NSGA II [2], where the entire population is sorted into various non-domination levels. This provides the means for selecting the individuals in the better fronts, hence providing the necessary selection pressure to push the population towards P^*. To maintain population diversity, we use a widely-used niching method [8], and also the crowding distance assignments adopted by NSGA II [9]. The following two sections describe these methods.

3.1 Selection Pressure Towards P^*

Instead of comparing solely on a particle's personal best with its potential offspring, the entire population of N particles' personal bests and N of these particles' offspring are first combined to form a temporary population of $2N$ particles. After this, domination comparisons among all the $2N$ individuals in this temporary population are carried out. This "combine-then-compare" approach will ensure more non-dominated solutions can be discovered through the domination comparison operations. Since updating the personal bests of the N particles represents "collectively" a step towards a better region in the search space, retaining them for domination comparison provides the needed selection pressure towards P^*. By comparing the combined $2N$ particles for non-domination relationships, we will be able to sort the entire population in different non-domination levels as used in NSGA II. This type of sorting can then be used to introduce the selection bias to the individuals in the populations, in favour of individuals closer to the true Pareto front P^*. At each iteration step, we choose only N individuals out of the $2N$ to the next iteration step, based on the non-domination levels. This process can be illustrated in Fig. 2. First the entire population is sorted into

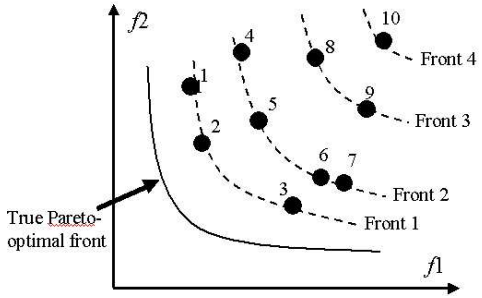

Fig. 2. Particles of a swarm population of 10 are classified into 4 successive non-dominated fronts

two sets, the non-dominated set and the remaining dominated set. In Fig.2, the non-dominated set is Front 1, which contains 3 particles labeled as 1, 2 and 3. Front 1 is the best non-dominated set, since all particles in Front 1 are not dominated by any other particles in the entire population. To obtain the next front,

Front 2, we temporarily remove Front 1 from the population, then find the non-dominated solutions of the remaining population, which is Front 2. Then again we remove Front 2 as well, in order to identify the non-dominated solutions of the next level. This procedure continues until all particles in the population are classified into different non-dominated front levels. For each particle, there are $O(mN)$ comparisons are required to find out if it is dominated by other partciles in a population of size N (m is the number of objectives). The complexity to find the first non-dominated front (i.e., Front 1) for the whole population is $O(mN^2)$. In the worst case where there is only one particle in each front, the complexity of the above procedure for classifying different non-dominated fronts is $O(mN^3)$.

Now we create the new particle population for the next iteration step, by selecting particles from fronts in ascending order, e.g., first from Front 1, then Front 2, etc, until N particles (or a specified threshold) are selected. Since the particles in the first few fronts get chosen first, this selection pressure will effectively drive the particle population towards the best front over many iteration steps. Note that Front 1 could have more than N particles (since the combined $2N$ particles are sorted), especially after a number of steps in a run. Setting a threshold may be necessary as it would allow opportunities of particles from other fronts to be selected as well, i.e., maintaining "lateral diversity" [2].

3.2 Parameter-free Niching Methods to Maintain a Diverse Q

Niching methods have been extensively studied and used as means of maintaining population diversity in Genetic Algorithms. One commonly used niching method is a sharing function model introduced by Goldberg and Richardson [10], where a niche is treated as a resource shared among other individuals in the niche. In NSPSO, to achieve the second goal of maintaining a diverse non-dominated solution set, two niching methods are tried, in order to see how effective each method is in maintaining solution diversity. The first method requires calculating a niche count for each particle, whereas the second method requires calculating a crowding distance value for each particle.

Using Niche Count. In NSPSO, the niche count m_i of a particle i is simply calculated as the number of other particles within a σ_{share} distance (i.e., Euclidean distance) from i. Note that σ_{share} can be calculated dynamically at each iteration step. Fig.3 shows how niche counts are calculated for two candidate solution A and B. Both A and B are on the current non-dominated front. However since A has a smaller niche count than B, A will be preferred over B. This choice has the effect of emphasizing a more diverse non-dominated front.

One of the undesirable features of the niching method employing σ_{share} is that σ_{share} has to be specified by a user, and the model's performance is highly dependant on the choice of value for this parameter. We adopted the dynamic update of σ_{share} proposed by Fonseca and Fleming [11] so that we do not have to specify σ_{share}. For two objective functions, the following equation is used to

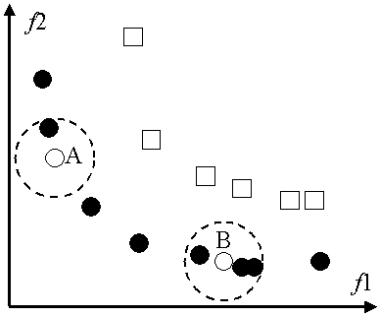

Fig. 3. Niche counts are calculated for particle A and B on the non-dominated front Q (indicated by circles), assuming minimization of f_1 and f_2.

determine the σ_{share} dynamically [2]:

$$\sigma_{share} = \frac{u_2 - l_2 + u_1 - l_1}{N - 1}, \tag{3}$$

where u_i and l_i are the upper and lower bounds for each of the two objective values for the entire population. Note that as population size increases, the σ_{share} is reduced to accommodate more niches. At each iteration step, we select from the current non-dominated solution set Q those particles with smallest niche counts. Then P_g for each particle is randomly chosen among these "less crowded" non-dominated particles.

Using Crowding Distance Assignments. The 2nd niching method employing crowding distance [9] is also free of choosing such a parameter. Deb et al. [9] in their NSGA II introduced this niching method that makes use of the density of solutions around a particular point on the non-dominated front. The density is estimated by calculating the so-called crowding distance of a point i , which is the average distance of the two point i-1 and i+1 on either side of this point i along each of the objectives. When we use crowding distance values for niching, we simply sort all the particles of the current Q in descending order, and then choose a particle randomly from the top part of the sorted list (i.e., a particle in the least crowded areas in the Pareto region) as P_g for each particle. This process is repeated for each particle in the population, and over many iteration steps. The complexity of this procedure in the worst case is $O(mNlogN)$, when all particles are in one front. For more information on crowding distance assignment, the readers can refer to [9].

Replacement of "Overcrowded" Particles with New Particles. Another method that can further promote diversity is to remove particles from overcrowded areas on the current non-dominated front Q, and replace them with new particles. We implement this in NSPSO by removing the particle with the

largest niche count (or the smallest crowding distance value), and replace it with a randomly generated new particle, at each iteration step. Since the new particle chooses the particle with the smallest niche count, i.e., least crowded (or the largest crowding distance value), from the current Q as its P_g, this particle should have a better chance to land somewhere "less crowded" on the current front Q.

3.3 NSPSO Algorithm

NSPSO can be summarized in the following steps:

1. Initialize the population and store the population in a list $PSOList$:
   (a) The current position of the i-th particle, X_i and its current velocity V_i, are initialized with random real numbers within the specified decision variable range; V_i has a probability of 0.5 being specified in a different direction; The personal best position P_i, is set to X_i ; V_{MAX} is set to the upper and lower bounds of the decision variable range.
   (b) Evaluate each particle in the population; iteration counter $t := 0$.
2. $t := t + 1$.
3. Identify particles that give non-dominated solutions in the population and store them in a list $nonDomPSOList$.
4. Calculate - a) niche count, or b) crowding distance value, for each particle.
5. Resort the $nonDomPSOList$ according to a) niche counts, or b) crowding distance values.
6. $For(i := 0; i < numParticles; i{+}{+})$ (step through $PSOList$):
   (a) Select randomly a global best P_g for the i-th particle from a specified top part (e.g. top 5%) of the sorted $nonDomPSOList$.
   (b) Calculate the new velocity V_i, based on the equation (1), and the new X_i by equation (2).
   (c) Add the i-th particle's P_i and the new X_i to a temporary population, stored in $nextPopList$. Note that P_i and X_i now coexist. Also note that $nextPopList$ now has a size of $2N$.
   (d) Go to a) if $i < numParticles$.
7. Identify particles that give non-dominated solutions from $nextPopList$, and store them in $nonDomPSOList$. Particles other than non-dominated ones from $nextPopList$ are stored in a list $nextPopListRest$.
8. Empty $PSOList$ for the next iteration step.
9. Select randomly members of $nonDomPSOList$ and add them to $PSOList$ (not to exceed $numParticles$).
10. Loop if $PSOList\ size < numParticles$:
    (a) Identify non-dominated particles from $nonDomListRest$ and store them in $nextNonDomList$.
    (b) Add members of $nextNonDomList$ to $PSOList$, if still the $PSOList\ size < numParticles$.
    (c) Copy $nextPopListRest$ to $nextPopListRestCopy$, then empty $nextPopListRest$.
    (d) Assign the vacant $nextPopListRest$ with the remaining particles other than non-dominated ones from $nextPopListRestCopy$.
    (e) Go back to a), if still the $PSOList\ size < numParticles$.
11. If $t < maxIterations$, go to 2.
12. Obtain Q from the final population, and calculate the performance metric values (see section 4).

4 Performance Metrics

Diversity of Q. We measure the diversity of solutions along the Pareto front in the final population by comparing the uniform distribution and the deviation of solutions as described by Deb [2]:

$$\Delta = \frac{\Sigma_{m=1}^{M} d_m^e + \Sigma_{i=1}^{|Q|} |d_i - \bar{d}|}{\Sigma_{m=1}^{M} d_m^e + |Q|\bar{d}}, \tag{4}$$

where d_i is the distance between two neighbouring solutions in the non-dominated solution set Q, $\bar{d}$ is the mean value of all the d_i. d_m^e is the distance between the

extreme solutions of the true Pareto-optimal set P^* and Q on the m-th objective. d_m^e is calculated by using Schott's difference distance measure [2]. By using d_m^e, equation (4) also takes into account of the extent of the spread. For an ideal distribution of solutions (uniform and d_m^e =0.0), Δ is 0.0. For functions with disconnected sectors on the Pareto front, e.g., ZDT3, Δ is calculated within each continuous sector and then averaged.

Number of non-dominated solutions found. The above diversity metric does not take into account the number of optimal solutions found. An ideal uniform distribution of Q with $\Delta = 0.0$ could have very few solutions. Obviously we prefer a uniform distribution with a larger number of solutions found in the final step of a run.

Closeness to P^*. Generational distance (GD) metric is used to measure the closeness of solutions in Q to P^*. The GD metric finds the average distance of the solutions of Q from P^* [2]:

$$GD = \frac{\left(\Sigma_{i=1}^{|Q|} d_i^p\right)^{1/p}}{|Q|}. \tag{5}$$

For a two objective problem ($p = 2$), d_i is the Euclidean distance between the solution $i \in Q$ and the nearest member of P^*.

5 Experiments

Four test functions ZDT1, ZDT2, ZDT3 and ZDT4 were used [2]. These functions are considered to be difficult because of the large number of decision variables, disconnectedness of P^*, and multiple local fronts. The initial population of the NSPSO was set to 200. 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 be the bounds of decision variable ranges, and χ simply to 1.0. NSPSO was run for 100 iteration steps. At the final iteration, the diversity metric Δ and closeness metric GD values were calculated according to equation (4) and (5), and also the number of non-dominated solutions found. A set of $|P^*| = 500$ uniformly distributed Pareto-optimal solutions is used to calculate the GD values. The results of NSPSO were compared with the real-parameter NSGA II. NSGA II also had an initial population of 200, and it was run for 100 generations. As suggested in [2], 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. NSPSO and NSGA II were both run 10 times. The results are averaged and summarised in Table 1 - 3. Four NSPSO variants were used: **NC**: using niche count only; **NC-R**: using niche count with replacement; **CD**: using crowding distance only; and **CD-R**: using crowding distance with replacement. Refer to section 3.2 about these four niching variants.

Table 1. Mean and variance values of the GD metric measuring convergence.

	ZDT1		ZDT2		ZDT3		ZDT4	
Algorithm	GD	δ^2	GD	δ^2	GD	δ^2	GD	δ^2
NC	7.97E-04	8.13E-05	8.05E-04	3.05E-05	3.40E-03	2.54E-04	7.82E-04	6.91E-05
NC-R	7.53E-04	4.18E-05	2.93E-02	9.03E-02	3.22E-03	5.31E-04	7.36E-04	5.16E-05
CD	1.05E-03	1.71E-04	8.48E-04	4.57E-05	3.54E-03	4.98E-04	9.03E-04	1.66E-04
CD-R	8.61E-04	1.55E-04	2.93E-02	9.02E-02	3.53E-03	3.79E-04	8.42E-04	1.50E-04
NSGA II	1.14E-03	7.64E-05	8.25E-04	3.26E-05	N/A	N/A	2.92E-02	4.67E-02

Table 2. Mean and variance values of the Δ metric measuring diversity.

	ZDT1		ZDT2		ZDT3		ZDT4	
Algorithm	Δ	δ^2	Δ	δ^2	Δ	δ^2	Δ	δ^2
NC	7.67E-01	3.00E-02	7.58E-01	2.77E-02	9.14E-01	3.93E-02	7.68E-01	3.57E-02
NC-R	7.72E-01	3.58E-02	7.67E-01	4.36E-02	9.04E-01	6.89E-02	8.06E-01	5.05E-02
CD	7.62E-01	3.83E-02	7.48E-01	4.66E-02	8.69E-01	5.81E-02	7.36E-01	1.84E-02
CD-R	7.62E-01	3.17E-02	7.60E-01	4.86E-02	8.65E-01	5.94E-02	7.89E-01	3.98E-02
NSGA II	3.86E-01	1.63E-02	3.90E-01	2.01E-02	N/A	N/A	6.55E-01	1.98E-01

6 Results and Discussion

Table 1 shows that for all 4 functions, NSPSO has no trouble reaching P^* for almost all of the 10 runs within 100 iterations. However an "outlier" was identified for NC-R (ZDT2) and CD-R (ZDT2) each. Closely examining the data, it was found that the poor GD value was the result of a single poor run, but 9 others were in fact very good. Note that NSGA II failed to converge in 100 iteration steps on ZDT3. NSGA II (ZDT4) has the worst variance and GD values, which are due to two runs reaching only local fronts (Fig. 4 (last on the bottom row). Table 2 shows that NSGA II has a better Δ value overall, whereas NSPSO's Δ values are higher. However from a typical run as shown in Fig. 4, we can see that NSPSO has a coverage of P^* just as good as NSGA II. The higher Δ values can

Table 3. Number of non-dominated solutions found in the final iteration step.

Algorithm	ZDT1	ZDT2	ZDT3	ZDT4
NC	274.3	297	201	276.4
NC-R	277.2	282.6	186.9	286.4
CD	187.9	239.7	134.1	159.6
CD-R	247.8	290.1	194	254.5
NSGA II	200	200	N/A	175.5

be attributed to a larger number of different Q found by NSPSO than NSGA II, as shown in Table 3. Since we can obtain the best front (Front 1) with possibly more than N non-dominated solutions in the combined population of $2N$ personal bests and offspring, NSPSO is often able to obtain more than 200 different non-dominated solutions in the final step. In contrast, NSGA II generally has a constant number of non-dominated solutions, which is its initial population size (except for ZDT4). To our surprise, NC-R and CD-R did not seem to make much difference in terms of Δ and GD values. However in Table 3, we can note that CD-R managed to find many more different non-dominated solutions than CD. For NC and NC-R, however, there is not much difference. Fig. 4 presents

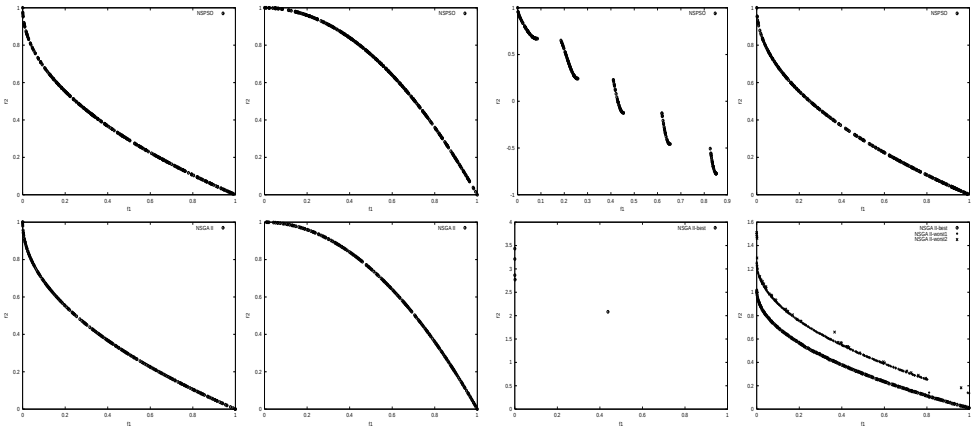

Fig. 4. Non-dominated solutions found by NSPSO (top row) and NSGA II (bottom row) for ZDT1, 2, 3, and 4 (from left to right).

that for a typical NSPSO and NSGA II run, the non-dominated solutions found in the final iteration step for all 4 test functions. Overall, the results show that NSPSO is very competitive in terms of solution spread, coverage and closeness to P^*. Fig. 4 (3rd on the top row) shows that NSPSO has no trouble converging on the different disconnected sectors of P^* for ZDT3, whereas in this case, NSGA II failed completely (3rd on the bottom row). For ZDT4, NSPSO's 10 runs have all converged to P^* with a good spread and coverage. In contrast, NSGA II had 2 out of 10 runs reaching only a local front (Fig. 4 (last on the bottom row)).

Fast and better convergence towards the global front. Fig. 5 presents 4 snapshots of the first few steps in a NSPSO's run for solving ZDT4. It can be noted that even in step 3, there were no particles that had found P^*, but by step 9 there were quite a few, though the majority of the particles were stuck at the 2nd best front. By step 15, all particles had reached P^* without difficulty. Considering the majority of current multiobjective evolutionary algorithms (with the exception of NSGA II) are unable to converge to P^* for ZDT4, this is a remarkable result. NSPSO also demonstrates its consistency as 10 out of 10

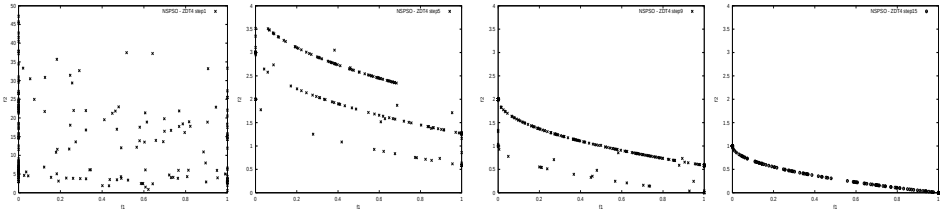

Fig. 5. Snapshots of a NSPSO run showing the entire particle population at step 1, 3, 9 and 15, for solving ZDT4.

runs have all converged to P^*, better than the NSGA II counterpart. NSPSO's impressive performance is due to the fact that we allow all particles' personal bests and offspring to be combined for non-domination comparison. Updating of the particles is then based on their mutual non-domination relationships with respect to the current Q in the population, not with respect to just another single particle (like the basic PSO). Each time, the best N particles in terms of non-domination levels of the entire population are selected for the next iteration step. Even if a single particle has found P^*, the particle's personal best and its offspring will be emphasised favourably in the next step. Since this personal best and the offspring coexist and they are most likely to be located near P^*, it is likely they will be selected again for the next step. This is more like a "self-proliferation" of good particles. The aggregated effect of this over many steps would produce a large number of fitter particles along P^*. Since at each step we only select the best N particles, particles on those local fronts are gradually phased out, replaced by the fitter ones on the global front.

Using a larger population size. It was found that a reasonably large population is necessary for a good convergence. If population size was too small (e.g., 20), NSPSO could converge to sub-optimal fronts, or to P^* but with a limited number of non-dominated solutions, which is insufficient in terms of solution spread and coverage. This is because a smaller number of particles do not sufficiently sample the search space, and as a result, certain existing particles could quickly become too dominant early on, and they would prevent other potentially good particles (in terms of different non-dominated solutions) from being produced. A large initial population size would allow for a better sampling of the search space, and from there onwards allow NSPSO to better use domination comparison operations to find a wide spread of solutions along P^*.

7 Conclusion

With an aim to improve PSO's effectiveness in utilising the domination comparison operations for solving multiobjective optimization problems, this paper has proposed a modified PSO, NSPSO. The model is able to discover more non-dominated relations by comparing the personal bests and offspring of all particles

in a combined swarm population, thereby providing a more appropriate selection pressure for the population to approach the true Pareto-optimal front. NSPSO adopts the non-dominated sorting concept, and uses two parameter-free niching techniques to promote solution diversity. It has been shown that NSPSO and its variants are able to perform remarkably well against some difficult test functions (functions with high-dimensional decision variables and multiple local fronts) found in the literature. NSPSO is also fast, more reliable, and often converging to the true Pareto-optimal front with a good solution spread and coverage within just a few steps. This once again proves that PSO is a powerful optimization technique that can be used efficiently not only for single objective, but also for multiobjective optimization. Future work will look into application of NSPSO to problems with more than 2 objectives, and also real-world multiobjective optimization problems.

References

1. Zitzler, E., Deb, K. and Thiele, L.: Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary Computation, **8**(2):173-195, April (2000).
2. Deb, K.: Multi-Objective Optimization using Evolutionary Algorithms, John Wiley & Sons, Chichester, UK (2001).
3. 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.
4. 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.
5. 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).
6. 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.
7. 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.
8. Horn, J., Nafpliotis, N., and Goldberg, D.E.: A Niched Pareto Genetic Algorithm for Multiobjective Optimization. In Proceedings of the First IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, vol: **1**, Piscataway, New Jersey. IEEE Service Center.(1994) 82-87.
9. Deb, K., Agrawal, S. Pratap, A. and Meyarivan, T.: A Fast Elitist NonDominated Sorting Genetic Algorithm for Multi-Objective Optimization: NSGA-II. In Proceedings of Parallel Problem Solving from Nature - PPSN VI, Springer(2000) 849-858.
10. Goldberg, D.E., and Richardson, J.J.: Genetic Algorithms with sharing for multimodal function optimization. Genetic Algorithms and Their Applications: Proceedings of the Second ICGA, Lawrence Erlbaum Associates, Hillsdale, NJ, (1987) 41-49.
11. Fonseca, C.M. and Fleming, P.J.: Genetic algorithms for multiobjective optimization: Formulation, discussion, and generalization. In Proceedings of the Fifth International Conference on Genetic Algorithms (1993) 355-365.