Particle Swarm with Extended Memory for Multiobjective Optimization

Xiaohui Hu [1, 2] Russell C. Eberhart [2] Yuhui Shi [3]

[1] Department of Biomedical Engineering
Purdue University, West Lafayette, Indiana, USA
hux@ecn.purdue.edu

[2] Department of Electrical and Computer Engineering
Purdue School of Engineering and Technology, Indianapolis, Indiana, USA
reberhar@iupui.edu

[3] EDS Embeded Systems Group
Kokomo, Indiana, USA
Yuhui.Shi@eds.com

Abstract: This paper presents a modified dynamic neighborhood Particle Swarm Optimization (DN-PSO) algorithm for multiobjective optimization problems. PSO is modified by using a dynamic neighborhood strategy, new particle memory updating, and one-dimension optimization to deal with multiple objectives. An extended memory is introduced to store global Pareto optimal solutions to reduce computation time. Several benchmark cases were tested and the results show that the modified DNPSO is much more efficient than the original DNPSO and other multiobjective optimization techniques.

I. INTRODUCTION

Multiobjective optimization (MO) is an important research topic for both scientists and engineers. Traditional optimization techniques such as gradient-based methods are difficult, if not impossible, to extend to true MO problems. Evolutionary computation algorithms deal with a population of candidate solutions. It seems natural to apply evolutionary computation algorithms to MO problems to find a group of Pareto optimal solutions simultaneously. In the past several years, many evolutionary algorithm based multiobjective optimization (MOEA) methods have been suggested and developed [1-3].

General MO problems can be defined in the following format:

Optimize $\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})]$

subject to $g_j(\vec{x}) \le 0 \quad for \quad j = 1, ..., p$

and $h_j(\vec{x}) = 0 \quad for \quad j = p+1, ..., m$

where $\vec{x} = (x_1, x_2 ... x_n) \in \Re^N$

The family of optimal solutions of this MO problem is composed of all those potential solutions such that the components of the corresponding objective vectors cannot be simultaneously improved. This is known as the concept of Pareto optimality. In a minimization problem, Pareto dominance and Pareto optimality are defined as follows [1]:

Definition 1 (Pareto Dominance): A given vector $\vec{x} = (x_1, x_2, \cdots, x_n)$ is said to dominate $\vec{x}' = (x'_1, x'_2, \cdots, x'_n)$ if and only if

$$\forall i \in \{1,2,\cdots n\}, x_i \le x'_i \text{ and } \exists i \in \{1,2,\cdots n\}, x_i < x'_i$$

Definition 2 (Pareto Optimality): For a general MO problem, a given solution $\vec{f}^* \in F$ (where F is the feasible solution space) is Pareto optimal if and only if there is no $\vec{f} \in F$ that dominates $\vec{f}^*$.

In other words, this definition of Pareto optimality says that $\vec{f}^*$ is Pareto optimal if there is no feasible vector $\vec{f}$ that would decrease some objective value without causing a simultaneous increase in at least one other objective value [3]. The Pareto optimum usually provides a group of solutions called non-inferior or non-dominated solutions instead of a single solution.

Definition 3 (Pareto front): the solutions which are Pareto optimal comprise the Pareto front.

II. PSO AND MO-PSO

As a new optimization method, particle swarm optimization (PSO) has been demonstrated to be an effective optimization tool in many areas [4, 5]. However, the application of PSO for solving multiobjective optimization problems (MO-PSO) is still under development. This paper reviews the current development of MO-PSO and proposes a new approach to multiobjective optimization using PSO.

Particle swarm optimization (PSO) is a population-based stochastic optimization technique originally developed by Kennedy and Eberhart [6, 7]. It is similar to evolutionary computation techniques in that: 1. It is initialized with a population of random solutions; 2. It searches for the optimum over generations; 3. Reproduction is based on the prior generations. The key point in PSO is that each potential

0-7803-7914-4/03/$10.00©2003 IEEE

solution, called a particle, "randomly" searches through the problem space by updating itself with its own memory and social information gathered from the other particles.

The updates of the particles are accomplished according to the following equations. Equation (1) calculates a new velocity for each particle (potential solution) based on its previous velocity (V_{id}), the particle's location at which the best fitness has been achieved (p_{id} , or *pBest*) so far, and the best particle location among the neighbors (p_{nd} , or *nBest*) at which the best fitness has been achieved so far. Equation (2) updates each particle's position in the solution hyperspace. The two random numbers *rand*() and *Rand*() are independently generated, while c_1 and c_2 are two learning factors. The use of the inertia weight w has provided improved performance in a number of applications [8].

$$V_{id} = w \times V_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{nd} - x_{id}) \quad (1)$$

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

In order to handle multiple objectives, PSO must be modified before being applied to MO problems. There are some studies reported in the literature that extend PSO to MO problems [9-13]. Due to the similarity of particle swarm and other evolutionary computation methods, many multiobjective handling techniques can be adopted to the modified PSO.

In most approaches, the major modifications of the PSO algirithm are the selection process of *nBest* and *pBest*.

Coello Coello *et al.* [9] developed a grid-based *nBest* slection process. He exployed a second population called a "repository" to store all the non-dominated solutions. The *nBest* is determined by choosing a non-dominant solution from the repository. In order to get better distribution and provent clustering, roulette-wheel selection is used to select one of the hypercubes from which the *nbest* will be randomly picked. The aim is to distribute the non-dominant solutions evenly among all the hypercubes. The selection of *pBest* is relatively easy, which is only updated according to the Pareto Dominance.

Parsopoulus *et al.* [11] tested a different strategy by using multiple populations to handle multiple objectives. It is called a vector evaluated particle swarm, which adopted the idea from vector evaluated genetic algorithms. Two swarms are used to solve a two-objective optimization problem; each swarm is evaluated according to one of the objectives. When one swarm updates the velocities of the particles, the other swarm is used to find the best particle to follow. Both methods show promising results. However a quantitative analysis of the performance was not reported

III. DYNAMIC NEIGHBORHOOD PSO WITH EXTENDED MEMORY

In a previous paper [10], a dynamic neighborhood particle swarm optimization algorithm for multiobjective optimization problems was introduced. Compared to the traditional PSO, following are the modifications in the DNPSO.

- *The selection of nBest*: first calculate the distances of the current particle from other particles in terms of F1 values (the multiple objectives are divided into two groups: F1 and F2. F1 is defined as neighborhood objective, F2 is defined as optimization objective, and the selections of F1 and F2 are arbitrary), then find the nearest m particles as the neighbors of the current particle based on the distances calculated above (m is the neighborhood size); finally find the *nBest* among the neighbors in terms of the F2 values.
- *The update of pBest*: *pBest* is the best position in a particle's history. Only when a new solution dominates the current *pBest*, is the *pBest* updated.

In the preliminary research of this approach, it was showed that DNPSO is a promising method. This paper is an extension to the previous paper. An important modification is made to significantly decrease the computation time. An extended memory is added to the DNPSO to memorize all potential Pareto optimal solutions.

Figure 1 shows the possible flying directions of a particle in a two-objective minimization scenario. Using the current particle as the origin, the hyperspace can be divided into four regions. All solutions in region I dominate the current particle, and all solutions in region III are dominated by current particle. The solutions in region II and IV are potential Pareto optimal solutions but do not dominate current solution. However, in the original DNPSO, the *pBest* is only updated when a new solution dominates the current *pBest*. The strategy might lose many potential good solutions in region II and region IV. Those solutions do not dominate the current particle, but they might dominate other particles. If those solutions are well utilized, it will significantly reduce the computation time. Thus an extended memory is introduced to store all the Pareto optimal solutions in the current generation. The selection process of *nbest* is same as that in DNPSO, but the candidate pool is the extended memory instead of all the *pBest*s of particles. This extended memory is similar to the concept of a repository proposed by Coello Coello and Fieldsend [9].

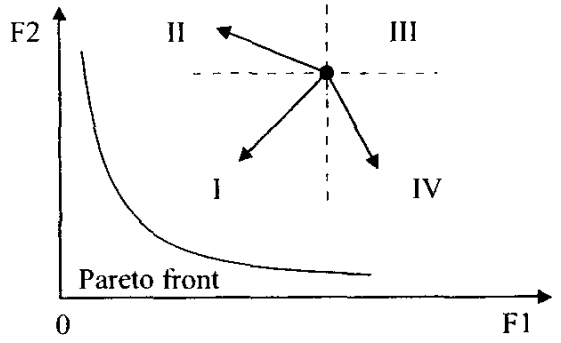

Figure 1: Possible flying directions of a particle

IV. EXPERIMENTAL RESULTS AND ANALYSIS

Zitzler *et al.* [14] proposed a group of benchmark test problems to compare evolutionary multiobjective approaches. Each of the test functions defined below is structured in the same manner and consists of three functions f_1, g, and h.

Minimize $t(\vec{x}) = (f_1(x_1), f_2(\vec{x}))$

Subject to $f_2(\vec{x}) = g(x_2,...,x_n) \cdot h(f_1(x), g(x_2,...,x_n))$

Where $\vec{x} = (x_1,...,x_n)$

Here test functions T1, T2, T3, T4, and T6 were used to test the performance of DNPSO. T5 was not tested because it is a binary string problem, which can not be handled by the current version of DNPSO. For detailed information about the functions please refer to the original paper [14].

In order to compare with other results, the same settings are used (the maximum iteration number was set to 250) however the population size was reduced to 20 due to the improved performance. For each test function, 30 runs were executed. The other parameters of PSO were set as follows: the inertia weight was [0.5 + (Rnd/2.0)]; the learning rates were 1.49445; the maximum velocity V_{MAX} was set to the dynamic range of the particle on each dimension. The size of the extended memory is 200 and the neighbor size is set to 10, which means every time nBest will be selected from the 10 nearest Pareto optimal solutions in the 200-solution pool.

TABLE 1: NUMBER OF FITNESS EVALUATIONS OF DIFFERENT ALGORITHMS

	SPEA	DNPSO	m-DNPSO
Population	100	100	20
Generations	250	250	250
Fitness evaluations	25000	25000	5000

To compare the results, the outcomes of the 30 runs were unified, and then the dominated solutions were removed from the union set. The simulation results of Strength Pareto Evolutionary Algorithm (SPEA) [15] were downloaded from http://www.tik.ee.ethz.ch/~zitzler/testdata.html [14]. DNPSO is the original dynamic neighbor PSO [10], and m-DNPSO is the modified DNPSO proposed in previous section. Figures 2-6 show the graphic results of the simulations.

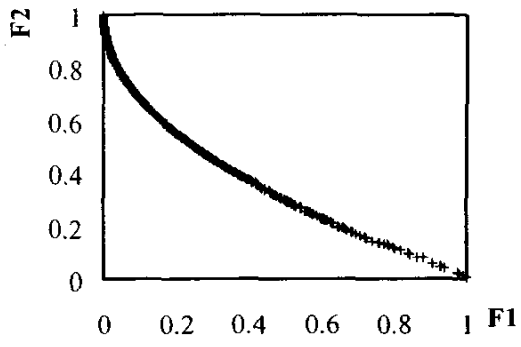

Figure 2: Test function T1

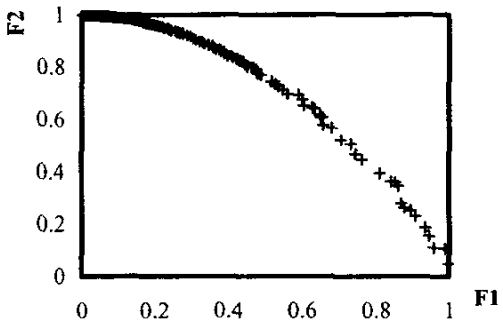

Figure 3: Test function T2

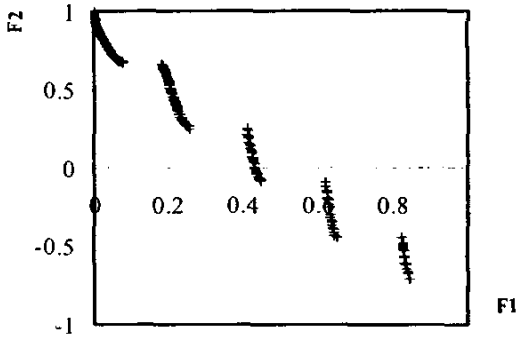

Figure 4: Test function T3

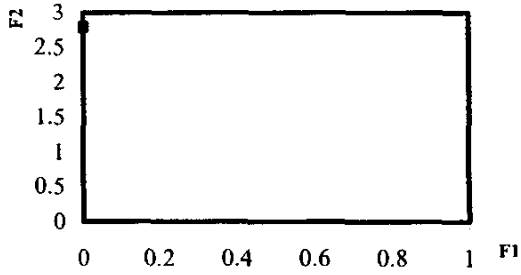

Figure 5: Test function T4

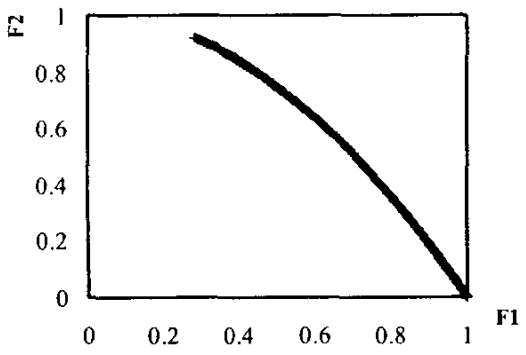

Figure 6: Test function T6

The goal of most evolutionary based optimization techniques is to get better results faster and cheaper. However, in multiobjective optimization, the definition of better results is more complicated. In general, the following creteria should be considered [14]:

1. The distance of the resulting non-dominated solution set to the Pareto front should be minimized.
2. A good (in most cases uniform) distribution of the resulting non-dominated solution set should occur.
3. A wide range of the Pareto front should be covered by the resulting non-dominated solutions set.

Quantitative metrics that formalize the above creteria are still under development. From Figures 2-6, it can be shown that the m-DNPSO can cover the majority of the Pareto Front and the distribution is satisfactory in test functions T1, T2 T3 and T6. Here another two quantitative methods are used to compare the performace: the number of non-dominated solutions and the C metric.

The C metric is defined as follows [14]:

Definition 4 (C Metric): Let X', $X'' \subseteq X$ be two sets of decision vectors, the function C maps the ordered pair (X', X'') to the interval [0, 1]

$$C(X', X'') = \frac{|a'' \in X''; \exists a' \in X' : a' \leq a''|}{|X''|}$$

The value C $(X', X'') = 1$ means that all solutions in X'' are dominated by or equal to solutions in X'. While, the value C $(X', X'') = 0$ means none of the solutions in X'' are covered by the set X'.

Table 2 shows the number of non-dominated solutions obtained by SPEA, DNPSO and m-DNPSO. From Table 2, it can be seen that m-DNPSO gets many more non-dominated solutions in a significantly fewer fitness evaluations for functions T1, T2, T3 and T6. Tables 3 and 4 show the C Metric among the different algorithms. They clearly show that almost all of solutions found by m-DNPSO are better than that in DNPSO and SPEA for functions T1, T2, T3, and T6. However, for function T4, neither m-DNPSO nor DNPSO got better results.

TABLE 2: NUMBER OF NON-DOMINATED SOLUTIONS FOUND IN 30 RUNS.

Function	SPEA	DNPSO	m-DNPSO
T1	204	925	1293
T2	112	424	515
T3	202	436	825
T4	156	6	2
T6	22	1847	5998

TABLE 3: C METRIC OF M-DNPSOPSO AND DNPSO

Function	C (DNPSO, m-DNPSO)	C(m-DNPSO, DNPSO)
T1	0.0	1.0
T2	0.014	1.0
T3	0.004	1.0
T4	0.0	1.0
T6	0.0	1.0

TABLE 4: C METRIC OF M-DNPSOPSO AND SPEA

Function	C (SPEA, m-DNPSO)	C(m-DNPSO, SPRA)
T1	0.0	1.0
T2	0.0	1.0
T3	0.0	0.980
T4	1.0	0.0
T6	0.0	1.0

V. CONCLUSION

This paper presents a modified dynamic neighborhood particle swarm optimization algorithm for multiobjective optimization problems. Compared to the traditional PSO, there are following modifications in the m-DNPSO.

- The update of *pBest: pBest* is the best position in a particle's history. Only when a new solution dominates the current *pBest*, is the *pBest* updated.
- An extended memory is used to store all the Pareto optimal solutions currently found by the population.
- The selection of *nBest*: First a neigborhood pool of current particles is selected from the extended memory by a distance measure. All but one fitness value is used to calculate the distance, and the *m* nearest solutions from current particle (m is the neighbor size) compose the neighborhood pool. The last fitness value in MO problems is used to determine the *nBest* from the neighbors.

Compared with previous methods, the m-DNPSO gets better results in a shorter time for the tested benchmark functions. It is demonstrated that dynamic neighborhood PSO with extended memory is an efficient and effective method to locate the Pareto front of MO problems. However, for some more complex problems, m-DNPSO still need to improve its performance. And the parameter settings under different circumstances are still under investigation.

REFERENCES

[1] Zitzler, Eckart, "Evolutionary algorithms for multiobjective optimizations: methods and applicaions." Ph. D. Thesis Swiss Federal Institute of Technology, Zurich, 1999.

[2] Fonseca, C. M. and Fleming, P. J., "An overview of evolutionary algorithms in multiobjective optimization." *Evolutionary Computation*, vol. 3, no. 1, pp. 1-16, 1995.

[3] Coello Coello, C. A., "A comprehensive survey of evolutionary-based multiobjective optimization techniques." *Knowledge and Information Systems*, vol. 1, no. 3, pp. 269-308, Aug. 1999.

[4] Eberhart, Russell C. and Shi, Yuhui. Particle swarm optimization: developments, applications and resources. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001), Seoul, Korea. 2001.

[5] Kennedy, J., Eberhart, R. C., and Shi, Y., *Swarm intelligence* San Francisco: Morgan Kaufmann Publishers, 2001.

[6] Eberhart, Russell C. and Kennedy, James. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995.

[7] Kennedy, James and Eberhart, Russell C. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995.

[8] Shi, Yuhui and Eberhart, Russell C. A modified particle swarm optimizer. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Piscataway, NJ. pp. 69-73, 1998.

[9] Coello Coello, Carlos A. and Lechuga, Maximino Salazar. MOPSO: a proposal for multiple objective particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. 2002.

[10] Hu, Xiaohui and Eberhart, Russell C. Multiobjective optimization using dynamic neighborhood particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA, 2002.

[11] Parsopoulos, Konstantinos E. and Vrahatis, Michael N. Particle swarm optimization method in multiobjective problems. Proceedings of the ACM Symposium on Applied Computing 2002 (SAC 2002), pp. 603-607, 2002.

[12] Ray, T. and Liew, K. M., "A swarm metaphor for multiobjective design optimization," *Engineering Optimization*, vol. 34, no. 2, pp. 141-153, 2002.

[13] Fieldsend, Jonathan E. and Singh, Sameer. 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. pp. 37-44, 2002.

[14] Zitzler, E., Deb, K., and Thiele, L., "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173-195, 2000.

[15] Zitzler, E. and Thiele, L. An evolutionary algorithm for multiobjective optimization: the strength pareto approach. Technical Report 43. 1998. Zurich, Switzerland, Computer Engineering and Communication Networks Lab (TIK), Swiss Federal Institute of Technology (ETH).