Multiobjective Optimization Using Dynamic Neighborhood Particle Swarm Optimization

Xiaohui Hu [1, 2] and Russell Eberhart [2]

[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

Abstract: This paper presents a Particle Swarm Optimization (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. Several benchmark cases were tested and showed that PSO could efficiently find multiple Pareto optimal solutions.

I. INTRODUCTION

In the real world, there are many problems involving multiple objectives, which should be optimized simultaneously. Thus multiobjective optimization is a very important research topic both for scientists and engineers and there are still many open questions in this area. General multiobjective optimization problems can be defined in the following format.

Optimize

$$\vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), ..., f_k(\vec{x})], \qquad \vec{x} = (x_1, x_2 ... x_n) \in \Re^N$$

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

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

For multiobjective optimization problems, objective functions may be optimized separately from each other and the best solution can be found for each objective dimension. However, suitable solutions for all the functions can seldom be found. This is because in most cases the objective functions are in conflict with each other. It results in there being a group of alternative solutions which must be considered equivalent in the absence of information concerning the relevance of each objective relative to the others. i.e., there is no single optimal value as in single-objective optimization.

The family of solutions of a multiobjective optimization problem is composed of all those potential solutions such that the components of the corresponding objective vectors cannot be all simultaneously improved. This is known as the concept of Pareto optimality.

The concept of Pareto optimality was formulated by Vilfredo Pareto in 19th century. We say that a point $\vec{x}^* \in F$ (where F is the feasible solution space) is Pareto optimal if for every $\vec{x} \in F$ either

$$(f_i(\vec{x}) = f_i(\vec{x}^*)) \ \forall i \in I$$

Or there is at least one $i \in I$ (where I is the objective dimension) such that

$$f_i(\vec{x}) > f_i(\vec{x}^*)$$

In words, this definition says that $\vec{x}^*$ is Pareto optimal if there is no feasible vector $\vec{x}$ that would decrease some objective values without causing a simultaneous increase in at least one other objective value [1]. The Pareto optimum usually gives a group of solutions called non-inferior or non-dominated solutions instead of a single solution.

Traditional optimization techniques, such as gradient-based methods, are difficult to extend to the true multiobjective case, because they were not designed to deal with multiple optimal solutions. In most cases, multiobjective problems have to be scaled to a single objective problem before the optimization [4]. Thus the result produces a single Pareto optimum for each run of the optimization process and the result is highly sensitive to the weight vector used in the scaling process. Moreover, in many cases, multiobjective optimizations prefer to provide a group of Pareto optimal solutions for the decision maker.

Because evolutionary computation algorithms deal with a group of candidate solutions, it seems natural to use evolutionary computation algorithms in multiobjective optimization problems to find a group of Pareto optimal solutions simultaneously. In the past several years, many evolutionary algorithms based optimizations have been developed. There are many papers that have reviewed the evolutionary based multiobjective optimization techniques [1,4,8]. Most of them are based on genetic algorithms. This paper shows a new approach to solve multiobjective optimization with particle swarm optimization.

II. MODIFIED PARTICLE SWARM OPTIMIZATION

Particle swarm optimization (PSO) is an evolutionary computation technique developed by Kennedy and Eberhart [2,6]. It uses the common evolutionary computation

0-7803-7282-4/02/$10.00 ©2002 IEEE

techniques: 1. It is initialized with a population of random solutions. 2. It searches for the optimum by updating generations. 3. The reproduction is based on the old generations. In PSO, the potential solutions, called particles, are "flown" through the problem space by following the current optimal 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 so far has been achieved (p_{id}, or *pBest*), and the population global (or local neighborhood, in the neighborhood version of the algorithm) location (p_{gd}, or *gBest*) at which the best fitness so far has been achieved. Equation 2 updates each particle's position in solution hyperspace. The two random numbers are independently generated. The use of the inertia weight w has provided improved performance in a number of applications [10].

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

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

Sharing many characteristics with other evolutionary algorithms, PSO could be a potential method for multiobjective optimization. However, basic global and local version PSO algorithms are not suitable for there is no absolute global optimum in multiobjective functions. It is not easy to define a single *gBest* or *lBest* during each generation.

Compared with genetic algorithms (GAs), the information sharing mechanism in PSO is significantly different. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only *gBest* (or *lBest*) gives out the information to others. It is a one-way information sharing mechanism. The evolution only looks for the best solution. All the particles tend to converge to the best solution quickly even in the local version.

However, due to the point-centered characteristics, global PSO is unable to locate the Pareto front, which includes multiple optimal points. The neighborhood (local) version does not work either, because the neighbors are predefined and they often only refine the search near the optimum.

Here a dynamic neighborhood PSO is presented. In each generation, after calculating distances to every other particle, each particle finds its new neighbors. Among the new neighbors, each particle finds the local best particle as the *lBest*. The problem is how to define the distance and how to define the local best particle.

Retaining generalization, two-objective continuous numeric optimizations are used to demonstrate the dynamic neighborhood PSO.

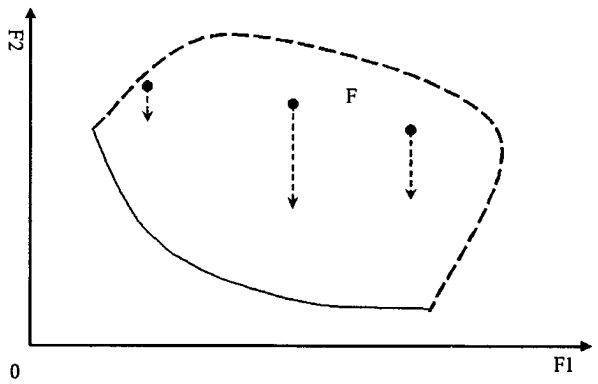

Figure 1: an example of problem with two objective functions. The Pareto front is marked with solid line

In two-dimensional fitness value space, the Pareto front is the boundary of fitness value region, it includes the combination of continuous or discontinuous lines and/or points. For a minimization problem, the boundary should be located at lower left side of the fitness space. If the first fitness values are fixed, only optimize the second objective function, and the final solution should be "dropped" onto the boundary line, which includes the Pareto front. So the algorithm used to search for local optima in each generation is defined as follows:

1. Calculate the distances of the current particle from other particles in the fitness value space of the first objective function (not the variable space).
2. Find the nearest m particles as the neighbors of the current particle based on the distances calculated above (m the neighborhood size).
3. Find the local optima among the neighbors in terms of the fitness value of the second objective function.

Another modification of the PSO is the update of the *pBest*. The *pBest* is the best position in particle's history. Only when a new solution dominates the current *pBest*, is the *pBest* updated.

Based on the above modifications, the PSO algorithm successfully finds multiple optimal solutions and locates the Pareto front.

III. PARAMETER CONTROL

The neighbor size m may affect the number of non-dominated solutions. We use $m = 2$, so a neighborhood is the particle and its two closest particles in fitness value space of the first fitness function.

Another question is how to pick up the objective function to optimize and the one to be fixed. For this empirical study, we always fix the relatively simple objective function and optimize the difficult one, here we always fix the first function f_1 and optimize f_2. We still do not know how this

process affects the final performance of the algorithm. It may highly depend on the problem itself.

In traditional PSO, the population size of PSO is often set between 10 and 40. However, under the multiobjective environment, the number of non-dominated solutions is directly linked to the population size. So a larger population size is preferred.

In PSO, there are not many parameters that need to be tuned. Only the following parameters in equation (1) and (2) need to be taken care of: maximum velocity V_{MAX} , inertia weight, cognition learning rate c_1 and social learning rate c_2 . Parameter settings were used as before [3,5]. 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.

IV. EXPERIMENTS AND RESULTS

This dynamic neighborhood PSO was applied to solve several test problems. The PSO was implemented in Java.

The first group of test problems was described by Lis and Eiben. [7]

- Test case 1,

  Minimize $f_1(x,y) = (x^2 + y^2)^{1/8}$

  $f_2(x,y) = ((x-0.5)^2 + (y-0.5)^2)^{1/4}$

  Where $-5 \le x \le 10$,

After 200 generations 94 non-dominated solutions out of 100 solutions were found, the results are shown in Figure 2.

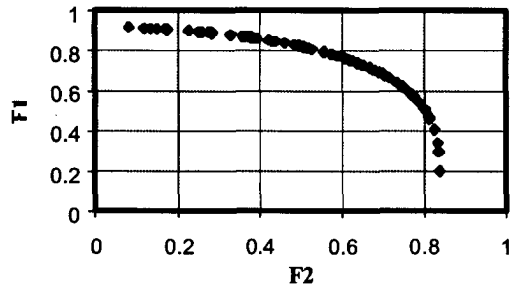

Figure 2: Results of test case 1 in terms of F1 and F2 values.

- Test case 2

  Minimize

$$f_1(x) = \begin{cases} -x & if \quad x \le 1 \\ -2+x & if \quad 1 < x \le 3 \\ 4-x & if \quad 3 < x \le 4 \\ -4+x & if \quad x > 4 \end{cases}$$

$$f_2(x) = (x-5)^2$$

Where $-100 \le x \le 100$,

After 200 generations 96 non-dominated solutions out of 100 solutions were found, the results are shown in Figure 3.

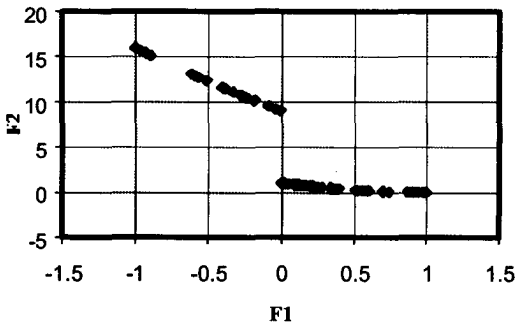

Figure 3: Results of test case 2 in terms of F1 and F2 values.

The second group of test problems was described by Zitzler [8,9]. Each of the test functions defined below is structured in the same manner and consists of three functions f1, 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 $\vec{x} = (x_1,...,x_n)$

The population size used in PSO was 200. And the iteration number was 500.

- Test function $\mathbf{t}_1$ has a convex Pareto-optimal front

$$f_1(x_1) = x_1$$

$$g(x_2,...,x_n) = 1 + 9 \cdot (\sum_{i=2}^{n} x_i)/(n-1)$$

$$h(f_1, g) = 1 - \sqrt{f_1 / g}$$

Where $n = 30$ and $x_i \in [0,1]$

After 500 iterations, 88 non-dominated solutions out of 200 particles were found. The results show in Figure 4.

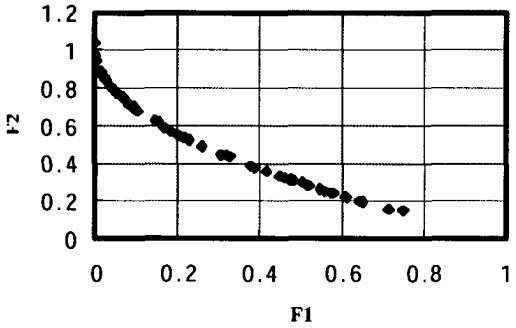

Figure 4: Results of test case $\mathbf{t}_1$ in terms of F1 and F2 values.

- Test function $\mathbf{t}_2$ has a convex Pareto-optimal front

$$f_1(x_1) = x_1$$
$$g(x_2,\ldots,x_n) = 1+9\cdot(\sum_{i=2}^{n} x_i)/(n-1)$$
$$h(f_1,g) = 1-(f_1/g)^2$$

Where $n=30$ and $x_i \in [0,1]$

After 500 iterations, 66 non-dominated solutions out of 200 particles were found. The results are shown in Figure 5.

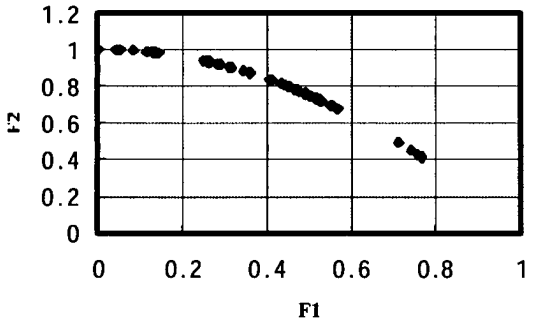

Figure 5: Results of test case t_2 in terms of F1 and F2 values.

- Test function t_3 has a convex Pareto-optimal front

$$f_1(x_1) = x_1$$
$$g(x_2,\ldots,x_n) = 1+9\cdot(\sum_{i=2}^{n} x_i)/(n-1)$$
$$h(f_1,g) = 1-\sqrt{f_1/g}-(f_1/g)\sin(10\pi f_1)$$

Where $n=30$ and $x_i \in [0,1]$

After 500 iterations, 63 non-dominated solutions out of 200 particles were found. The results are shown in Figure 6.

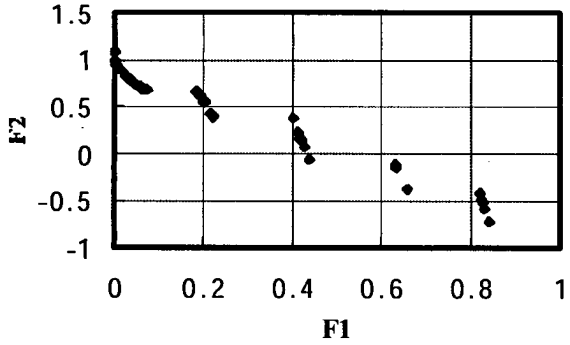

Figure 6: Results of test case t_3 in terms of F1 and F2 values.

- Test function t_4 has a convex Pareto-optimal front

$$f_1(x_1) = x_1$$
$$g(x_2,\ldots,x_n) = 1+10(n-1)+\sum_{i=2}^{n}(x_i^2-10\cos(4\pi x_i))$$
$$h(f_1,g) = 1-\sqrt{f_1/g}$$

Where $n=30$, $x_1 \in [0,1]$ and $x_2,\ldots x_n \in [-5,5]$

After 500 iterations, 152 non-dominated solutions out of 200 particles were found. The results are shown in Figure 7.

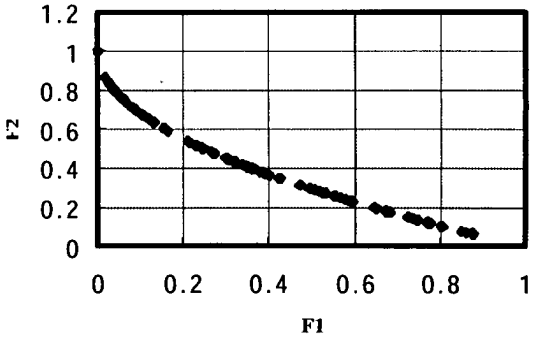

Figure 7: Results of test case t_4 in terms of F1 and F2 values.

- Test function t_6 has a convex Pareto-optimal front

$$f_1(x_1) = 1-\exp(-4x_1)\sin^6(6\pi x_1)$$
$$g(x_2,\ldots,x_n) = 1+9\cdot((\sum_{i=2}^{n} x_i)/(n-1))^{0.25}$$
$$h(f_1,g) = 1-(f_1/g)^2$$

Where $n=10$ and $x_i \in [0,1]$

After 500 iterations, 145 non-dominated solutions out of 200 particles were found. The results are shown in Figure 8.

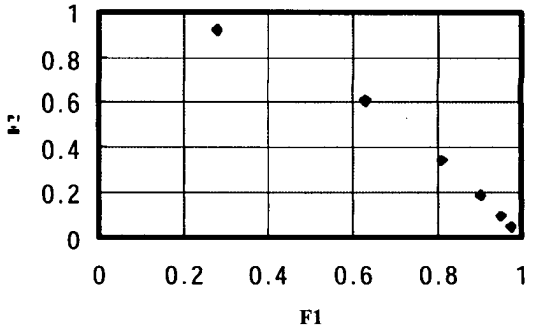

Figure 8: Results of test case t_6 in terms of F1 and F2 values.

In order to compare different optimization techniques, some notion of performance must be involved. However, there is no well-established ubiquitous performance metric for multiobjective optimization problems due to complexity and different user demands. Shown above are some graphical results. Visually compared to other results [7, 8, 9], PSO showed the potential to be a method to multiobjective optimization problems. The performance metrics are still under investigation.

V. CONCLUSION

This paper presents a particle swarm optimization algorithm for multiobjective optimization. Compared to the traditional PSO, there are three modifications in this dynamic neighborhood version:

1. Dynamic neighbors: Each particle has different neighbors in each generation based on the fitness values.
2. New *pBest* updating strategy: Only those solutions which dominate the current *pBest* will be counted.
3. One-dimension optimization: the algorithm only optimizes on one objective in each run.

It is demonstrated that dynamic neighborhood PSO is an efficient and general method to locate the Pareto front of multiobjective optimization problems. The advantage of the PSO method is that it is easy to implement and has few parameters that need to be adjusted.

This paper represents only the first step in the investigation of solving multiobjective parameter optimization problems using particle swarm optimization. Further study and investigation are needed to test the ability of PSO. The following problems need to be considered. 1. The parameters and their affect on the performance of the optimization should be studied in more detail. 2. In the process, PSO only optimize on one objective, thus how to select the objective and how it affects the result should be considered. 3. The distribution of the solutions, a better algorithm will give out evenly distributed Pareto optimal solutions or desired distributed solutions. 4. The current PSO version only deals with multiobjective optimization without constraints. How to deal with constraints is still a problem. 5. The hypothesis used about one-dimension optimization is based on two-objective function optimization. Can it be applied to multiple (greater than two) objective optimization problems? The comparison between the PSO algorithm and other evolutionary algorithm approaches also should be investigated.

REFERENCES

[1] 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.

[2] Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory," Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, pp. 39-43, 1995.

[3] Eberhart, R. C. and Shi, Y. Tracking and optimizing dynamic systems with particle swarms, Proc. Congress on Evolutionary Computation 2001, Seoul, Korea, pp. 94-97, 2001.

[4] 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.

[5] Hu, X. and Eberhart, R. C. Tracking dynamic systems with PSO: where's the cheese?, Proceedings of the Workshop on Particle Swarm Optimization, Indianapolis, IN. 2001.

[6] Kennedy, J. and Eberhart, R. C. Particle swarm optimization, Proc. IEEE Int'l Conf. on Neural Networks, Piscataway, NJ, pp. 1942-1948, 1995.

[7] Lis, J. and Eiben, A. E. A multi-sexual genetic algorithm for multiobjective optimization, Proceedings of the 1997 International Conference on Evolutionary Computation, Indianapolis, IN, USA, pp. 59-64, 1997.

[8] Zitzler, E. Evolutionary algorithms for multiobjective optimizations: methods and applications, Ph.D. thesis, Swiss Federal Institute of Technology, Zurich, 1999.

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

[10] Shi, Y. Eberhart R. C. A modified particle swarm optimizer. Proceedings of the IEEE International Conference on Evolutionary Computation. Piscataway, NJ, pp. 69-73, 1998