Multiobjective Particle Swarm Optimization

Jacqueline Moore
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama 36849
jmoore@eng.auburn.edu

Richard Chapman
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama 36849
chapman@eng.auburn.edu

Gerry Dozier
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama 36849
gvdozier@eng.auburn.edu

Abstract-Evolutionary algorithms (EAs) are search procedures based on natural selection [2]. They have been successfully applied to a wide variety of optimization problems [4]. Particle Swarm Optimization (PSO) [1,7] is a new type of evolutionary paradigm that has been successfully used to solve a number of single objective optimization problems (SOPs). However, to date, no one has applied PSO in an effort to solve multiobjective optimization problems (MOPs). The purpose of our research is to demonstrate how PSO can be modified to solve MOPs. In addition to showing how this can be done, we demonstrate its effectiveness on two MOPs.

1.0 Particle Swarm Optimization

PSO [6] is based on the hypothesis that members of a population (swarm) can profit from their past experiences and the experiences of other individuals (particles). During the exploration of a search space, each particle has access to two pieces of information: the best potential solution (PS) that it has encountered and the best PS contained within its neighborhood. This information is used to direct the search.

The particles of a swarm are arranged in a ring topology [6]. In this topology, the neighborhood of particle *i* consists of particles i-1, i, and i+1. Figure 1 shows the topology of a five particle swarm. Using Figure 1 as an example, the neighborhood of 0 would be 4, 0, and 1. The neighborhood of 4, would be 3, 4, and 0.

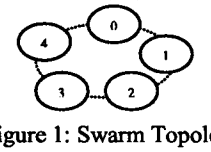

Figure 1: Swarm Topology

Figure 2 provides an illustration of a particle. A particle is composed of three vectors, *x*, *p* and *v* [6]. The *x*-vector contains the current PS. The value, χ_i ,represents the fitness assigned to x_i by the objective function. The *p*-vector contains the location of the best PS discovered by a particle. The value, ρ_i, is the fitness assigned to the *p*-vector. Finally, the *v*-vector, known as the velocity vector, is used to determine the next PS to be evaluated.

x-vector:
$x_i=(x_{i,1},x_{i,2},\ldots,x_{i,m})$
χ_i= evaluate(x_i)
p-vector:
$p_i=(p_{i,1}, p_{i,2},\ldots,p_{i,m})$
ρ_i= evaluate(p_i)
v-vector:
$v_i=(v_{i,1}, v_{i,2},\ldots,v_{i,m})$

Figure 2: Elements Of A Particle

PSO begins by randomly initializing the *x* and *v* vectors. Initially, the *p*-vector is set equal to the *x*-vector. Each time the *x*-vector of a particle is updated, χ_i is compared to ρ_i. If χ_i is less than ρ_i, p_i is set equal to x_i. Therefore, the *p*-vector always contains the best PS discovered by a particle. To find the best PS in the neighborhood of some particle *i*, the ρ values of i's neighborhood are compared, and the index of the particle that has the best ρ is returned. Each parameter, *d*, of the *x*-vector is then updated using the following equation: $v_{i,d} = v_{i,d} + \eta\,(\,(\varphi_{i,d}\,(p_{i,d}-x_{i,d})) + (\psi_{i,d}\,(p_{g,d}-x_{i,d}))\,)$ [1,5]. In this equation, $\varphi_{i,d}$ and $\psi_{i,d}$ are random numbers between 0.0 and 1.0, η represents the learning rate, and *g* represents the particle in the neighborhood with the best ρ. The new velocity value, $v_{i,d}$, is then checked to see if it is within the bounds [-*vmax*, *vmax*]. The value, *vmax*, denotes the maximum amount of change that can be applied to the values of the parameters. The *x*-vector is then updated by adding the *v*-vector. The updated *x*-vector represents a new PS that is located somewhere between x_i, p_i, and p_g. Figure 3 provides a pseudocode version of PSO.

```
Update_Particle(i) {
  χi = evaluate(i);
  If χi < ρi then
    {pi = xi; ρi = χi;}
  g = best_of((i-1 +n ) mod n, i, (i+1) mod n);
       (∀d  vi,d = vi,d +η ( (φi,d (pi,d-xi,d)) + (ψi,d (pg,d-xi,d) ) ));
  If vi,d > vmax Then vi,d = vmax
   Else If vi,d < -vmax  Then vi,d = -vmax
  xi,d=xi,d+vi,d ;
  }
```

Figure 3: Particle Swarm Optimization Algorithm

1.1 A Multiobjective Particle Swarm Optimizer

In multiobjective optimization, one is faced with the problem of simultaneously optimizing a set of objective functions. To accomplish this, the notion of preference must be initially established in order to determine when one PS dominates (or is better than) another [10]. One type of preference that has been widely used by many multiobjective optimizers is known as Pareto preference [3]. In Pareto preference, a PS, q_0, is said to dominate another PS, q_1, if $\forall_j\ f_j(q_o) \leq f_j(q_1) \wedge \exists_k\ f_k(q_0) < f_k(q_1)$ where f_j represents the j^{th} objective to be optimized [10]. When using Pareto preference, the set of nondominated PSs discovered is referred to as the Pareto set. In order to adapt PSO for multiobjective optimization, the *p*-vector was modified to keep track of all nondominated solutions (using Pareto preference) that a particle encountered as it explored the search space.

2.0 Results and Conclusions

Two experiments were conducted in order to test the effectiveness of our multiobjective particle swarm optimizer (MPSO). The functions used in the first experiment were $f_{1,1}(x,y) = (x^2+y^2)^{1/8}$ and $f_{1,2}(x,y) = ((x-0.5)^2 + (y-0.5)^2)^{1/4}$ where x, y $\in$(-5.0,10.0). Similarly the second experiment also contained two functions. The first function $f_{2,1}(x)$ was a piecewise function that returned –x when x<=1, x-2 when 1<x<=3, 4-x when 3<x<=4, and x-4 when 4<x. The second function was $f_{2,2}(x) = (x-5)^2$. For both functions x$\in$(0.0, 5.0). The functions for our experiments were taken from [8].

The performances of our MPSO for the two experiments are shown in Tables 1 and 2. In both experiments, a swarm size of 20 particles was used. In the Tables, the leftmost column indicates the values that were used for *vmax* (1, 2, 4, 8, and 16) and the top row indicates the values used for η (again we chose the values 1, 2, 4, 8, and 16). A total of 25 instances of our MPSO were compared. The value within each cell of the tables represents the average number of nondominated solutions found over 121 runs. For both experiments, one run consisted of 296 iterations.

In Table 1, one can see that the greatest average number of nondominated solutions, 137, was found with *vmax* =1, and η=8. In [8], the authors report finding only 129 nondominated solutions to the MOP of Experiment 1 using a "traditional" EA. Table 2 contains the results obtained from Experiment 2. The greatest average number of nondominated solutions, 4425, was found with *vmax* =16, and η=16. For Experiment 2, the authors of [8] reported finding only 494 nondominated solutions. Figures 4 and 5 show the graphed results of one run of our best MPSOs for the two experiments. In the Figures, the x and y axes correspond to the objective functions being optimized.

In this paper we have shown how PSO can be modified to solve MOPs. We tested our MPSO on two MOPs that were taken from [8]. Our results show that for Experiment 1 our best MPSO produced results that are comparable to those presented in [8]. The results for Experiment 2 show that our best MPSO produced results that are superior to those presented in [8]. Our future work will be devoted towards the application of our MPSO to other real world problems, particularly those in the area of Hardware/Software Codesign [9].

	1	2	4	8	16
1	63	79	108	137	106
2	37	37	49	62	39
4	30	23	22	28	17
8	30	21	15	10	4
16	29	21	15	7	3

Table 1: Results of Experiment 1

	1	2	4	8	16
1	2849	2770	2906	3017	2957
2	3249	3287	3234	3230	3089
4	3233	3487	3817	4112	4406
8	3216	3477	3838	4160	4222
16	3211	3484	3838	4255	4425

Table 2: Results of Experiment 2

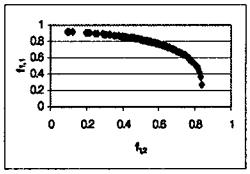

Figure 4: Graphed Results of Experiment 1

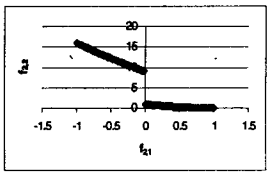

Figure 5: Graphed Results of Experiment 2

References

[1] Angeline, P. "Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences", In 7th International Conference on Evolutionary Programming, San Diego, California, Springer, 1998, pp. 601-610.

[2] Back, T. Evolutionary Algorithms in Theory and Practice, Oxford University Press, New York, 1996, pp. 7-11.

[3] Coello Coello, C. "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques", In Knowledge and Information Systems, August 1999, pp. 269-308.

[4] Goldberg, D. Genetic Algorithms in Search, Optimization &Machine Learning, Addison-Wesley, Massachusetts, 1989, pp. 106-120.

[5] Kennedy, J. "The Behavior of Particles", In 7th International Conference on Evolutionary Programming, San Diego, California, Springer, 1998, pp. 582-589.

[6] Kennedy, J. "The Particle Swarm, Social Adaptation of Knowledge", In Proceedings of the 1997 International Conference on Evolutionary Computation, IEEE, NJ, pp. 303-308.

[7] Kennedy, J., and Eberhart, R. "Particle Swarm Optimization", In Proceedings of the 1995 IEEE International Conference on Neural Networks, IEEE, NJ, pp. 1942-1948.

[8] Lis, J. and Eiben, A. "A MultiSexual Genetic Algorithm for Multiobjective Optimization", In Proceedings of the 1997 International Conference on Evolutionary Computation, Indianapolis, Indiana, 1997, pp. 59-64.

[9] Wolf, W. "Hardware-Software Co-Design of Embedded Systems", In Proceedings of the IEEE, Vol. 82, No. 7, July 1994, pp. 967-989.

[10] Yu, P. Multiple-Criteria Decision Making, Plenum Press, New York, 1985, pp. 7-10.