Application Of Particle Swarm To Multiobjective Optimization

Jacqueline Moore
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama, 36849-5347
jmoore@eng.auburn.edu

Richard Chapman
Department of Computer Science and Software Engineering
Auburn University
Auburn, Alabama, 36849-5347
chapman@eng.auburn.edu

Abstract- Evolutionary algorithms (EAs), search procedures based on natural selection (Back 1996), have been used to solve a wide variety of single and multiple objective optimization problems (Goldberg 1989). Particle Swarm Optimization (PSO) (Kennedy and Eberhart 1995) is a new type of evolutionary paradigm that has been successfully used to solve a number of single objective optimization problems (SOPs). However, there has been no application of PSO to multiobjective optimization. This paper demonstrates how PSO can be modified to solve multiobjective optimization problems (MOPs).

1 Particle Swarm Optimization

PSO (Kennedy 1997) 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 the search space, each particle has access to two pieces of information: the best potential solution (PS) that it has encountered and the best PS encountered by its neighborhood. This information is used to direct the search.

The particles of a swarm are arranged in a ring topology (Kennedy 1997). Figure 1 illustrates the topology of a five particle swarm. In this topology, the neighborhood of particle *i* consists of the particles *i*-1, *i*, and *i*+1. Using Figure 1 as an example, the neighborhood of particle 0 would consist of particles 4, 0, and 1. Similarly, the neighborhood of particle 4 would consist of particles 3, 4 and 0.

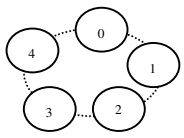

Figure 1: Swarm Topology

Figure 2 provides an illustration of the elements of a particle. A particle is composed of three vectors, *x*, *p* and *v* (Kennedy 1997). The *x*-vector contains the current PS of the particle. The value, χ_i , represents the fitness assigned to x_i by the objective function. The *p*-vector contains 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. The length of the vectors corresponds to the number of parameters in the problem.

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. For minimization problems, p_i is set equal to x_i if χ_i is less than ρ_i. For maximization problems, p_i is set equal to x_i if χ_i is greater than ρ_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: $v_{i,d} = v_{i,d} + \eta ((\varphi_{i,d} (p_{i,d}-x_{i,d})) + (\psi_{i,d} (p_{g,d}-x_{i,d})))$ (Angeline 1998, Kennedy 1998) where $\varphi_{i,d}$ and $\psi_{i,d}$ are random numbers between 0.0 and 1.0. The variable, *g*, denotes the particle in the neighborhood that has the best PS, $p_{i,d}$-$x_{i,d}$ denotes the distance between the best PS of particle *i* and its current PS, and $p_{g,d}$-$x_{i,d}$ denotes the distance between the best PS of particle's *i* neighborhood and its current PS, and η represents the learning rate.

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 to it. 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) {
    χ_i = evaluate(i);
    If χ_i < ρ_i then
      {p_i = x_i; ρ_i = χ_i;}
    g = best_of((i-1 +n ) mod n, i, (i+1) mod n);
    (∀d  v_i,d = v_i,d +η ( (φ_i,d (p_i,d-x_i,d)) + (ψ_i,d (p_g,d-x_i,d) ) ));
    If v_i,d > vmax
          Then v_i,d = vmax
    Else
          If v_i,d < -vmax
            Then v_i,d = -vmax
    x_i,d=x_i,d+v_i,d ;
} note: n represents the total number of particles
```

Figure 3: Pseudocode for Particle Swarm

The particle swarm update equation contains two components (Kennedy 1997). The cognitive component ($p_{i,d}$-$x_{i,d}$) represents the attraction that individuals have toward their own successes and the social component ($p_{g,d}$-$x_{i,d}$) represents the attraction that individuals have toward the successes found by their neighborhood. These two components are used to derive the four models of particle swarm: Full, Cognition-Only, Social-Only and Selfless. The Full model uses both components. The Cognition-Only model uses the cognitive component and Social-Only model uses the social component. The Selfless model uses the social component with one exception, g≠i. Therefore, the particles are strictly influenced by the performance of their neighbors.

1.1 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 (Yu 1985). One type of preference that has been widely used by many multiobjective optimizers is known as Pareto preference (Coello 1999). 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 is the j^{th} objective to be optimized (Yu 1985). When using Pareto preference, the set of nondominated PSs discovered is referred to as the Pareto set.

To adapt PSO for multiobjective optimization, the *p*-vector was modified to keep track of all nondominated solutions (according to Pareto preference) that a particle encountered as it explored the search space. The *p*-vector has now become a list of solutions. The updated view of a particle is shown in Figure 4.

x-vector:
$x_i=(x_{i,1},x_{i,2},\ldots,x_{i,m})$
$\chi_{i,n}$= evaluate(x_i)
p-list:
$p_{k,i}=(p_{i,1}, p_{i,2}, \ldots, p_{i,m})$
$\rho_{k,i,n}$= evaluate($p_{k,i}$)
v-vector:
$v_i=(v_{i,1}, v_{i,2},\ldots,v_{i,m})$

Figure 4 Updated View of A Particle

PSO for multiobjective optimization begins by randomly initializing the *x* and *v* vectors. Each time the *x*-vector of a particle is updated, it is compared with the solutions in the p-list to determine if it is a nondominated solution. If it is a nondominated solution, it is added to the p-list. The p-list is also constantly updated to ensure that it contains only nondominated solutions.

The exploration of the search space is guided by at most two pieces of information: the best PS discovered by an individual and the best PS discovered by its neighborhood. Since the *p*-list, formally the *p*-vector, can now contain numerous solutions, the best PS discovered by an individual is randomly selected from its list of nondominated solutions. To determine the best PS in the neighborhood, we compare the nondominated solutions found in the *p*-lists to find one that is not dominated within the neighborhood. The remainder of the algorithm proceeds as mentioned earlier.

2 Results

Two experiments were conducted in order to test the effectiveness of our multiobjective particle swarm optimizer (MPSO). The functions used for 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). 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 (Lis and Eiben 1997). The results of our MPSO are shown in Tables 1-8.

In both experiments the swarm size was set to 20. In the tables, the leftmost column indicates the values 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). The value within each cell represents the average number of nondominated solutions found over 121 runs. For both experiments, one run consisted of 296 iterations.

2.1 Results of Experiment 1

Tables 1-4 contain the results for Experiment 1. Table 1 contains the results of the Cognition-Only model, Table 2 contains the results of the Selfless model, Table 3 contains the results of the Social-Only model and Table 4 contains the results of the Full model. The best result (136) was obtained when the Full model was used with the parameters *vmax* =1, and η=8. Figure 5 shows the graphed result of one run with the best parameters.

vmax \ η	**1**	**2**	**4**	**8**	**16**
1	64	56	65	79	95
2	59	55	46	42	44
4	63	48	38	25	21
8	61	52	38	19	9
16	63	61	38	19	7

Table 1: Cognition-Only Results Experiment 1

vmax \ η	**1**	**2**	**4**	**8**	**16**
1	29	44	65	93	111
2	14	18	27	41	49
4	12	12	12	18	21
8	13	11	10	9	7
16	12	11	10	7	5

Table 2: Selfless Model Results Experiment 1

vmax \ η	**1**	**2**	**4**	**8**	**16**
1	39	47	68	94	110
2	21	23	29	42	49
4	20	18	15	19	21
8	20	17	13	10	7
16	19	16	14	9	5

Table 3: Social-Only Model Results Experiment 1

vmax \ η	**1**	**2**	**4**	**8**	**16**
1	64	80	110	136	97
2	35	38	49	60	37
4	31	22	23	27	15
8	28	20	14	10	5
16	30	21	14	7	3

Table 4: Full Model Results Experiment 1

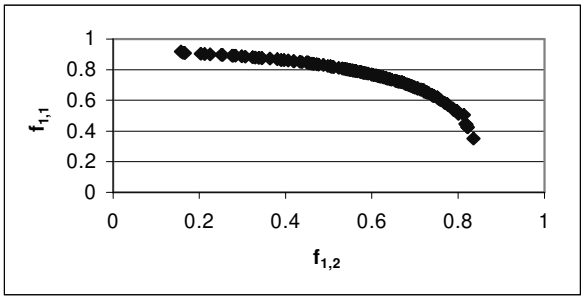

Figure 5: Graphed Results of Experiment 1

2.2 Results of Experiment 2

Tables 5-8 contain the results for Experiment 2. Table 5 contains the results of the Social-Only model, Table 6 contains the results of the Full model, Table 7 contains the results of the Selfless model and Table 8 contains the results of the Cognition-Only model. The best result (3124) was obtained when the Cognition-Only model was used with the parameters *vmax* =1, and η=1. Figure 6 shows the graphed result of one run with the above mentioned parameters.

vmax \ η	**1**	**2**	**4**	**8**	**16**
1	2227	2103	2037	1999	1750
2	2137	2063	1878	1661	1304
4	2114	2010	1798	1421	1034
8	2168	2038	1784	1332	837
16	2123	2006	1788	1316	792

Table 5: Social-Only Model Results Experiment 2

vmax \ η	**1**	**2**	**4**	**8**	**16**
1	2349	2274	2259	1909	1244
2	2346	2199	1874	1494	923
4	2362	2132	1840	1209	749
8	2352	2094	1771	1107	578
16	2306	2070	1662	1024	548

Table 6: Full Model Results Experiment 2

vmax	η				
	1	**2**	**4**	**8**	**16**
1	2683	2700	2702	2776	2671
2	2705	2687	2603	2437	1893
4	2721	2676	2595	2225	1601
8	2698	2635	2564	2132	1407
16	2658	2681	2538	2114	1381

Table 7: Selfless Model Results Experiment 2

vmax	η				
	1	**2**	**4**	**8**	**16**
1	3124	2716	2433	2033	1558
2	3061	2682	2077	1268	665
4	3045	2680	1986	1008	441
8	3084	2648	2016	915	286
16	3056	2688	2022	909	270

Table 8: Cognition-Only Model Results Experiment

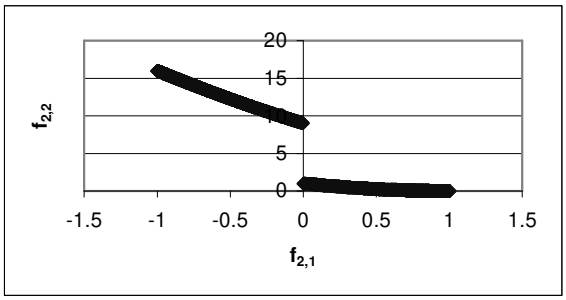

Figure 6 Graphed Results of Experiment 2

3 Conclusions

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 (Lis and Eiben 1997). We tested each model and found that for Experiment 1 the best results, in terms of the average number of nondominated solutions, was found when the Full model was used. For Experiment 2 we found that the best results were found when the Cognition-Only model was used. Our results also show that MPSO produces results that are comparable to those presented in (Lis and Eiben 1997) for Experiment 1. We also noticed that our MPSO produces results that surpass those presented in (Lis and Eiben 1997) for Experiment 2.

Our future work will be devoted towards the application of MPSO to other real world problems, particularly those in the area of Hardware/Software Codesign (Wolf 1994).

Acknowledgments

The authors would like to thank Dr. Gerry Dozier (Auburn University) for his involvement with this project.

Bibliography

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

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

Coello, Coello C. (1999) "A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniquies", In Knowledge and Information Systems, pp. 206-308.

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

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

Kennedy, J. (1997) "The Particle Swarm: Social Adaptation of Knowledge", In Proceedings of the 1997 Interational Conference on Evoluionary Computation, IEEE, NJ, pp. l303-308.

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

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

Wolf, W. (1994) "Hardware-Software Codesign of Embedded Systems", In Proceedings of the IEEE, Vol, 82, No. 7, pp. 967-989.

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