Proceedings of the 5th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases, Madrid, Spain, February 15-17, 2006 (pp85-89)

Multi-Speed Particle Swarm Optimization

BUTHAINAH S. AL-KAZEMI
Kuwait University
Department of Computer Engineering
P.O. Box 5969 Safat, 13060
KUWAIT
http://www.faculty.eng.kuniv.edu.kw/staff/bsnak.jsp

Abstract: - : This paper presents a modified version of the Particle Swarm Optimization (PSO). In the new version, particles are allowed to pick from among many positions to where it will move in the next generation. The new version outperforms the PSO algorithm on many benchmarks problems.

Key-Words: - PSO, MSPSO, Evolutionary Computation, Convergence.

1. Introduction

The *Particle Swarm Optimization (PSO)* algorithm has been successful in solving a number of continuous optimization problems 0. Unlike other evolutionary algorithms, each particle (solution) in PSO moves in the search space by constantly updating its velocity vector based on the best solutions found so far by that particle as well as others in the population (*swarm*) [1] [3]. Many authors add some features to the original PSO to produce a hybrid algorithm that is more efficient [4][5].

This paper presents a modified version of PSO, in which each particle is allowed to pick the best velocity among many that will move it to a better position. This new version will be compared with the original PSO. In the future the algorithm will be compared with the modified versions of PSO.

In section 2, we explain a modified version of the original PSO algorithm which is used as the original PSO by all users. Section 3 describes the MSPSO algorithm. Section 4 contains details of the experiments, and Section 5 describes the results obtained on benchmark problems.

2. PSO Algorithm

The PSO algorithm evolves a population of particles called swarm. Each particle updates its current velocity and position in the search space using historical information regarding its own previous best position as well as the best position discovered by all other particles or neighbouring particles. As both local and global serach were invloved, inertia weights was added by Shi and Eberhart [6] to control those searches. As particles represent solution, the size of a particle depends on the problem. Each particle needs to calculate its new velocity for each of its dimension, then this velocity is used to move the particle to a new position. Figure 1 shows the PSO algorithm.

```
Particle Swarm Optimization:
begin
    t =0;
    initialize particles P(t);
    evaluate particles P(t);
    while (termination conditions are unsatisfied)
    begin
            t = t + 1;
            update weights
            select pBest for each particle
            select gBest from P(t-1);
            calculate particle velocity P(t);
            update particle position P(t);
            evaluate particles P(t);
    end
end
```

Figure 1: Particle Swarm Optimization algorithm

In each dimension (depending on the problem being solved), particle n moves in the space using the following two equations:

$$V_{i,n}(t+1) = w * V_{i,n}(t) + C_1 * (G_i(t) - X_{i,n}(t)) + C_2 * (l_{i,n}(t) - X_{i,n}(t))$$

$$X_{i,n}(t+1) = X_{i,n}(t) + V_{i,n}(t+1)$$

Where C_1 and C_2 are random numbers, G_i is the best particle found so far (by all particles) in dimention i, and $l_{i,n}$ is the best position discovered so far by particle n in dimension i. Velocity magnitude are clipped to a predetermined maximumj value, V_{max} [7][8].

3. MSPSO

The MSPSO algorithm use the same equation for the velocity and the position as the original PSO. The difference is in the way the velocity being updated. Instead of having only one value that is used directly to update the velocity, number of velocities (m) are produced. Then, those velocities are used to calculate (m) new positions. The new positions are compared together to find the best position that a particle can move to inorder to reach the optima. By best position we mean the position that can move closer to the optimum. This is done by evaluationg the new m positions and pick the one with the best fitness.

In addition, in the original PSO, the new fitness is calculated after updating the position in all the dimension. In the MSPSO, the fitness is updated after each dimension change. Figure 2 shows the MSPSO algorithm

Figure 2: MSPSO algorithm

```
Particle Swarm Optimization:
begin
    t =0;
    initialize particles P(t);
    evaluate particles P(t);
    while (termination conditions are unsatisfied)
    begin
        t = t + 1;
        update weights
        select pBest for each particle
        select gBest from P(t-1);
        for i=1 to i=n
            for k=1 to k=m
                calculate particle velocity Vi,k(t)
                calculate particle position Pi,k(t)
            choose Vi,k(t) that optimize the fitness
        update particle position Pi(t);
        evaluate particles P(t);
        i=i+1
    end
end
```

4. Experiments

The modified PSO and the MSPSO were tested on four different benchmark problems described below. Both algorithms have the same parameters settings: population size = 10, Vmax = 5, and the maximum generation = 200. For MSPSO, the new parameter m = 10. That is, in each dimension, 10 new velocities are being calculated and the best one of them is chosen.

The following functions were used as test functions, each of which was to be minimized:

1. Generalized Sphere Function:

$$f(x) = \sum_{i=1}^{n} x_i^2$$

Where x is an n-dimensional real-valued vector and xi is the ith element of that vector.

2, Generalized Rastrigin Function :

$$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$$

3. De Jong Function F2:

$$f(x_1, x_2) = 100(x_1^2 - x_2)^2 + (1 - x_1)^2,$$

Where $-2.048 \le Xi \le 2.048$

4. De Jong Function F5:

$$f(x_1, x_2) = \frac{1}{1/k + \sum_{j=1}^{25} f_j^{-1}(x_1, x_2)},$$

Where $f_j(x_1, x_2) = c_j + \sum_{i=1}^{2} (x_i - a_{ij})^6,$

Where $-65.536 \le Xi \le 65.536$, k = 500,

cj = j

$$[a_{ij}] = \begin{bmatrix} -32 & -16 & 0 & 16 & 32 & -32 & \dots & 0 & 16 & 32 \\ -32 & -32 & -32 & -32 & -32 & -16 & \dots & 32 & 32 & 32 \end{bmatrix}$$

5. Results

Table 1 lists the average results for both PSO and MSPSO for the computational problem of Circle and Rastrigin 2D. Table 2 displays average results for PSO and MSPSO algorithms for the DeJongF2 and DeJongF5 equations.

	Circle		Rastrigin 2D	
Gen	PSO	MSPSO	PSO	MSPSO
5	1117.8698	952.9940	10.7374	4.2734
10	582.7662	176.7972	8.3023	0.4683
15	229.4220	3.2271	7.1324	0.0906
20	37.7236	0.0052	6.2764	0.0037
25	4.0610	0.0000	4.8099	0.0000

Table 1 Average fitness for Circle, Rastrigin 2D over 25 Generations

	DeJongF2		DeJongF5	
Gen	PSO	MSPSO	PSO	MSPSO
5	7.4910	3.0625	222.7761	106.9031
10	2.2978	0.0387	40.4220	4.3363
15	1.9998	0.0034	23.6380	2.0362
20	1.0349	0.0003	13.9231	1.7920
25	0.7210	0.0001	12.6756	1.7917

Table 2 Average fitness for DeJong F2 and DeJong F5 over 25 Generations

Figure 3, 4, 5, and 6 shows the behaviour of both PSO and MSPSO for functions: Circle, Rastrigin, DeJong F2, and DeJong F5, respectively.

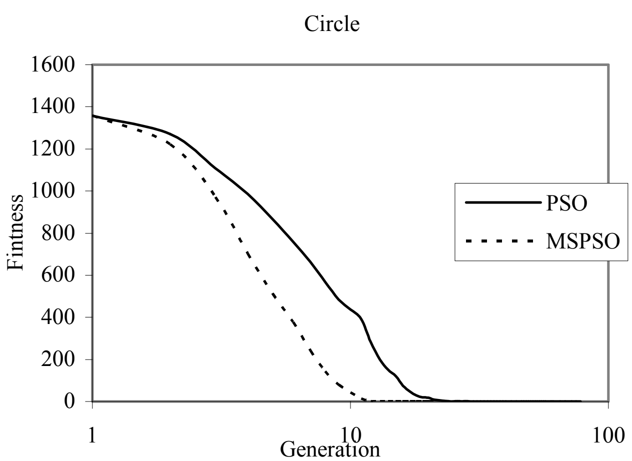

Figure 3 Comparison of evolution of best fitness for PSO and MSPSO for circle function

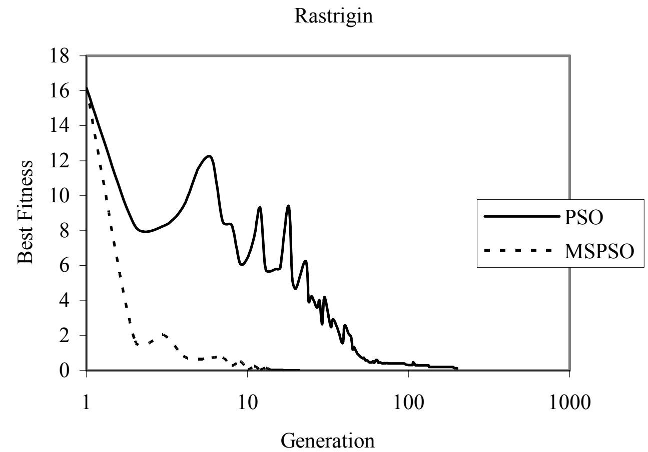

Figure 4 Comparison of evolution of best fitness for PSO and MSPSO for Rastrigin function (2D)

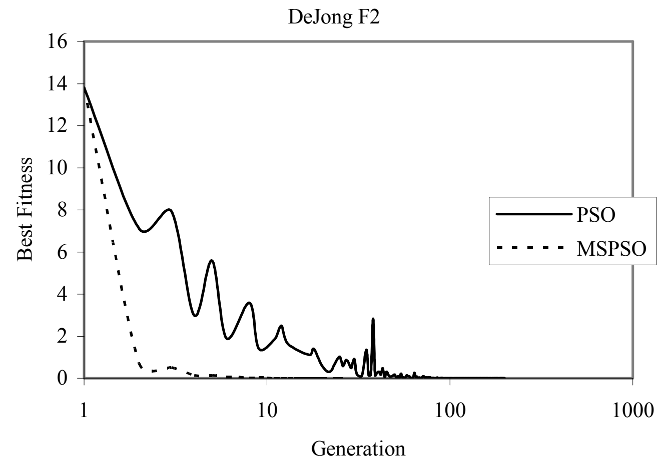

Figure 5 Comparison of evolution of best fitness for PSO and MSPSO for DeJongF2

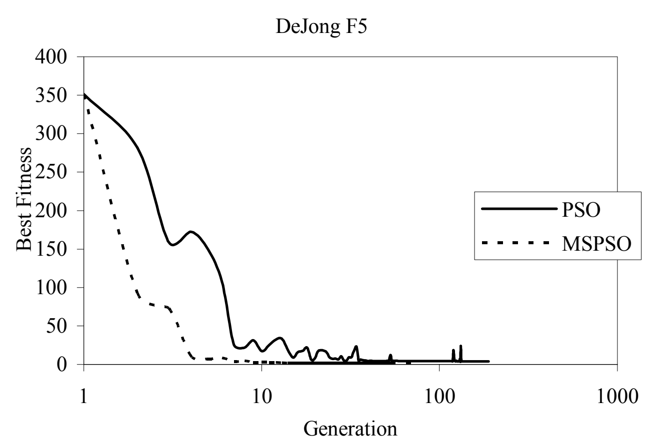

Figure 6 Comparison of evolution of best fitness for PSO and MSPSO for DeJongF5

Table 3 shows the result of applying PSO and MSPSO to Rastrigin function in three different dimensions: 10, 20, and 30. Figures 7, 8, and 9 shows the performance of algorithms in Rastrigin 10, 20, and 30 respectively.

	Rastrigin 10		Rastrigin 20		Rastrigin 30	
	Gen	Best Fitness	Gen	Best Fitness	Gen	Best Fitness
PSO	200	14.59906	200	49.98067	200	116.0156
MSPSO	43	0.000045	60	0.497507	70	0.994974

Table 3 Average fitness for Rastrigin 10, Rastrigin 20, Rastrigin 30 over 25 Generations

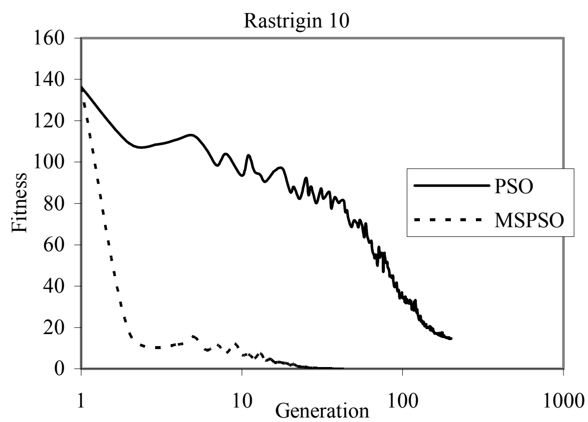

Figure 7 Comparison of Evolution of Best fitness for PSO and MSPSO for function – Rastrigin 10

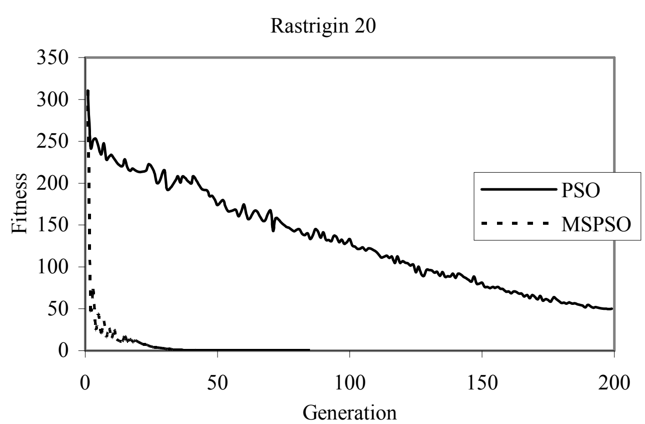

Figure 8 Comparison of Evolution of Best fitness for PSO and MSPSO for function – Rastrigin 20

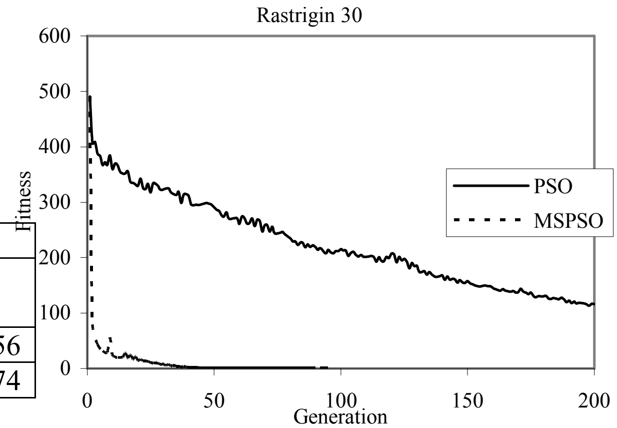

Figure 9 Comparison of Evolution of Best fitness for PSO and MSPSO for function – Rastrigin 30

6. Problem Solution

As table 1 and 2 show, MSPSO was able to reach the optimum faster than PSO for all the test functions. In circle function, PSO was able to reach the optimum in mean generation 78, while MSPSO already reached the optimum in generation 21. In addition, for Rastrigin function, PSO did not reached the optimum with the experimental settings, while MSPSO reached the optimum in generation 2. For DeJong F2, PSO reached the optimum in generation 177 and for DeJong F5 it did not reached the optimum, while MSPSO reached the optimum in generation 25 for DeJong F2, and near the optimum in DeJong F5. Figures 3 to 6 clearly shows the convergence of MSPSO to the optimum faster then the PSO algorithm.

From table 3, it is observed that MSPSO was able to maintain the convergence to the optimum faster than PSO when the problem size increased. For Rastrigin 10, MSPSO was able to reach the optimum, while PSO failed even to come closer to the optimum. As the problem size increased, MSPSO was still able to reach near the optimum (according to the problem settings), but PSO slowed down and was unable to converge. Figures 7 to 9, clearly shows the performance of both MSPSO and PSO.

Not shown in the paper, more tests were made in Rastrigin 10, 20, and 30 by changing the population's size and the maximum generation for both PSO and MSPSO. PSO still used many generations to reach near the optimum, while MSPSO reached the optimum with les number of generation.

7. Conclusions

The results presented in the previous section show that MSPSO is a powerful algorithm that succeeds in solving the benchmark optimization problems using a small number of generation to reach the best fitness.

The MSPSO algorithm was faster in converging to the optimum comparing it to the modified version of PSO. The new metric, which was picking the best velocity among multiple speeds, made the convergence faster. In addition, updating the fitness in each dimension will increase the diversity, and thus escape from local and global optima.

8. Future Work

As the MSPSO overcomes the PSO algorithm, it will be tested to the other modified versions to find how good it is as compared to MSPSO algorithm.

References

[1] J. Kennedy and R. C. Eberhart, *Particle Swarm Optimization*, Proc. IEEE International conference on Neural Networks, 1995.

[2] R. C. Eberhart and J. Kennedy, *A new optimizer using particle swarm theory*, Proc. the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 1995.

[3] Shi, Y. and Eberhart, R. C., *Empirical study of particle swarm optimization*, Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Piscataway, NJ, pp. 1945-1950, 1999.

[4] Suganthan, P. N., *Particle swarm optimiser with neighbourhood operator*, Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), Piscataway, NJ. pp, 1958-1962, 1999.

[5] Al-kazemi, B. and Mohan, C. K., *Multi-phase generalization of the particle swarm optimization algorithm*, Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA, 2002.

[6] Y. Shi and R. Eberhart, *A Modified Particle Swarm Optimizer*, Proc. IEEE International conference on Evolutionary Computation, Anchorage, Alaska, 1998.

[7] Shi, Y. and Eberhart, R. C. *Parameter selection in particle swarm optimization*, Evolutionary Programming VII, Proceedings of the Seventh Annual Conference on Evolutionary Programming, New York, 1998.

[8] M. Clerc, *The Swarm and the Queen: Towards a Deterministic and Adaptive Particle Swarm Optimization*, Proc. Congress on Evolutionary Computation, Washington, DC, 1999.