Optimization Using Particle Swarms with Near Neighbor Interactions

Kalyan Veeramachaneni, Thanmaya Peram, Chilukuri Mohan,
and Lisa Ann Osadciw

Department of Electrical Engineering and Computer Science
Syracuse University
Syracuse, NY 13244-1240
(315)443-3366(office)/(315)443-2583(fax)
kveerama,tperam,mohan,laosadci@ecs.syr.edu

Abstract. This paper presents a modification of the particle swarm optimization algorithm (PSO) intended to combat the problem of premature convergence observed in many applications of PSO. In the new algorithm, each particle is attracted towards the best previous positions visited by its neighbors, in addition to the other aspects of particle dynamics in PSO. This is accomplished by using the ratio of the relative fitness and the distance of other particles to determine the direction in which each component of the particle position needs to be changed. The resulting algorithm, known as Fitness-Distance-Ratio based PSO (FDR-PSO), is shown to perform significantly better than the original PSO algorithm and several of its variants, on many different benchmark optimization problems. Avoiding premature convergence allows FDR-PSO to continue search for global optima in difficult multimodal optimization problems, reaching better solutions than PSO and several of its variants.

1 Introduction

The Particle Swarm Optimization algorithm (PSO), originally introduced in terms of social and cognitive behavior by Kennedy and Eberhart in 1995 [1], [2], has proven to be a powerful competitor to other evolutionary algorithms such as genetic algorithms [3].
The PSO algorithm simulates social behavior among individuals (particles) "flying" through a multidimensional search space, each particle representing a single intersection of all search dimensions[7]. The particles evaluate their positions relative to a goal (fitness) at every iteration, and particles in a local neighborhood share memories of their "best" positions, then use those memories to adjust their own velocities and positions as shown in equations (1) and (2) below. The PSO formulae define each particle as a potential solution to a problem in a D-dimensional space, with the *ith* particle represented as $X_i = (x_{i1}, x_{i2}, x_{i3}, ... x_{iD})$. Each particle also remembers its previous best position, designated as *pbest*, $P_i = (p_{i1}, p_{i2}, p_{i3}, ...p_{iD})$ and its velocity $V_i = (v_{i1}, v_{i2}, v_{i3}, ...v_{iD})$ [7]. In each generation, the velocity of each particle is updated, being pulled in the direction of its own previous best position (p_i) and the best of all positions (p_g) reached by all particles until the preceding generation.

E. Cantú-Paz et al. (Eds.): GECCO 2003, LNCS 2723, pp. 110–121, 2003.

The original PSO formulae developed by Kennedy and Eberhart were modified by Shi and Eberhart [4] with the introduction of an inertia parameter, ω, that was shown empirically to improve the overall performance of PSO.

$$V_{id}^{(t+1)} = \omega \times V_{id}^{(t)} + \psi_1 \times (p_{id} - X_{id}^{(t)}) + \psi_2 \times (p_{gd} - X_{id}^{(t)}) \quad (1)$$

$$X_{id}^{(t+1)} = X_{id}^{(t)} + V_{id}^{(t+1)} \quad (2)$$

Several interesting variations of the PSO algorithm have recently been proposed by researchers in [12], [13], [14], [15], [16], [17]. Many of these PSO improvements are essentially extrinsic to the particle dynamics at the heart of the PSO algorithm and can be applied to augment the new algorithm presented in this paper. By contrast to most other PSO variations, this paper proposes a significant modification to the dynamics of particles in PSO, moving each particle towards other nearby particles with a more successful search history, instead of just the best position discovered so far. This is in addition to the terms in the original PSO update equations.

Section 2 motivates and describes the new Fitness- Distance-Ratio based PSO (FDR-PSO) algorithm. Section 3 defines the benchmark continuous optimization problems used for experimental comparison of the algorithms, and the experimental settings for each algorithm. Section 4 presents and discusses the results. Conclusions and future work are presented in Section 5.

2 FDR-PSO Algorithm

Theoretical results [10][13] have shown that the particle positions in PSO oscillate in damped sinusoidal waves until they converge to points in between their previous best positions and the global best positions discovered by all particles so far. If some point visited by a particle during this oscillation has better fitness than its previous best position (as is very likely to happen in many fitness landscapes), then particle movement continues, generally converging to the global best position discovered so far. All particles follow the same behavior, quickly converging to a good local optimum of the problem. However, if the global optimum for the problem does not lie on a path between original particle positions and such a local optimum, then this convergence behavior prevents effective search for the global optimum. It may be argued that many of the particles are wasting computational effort in seeking to move in the same direction (towards the local optimum already discovered), whereas better results may be obtained if various particles explore other possible search directions. This paper explores an alternative in which each particle is influenced by several other particles, not just moving towards or away from the best position discovered so far.

The most logical choices, for deciding which other particles ought to influence a given particle, are drawn from natural observations and expectations of animal behavior:

1. An organism is most likely to be influenced by others in its neighborhood.

2. Among the neighbors, those that have been more successful (than itself) are likely to affect its behavior.

Attempting to introduce the effects of multiple other (neighboring) particles on each particle must face the possibility of crosstalk effects encountered in neural network learning algorithms. In other words, the pulls experienced in the directions of multiple other particles may mostly cancel each other, reducing the possible benefit of all the associated computations. To counteract this possibility, the FDR-PSO algorithm selects only one other particle when updating each velocity dimension, which is chosen to satisfy two criteria:

1. It must be near the particle being updated.
2. It should have visited a position of higher fitness.

Experiments have been conducted with several possible ways of selecting particles that satisfy these criteria, without significant difference in the performance of the resulting algorithm. The simplest and most robust variation was to update each velocity dimension by selecting a particle that maximizes the ratio of the fitness difference to the one-dimensional distance. In other words, the *dth* dimension of the *ith* particle's velocity is updated using a particle called the *nbest*, with prior best position P_j, chosen to maximize

$$FDR(j, i, d) = \frac{Fitness(P_j) - Fitness(X_i)}{|P_{jd} - X_{id}|} \tag{3}$$

where |...| denotes the absolute value, and it is presumed that the fitness function is to be maximized. The above expression is called the Fitness-Distance-Ratio, suggesting the name FDR-PSO for the algorithm; for a minimization problem, we would instead use *(Cost (P_j) - Cost(X_i))* in the numerator of the above expression.

This version of the algorithm has been more successful than variations such as selecting a single particle in whose direction all velocity components are updated. The pseudocode for this algorithm is given in Figure 1.

3 Experimental Settings and Benchmark Problems

Experiments were conducted with several variations of FDR-PSO, obtained by changing the parameter values ψ_1, ψ_2, ψ_3. The results in the tables and figures use the notation "FDR-PSO(ψ_1, ψ_2, ψ_3)". Note that FDR-PSO(1,1,0) is the same as the usual PSO algorithm described by Kennedy and Eberhart. On the other hand, FDR-PSO(0,1,ψ_3) and FDR-PSO(1,0, ψ_3) correspond to the variations in which one of the main components of the old PSO algorithm is completely deleted.

"FDR-PSO (1,1,ψ_3)" refers to an instance of the new algorithm in which the relative weightage of the new term is "ψ_3" and the terms of the old PSO algorithm remain unchanged. In all the implementations, the inertia parameter is decremented with number of iterations as in [11].

$$\omega(i) = \frac{(\omega - 0.4) \times (gsize - i)}{gsize + 0.4} \quad (4)$$

where $\omega = 0.9$;

where *gsize* is the maximum number of generations for which the algorithm runs, *i* is the present generation number.

FDR-PSO was compared against two variants of random search algorithms, to verify whether the particle dynamics are of any use at all. In the "Random Velocity Update algorithm" the new velocity term = old velocity term + a number chosen from the interval [-*width*/10, *width*/10], where "*width*" is the difference between the max. and min. possible values for that dimension. In the "Random Position Update algorithm", with no explicit velocity contributing, new position = old position + a random number chosen in the same manner.

Algorithm FDR-PSO:
For t= 1 to the max. bound of the number on generations,
For i=1 to the population size,
For d=1 to the problem dimensionality,
Apply the velocity update equation:

$$V_{id}^{t+1} = \omega \times V_{id}^{t} + \psi_1 \times (p_{id} - X_{id}) + \psi_2 \times (p_{gd} - X_{id}) + \psi_3 \times (p_{nd} - X_{id})$$

where P_i is the best position visited so far by X_i,
P_g is the best position visited so far by any particle
and P_n is chosen by maximizing

$$\frac{Fitness(P_j) - Fitness(X_i)}{|P_{jd} - X_{id}|};$$

Limit magnitude:

$$V_{id}^{(t+1)} = min(V_{max}, max(-V_{max}, V_{id}^{(t+1)}));$$

Update Position:

$$X_{id}^{(t+1)} = min(Max_d, max(-Min_d, X_{id}^{(t)} + V_{id}^{(t+1)}));$$

End- for-d;
Compute fitness of ($X_i^{(t+1)}$);
If needed, update historical information regarding *Pi* and *Pg*;
End-for-i;
Terminate if *Pg* meets problem requirements;
End-for-t;
End algorithm.

Fig. 1. Pseudocode for FDR-PSO algorithm

All the experiments were conducted using a population size of 10, with each algorithm executed for a maximum of 1000 generations. Experiments were conducted with the following benchmark problems for a dimensionality of n=20. All the benchmarks have global minima at the origin.

3.1 De Jong's function 1

$$f(x) = \sum_{i=1}^{n} x_i^2 \quad \text{where } -5.12 \le x_i \le 5.12 \tag{5}$$

3.2 Axis parallel hyper-ellipsoid

$$f(x) = \sum_{i=1}^{n} i \times x_i^2 \quad \text{where} -5.12 \le x_i \le 5.12 \tag{6}$$

3.3 Rotated hyper-ellipsoid

$$f(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2 \quad \text{where } -65.536 \le x_i \le 65.536 \tag{7}$$

3.4 Rosenbrock's Valley (Banana function)

$$f(x) = \sum_{i=1}^{n-1} 100 \times (x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \quad \text{where } -2.048 \le x_i \le 2.048 \tag{8}$$

3.5 Griewangk's function

$$f(x) = \sum_{i=1}^{n} \frac{x_i^2}{4000} - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \quad \text{where } -600 \le x_i \le 600 \tag{9}$$

3.6 Sum of different powers

$$f(x) = \sum_{i=1}^{n} |x_i|^{i+1} \quad \text{where } -1 \le x_i \le 1 \tag{10}$$

4 Results and Discussion

Figures 2 through 7 present the results on the optimization functions defined in the previous section. The graphs show results averaged over 30 trials. In each trial, the population is randomly initialized and the same population is used for PSO and FDR-PSO.

As shown in Figures 2, 3, 4, 5, 6, 7 and Table 1, the new FDR-PSO algorithm outperforms the classic PSO algorithm on each of the benchmark problems on which the experiments have been conducted so far. In each case, the original PSO algorithm performs well in initial iterations but fails to make further progress in later iterations.

The significant improvement achieved by the FDR-PSO algorithm can be attributed to the near neighbor interactions. Population diversity is achieved by allowing particles to learn from their nearest best neighbor which may be of poorer fitness than the global best. FDR-PSO's learning is consistent with the social behavior of the individuals in groups, i.e., learning from the nearest best neighbors with successful search history rather than learning from only the global best. In some cases, the nearest best neighbors can be the

global best itself and hence can imply re-emphasizing the social learning. However, the probability of the nearest best neighbor being the global best decreases with the increase in population as well as the dimensionality of the problem. The algorithm in this paper has been implemented for a population of 10. The probability of the global best being the nearest best neighbor was observed to be as low as 0.4. Increasing the population size would result in a more robust implementation of this algorithm and is expected to result in a more diverse population. Such an implementation can be used for a more difficult multimodal search space where a diverse and localized PSO is a requirement.

The population diversity that is achieved can be demonstrated by the fact that the best fitness and average population fitness became identical within 500 generations when the PSO algorithm was applied to Rosenbrock's problem, whereas this did not occur until about 1000 generations when the FDR-PSO algorithm was applied. Similar results were observed for all the other benchmark problems, this shows that the new algorithm is less plagued by the premature convergence problem faced by the PSO.

Table 1. Minima achieved for different optimization functions using different algorithms

Algorithm	De Jong's	Rosenbrock's	Axis Parallel Hyper-Ellipsoid	Rotated Hyper-Ellipsoid	Griewangk's	Sum of Powers
PSO	0.0239	6.8309	0.1250	55.85	5.0501	1.8e-7
FDR(111)	0.0027	6.0802	0.0230	20.5686	3.6946	7.32e-11
FDR (112)	2.02e-5	**4.8717**	**1.07e-5**	1.2776	0.0475	**5.3e-19**
FDR (102)	**2.63e-7**	5.7389	7.6e-5	365.0034	0.4172	4.8e-17
FDR (012)	8.36e-6	5.0130	3.6e-4	**0.9080**	**0.0308**	3.8e-11
FDR(002)	0.0010	8.2869	0.0035	1513.2	2.1735	3.3e-12

The results also show that the FDR-PSO algorithm can perform well in the absence of the social or cognitive terms. By ranking the algorithm in the decreasing order of their performance, it can be seen that the top three positions are shared by FDR-PSO(112), FDR-PSO(012), FDR-PSO(102) with FDR-PSO(112) being the best in most of the benchmark problems. It is interesting to note that FDR-PSO(111) has always been a poor performer in the FDR-PSO family. This demonstrates the sensitivity of the algorithm to the weight given to the "nbest" term. The "near neighbor" term, however, remains the most important term of the new algorithm with PSO related terms adding a little more to the performance. This can be seen from the fact that the FDR-PSO(002) outperforms the standard PSO and the FDR-PSO(111) in four benchmark problems. The versions, random velocity update and random position updates are worst performers of all.

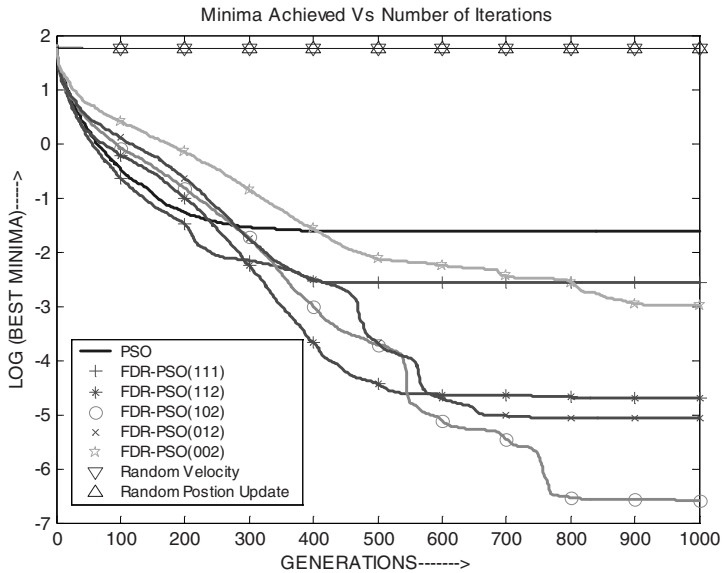

Fig. 2. Best minima plotted against the number of generations for each algorithm, for DeJong's function, averaged over 30 trials

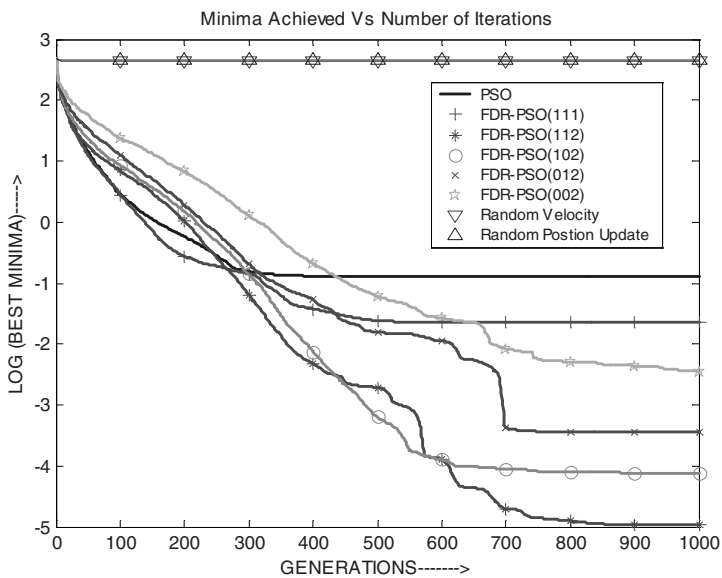

Fig. 3. Best minima plotted against the number of generations for each algorithm, for Axis parallel hyper-ellipsoid, averaged over 30 trials

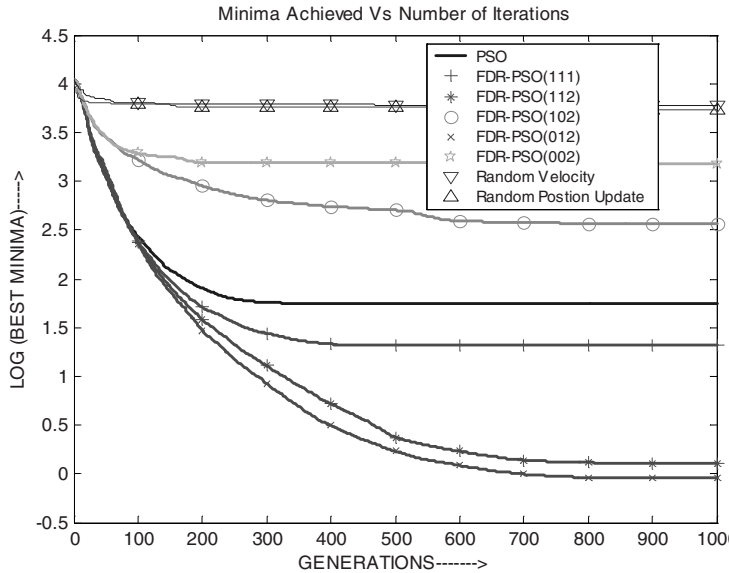

Fig. 4. Best minima plotted against the number of generations for each algorithm, for Rotated hyper-ellipsoid, averaged over 30 trials

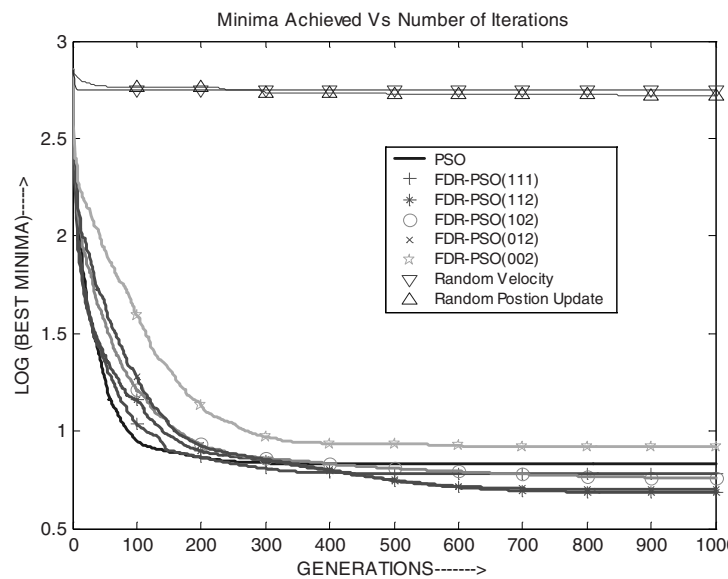

Fig. 5. Best minima plotted against the number of generations for each algorithm, for Rosenbrock's Valley, averaged over 30 trials

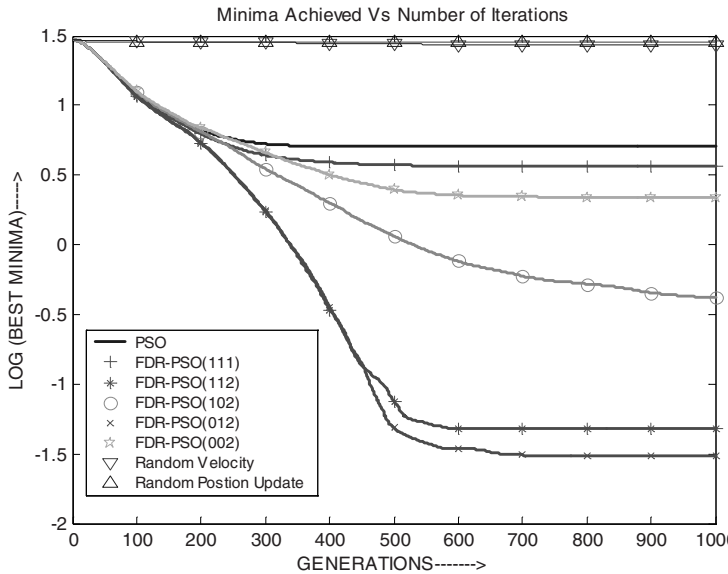

Fig. 6. Best minima plotted against the number of generations for each algorithm, for Griewangk's Function, averaged over 30 trials

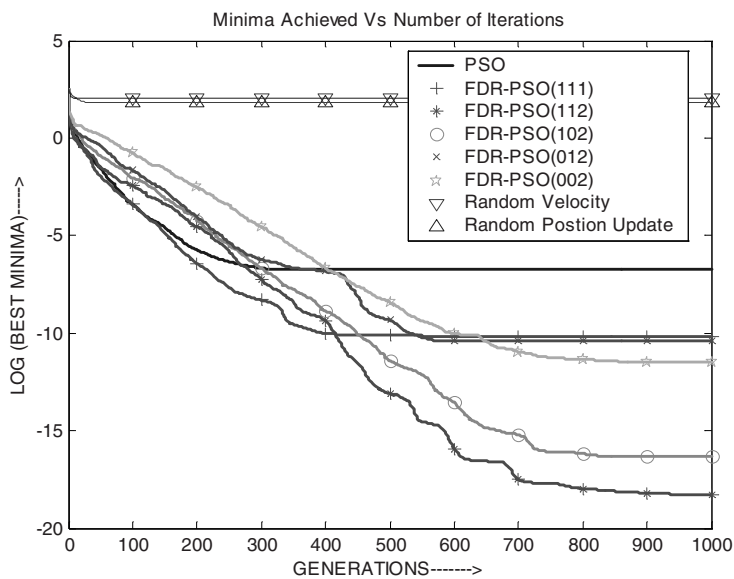

Fig. 7. Best minima plotted against the number of generations for each algorithm, for Sum of Powers, averaged over 30 trials

Several other researchers have proposed different variations of PSO. For example, ARPSO[17] uses a diversity measure to have the algorithm alternate between two phases i.e., attraction and repulsion. In this algorithm, 95% of the fitness improvements were achieved in the attraction phase and the repulsion phase merely increases the diversity. In the attraction phase the algorithm runs as the basic PSO, while in the repulsion phase the particles are merely pushed in opposite direction of the best solution achieved so far.

The random restart mechanism has also been proposed under the name of "PSO with Mass Extinction"[15]. In this, after every "I_e" generations, called the extinction interval, the velocities of the swarm are reinitialised with random numbers. Researchers have also explored increasing diversity by increasing randomness associated with velocity and position updates, thereby discouraging swarm convergence, in the "Dissipative PSO"[16]. Lovbjerg and Krink have explored extending the PSO with "Self Organized Criticality"[14], aimed at improving population diversity. In their algorithm, a measure, called "criticality", describing how close to each other are the particles in the swarm, is used to determine whether to relocate particles. Lovbjerg, Rasmussen, and Krink also proposed in [6], an idea of splitting the population of particles into subpopulations and hybridizing the algorithm, borrowing the concepts from Genetic algorithms. All these variations perform better than the PSO. These variations however seem to add new control parameters, such as, extinction interval in [15], diversity measure in [17], criticality in[14], and various genetic algorithm related parameters in [6], which can be varied and have to be carefully decided upon. The beauty of FDR-PSO lies in the fact that it has no more additional parameters than the PSO and achieves the objectives achieved by any of these variations and reaches a better minima. Table 2 compares the FDR-PSO algorithm with these variations. The comparisons were performed by experimenting FDR-PSO(1, 1, 2) on the benchmark problems with approximately the same settings as reported in the experiments of those variations. In all the cases the FDR-PSO outperforms the other variations.

Table 2. Minima achieved by different variations of PSO and FDR-PSO

Algorithm	Dimensions	Generations	Griewangk's Function	Rosenbrock's Function
PSO	20	2000	0.0174	11.16
GA	20	2000	0.0171	107.1
ARPSO	20	2000	0.0250	2.34
FDR-PSO(112)	20	2000	**0.0030**	**1.7209**
PSO	10	1000	0.08976	43.049
GA	10	1000	283.251	109.81
Hybrid(1)	10	1000	0.09078	43.521

Algorithm	Dimensions	Generations	Griewangk's Function	Rosenbrock's Function
Hybrid(2)	10	1000	0.46423	51.701
Hybrid(4)	10	1000	0.6920	63.369
Hybrid(6)	10	1000	0.74694	81.283
HPSO1	10	1000	0.09100	70.41591
HPSO2	10	1000	0.08626	45.11909
FDR-PSO(112)	10	1000	**0.0148**	**9.4408**

5 Conclusions

This paper has proposed a new variation of the particle swarm optimization algorithm called FDR-PSO, introducing a new term into the velocity component update equation: particles are moved towards nearby particles' best prior positions, preferring positions of higher fitness. The implementation of this idea is simple, based on computing and maximizing the relative fitness-distance-ratio. The new algorithm outperfoms PSO on many benchmark problems, being less susceptible to premature convergence, and less likely to be stuck in local optima. FDR-PSO algorithm outperforms the PSO even in the absence of the terms of the original PSO.

From one perspective, the new term in the update equation of FDR-PSO is analogous to a recombination operator where recombination is restricted to individuals in the same region of the search space. The overall evolution of the PSO population resembles that of other evolutionary algorithms in which offspring are mutations of parents, whom they replace. However, one principal difference is that algorithms in the PSO family retain historical information regarding points in the search space already visited by various particles; this is a feature not shared by most other evolutionary algorithms.

In current work, a promising variation of the algorithm, with the simultaneous influence of multiple other neighbors on each particle under consideration, is being explored. Future work includes further experimentation with parameters of FDR-PSO, testing the new algorithm on other benchmark problems, and evaluating its performance relative to EP and ES algorithms.

References

1. Kennedy, J. and Eberhart, R., "Particle Swarm Optimization", IEEE International Conference on Neural Networks, 1995, Perth, Australia.
2. Eberhart, R. and Kennedy, J., "A New Optimizer Using Particles Swarm Theory", Sixth International Symposium on Micro Machine and Human Science, 1995, Nayoga, Japan.
3. Eberhart, R. and Shi, Y., "Comparison between Genetic Algorithms and Particle Swarm Optimization", The 7th Annual Conference on Evolutionary Programming, 1998, San Diego, USA.

4. Shi, Y. H., Eberhart, R. C., “A Modified Particle Swarm Optimizer”, IEEE International Conference on Evolutionary Computation, 1998, Anchorage, Alaska.
5. Kennedy J., “Small Worlds and MegaMinds: Effects of Neighbourhood Topology on Particle Swarm Performance”, Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1931-1938. IEEE Press.
6. Lovbjerg, M., Rasmussen, T. K., Krink, T., “ Hybrid Particle Swarm Optimiser with Breeding and Subpopulations”, Proceedings of Third Genetic Evolutionary Computation, (GECCO 2001).
7. Carlisle, A. and Dozier, G.. “Adapting Particle Swarm Optimization to Dynamic Environments”, Proceedings of International Conference on Artificial Intelligence, Las Vegas, Nevada, USA, pp. 429-434, 2000.
8. Kennedy, J., Eberhart, R. C., and Shi, Y. H.,_Swarm Intelligence, Morgan Kaufmann Publishers, 2001.
9. GEATbx: Genetic and Evolutionary Algorithm Toolbox for MATLAB, Hartmut Pohlheim, http://www.systemtechnik.tu-ilmenau.de/~pohlheim/GA_Toolbox/index.html.
10. E. Ozcan and C. K. Mohan, “Particle Swarm Optimzation: Surfing the Waves”, Proceedings of Congress on Evolutionary Computation (CEC’99), Washington D. C., July 1999, pp 1939-1944.
11. Particle Swarm Optimization Code, Yuhui Shi, www.engr.iupui.edu/~shi
12. van den Bergh, F., Engelbrecht, A. P., “Cooperative Learning in Neural Networks using Particle Swarm Optimization”, South African Computer Journal, pp. 84-90, Nov. 2000.
13. van den Bergh, F., Engelbrecht, A. P., “Effects of Swarm Size on Cooperative Particle Swarm Optimisers”, Genetic and Evolutionary Computation Conference, San Francisco, USA, 2001.
14. Lovbjerg, M., Krink, T., “Extending Particle Swarm Optimisers with Self-Organized Criticality”, Proceedings of Fourth Congress on Evolutionary Computation, 2002, vol. 2, pp. 1588-1593.
15. Xiao-Feng Xie, Wen-Jun Zhang, Zhi-Lian Yang, “Hybrid Particle Swarm Optimizer with Mass Extinction”, International Conf. on Communication, Circuits and Systems (ICCCAS), Chengdu, China, 2002.
16. Xiao-Feng Xie, Wen-Jun Zhang, Zhi-Lian Yang, “A Dissipative Particle Swarm Optimization”, IEEE Congress on Evolutionary Computation, Honolulu, Hawaii, USA, 2002.
17. Jacques Riget, Jakob S. Vesterstorm, “A Diversity-Guided Particle Swarm Optimizer - The ARPSO”, EVALife Technical Report no. 2002-02.