Particle Swarm Optimisation with Spatial Particle Extension

Thiemo Krink, Jakob S. Vesterstrøm and Jacques Riget
EVALife Group, Department of Computer Science
Ny Munkegade, Bldg. 540, University of Aarhus
DK-8000 Aarhus C, Denmark
www.evalife.dk krink,jve,riget@daimi.au.dk

Abstract - In this paper, we introduce spatial extension to particles in the PSO model in order to overcome premature convergence in iterative optimisation. The standard PSO and the new model (SEPSO) are compared w.r.t. performance on well-studied benchmark problems. We show that the SEPSO indeed managed to keep diversity in the search space and yielded superior results.

I. Introduction

The PSO idea was originally introduced by J. Kennedy et al. in 1995 as an optimisation technique inspired by swarm intelligence and theory in general such as bird flocking, fish schooling and even human social behavior [4], [7], [8]. Furthermore, the whole idea and structure of the algorithm is inspired by evolutionary computation. Later PSOs has turned out to be a worthy alternative to the standard Genetic Algorithm (GA) and other iterative optimisation techniques.

Since the introduction of PSOs in 1995 several research articles have been published on the subject. Shi and Eberhart investigated the problem of finding good parameter setting with the PSO model [11]. Further, the interaction between particles has been researched in 1999 regarding the "swarm neighborhood topology" [10]. Even though much work has been done in this area during the last six years, only a small part of it has been concerned with testing and comparison with traditional optimisation techniques like GAs and other evolutionary approaches. There has been an empirical study concerning the performance of the PSO and an analytical comparison between the PSO model and the GA approach [5], [13]. P. J. Angeline compared the performance between a standard Evolutionary Algorithm (EA) and a basic PSO and concluded that the performance of the two different methods is competitive [1]. Angeline's main point was that the PSO often converges significantly faster to the global optimum than the EA but has difficulties in fine tuning solutions. Hence, the performance of the PSO flattens out with a loss of diversity in the search space as the overall result.

Quite a few articles have been concerned with improving the PSO model. However, conceptually these attemps have not differed much from the basic PSO, since the main issue has been to investigate, optimise and fit different parameters in the velocity update formula to achieve better and faster convergence [2], [9], [10], [11].

To our knowledge there has been no research on the individual particle behavior. We have implemented a graphical visualisation of the swarm mainly for the purpose of exploring the behavior of the individual particle and its movement in the search space; hence, getting more information about the swarm behavior and identifying problems with the PSO strategy.

The purpose of the work presented in this paper is to investigate the problems with the PSO model and suggest a solution to the drastic performance loss around local optima. Compared with the basic PSO, the simple innovation in our new model is to give particles a spatial extension. In the modified model it is possible to avoid particle collisions, hence avoiding clustering of particles and keeping diversity in the search space.

The SEPSO was tested in comparison with the basic PSO model and the std. GA. In the next section the original PSO model and the SEPSO are introduced. In section III the experimental settings and test cases are presented. Section IV and V describe and discuss the results from our experiment. Finally we summarize our results in section VI.

II. The PSO Models

A. The Basic PSO model

The basic PSO model consists of a swarm of particles moving in a n-dimensional search space where a certain quality measure, the fitness, can be calculated. Each particle has a position represented by a position-vector $\vec{x}$ and a velocity represented by a velocity-vector $\vec{v}$. Each particle remembers its own best position so far in a vector $\vec{p}$ (i.e. the position where it achieved its best fitness). Further, a neighborhood relation is defined for the swarm. The best position-vector among all the neighbors of a particle is then stored in the particle as a vector $\vec{n}$.

At each timestep t the velocity is updated and the particle is moved to a new position. The new position is determined by the sum of the previous position and the new velocity by

$$\vec{x}(t+1) = \vec{x}(t) + \vec{v}(t+1). \tag{1}$$

The update of the velocity from the previous velocity to the new velocity is (in its simplest form) determined by

$$\vec{v}(t+1) = \vec{v}(t) + \phi_1(\vec{p} - \vec{x}) + \phi_2(\vec{n} - \vec{x}), \tag{2}$$

where ϕ_1 and ϕ_2 are real numbers chosen uniformly and at random in some interval, usually $[0, 2]$. Most attempts to improve the velocity update formula (2), can be captured by the following formula (as presented in [10]):

$$\vec{v}(t+1) = \chi(\omega \cdot \vec{v}(t) + \phi_1(\vec{p} - \vec{x}) + \phi_2(\vec{n} - \vec{x})). \tag{3}$$

The two new parameters χ and ω are also real numbers. The parameter χ controls the magnitude of $\vec{v}$, whereas the inertia weight ω weights the magnitude of the old velocity $\vec{v}(t)$ in the calculation of the new velocity $\vec{v}(t+1)$. In the same way, ϕ_1 and ϕ_2 determine the significance of $\vec{p}$ and $\vec{n}$ in the calculation of $\vec{v}(t+1)$.

There are many ways to define the neighborhood relation on a swarm. At least three of them have been used and tested to give good results [10].

1) **Star**, also known as *gbest*, is a fully connected neighborhood relation. Each particle has all the other particles as neighbors; this implies that the global best particle-position for all particles is identical.

2) **Circle**, also known as *lbest*, connects each particle to its K immediate neighbors. For $K = 2$ the circle topology corresponds to a ring. The "flow of information" is heavily reduced compared to the star topology. It will for instance take $\frac{swarmsize}{2}$ timesteps for a new global best position to propagate to the other side of the ring.

3) **Wheel** connect one central particle to all the others, where all peripheral particles are not connected with each other. Even though the number of connections is smaller in this topology than in the circle topology ($swarmsize - 1$ compared to $swarmsize$), the information will flow faster since it will not take more than two timesteps for one new global best position to reach any other particle.

In section IV we will discuss the implication the choice of topology has on the behavior and convergence of the PSO algorithm.

B. PSO model with Spatial Particle Extension

The main motivation for giving the particles an extension in space is that the particles in the basic PSO tend to cluster too closely. When an optimum (local or global) is found by one particle the other particles will be drawn towards it. If all particles end up in this optimum, they will stay at this optimum without much chance to escape. This simply happens because of the way the basic PSO (in particular the velocity update formula) works. If the identified optimum is only local it would be advantageous to let some of the particles explore other areas of the search space while the remaining particles stay at this optimum to finetune the solution.

In our spatial particle extension model, we tried to increase the diversity when particles started to cluster. For this we added a radius r to each particle in order to check whether two particles would collide. If they collide, action can be taken to make them *bounce* off to avoid the collision and thus the clustering.

An important issue is to determine the direction in which the particles should bounce away and at what speed. We investigated three strategies: 1) Random bouncing, where the particles are sent away from the collision in a random direction preserving the old speed. 2) Realistic physical bouncing. 3) Simple velocity-line bouncing in which the particles continue to move in the direction of their old velocity-vector, but with a scaled speed. This gives the particle the possibility of making an U-turn and return to where it came from (by scaling with a negative bounce-factor). Particles can be slowed down (bounce-factor between 0 and 1) or speeded up to avoid the collision (bounce-factor greater than 1). A particle collision avoidance with velocity-line bouncing is illustrated in figure 1.

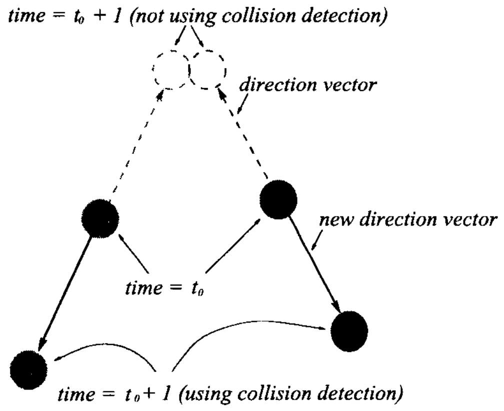
time = t_0 + 1 (not using collision detection)
direction vector
new direction vector
time = t_0
time = t_0+ 1 (using collision detection)

Fig. 1. A particle-collision. The solid circles are the particle positions at time t_0 and $t_0 + 1$ using collision detection. The dotted rings are their positions at time $t_0 + 1$ without collision detection. The bounce-factor is set to -1.

III. Experimental Settings

We have used four functions to test our SEPSO model and the neighborhood topologies. These four functions are widely known and are used as benchmark functions for optimisation strategies such as GAs and PSOs. The performance of the basic PSO, SEPSO and GA were compared using the multi-modal Griewank and Rastrigin problems in 10, 20 and 30 dimensions. Furthermore, the performance of the basic PSO and the SEPSO were compared using the uni-modal Sphere and Rosenbrock functions in 10, 20 and 30 dimensions.

$$\text{Sphere: } f_1(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 \tag{4}$$

$$\text{Griewank: } f_2(\mathbf{x}) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1 \tag{5}$$

$$\text{Rosenbrock: } f_3(\mathbf{x}) = \sum_{i=1}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2 \tag{6}$$

$$\text{Rastrigin: } f_4(\mathbf{x}) = \sum_{i=1}^{n} x_i^2 + 10 - 10 \cdot \cos(2\pi x_i) \tag{7}$$

The domain of the functions and the intialisation domain of the particles are shown in table I below.

Function	Func. domain	Init. domain
Sphere	$(-100, 100)^n$	$(50, 100)^n$
Rosenbrock	$(-100, 100)^n$	$(15, 30)^n$
Griewank	$(-600, 600)^n$	$(300, 600)^n$
Rastrigin	$(-10, 10)^n$	$(2.56, 5.12)^n$

TABLE I
FUNCTION AND INITIALISATION DOMAIN.

In accordance with Angeline we used asymmetric initialisation since the search spaces are symmetric around the origin [1]. This is to prevent a centre-seeking optimiser from "accidently" finding the global optimum.

All experiments were repeated 50 times. The number of timesteps was set to 10000 for all functions in order to investigate whether the SEPSO continued the optimisation or stagnated at a local optimum. The parameters ϕ_1 and ϕ_2 were set to 2.0 for both the basic PSO and the SEPSO. The problem dependent linearly decreasing inertia weight ω was set at intervals ranging from (0.9 to 0.6) to (0.3 to 0.1). The maximum velocity v_{max} was different for each problem. As a rule of thumb, the larger the function domain, the higher v_{max} was set. The swarm size was 20 for all problems. In all experiments the bounce-factor was set to -1 using simple velocity-line bouncing.

The genetic algorithm used in the comparison is a more or less straight-forward implementaion of the standard, real-valued GA. It uses tournament-selection with *populationsize* = 2, one-point crossover (applied with probability p_c) and a N(0,σ^2)-distributed mutation operator (applied independently at each gene-coordinate with probability p_m. We optimized the variance σ^2-parameter to be a decreasing function of time to emphasize local search towards the end of the test run. The variance was set to $\sigma^2(t) = 1 - \frac{t}{t_{max}}$ of time t. Moreover we use elitism with size 1. The problem dependent parameters p_m and p_c was tuned for optimal performance on each test problem. The population size was set at 20 to make the performance of the GA and the PSO directly comparable.

IV. Experimental Results

Table II shows the results from the conducted experiments. The table lists the dimension of the test function and the average best fitness found for the 50 runs of each of the functions. Figures 2 to 5 show the performance over time for some of our experiments. All with average best fitness plotted for each timestep for the PSO, SEPSO and GA. Figure 6 shows the performance of the basic PSO with the different neighborhood topologies on the Rastrigin function in 30 dimensions.

Dim.	10	20	30
Sphere			
GA	$0.10 \cdot 10^{-5}$	$0.56 \cdot 10^{-5}$	$0.18 \cdot 10^{-2}$
bPSO	$0.00 \cdot 10^{-8}$	$0.00 \cdot 10^{-8}$	$0.00 \cdot 10^{-8}$
SEPSO	$0.04 \cdot 10^{-8}$	$0.27 \cdot 10^{-8}$	$0.59 \cdot 10^{-8}$
Griewank			
GA	0.0462	0.0526	0.1133
bPSO	0.0326	0.0016	0.0590
SEPSO	0.0338	0.0001	0.0118
Rosenbrock			
GA	14.0504	85.7777	106.2853
bPSO	1.4197	12.1450	26.9697
SEPSO	1.5379	9.4112	15.4375
Rastrigin			
GA	0.0018	0.6050	14.2025
bPSO	1.1741	10.7257	32.7142
SEPSO	0.0106	0.1026	0.4541

TABLE II
AVERAGE BEST FITNESS VALUES FOR BENCHMARK PROBLEMS

With Regards to the "simplest" problem, the Sphere function, there is practically no difference on the performance of basic PSO and the SEPSO. Regarding convergence time and best particle found, they performed very similar in all instances. The results indicate that bouncing is not needed on a very simple problem, since clustering does not become a problem with only one optimum.

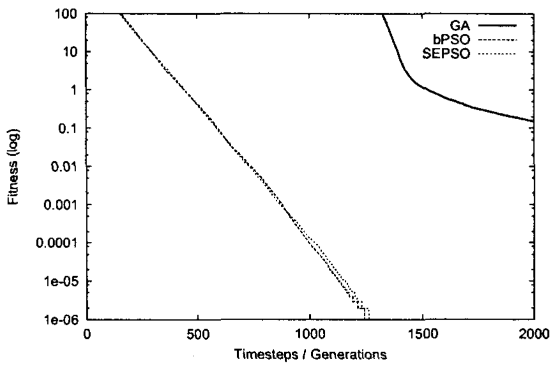

Fig. 2. Sphere, 30 dimensions

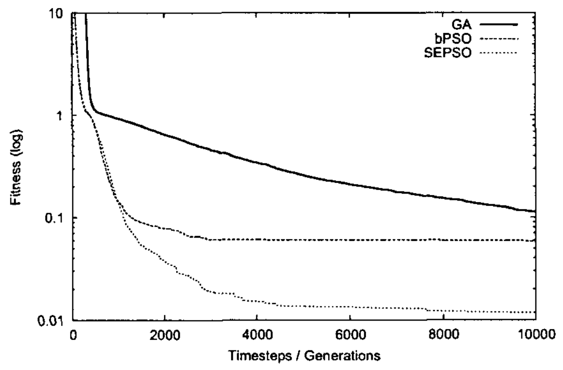

Fig. 3. Griewank, 30 dimensions

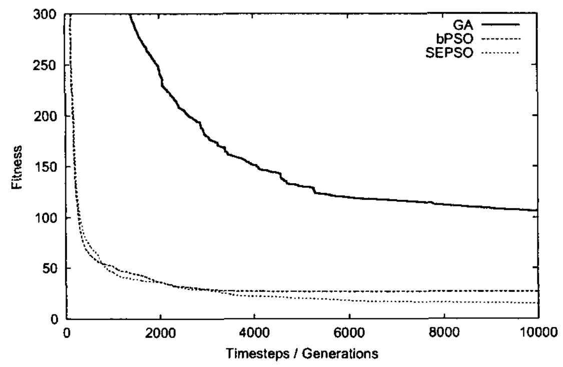

Fig. 4. Rosenbrock, 30 dimensions

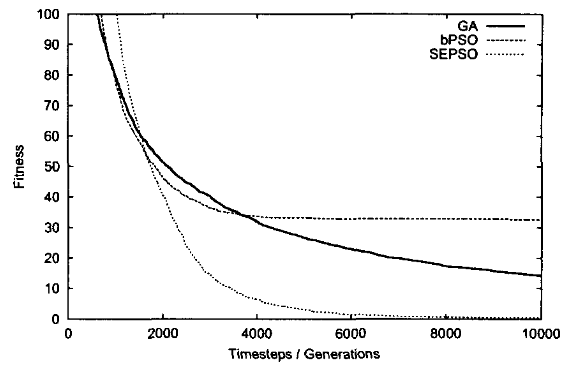

Fig. 5. Rastrigin, 30 dimensions

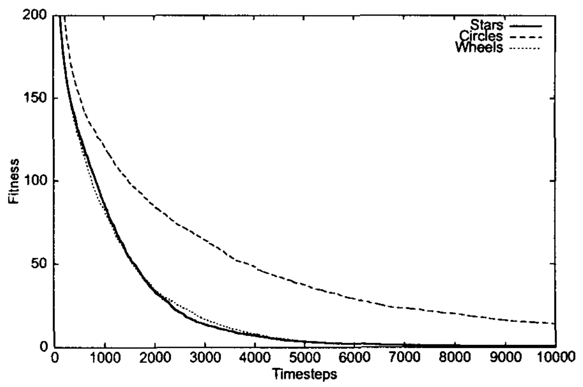

Fig. 6. Comparison of the three topologies on Rastrigin, 30 dimensions.

Regarding the Griewank and Rosenbrock functions there was an interesting tendency. In 10 and 20 dimensions, the basic PSO and the SEPSO performed similary both with respect to convergence time and best fitness value found. The benchmark problems were not sufficiently hard for the SEPSO to outperform the basic PSO. However, in 30 dimensions, we found clear differences: The SEPSO kept on optimising towards a better fitness, whereas the basic PSO algorithm stagnated and flattened out with no further improvement.

Regarding the Rastrigin function the SEPSO outperformed the basic PSO even on the 10 dimensional problem. It converged faster and obtained a significantly better end-result. Interestingly, the standard GA followed the SEPSO very closely in performance in 10 dimensions. In 20 and especially in 30 dimensions the SEPSO clearly outperformed the basic PSO and the GA. After 4000 timesteps the basic PSO obtained no further improvements, whereas the SEPSO was still improving notably.

A. Visual Analysis

We have implemented a visualisation tool for the PSO, as mentioned in section I, allowing us to see each particle position, velocity and movement in the search space in 2 dimensions. With this interface we conducted an analysis on all test cases with the most common neighborhood topologies: star, wheel, and circle. This could of course only be done in the two-dimensional case. Without loss of generality, we made two major observations:

- Premature convergence due to clustering of particles
- Lack of dynamic velocity adjustment

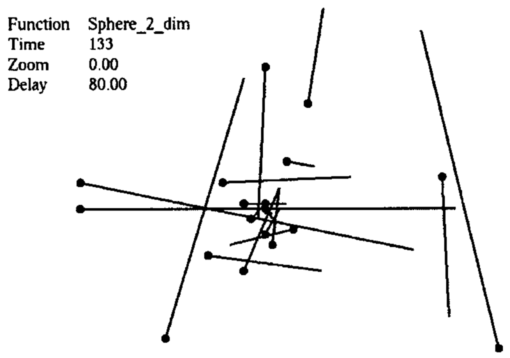

Fig. 7. Lack of dynamic velocity adjustment.

Premature convergence is known to be a huge problem in all (iterative) optimisation techniques including the particle swarm. This result is also in accordance with earlier results by Angeline [1]. The investigations on neighborhood topologies by Kennedy suggested different neighborhood settings which should control the communication flow between particles [10]. Our experiments on these topologies indicate that this has no significant effect on the clustering. In fact, it seems that a neighborhood topology with slower information flow between particles just tends to slow down the convergence towards local and global optima. This is evident from our experiment on the Rastrigin test function shown in figure 6. The circle topology leads to slower convergence. This is simply because the maximum time for information to flow to every particle in the swarm is $\frac{swarmsize}{2}$ compared to 1 in the star topology and 2 in the wheel case. No positive effect was recorded on any test function when using the circle topology. Conslusively, the fastest information flow clearly leads to the best performance. Slower information flow does not solve the problem with premature convergence.

The graphical illustration (figure 7) examplifies the problem of a lack of adequate velocity adjustment throughout the run, which confirms the findings by Angeline [1]. When the particles settled near an optimum they seemed to flicker around aimlessly. The velocity vector never seemed to dynamically adjust and settle – or in the best case adjusted the velocity vector to slow, resulting in the performance near optimum to flatten out drastically. One way to patch this problem is through the inertia weight ω introduced in [12]. Decreasing ω linearly throughout the run will also decrease the overall particle velocity. However, this solution is unfortuneately highly problem dependent which was also the case in the experiments presented in this paper. The optimal value of the inertia weight ω varied in our test cases from 0.9 decreasing to 0.6 – to a setting with 0.3 decreasing to 0.1.

V. Discussion

A. Diversity measure

In this paper we have introduced the SEPSO, which is an extension of a basic PSO with collision detection. Our experiments show that the SEPSO is a much stronger optimiser. Further, we found that the SEPSO seemed to optimise slightly slower than the basic PSO in the first couple of hundred generations, but then quickly caught up and outperformed the basic PSO by its ability to improve further. The basic PSO tended to stagnate for instance on the Rastrigin function in 30 dimensions after about 4000 timesteps. In contrast the SESPO was still improving after 10000 timesteps. The main factor behind this difference is that the SEPSO maintains a high diversity throughout the run. A common measure for this is the "distance-to-average-point" measure defined as:

$$diversity(S) = \frac{1}{|S|} \cdot \sum_{i=1}^{|S|} \sqrt{\sum_{j=1}^{N} (p_{ij} - \overline{p}_j)^2}, \qquad (8)$$

where S is the population, $|S|$ is the swarmsize, N is the dimensionality of the problem, p_{ij} is the j'th value of the i'th particle and $\overline{p}_j$ is the j'th value of the average point $\overline{p}$. We have compared the diversity of the swarm on the Rastrigin function shown in figure 8. The diversity for the SEPSO increases to a level of about 1.5 after approximately 5000 timesteps, whereas the diversity of the basic PSO decreaceses to a level of less than 0.5.

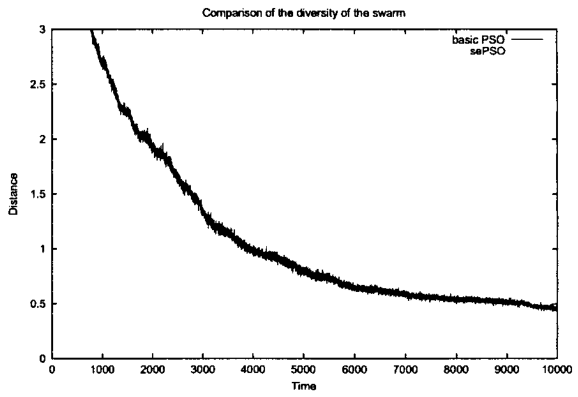

Fig. 8. Comparison of the diversity of the swarm for the basic PSO and the SEPSO on Rastrigin in 30 dimensions.

B. Collision detection and bouncing

The visualiser helped us discover that bouncing is a mechanism that spread out clustered particles. In a preliminary experiment we investigated the effect of random instead of directed bouncing which turned out to be nearly as "bad" as the basic PSO without bouncing. This is not at all surprising when considering the actual purpose of bouncing. While random bouncing does not work, almost all other forms of consistent bouncing work fine – and almost equally well! In other words, for rigid velocity-line bouncing all values from -1 to about -100 and 1 to 100 worked fine.

What is it that makes bouncing successful? We have already seen that the SEPSO model with simple bouncing can maintain higher diversity than the basic PSO. With respect to GAs one could compare this idea with methods such as crowding or sharing [3], [6]. The objective in crowding is to maintain the diversity of the initial population for a longer time period. The idea is to renew the population so that old solutions are replaced by the most similar new solutions. Likewise with sharing the idea is to maintain diversity of the genomes so that they cover a larger area of the search space. This is achieved by a fitness scaling mechanism that directly punishes individuals with too similar genomes.

In crowding we have the major drawback of the risk of replacement errors. With sharing we experience another disadvantage. The performance of sharing is extremely sensitive to the range of fitness values imposed by the objective problem [14]. Of course bouncing will sometimes, but not often, be applied in the early timesteps, when particles collide but without any significance in the overall run.

VI. Conclusion

We can conclude that on all four test problems, the SEPSO performed well. Especially on Rastrigin and in higher dimensions on Rosenbrock and Griewank, the SEPSO outperformed both the implemented GA and the basic PSO significantly. The main advantage of the SEPSO is that it kept improving, whereas the other algorithms stagnated. Actually, we have not been able to observe a total stagnation on the SEPSO in our experiments.

By now, these four cases are the only problems on which the SEPSO has been tested and more benchmark tests must be investigated in the future. However, the current results obtained with the Spatial Extended Particle Swarm Optimiser are extremely promising. Our experiment clearly show that the SEPSO model contain great potential with regards to unimodal and multimodal optimisation.

VII. Future Work

We have thought about how to take the SEPSO idea even further. There is still room for improvement regarding the lack of dynamic velocity adjustment in our new PSO model. A way to tackle this problem could be to let particles spawn extra particles with very low and decreasing velocity vectors around an optimum. A promising additional result of our experiment is that a measurement of the number of collisions suggest an intelligent choice for an independent gradient descent search algorithm for finetuning. How this should be implemented and how well the results will form out is still an open question that should be investigated.

References

[1] Angeline, P. J., "Evolutionary Optimization Versus Particle Swarm Optimization: Philosophy and Performance Differences", Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 601–610. Springer.

[2] Clerc, M. and Kennedy, J., "The Particle Swarm: Explosion, Stability, and Convergence in a Multi-Dimensional Complex Space", 1999.

[3] DeJong, K. A., "An analysis of the Behavior of a Class of Genetic Adaptive Systems. PhD Thesis, University of Michagan, 1975.

[4] Eberhart, R. C., Simson, P. and Dobbins, R., "Computational Intelligence PC Tools", Boston Academic Press, 1996.

[5] Eberhart, R. C., and Shi, Y., "Comparison between Genetic Algorithms and Particle Swarm Optimization", Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 611–616. Springer.

[6] Goldberg, D. E. and Richardson, J., "Genetic Algorithms with Sharing for Multimodal Function Optimization", In Grefenstette, editor, "Genetic Algorithms and their Applications" (ICGA'87), pages 41-49.

[7] Kennedy, J. and Eberhart, R. C., "Particle swarm optimization", Proceedings of the 1995 IEEE International Conference on Neural Networks, vol. 4, 1942–1948. IEEE Press.

[8] Kennedy, J., "The Particle swarm: Social adaptation of Knowledge", Proceedings of the 1997 International Conference on Evolutionary Computation (Indianapolis, Indiana), 303–308, IEEE Service Center, Piscataway, NJ.

[9] Kennedy, J., The Behavior of Particles", Proceeding of the 7th International Conference on Evolutionary Programming, San Diego, California, USA, March 1998.

[10] Kennedy, J. "Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1931–1938. IEEE Press.

[11] Shi, Y. and Eberhart, R. C., "Parameter Selection in Particle Swarm Optimization", Evolutionary Programming VII (1998), Lecture Notes in Computer Science 1447, 591–600. Springer.

[12] Shi, Y. and Eberhart, R. C., "A modified Particle Swarm Optimiser", IEEE International Conference on Evolutionary Computation, Anchorage, Alaska, May 4-9, 1998.

[13] Shi, Y. and Eberhart, R. C., "Empirical Study of Particle Swarm Optimization", Proceedings of the 1999 Congress of Evolutionary Computation, vol. 3, 1945–1950. IEEE Press.

[14] Ursem, R. K., "When sharing fails", Proceedings of the Third Congress of Evolutionary Computation (CEC 2001). IEEE Press.