International Journal of Innovative Computing, Information and Control
Volume **6**, Number **1**, January **2010**

ICIC International ©2010 ISSN 1349-4198
pp. **287–296**

PSOGP: A GENETIC PROGRAMMING BASED ADAPTABLE EVOLUTIONARY HYBRID PARTICLE SWARM OPTIMIZATION

Muhammad Rashid and A. Rauf Baig

Department of Computer Science
National University of Computer and Emerging Sciences
A.K. Brohi Road, Sector H-11/4, Islamabad, Pakistan
rashid.nuces@gmail.com; rauf.baig@nu.edu.pk

Received October 2008; revised March 2009

Abstract. *In this study we describe a method for extending particle swarm optimization. We have presented a novel approach for avoiding premature convergence to local minima by the introduction of diversity in the swarm. The swarm is made more diverse and is encouraged to explore by employing a mechanism which allows each particle to use a different equation to update its velocity. This equation is also continuously evolved through the use of genetic programming to ensure adaptability. We compare two variations of our algorithm, one utilizing random initialization while in the second one we utilize partial non-random initalization which forces some particles to use the standard PSO velocity update equation. Results from experimentation suggest that the modified PSO with complete random initialization shows promise and has potential for improvement. It is particularly very good at finding the exact optimum.*

Keywords: Particle swarm optimization, Genetic programming, Function optimization, Evolution, Velocity update equation

1. Introduction.

This paper presents an extension to the well known particle swarm optimization algorithm. Unlike other population-based optimization approaches which are motivated by evolution, particle swarm optimization is motivated from the simulation of social behavior. There have been numerous attempts to incorporate evolution into PSO. This is usually done by creating a hybrid algorithm which combines aspects from evolutionary computation with PSO. These can be categorized into two broad categories. The first category consists of those hybrid algorithms which evolve the particles of PSO. The second category includes those hybrid algorithms in which the PSO algorithm itself is evolved. There have also been numerous approaches in which a different velocity update equation has been used for different particles in a swarm. Generally in these approaches, the particles are divided into multiple subgroups and each subgroup shows a different behavior achieved by using a separate update equation for that subgroup. There have also been numerous other modifications made to PSO and evolutionary algorithms in general. We introduce a novel approach which incorporates evolution into PSO [30]. In this approach a learning set is not required and the PSO adapts itself to the target problem at runtime by evolving the update equation through genetic programming [20]. This allows the PSO to perform better by adjusting itself according to the problem. We also introduce the concept of using a different update equation for each particle to avoid premature convergence to local minima because this introduces diversity in the swarm and encourages exploration. Each particle's velocity update equation is evolved separately by using genetic operators. We also present another variant which utilizes partial non-random initialization. We present a comparison of the two approaches and compare their performance with each other as well as with the standard PSO. In the next section, we to incorporate evolution in PSO. Then we look at some other existing PSO algorithms in which multiple velocity update equations have been used. We also mention a few of the recent innovations made in PSO and evolutionary algorithms. We then proceed to present our proposed extensions. In the end the experimentation is described followed by a discussion of the results.

2. **Particle Swarm Optimization.**

Particle swarm optimization (PSO) is a population based computational technique inspired from the simulation of social behavior of flock of birds. PSO was originally designed and developed by Eberhart and Kennedy [18]. A newer version was introduced in [33] by incorporating inertia weight. In PSO a swarm of particle is used to represent the population of candidate solutions. Each particle is a point in the N-dimensional search space. A particle is represented by its current position x_i, and its current velocity v_i (both x_i and v_i are N-dimensional vectors). PSO tries to find the optimal solution to the problem by moving the particles and evaluating the fitness of the new position. A particle's position is updated by the following equation.

$$x_i^{t+1} = x_i^t + v_i^t$$

Along with updating the particle's position, its velocity is updated with every iteration as well. The amount by which the velocity is changed is dependent upon three factors; the particle's personal best position, the swarm's best position and the particle's previous velocity. The velocity is updated according to the following equation.

$$v_i^{t+1} = w \times v_i^t + c_1 \times r_1 \times (p_i^t - x_i^t) + c_2 \times r_2 \times (p_g^t - x_i^t)$$

Where w is the inertia weight and c_1 and c_2 are constants and r_1 and r_2 are random numbers in the range [0...1]. p_i is the personal best position of the particle and p_g is the global best position of the swarm. The second part of the above equation is usually referred to as the cognition part because due to this the particle's new velocity is influenced by its personal best position; whereas the third part is referred to as the social part because due to this part the particle's new velocity is influenced by the swarms best position. PSO has been the focus of many research efforts. Many improvements have been suggested and many more are still possible. Even in its current state, PSO is considered a fast and robust method for optimization of continuous nonlinear functions.

2.1. **PSO variations incorporating evolution.**

Angeline [4] incorporated a selection process similar to that of evolutionary algorithms into PSO and showed that the selection based PSO algorithm exhibits better performance on some classes of problems. Clerc [8] was one of the first ones to use reproduction in PSO. In this approach the particles of PSO were capable of generating new particles, killing themselves or modify the inertia and acceleration coefficient. Løvberg et al. [24, 25] suggested the use of an arithmetic cross-over operator for reproduction, which selected two particles randomly to create an offspring. Numerous other authors [13, 16, 26, 27, 39, 41, 42, 43] have used mutation with PSO to create hybrid PSO algorithms which can avoid premature convergence. Hendtlass [15], Kannan et al. [17], Zhang and Xie [44] have all made use of an arithmetic cross-over operator which performs cross-over between three randomly selected parents.

Diosan and Oltean [12] suggest the use of GA to create a hybrid technique in which the PSO algorithm is evolved. Poli et al. [28] explore the possibility of evolving the velocity update equation through the use of genetic programming. In this approach the PSO is specialized for a certain class of problem by training on learning set data. Once trained, the newly evolved equation is used to update the velocity of all the particles for that class of problems and the update equation doesn't change afterwards.

2.2. **PSO variations utilizing multiple velocity update equations.** Al-Kazemi and Mohan [1, 2, 3] suggested that the main swarm be divided into two sub-swarms. Each sub-swarm can be in either attraction or repulsion phase and use a different velocity update equation for each phase. Rigit and Vesterstrøm [30, 31, 40] followed a similar approach except for the fact that they didn't use sub-swarms and the whole swarm alternated between attraction and repulsion phase. During attraction the standard velocity update equation is used. Whereas during repulsion a modified velocity update equation is used. Thompson et al. [38] have used a cluster-based PSO in which the velocity update equation used for the cluster centers are different from those used for particles within the clusters. Silva et al. [35] implemented the predator-prey PSO in which they used one predator particle to pursue the global best prey particle. The velocity update equation of the predator particle is different from that of the prey particles.

2.3. **Other PSO variations.** Deng et al. [10] incorporated double justification in the PSO for the resource-constrained project scheduling problem. Cai et al. [5] proposed the performance-dependent adaptive PSO. Cui et al. [9] introduced a new fast PSO that does not evaluate all new positions owning a fitness and associated reliability value of each particle of the swarm. Lin et al. [22] proposed the immune PSO which combines the immune algorithm and the PSO to perform the parameter learning for a functional-link-based neural fuzzy network. Cai et al. [6] proposed a modified version of PSO combined with a self-adjusting cognitive selection strategy and mutation strategy. Shen and Tsai [32] developed a systematic design strategy for Co-Planar Waveguide fed monopole antennas by using an improved PSO to determine the optimized values of an equivalent circuit's components. Li et al. [21] proposed a Fine-grained parallel particle swarm optimization method based on GPU acceleration, which maps a parallel PSO algorithm to texture-rendering on consumer-level graphics cards. Su et al. [37] trained the weights of the Artificial Neural Networks by using PSO. Chang et al. [7] presented a parallel version of the PSO algorithm together with three communication strategies which can be used according to the independence of the data. Liu et al. [23] proposed the real-coded genetic algorithm to resolve the design of automobile fixtures optimization problem. Furusho et al. [14] proposed a decentralized optimization method for production scheduling, transportation routing for AGVs and motion planning for material handling robots simultaneously.

3. **Proposed extension.**

In the classical PSO presented by Kennedy and Eberhart [18], a swarm of particles is used to explore the search space. The individual particles of the swarm are allowed to move through the whole search space in an effort to find the optimum. As the particles move about the search space, they take guidance from other particles which have higher fitness. The particles are also influenced by their own experiences. To achieve this, each particle keeps information about itself and the rest of the swarm. Each particle keeps track of its current position, current velocity and personal best position. In each of the iterations a velocity update equation is used to determine the change in the velocity using information about that particle's best position, the global best position and random values. The new velocity is then used to update the position of the particle thus enabling it to move through the search space. Taking inspiration from existing studies and revisiting the swarm in nature we suggest that both evolution and diversity are essential for PSO. Hence we propose a novel approach in which we incorporate both the concepts into PSO. We allow our PSO to utilize different velocity update equations for different particles to ensure diversity in the swarm. We also constantly evolve these velocity update equations to allow the PSO to adapt to the specific problem.

We first discuss the use of a different velocity update equation for each particle. We have seen many researchers [1, 2, 3, 30, 31, 35, 38, 40] experimenting with PSO having more than one velocity update equations. In all these approaches, the researchers have used only two velocity update equations. Also these equations are predefined and do not change. We however suggest the use of a different velocity update equation for each particle. So in addition to keeping track of its current position, current velocity, personal best position and global best position the particle will also keep track of the velocity update equation used to update its velocity. The logic behind it is based upon inspiration from nature. The particles within a swarm tend to follow a generally similar path towards a common goal but there is slight differences in the manner in which they proceed, this difference can be attributed to the individual's own personality. Each individual in the swarm tends to have a slightly different personality from the rest. So by keeping a separate update equation for each individual, we are giving each particle a personality. This addition introduces a much need property to the PSO which is diversity.

Now coming towards incorporating evolution into PSO. We have learned from literature [4, 8, 11, 13, 15, 16, 17, 24, 25, 26, 27, 28, 29, 36, 39, 41, 42, 43, 44] that evolution when incorporated into PSO can result in improved performance. It has also been observed that different update equations give different results for different class of problems [28]. We also suggest that each velocity update equation be evolved throughout the execution of the PSO. Thus, by allowing the velocity update equations of each particle to be evolved, we are letting the PSO adapt to the specific problem. In nature we observe that the personality of an individual changes over time, this change in the personality of an individual is influenced by personalities of other individuals in the swarm. By allowing each particle to have a different velocity update equation and by evolving those equations through GP we have developed a PSO hybrid which is, not only diverse, but can also adapt to the optimization problem.

The GP that we have utilized will differ from classical GP in the sense that the population size will be limited (i.e. the number of programs in a generation will be equal to the number of particles). The function and terminal set used for GP is similar to that of [28]. The function set includes the functions $+$, $-$, $\times$ and the protected division DIV. The terminal set includes the position of the particle x, the velocity of the particle v, the personal best position of the particle p, the global best of the swarm g and a constant c. In the beginning while initializing the position of the particle the velocity update equation of each particle will also be initialized. Within each iteration of PSO algorithm, along with updating the position of the particle and the velocity, the velocity update equation will also be updated using the cross over and mutation operators of genetic programming. By introducing these two enhancements we hope to achieve diversity and adaptability in the classical PSO. The new PSOGP algorithm is presented in Figure 1. In line 1 the task of initializing the swarm includes initializing the positions, velocities and velocity update equation of the particles. In line 4 the velocity update equation is evolved by either using cross over operator or the mutation operator. One or two parents are selected from the original swarm at random depending upon the operator that is going to be applied. The particles having better fitness have a greater probability of being selected.

We also compare the PSOGP with a modified version of itself. We call this variant PSOGPNR. PSOGPNR differs from PSOGP in the sense that we have used a partial non random initialization technique for the GP, wherein some individuals in the GP population are not randomly initialized but are rather initialized with an individual which is known to have a high fitness. In PSOGPNR we have also employed a generation gap strategy in order to protect the non-randomly initialized individuals in the GP population from becoming extinct. The standard velocity update equation of PSO is considered to be a

```
1. Initialize the swarm
2. While termination criteria is not meet do
3. Begin
       For each particle evolve velocity update equation
       Update velocity of each particle
       Update position of each particle
       Update personal best and global best
4. End
```

FIGURE 1. PSOGP Algorithm

good function for updating the velocity of the particles in a swarm. We were interested in determine if we could achieve better performance by having some of the particles utilize the standard velocity update equation of PSO. To this end we employed a partial non random initialization technique for initializing the velocity update equations of the particles. According to this technique, not all of the individuals were randomly initialized. A small percentage of the individuals were initialized with the standard velocity update equation of PSO. In order to protect these individuals from becoming extinct during the execution of the algorithm, a generation gap was used which only allowed the randomly initialized individuals to be replaced with their children and preserved the non-randomly initialized particle till the end. A point to be noted is that this only applies to the velocity update equation; the position and velocity of the particles were randomly initialized just as in the standard PSO.

4. **Experiment.**

We compared the performance of PSOGP with its own variant, PSOGPNR, and with the standard PSO. The three algorithms were tested on four benchmark problems which have been commonly used in other studies of PSO [4, 19, 34]. These are given in Table 1.

TABLE 1. Test Functions

Function	Equation	Range
Sphere	$f(x) = \sum_{i=1}^{n} x_i^2$	[-5,5]
Rosenbrock	$f(x) = \sum_{i=1}^{n}(100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	[-2.048, 2.048]
Rastrigin	$f(x) = \sum_{i=1}^{n}(x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12, 5.12]
Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600, 600]

The 10 dimensional versions of these functions were employed. Each PSO was executed 100 times. During each execution a maximum of 200 iterations were allowed. The initial positions of the particles, in the swarm, were uniformly distributed over the entire search space. No restriction was placed on the velocity. However the positions of the particles were restricted to be within the limits specified for each function. 10% of the particles were initialized with the standard velocity update equation of PSO, whereas the velocity update equation of the other 90% particles were randomly initialized. The cross over operator was applied with the probability of 0.9 and the mutation operator was applied with the probability of 0.1. A generation gap of 0.9 was used, that is 10% of the particles which were initialized with the standard velocity update equation of PSO did not change their velocity update equation. The parents were selected based on their fitness (i.e. particles with greater fitness had a greater chance of being selected for reproduction). With each iteration the velocity update equation of 90% of the particles were changed.

The size of the swarm was kept constant. The number of particles for the 10 dimensional problems was 16. The acceptable error was set at 0.0001. Any PSO with an error less than the threshold was considered successful. PSO was terminated when either the optimal solution was found or it had executed 200 iterations. The parameters for the experiments have been taken from literature and were empirically determined to be good values for these test functions. When using PSOGP on real world problems, these can be adjusted according to the specific problem.

To compare the performance of the three PSO algorithms, we made use of four different performance measures. These are mean error, success rate, optimum rate and iterations. To calculate the mean error, we calculated the error of the global best position at the end of each run and took the average of 100 runs. We also calculated the standard deviation of mean error. The success rate was calculated by counting the number of runs in which the global best position had an error value less than the threshold at the end of the run. Similarly the optimum rate was calculated by counting the number of runs in which the PSO was able to find the exact optimum, i.e. the global best position at the end of the run had an error value of zero. The number of iterations were also counted for each PSO and averaged over 100 runs. In addition to comparing the performance of PSOGP with other PSO algorithms, we were also interested in observing the velocity update equations that were evolved during the execution of PSO. To this end we recorded all of the velocity update equations which were used for each particle for a single run of the algorithm for all four benchmark functions. We were interested in seeing whether certain equations were used more frequently than others or not. We calculated the frequency of use of each velocity update equations and analyzed the more frequently occurring ones.

5. **Results.** During the experiment six different measures were collected for each of the PSO. Table 2 lists results from the conducted experiment. This table lists the test functions and the mean and standard deviation of the error. The results show that PSOGP was able to achieve significantly better performance over PSOGPNR and PSO. Although PSOGPNR gave a better performance as compared to PSO but its results were worse than that of PSOGP. This leads us to believe that the non random initialization technique did not give improvements in terms of mean error. For PSOGP the mean and standard deviation of all but Rosenbrock function are zero because in all other cases PSOGP was able to find the optimum solution 100% of the times. In Table 3 the percentage of success and the percent of times the optimum was found are listed for PSOGP, PSOGPNR and PSO. Again with the exception of the Rosenbrock function, PSOGP was able to find the optimum 100% of the time. This is a significant improvement from the standard PSO. There are many optimization problems in which it is important that the exact optimum is found instead of just finding a near optimum. Although the standard PSO is good at finding near optimum solutions but it has trouble when it comes to finding the exact optimum as is evident from the results. This is the main strength of PSOGP as it is very good at finding the exact optimum. PSOGPNR did give better performance as compared to PSO but its performance was worse than that of PSOGP which further strengthens our claim that the non random initialization strategy which forced some of the particles to use the standard PSO velocity update equation does not give better performance. Table 4 presents the number of iterations required by each PSO variant. Once again we see that on the average a lesser number of iterations were required by PSOGP to find the optimum.

Table 5 present the 10 most frequently evolved velocity update equations during a single execution of PSOGP for all of the test functions. The velocity update equations are listed in pre-order notation and represent the change in the velocity of the particle. An

Table 2. Mean Error

Function	PSOGP	PSOGPNR	PSO
Sphere	0.00 ± 0.00	46.93E-14 ± 1.37E-13	4.43E-12 ± 2.76E-11
Rosenbrock	5.38E-02 ± 1.57E-01	7.30E+00 ± 2.66E+00	4.82E+00 ± 1.58E+00
Rastrigin	0.00 ± 0.00	1.13E-07 ± 2.22E-07	1.56E+01 ± 6.57E+00
Griewank	0.00 ± 0.00	4.86E-08 ± 8.00E-08	1.31E-01 ± 7.58E-02

Table 3. Success and Optimum Rate

Function	PSOGP		PSOGPNR		PSO	
	Success	Optimum	Success	Optimum	Success	Optimum
Sphere	100.00%	100.00%	100.00%	99.00%	100.00%	0.00%
Rosenbrock	80.00%	78.00%	17.00%	16.00%	0.00%	0.00%
Rastrigin	100.00%	100.00%	100.00%	97.00%	0.00%	0.00%
Griewank	100.00%	100.00%	100.00%	38.00%	0.00%	0.00%

Table 4. Iterations

Function	PSOGP	PSOGPNR	PSO
Sphere	48.51	112.31	200
Rosenbrock	109.89	174.2	200
Rastrigin	45.27	149.14	200
Griewank	35.02	194.08	200

Table 5. Frequency of occurrence of equations for test functions

Equation	Sphere	Rosenbrock	Rastrigin	Griewank
x	2305	6502	1932	1726
$-gg, -pp, -vv, -xx, -cc$	1464	7430	782	1415
g	2275	3546	2061	1642
p	1756	3275	1862	1116
c	1999	1990	1709	1698
v	1376	3366	1670	933
$-gx$	500	2363	201	388
$-px$	345	1663	145	230
$-vx$	219	1694	192	172
$-cx$	562	700	189	745

explanation of the evolved equations is given in Table 6. As we can see that for all of the test functions similar velocity update equations were evolved more frequently. One of the frequently occurring velocity update equations is c, meaning that a random number was used to update the velocity. This is interesting because as a result of this the velocities of the particles were randomly changed with a higher frequency. We can also see the social component of the classical PSO ($-gx$) being used quite frequently. The cognition component of the classical PSO ($-px$) also occurs with higher frequency. However we rarely see both the cognition and social components being used simultaneously in PSOGP. This together with that fact that PSOGPNR, in which we forced some particles to use the standard PSO velocity update equation which has both social and cognition components, gave poorer performance as compared to PSOGP shows that the key to PSOGP success

Table 6. Explanation of frequently occurring equations

Equation	Description
$-cx$	Calculates the distance of the particle from a random point
$-gx$	Calculates the distance of the particle from global best position
$-px$	Calculates the distance of the particle from personal best position
$-vx$	Calculates the difference of the particle's velocity and position
x	The current position of the particle
v	The current velocity of the particle
p	The current personal best position of the particle
g	The current global best position
c	A random number
$-gg, -pp, -vv, -xx, -cc$	Evaluates to 0

is its ability to use different velocity update equations and to evolve them. Another interesting equation which is evolved frequently is v, meaning that the current velocity of the particle is added to its velocity causing the velocity to be doubled. Hence we can deduce that quite often the particle is looking to accelerate and double its velocity. We also see the equations $-gg, -pp, -cc, -vv, -xx$ being evolved quite often. They all can be considered the same because all five evaluate to zero and thus cause the velocity of the particle to remain unchanged. In classical PSO the velocity of every particle has to be changed but in PSOGP we see that during many iterations the velocities of particles are left unchanged. We can conclude that although a large number of random velocity update equations were evolved during the execution of PSOGP, the useful equations were evolved more often than others.

6. Conclusion.

By using a separate velocity update equation for each particle we were able to introduce enough diversity in the swarm to allow the swarm to thoroughly explore the whole search space and find the optimum with great success. With the addition of evolution of these functions we were able to achieve significantly faster convergence. The main advantage of PSOGP is that it is capable of finding the exact optimum which is very crucial for some optimization problems. We have tried a partial non random initialization strategy which forced the algorithm to utilize the standard PSO velocity update equation. We concluded that this approach did not improve the performance of the algorithm. However in future research we would like to try a partial non random initialization strategy using some of the equations which we have found in this study to have been used more frequently. It would be interesting to see if we can improve the performance of algorithm in terms of execution time by using these equations directly instead of letting the algorithm evolve them. We might also consider limiting the GP population to only these equations. A point worth mentioning is that we have not taken into account the concepts from other extensions of PSO such as neighborhoods. It will be interesting to see whether we can achieve even better performance by including neighborhoods in our PSO. We can also incorporate other ingredients in the velocity update equation as in [29] and [11]. Also only the basic GP is used. Another research direction can be to use different variations of GP to see if an improvement can be made.

Acknowledgment. This work is partially supported by Higher Education Commission of Pakistan. The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] B. Al-Kazemi, C. K. Mohan, Multi-Phase Discrete Particle Optimization, *Proc. of the Int. Workshop on Frontiers in Evolutionary Algorithms*, pp.622-625, 2002

[2] B. Al-Kazemi, C. K. Mohan, Multi-Phase Generalization of the Particle Swarm Optimization Algorithm, *Proc. of the IEEE Congress on Evolutionary Computation*, pp.489-494, 2002

[3] B. Al-Kazemi, C. K. Mohan. Training Feedforward Neural Networks using Multi-Phase Particle Swarm Optimization, *Proc. of the Ninth Int. Conf. on Neural Information Processing*, vol.5, pp.2615-2619, 2002

[4] P. J. Angeline, Using Selection to Improve Particle Swarm Optimization, *Proc. of the IEEE World Congress on Computational Intelligence*, Anchorage, Alaska, USA, pp.84-89, 1998

[5] X. Cai, Z. Cui, J. Zeng and Y. Tan, Performance-Dependent Adaptive Particle Swarm Optimization, *Int. J. of Innovative Computing, Information and Control*, vol.3, no.6(B), pp.1697-1706, 2007.

[6] X. Cai, Z. Cui, J. Zeng and Y. Tan, Particle Swarm Optimization with Self-Adjusting Cognitive Selection Strategy, *Int. J. of Innovative Computing, Information and Control*, vol.4, no.4, pp. 943-952, 2008.

[7] J. Chang, S. Chu, J. F. Roddick and J. Pan, A Parallel Particle Swarm Optimization Algorithm with Communication Strategies, J. of Information Science and Engineering, vol.21, no.4, pp. 809-818, 2005

[8] M. Clerc, Think Locally, Act Locally: The Way of Life of Cheap-PSO, an Adaptive PSO, Technical report, http://clerc.maurice.free.fr/pso/,2001

[9] Z. Cui, J. Zeng and G. Sun, A Fast Particle Swarm Optimization, *Int. J. of Innovative Computing, Information and Control*, vol.2, no.6, pp.1365-1380, 2006.

[10] L. Deng, Y. Lin, W. Zheng and Y. Xi, Incorporating Justification in the Particle Swarm Optimization for the RCPSP, *Int. J. of Innovative Computing, Information and Control*, vol.4, no.9, pp.2315-2324, 2008.

[11] C. Di Chio, R. Poli, W. B. Langdon, Evolution of Force-Generating Equations for PSO using GP, *AI*IA Workshop on Evolutionary Computation, Evoluzionistico GSICE05*, University of Milan Bicocca, Italy, 2005

[12] L. Diosan, M. Oltean, Evolving the Structure of the Particle Swarm Optimization Algorithms, *Proc. of the European Conf. on Evolutionary Computation in Combinatorial Optimization EcoCOP2006, LNCS*, Budapest, Hungary, vol.3906, pp.25-36, 2006

[13] S. C. Esquivel, C. A. Coello Coello, On the Use of Particle Swarm Optimization with Multimodal Functions, *Proc. of the IEEE Transactions on Evolutionary computation*, vol.2, pp.1130-1136, 2003

[14] T. Furusho, T. Nishi and M. Konishi, Distributed Optimization Method for Simultaneous Production Scheduling and Transportation Routing in Semiconductor Fabrication Bays, *Int. J. of Innovative Computing, Information and Control*, vol.4, no.3, pp.559-575, 2008.

[15] T. Hendtlass, A Combined Swarm Differential Evolution Algorithm for Optimization Problems, *Proc. of the Fourteenth Int. Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, Lecture Notes in Computer Science, vol.2070, pp.11-18, Springer-Verlag, 2001

[16] H. Higashi, H. Iba, Particle Swarm Optimization with Gaussian Mutation, *Proc. of the IEEE Swarm Intelligence Symposium*, pp.72-79, 2003

[17] S. Kannan, S. M. R. Slochanal, P. Subbaraj, and N. P. Padhy, Application of Particle Swarm Optimization Technique and its Variants to Generation Expansion Planning, Electric Power Systems Research, vol.70, no.3, pp.203-210, 2004

[18] J. Kennedy, R. Eberhart, Particle Swarm Optimization, *Proc. of the IEEE Int. Conf. on Neural Networks*, Perth, Australia, vol.4, pp.1942-1948, 1995

[19] J. Kennedy, Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance, *Proc. of the IEEE Congress on Evolutionary Computation*, Washington DC, USA, vol.3, pp.1938, 1999

[20] J. R. Koza, *Genetic Programming: On the programming of Computers by Means of Natural Selection*, MIT Press, 1992

[21] J. Li, D. Wan, Z. Chi and X. Hu, An Efficient Fine-Grained Parallel Particle Swarm Optimization Method Based on GPU-Acceleration, *Int. J. of Innovative Computing, Information and Control*, vol.3, no.6(B), pp. 1707-1714, 2007

[22] C. Lin, Y. Liu and C. Lee, An Efficient Neural Fuzzy Network based on Immune Particle Swarm Optimization for Prediction and Control Applications, *Int. J. of Innovative Computing, Information and Control*, vol.4, no.7, pp.1711-1722, 2008.

[23] T. Liu, C. Chen and J. Chou, Intelligent Design of the Automobile Fixtures with Genetic Algorithms, *Int. J. of Innovative Computing, Information and Control*, vol.4, no.10, pp.2533-2550, 2008.

[24] M. Løvberg, Improving Particle Swarm Optimization by Hybridization of Stochastic Search Heuristics and Self-Organized Critically, Master's thesis, Department of Computer Science, University of Aarhus, Denmark, 2002

[25] M. Løvbjerg, T. K. Rasmussen, T. Krink, Hybrid Particle Swarm Optimiser with Breeding and Subpopulations, *Proc. of the Genetic and Evolutionary Computation Conf., GECCO-2001*, San Francisco, California, USA, pp.469-476, 2001

[26] V. Miranda, N.Fonseca, EPSO - Best-of-two-worlds Mets-heuristic Applied to Power System Problems, *Proc. of the IEEE Congress on Evolutionary Computation*, vol.2, pp.1080-1085, 2002

[27] V. Miranda, N. Fonseca, EPSO - Evolutionary Particle Swarm Optimization, a New Algorithm with Applications in Power Systems, *Proc. of the Asia Pacific IEEE/PES Transmission and Distribution Conf. and Exhibition*, Yokohama, Japan, vol.2, pp.745-750, 2002

[28] R. Poli, W. B. Langdon, O. Holland, Extending Particle Swarm Optimisation via Genetic Programming, *Proc. of the 8th European Conf. on Genetic Programming*, Lausanne, Switzerland, pp.291-300, 2005

[29] R. Poli, C. Di Chio, W. B. Langdon, Exploring Extended Particle Swarms: A Genetic Programming Approach, *Proc. of the 2005 Conf. on Genetic and Evolutionary Computation*, Washington DC, USA, pp.169-176, 2005

[30] J. Riget, J. S. Vesterstrøm, A Diversity-Guided Particle Swarm Optimizer - The ARPSO, Technical report, Department of Computer Science, University of Aarhus, 2002

[31] J. Riget, J. S. Vesterstrøm, Controlling Diversity in Particle Swarm Optimization, Master's thesis, University of Aarhus, Denmark, 2002

[32] M. Shen and Y. Tsai, CPW-FED Monopole Antenna Characterized by using Particle Swarm Optimization Incorporating Decomposed Objective Functions, *Int. J. of Innovative Computing, Information and Control*, vol.4, no.8, pp. 1897-1919, 2008

[33] Y. Shi ,R. Eberhart, A Modified Particle Swarm Optimizer, *Proc. of the IEEE World Congress on Evolutionary Computation*, Anchorage, Alaska, USA, pp.69-73, 1998

[34] Y. Shi, R. Eberhart, Empirical Study of Particle Swarm Optimization, *Proc of the IEEE Congress on Evolutionary Computation*, Washington DC, USA, vol.3, pp.1950, 1999

[35] A. Silva, A. Neves, E. Costa, An Empirical Comparison of Particle Swarm and Predator Prey Optimisation, *Proc. of the Thirteenth Irish Conf. on Artificial Intelligence and Cognitive Science, In: Lecture Notes in Artificial Intelligence*, vol.2464, pp.103-110, Springer-Verlag, 2002

[36] A. Stacey, M. Jancic, I. Grundy, Particle Swarm Optimization with Mutation, *Proc. of the IEEE Congress on Evolutionary Computation*, vol.2, pp.1425-1430, 2003

[37] T. Su, J. Jhang and C. Hou, A Hybrid Artificial Neural Networks And Particle Swarm Optimization For Function Approximation, *Int. J. of Innovative Computing, Information and Control*, vol.4, no.9, pp. 2363-2374, 2008.

[38] B. B. Thompson, R. J. Marks, M. A. El-Sharkawi, W. J. Fox, R. T. Miyamoto, Inversion of Neural Network Underwater Acoustic Model for Estimation of Bottom Parameters using Modified Particle Swarm Optimizer, *Proc. of the Int. Joint Conf. on Neural Networks*, pp.1306, 2003

[39] T-O. Ting, M. V. C. Rao, C. K. Loo, S-S. Ngu, A New Class of Operators to Accelerate Particle Swarm Optimization, *Proc. of the IEEE Congress on Evolutionary Computation*, vol.4, pp.2406-2410, 2003

[40] J. S. Vesterstrøm, J. Riget, Particle Swarms: Extensions for Improved Local, Multi-Modal, and Dynamic Search in Numerical Optimization, Master's thesis, Department of Computer Science, Univerisyt of Aarhus, 2002

[41] C. Wei, Z. He, Y. Zheng, W. Pi, Swarm Directions Embedded in Fast Evolutionary Programming, *Proc. of the IEEE Congress on Evolutionary Computation*, vol.2, pp.1278-1283, 2002

[42] X. Yao, Y. Liu, Fast Evolutionary Programming, *Proc. of the Fifth Annual Conf. on Evolutionary Programming*, pp.451-460, 1996

[43] X. Yao, Y. Liu, G. Liu, Evolutionary Programming made Faster, IEEE Transactions on Evolutionary Computation, vol.3, no.2, pp.82-102, 1999

[44] W-J. Zhang, X-F. Xie, DEPSO: Hybrid Particle Swarm with Differential Evolution Operator, *Proc. of the IEEE Int. Conf. on System, Man, and Cybernetics*, vol.4, pp.3816-3821, 2003