Recent Advances in Particle Swarm

Xiaohui Hu
Department of Biomedical Engineering
Purdue Univesity.
West Lafayette, IN 47907
xhu@purdue.edu

Yuhui Shi
EDS Embeded Systems Group
Kocomo, Indiana, USA
Yuhui.Shi@eds.com

Russ Eberhart
Purdue School
of Engineering and Technogloy
Indianapolis, IN 46202
reberhar@iupui.edu

Abstract- This paper reviews the development of the particle swarm optimization method in recent years. Included are brief discussions of various parameters. Modifications to adapt to different and complex environments are reviewed, and real world applications are listed.

I. INTRODUCTION

Particle swarm optimization (PSO) is population based stochastic optimization technique developed by Kennedy and Eberhart in 1995 [1, 2]. As a relatively new evolutionary paradigm, PSO has grown in the past several years, and over 300 papers related to PSO have been published. More and more researchers are interested in this new algorithm and it has been investigated from various perspectives (Figure 1).

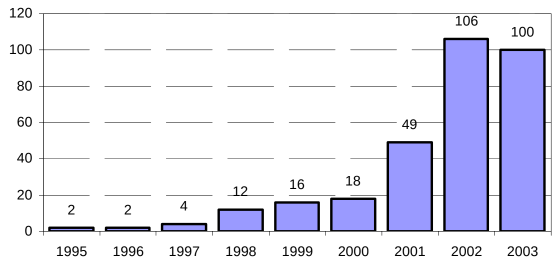

Figure 1: Number of papers published in each year
(Incomplete data for year 2003)

Following the introduction, major developments in PSO are reviewed in section II. The original version is presented. Following are discussions of various parameters used in PSO. Modifications to improve the algorithms are also discussed. In section III, PSO in various scenarios is discussed, including training neural networks, multiobjective optimization, dynamic tracking, and constraint optimization. Finally, some typical real world applications are presented.

II. BASIC ALGORITHM

PSO is inspired by the behavior of bird flocking. Assume the following scenario: a group of birds are randomly searching food in an area. There is only one piece of food in the area being searched. The birds do not know where the food is. But they know how far the food is and their peers' positions. So what's the best strategy to find the food? An effective strategy is to follow the bird which is nearest to the food.

PSO learns from the scenario and uses it to solve the optimization problems. In PSO, each single solution is like a "bird" in the search space, which is called "particle". All particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the particles with the best solutions so far.

A. Basic algorithm

PSO is initialized with a group of random particles (solutions) and then searches for optima by updating each generation. In every iteration, each particle is updated by following two "best" values. The first one is the location of the best solution (fitness) a particle has achieved so far. This value is called *pBest*. Another "best" value is the location of the best solution that any neighbor of a particle has achieved so far. This best value is a neighborhood best and called *nBest*. When a particle takes all the population as its neighbors, the best location is a global best and is called *gBest*.

The general process of PSO is as follows

```
Initialize the particle population randomly

Do
   Calculate fitness values of each particle
   Update pbest if the current fitness value is better
   than pBest
   Determine nBest for each particle: choose the
   particle with the best fitness value of all the
   neighbors as the nBest
   For each particle
      Calculate particle velocity according to (a)
      Update particle position according to (b)
While maximum iterations or minimum criteria is
not attained
```

Figure 2: The process of particle swarm

The core of PSO is the updating formulae of the particle, which can be represented as follows. Equation (a) calculates a new velocity for each particle (potential solution) based on its previous velocity (V_{id}), the particle's location at which the best fitness so far has been achieved (p_{id} , or *pBest*), and the neighbor's best location

(p_{nd}, or *nBest*) at which the best fitness in a neighborhood so far has been achieved. Equation (b) updates each particle's position in the solution hyperspace. *Rand*() and *rand*() are two random numbers independently generated. c_1 and c_2 are two learning factors, which control the influence of *pBest* and *nBest* on the search process. The use of the inertia weight *w* has provided improved performance in a number of applications [3].

$$V_{id} = w \times V_{id} + c_1 \times rand() \times (p_{id} - x_{id}) + c_2 \times Rand() \times (p_{nd} - x_{id}) \quad (a)$$
$$x_{id} = x_{id} + V_{id} \quad (b)$$

B. The selection of pBest

pBest is the best location the particle has achieved so far. It can be viewed as the particle’s memory. Currently only one memory slot is allocated to each particle,

The best location does not always depend on the value of the fitness function only. Many constraints can be applied to the definition of the best location to adapt to different problems. And this won’t lower the search ability and performance. For example, in nonlinear constrained optimization problems [4, 5], the particles only remember those positions in the feasible space and disregard unfeasible solutions. This simple modification successfully locates the optimum of a series of benchmark problems. In a multiobjective optimization (MO) environment [6, 7], the best position is determined by Pareto dominance (if solution A is not worse than solution B in every objective dimension and is better than solution B in at least one objective dimension, solution B is dominated by solution A). Another popular technique is memory reset. In dynamic environments [8, 9], particles *pBest* will be reset to the current value if the environment changes.

C. The selection of nBest

nBest is the best position that neighbors of a particle have achieved so far. The neighborhood of a particle is the social environment a particle encounters. The selection of *nbest* consists of two steps, which are to determine the neighborhood and select the *nbest* among the neighbors.

Traditionally, PSO takes certain predetermined conjunct particles as the neighbors. The number of neighbors or the size of the neighborhood will affect the convergence speed of the algorithm. It is generally accepted that a larger neighbor size will make the particles converge faster, while a small neighbor size will help to prevent the particle from premature or pre-convergence. *gBest* is an extreme situation of the *nBest* version. It takes the whole population as the neighbors of each particle [1]. Although Kennedy *et al.* investigated various neighborhood structures and their influences on performance [10, 11], no conclusive results have been reached so far. In multiobjective optimization problems, an external repository of Pareto optimal solutions is used as the neighborhood in several papers discussed in section III. Brits *et al.* [1, 12] choose topological neighborhood method to deal with multimodal functions. However, it is doubtful that a topological neighborhood structure can escape from local optima easily.

The selection of the *nBest* is usually determined by comparing fitness values among neighbors. However, in a multi-objective optimization environment, the situation is more complicated when there are multiple fitness values for each particle. Section III provides a more detailed discussion of this topic. Peram *et al.* [13]used the ratio of the fitness and the distance of other particles to determine the *nbest* and claimed it outperforms the original version of PSO.

D. Learning factors

The learning factors c_1 and c_2 in equation (a) represent the weighting of the stochastic acceleration terms that pull each particle toward *pBest* and *nBest* positions [14]. From a psychological standpoint, the cognitive term represents the tendency of individuals to duplicate past behaviors that have proven successful, whereas the social term represents the tendency to follow the successes of others.

Both c_1 and c_2 are sometimes set to 2.0. The obvious reason is it will make the search cover all surrounding regions which is centered at the *pBest* and *nBest*. 1.49445 is also used according to the work by Clerc [15] which indicates that a constriction factor may be necessary to insure convergence of PSO [14]

In most cases, the learning factors are identical. That puts the same weights on social searching and cognitive searching. Kennedy investigated two extreme cases: social-only model and cognitive-only model and found out that both parts are essential to the success of particle swarm searching [16]. No definitive conclusions about asymmetric learning factors have been reported.

E. Inertia weight

Inertia weight was first introduced by Shi and Eberhart [3]. The function of inertia weight is to balance global exploration and local exploitation. Linearly decreasing inertia weights were recommended by the authors. Zheng *et al.* [17] claimed that PSO with increasing inertia weight performs better. However, the authors used a different set of learning factors, and it is not clear from the paper how this affects the performance. A randomized inertia weight is also used in several reports. The inertia weight can be set to [0.5 + (Rnd/2.0)], which is selected in the spirit of Clerc’s constriction factor [14].

F. Other parameters

Particles’ velocities are clamped to a maximum velocity Vmax, which serves as a constraint to control the global exploration ability of particle swarm. Generally, Vmax is set to the value of the dynamic range of each variable.

The population size selected is problem-dependent. Sizes of 20-50 are most common. In some situations, large population sizes may be used to adapt to different requirements.

III. DISCRETE VERSION OF PSO

Many optimization problems involve in discrete or binary variables. Typical examples are scheduling problems or routing problems. The updating formula of POS and procedures are oriented from and designed for continuous spaces. Some changes have to be made to adapt to the discrete spaces. The coding changes may be simple, but it is hard to define the meanings of velocities and determine the changes of trajectories.

Kennedy *et al.* [18] defined the first discrete binary version of PSO. The particles are coded as binary strings. The velocities are constrained to the interval [0, 1] by using the sigmoid function and are interpreted as "changes of probabilities". Chang *et al.* [19] applied the method to feeder reconfiguration problems and showed it is efficient in searching for the optimal solutions.

Mohan *et al.* [20] proposed several binary approaches (direct approach, quantum approach, regularization approach, bias vector approach, and mixed approach) and no conclusion was drawn from limited experimentation.

Hu *et al.* [21] introduced a modified PSO to deal with permutation problems. Particles were defined as permutations of a group of unique values. Velocities are redefined based on the similarity of two particles. Particles make switches to get a new permutation with a random rate defined by their velocities. A mutation factor is also introduced to prevent the current *pBest* from being stuck at local minima. Preliminary study on n-queens problem showed that the modified PSO is promising in solving the constraint satisfaction problems.

When dealing with integer variables, PSO sometimes are easily trapped into local minima. It seems PSO can locate the optimal area but fails to exploit more details. The philosophy behind the original PSO is to learn from individual's own experience and his peers' experience. How to effectively apply these rules to discrete problems is still an open issue. A direct translation of the original PSO might not be the only choice.

IV. PSO IN COMPLEX ENVIRONMENTS

A. Multiobjective optimization

In recent years, multi-objective optimization has been a very active research area. In multi-objective optimization (MO) problems, objective functions may be optimized separately from each other and the best solution may be found for each objective. However, perfect solutions which are optimal on all the objective dimensions can seldom be found due to the fact that the objective functions are often in conflict among themselves. This results in there being a group of alternative solutions which must be considered equivalent in the absence of information concerning the relevance of each objective relative to the others. The group of alternative solutions is known as a Pareto optimal set or Pareto front.

The information sharing mechanism in PSO is significantly different from other population based optimization tools. In genetic algorithms (GAs), chromosomes exchange information with each other by crossover, which is a two-way information sharing mechanism. In PSO, only *nBest* gives out information to others. It is a one-way information sharing mechanism. Due to the point-attraction characteristics, traditional PSO is not able to locate multiple optimal points simultaneously, which represent the Pareto front. Although multiple optimal solutions could be obtained through multiple runs with different weighted combinations of all the objectives, a method that could find a group of Pareto optimal solutions simultaneously is preferred.

In PSO, a particle is an independent intelligence agent, which searches the problem space based on its own experience and the experiences of peer particles. The former is the cognitive part of the particle update formula, while the latter is the social part of particle update formula. Both play crucial rules to guide the particle's searching. Thus, the selection of social and cognitive leaders (*nBest* and *pBest*) is the key point of MO-PSO algorithms [6, 7, 22-29].

The selection of the cognitive leader follows the same rule in traditional PSO. The only difference is that the leader is determined by Pareto dominance. It is possible to let each particle have multiple memory slots to store more Pareto optimal solutions. No report has been found in the literature.

The selection of the social leader includes two steps, which are similar to the selection of *nBest*. The first step is to formalize a candidate pool from which the leader is chosen. In traditional PSO, the leader is chosen from the *pbest* values of neighbors. A more popular method in MO-PSO is to use an external pool to store more Pareto-optimal solutions.

The second step is the process to choose the leader. The selection of *nBest* should satisfy the following two criteria: first, it should provide effective guidance to the particle to get better convergence speed; second, it also needs to provide a balanced search along the Pareto front to maintain population diversity. Two typical approaches have been employed in the literature: 1. Roulette wheel selection scheme: In this approach, all candidates are assigned weights based on some criteria, which can be crowding radius [29], crowding factor [24], niche count [24] or other measurements. Then random selection is used to choose the social leader [6, 24, 29, 30]. The aim of this process is to maintain population diversity. 2. Quantitative standards: In this approach, the social leader is determined by some procedure without any random selection involved [7, 22, 23, 27].

Ray *et al.* [29] combined the Pareto ranking scheme and PSO to handle MO problems. A set of leaders (SOL), which are better performing particles, are selected based on the Pareto ranks during each generation. The remaining particles will select a leader from the set of leaders as *nBest* and move to a new location. The selection of a leader from the SOL is based on a roulette wheel scheme that ensures SOL members with a larger crowding radius have a higher probability of being selected as a leader.

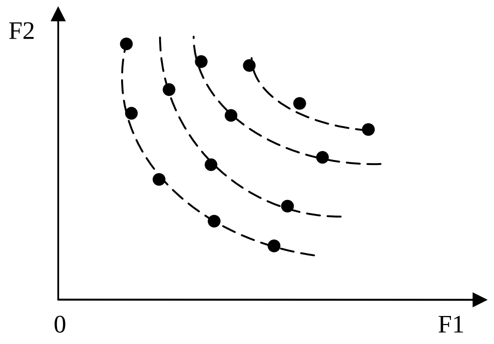

Figure 3: Pareto ranking scheme in Ray *et al.*

Coello Coello *et al.* [6] used a two-step selection process to get the social leader. First fitness hyperspace is divided into small hypercubes, and each cube was assigned a weight which is inversely proportional to the number of non-dominated solutions inside the cube. Then roulette-wheel selection is used to select one of the hypercubes from which the *nBest* will be picked. In the second step, a social leader will be randomly picked from the selected hypercube.

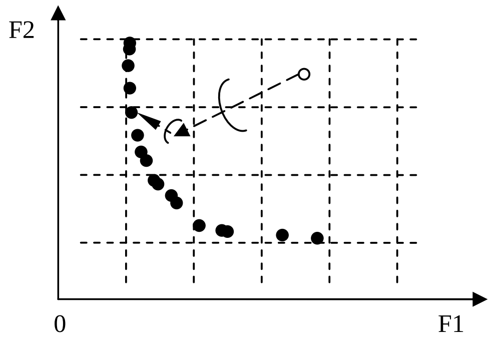

Figure 4: Particle search scheme in Coello Coello *et al.*

Mostaghim *et al.* [27] introduces the sigma method as a new method for finding the best social leader for each particle of the population. Sigma values are calculated for each individual in the candidate pool as well as the particle, while the sigma value for a two-objective optimization problem is defined as in (c). Then the particle will find a social leader with minimal sigma distance to the particle.

$$\sigma = \frac{f_1^2 - f_2^2}{f_1^2 + f_2^2} \quad (c)$$

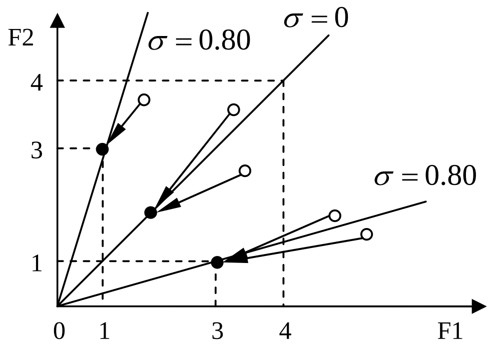

Figure 5: Sigma method in Mostaghim *et al.*

Fieldsend [22] *et. al* proposed a dominated tree for storing the particles shown in figure 6.

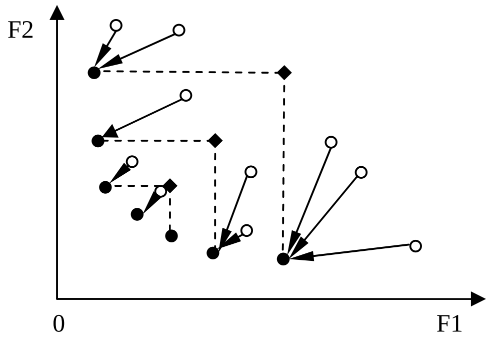

Figure 6: Pareto ranking scheme in Ray *et al.*

Hu *et. al* developed a dynamic neighborhood strategy to pick the social leader in MOPSO. The objectives are divided into two groups. One objective is defined as the optimization objective, while all the other objectives are defined as neighborhood objectives. First the distances between the current particle and all the candidates are calculated, and then a group of neighbors are picked based on the distances. Then the candidate with the minimum fitness value of the optimization objective becomes the social leader. The method is simple and intuitive. However, the drawback is that it is sensitive to the selection of objectives.

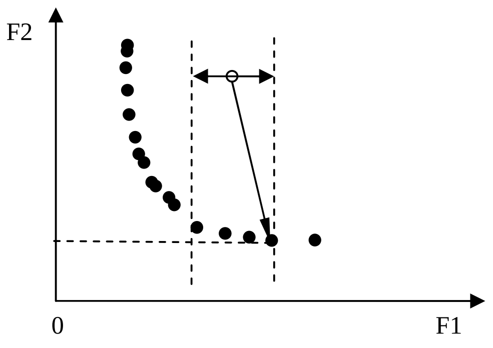

Figure 7: Dynamic neighbor PSO in Hu *et al.*

B. Constraint Optimization

There are some studies reported in the literature that extend PSO to constrained optimization problems. The goal of constrained optimization problems is to find the solution that optimizes the fitness function while satisfying a group of linear and non-linear constraints. For constrained optimization problems, the original PSO method needs to be modified to deal with constraints.

Hu and Eberhart [5] introduced a fairly easy but effective method to solve the constrained optimization problems. The preserving feasibility strategy is employed to deal with constraints. Two modifications were made to PSO algorithms. 1. When updating the memories, all the particles only keep feasible solutions in their memory, 2. During the initialization process, all particles are started with feasible solutions. Various test cases showed that PSO is much faster and better than other evolutionary constrained optimization techniques when dealing with optimization problems with linear or nonlinear inequities constraints. The disadvantage is that the initial feasible solution set is sometimes hard to find.

El-Gallad *et al.* [4] introduced a similar method. The only difference is that when a particle is outside of feasible space, it will be reset to the last best value found. The potential problem is that it may limit the particles to the region where they are initialized.

Parsopoulos *et al.* [31] converted the constrained optimization problem into a non-constrained optimization problem by adopted a non-stationary multi-stage assignment penalty function and then applied PSO to the converted problems. Several benchmark problems were tested and the author claimed it outperformed other evolutionary algorithms, such as Evolution Strategies and GAs.

Ray *et al.* [32] proposed a swarm metaphor with a multilevel information sharing strategy to deal with optimization problems. In a swarm, there are some better performers (leaders) that set the direction of search for the rest of the individuals. An individual that is not in the better performer list (BPL) improves its performance by deriving information from its closest neighbor in the BPL. The constraints are handled by a constraint matrix. A multilevel Pareto ranking scheme is implemented to generate the BPL based on the constraint matrix. It should be noted that the update of each particle uses a simple generational operator instead of the regular PSO formula. Test cases showed much faster convergence and much lower number of function evaluations.

C. PSO in dynamic environments

A dynamic system changes state frequently or even perhaps almost continuously. Many real-world systems involve in dynamic environments. For example, most of the computational time in scheduling systems is spent in rescheduling, caused by changes in customer priorities, unexpected equipment maintenance, etc. In real-world applications, then, these system state changes result in a requirement for frequent re-optimization.

Initial work in tracking dynamic systems with particle swarm optimization was reported in Eberhart and Shi [33]. The follow-up paper [9] introduces an adaptive PSO, which automatically tracks various changes in a dynamic system. Different environment detection and response techniques are tested on the parabolic benchmark functions, and re-randomization is introduced to respond to the dynamic changes. The detection method used is to monitor the behavior of the best particle in the population. Carlisle [8] used a random point in the search space to determine if the environment changes.

Blackwell [34] added a charged term to the particle updating formula, which keeps the particles in a extended swarm shape to deal with fast changing dynamic environments. No detection is needed in the proposed method.

V. APPLICATIONS

A. Artificial nerual network training

PSO has been applied to three main attributes of neural networks: network connection weights, network architecture (network topology, transfer function), and network learning algorithms. Most of the work involving the evolution of ANNs has focused on the network weights and topological structure. Usually the weights and/or topological structure are encoded as a chromosome in GAs. The selection of the fitness function depends on the research goals. For a classification problem, the rate of mis-classified patterns can be viewed as the fitness value.

Compared with the back-propagation training method, the advantage of PSO is that it can be used in cases with non-differentiable PE transfer functions and no gradient information available. There are several papers reporting use of PSO to replace the back-propagation learning algorithm in ANN in the past several years [35-40]. They showed that PSO is a promising method to train ANNs. It is faster and gets better results in most cases. It also avoids some of the problems GAs encounter.

B. Interaction with other computational method

Besides neural network training, PSO has been combined with various techniques to solve different problems such as fuzzy system [41, 42], self-organizing maps [43], support vector machine [44], and hidden Markov model training [45] etc.

C. Parameter Optimization

As an optimization method, PSO has been applied to various parameter estimation and optimization problems. For example, Abido *et al.* [46, 47] applied PSO to optimize parameter settings of power system stabilizers as well as the optimal power flow problem [48].

D. Feature Selection

Agrafiotis [49] adapted PSO to the problem of feature selection by viewing the location vectors of the particles as probabilities and employing roulette wheel selection to construct candidate subsets. Testing results showed that the method compares favorably with simulated annealing. The authors also claimed that PSO does not converge as reliably to the same minimum. One possible reason is the selection of learning factors. Both factors are set to 1 in the experiments, which might be too small according to previous discussion.

VI. SUMMARY

As an emerging technology, particle swarm has gained a lot of attention in recent years. The first IEEE Swarm Intelligence Symposium was held in Indianapolis, USA in April, 2003. Authors from various countries presented over 30 papers in related areas. The response was encouraging. Nevertheless, there are still many unsolved issues in particle swarm including but not limited to:

- Convergence analysis. It is still not clear that why and how PSO converges. It is also important to the theoretical research of swarm intelligence and chaos systems.
- Discrete/binary PSO. Most of the research projects reported in the literature deal with continuous variables. Limited research showed that PSO has some difficulties dealing with discrete variables.
- Combination of various PSO techniques to deal with complex problems.
- Agent-based distributed computation. PSO can be viewed as a distributed agent model and many agent computing characteristics are still uncovered.
- Interaction with biological intelligence. Rooted in artificial life, PSO is successful even only simple rules in biological worlds are adopted. What if more complicated rules are included?

BIBLIOGRAPHY

[1] Eberhart, R. C. and Kennedy, J. A new optimizer using particle swarm theory. Proceedings of the Sixth International Symposium on Micromachine and Human Science, Nagoya, Japan. pp. 39-43, 1995

[2] Kennedy, J. and Eberhart, R. C. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks, Piscataway, NJ. pp. 1942-1948, 1995

[3] Shi, Y. and Eberhart, R. C. A modified particle swarm optimizer. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1998), Piscataway, NJ. pp. 69-73, 1998

[4] El-Gallad, A. I., El-Hawary, M. E., and Sallam, A. A., "Swarming of intelligent particles for solving the nonlinear constrained optimization problem," *Engineering Intelligent Systems for Electrical Engineering and Communications*, vol. 9, no. 3, pp. 155-163, Sept.2001.

[5] Hu, X. and Eberhart, R. C. Solving constrained nonlinear optimization problems with particle swarm optimization. Proceedings of the Sixth World Multiconference on Systemics, Cybernetics and Informatics 2002 (SCI 2002), Orlando, USA. 2002

[6] Coello Coello, C. A. and Lechuga, M. S. MOPSO: a proposal for multiple objective particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. 2002

[7] Hu, X. and Eberhart, R. C. Multiobjective optimization using dynamic neighborhood particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. pp. 1677-1681, 2002

[8] Carlisle, A. and Dozier, G. Tracking changing extrema with adaptive particle swarm optimizer. Proceedings of the 5th Biannual World Automation Congress, 2002, Orlando, Florida, USA. pp. 265-270, 2002

[9] Hu, X. and Eberhart, R. C. Adaptive particle swarm optimization: detection and response to dynamic systems. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. pp. 1666-1670, 2002

[10] Kennedy, J. and Mendes, R. Neighborhood topologies in fully-informed and best-of-neighborhood particle swarms. Proceedings of the 2003 IEEE International Workshop on Soft Computing in Industrial Applications 2003 (SMCia/03), pp. 45-50, 2003

[11] Kennedy, J. and Mendes, R. Population structure and particle swarm performance. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2002), Honolulu, Hawaii USA. 2002

[12] Brits, R., Engelbrecht, A. P., and van den Bergh, F. Solving systems of unconstrained equations using particle swarm optimization. Proceedings of IEEE Conference on Systems, Man, and Cybernetics 2002 (SMC 2002), Hammamet, Tunisa. pp. 102-107, 2002

[13] Peram, T., Veeramachaneni, K., and Mohan, C. K. Fitness-distance-ratio based particle swarm optimization. Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana, USA. pp. 174-181, 2003

[14] Eberhart, R. C. and Shi, Y. Particle swarm optimization: developments, applications and resources. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001), Seoul, Korea. 2001

[15] Clerc, M. The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 1999), pp. 1951-1957, 1999

[16] Kennedy, J. Minds and cultures: particle swarm implications. Socially Intelligent Agents: Papers from the 1997 AAAI Fall Symposium, Menlo Park, CA. pp. 67-72, 1997

[17] Zheng, Y., Ma, L., Zhang, L., and Qian, J. On the convergence analysis and parameter selection in particle swarm optimization. Proceedings of International Conference on Machine Learning and Cybernetics 2003, pp. 1802-1807, 2003

[18] Kennedy, J. and Eberhart, R. C. A discrete binary version of the particle swarm algorithm. Proceedings of the World Multiconference on Systemics,Cybernetics and Informatics 1997, Piscataway, NJ. pp. 4104-4109, 1997

[19] Chang, R. F. and Lu, C. N. Feeder reconfiguration for load factor improvement. Proceedings of the IEEE Power Engineering Society Transmission and Distribution Conference, pp. 980-984, 2002

[20] Mohan, C. K. and Al-kazemi, B. Discrete particle swarm optimization. Proceedings of the Workshop on Particle Swarm Optimization 2001, Indianapolis, IN. 2001

[21] Hu, X., Eberhart, R. C., and Shi, Y. Swarm intelligence for permutation optimization: a case study on n-queens problem. Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana, USA. pp. 243-246, 2003

[22] Fieldsend, J. E. and Singh, S. A multi-objective algorithm based upon particle swarm optimisation, an efficient data structure and turbulence. Proceedings of the 2002 U.K. Workshop on Computational Intelligence, Birmingham, UK. pp. 37-44, 2002

[23] Hu, X., Eberhart, R. C., and Shi, Y. Particle swarm with extended memory for multiobjective optimization. Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana, USA. pp. 193-197, 2003

[24] Li, X. A non-dominated sorting particle swarm optimizer for multiobjective optimization. Lecture Notes in Computer Science (LNCS) No. 2723: Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO 2003), Chicago, IL, USA. pp. 37-48, 2003

[25] Lu, H. Dynamic population strategy assisted particle swarm optimization in multiobjective evolutionary algorithm design. 2003. IEEE Neural Network Society. IEEE NNS Student Research Grants 2002 - Final Reports.

[26] Moore, J. and Chapman, R. Application of particle swarm to multiobjective optimization. 1999. Department of Computer Science and Software Engineering, Auburn University.

[27] Mostaghim, S. and Teich, J. r. Strategies for finding local guides in multi-objective particle swarm optimization (MOPSO). Proceedings of the IEEE Swarm Intelligence Symposium 2003 (SIS 2003), Indianapolis, Indiana, USA. pp. 26-33, 2003

[28] Parsopoulos, K. E. and Vrahatis, M. N. Particle swarm optimization method in multiobjective problems. Proceedings of the ACM Symposium on Applied Computing 2002 (SAC 2002), pp. 603-607, 2002

[29] Ray, T. and Liew, K. M., "A swarm metaphor for multiobjective design optimization," *Engineering Optimization*, vol. 34, no. 2, pp. 141-153, 2002.

[30] Coello Coello, C. A., Toscano Pulido, G., and Salazar Lechuga, M. An extension of particle swarm optimization that can handle multiple objectives. Workshop on Multiple Objective Metaheuristics, Paris, France. 2002

[31] Parsopoulos, K. E. and Vrahatis, M. N. Particle swarm optimization method for constrained optimization problems. Proceedings of the Euro-International Symposium on Computational Intelligence 2002, 2002

[32] Ray, T. and Liew, K. M. A swarm with an effective information sharing mechanism for unconstrained and constrained single objective optimization problem. Proceedings of IEEE Congress on Evolutionary Computation (CEC 2001), Seoul, Korea. pp. 75-80, 2001

[33] Eberhart, R. C. and Shi, Y. Tracking and optimizing dynamic systems with particle swarms. Proceedings of the IEEE Congress on Evolutionary Computation (CEC 2001), Seoul, Korea. pp. 94-97, 2001

[34] Blackwell, T. M. Swarms in dynamic environments. Lecture Notes in Computer Science (LNCS) No. 2723: Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO 2003), Chicago, IL, USA. pp. 1-12, 2003

[35] Eberhart, R. C. and Shi, Y. Evolving artificial neural networks. Proceedings of International Conference on Neural Networks and Brain, 1998, Beijing, P. R. China. pp. PL5-PL13, 1998

[36] Engelbrecht, A. P. and Ismail, A., "Training product unit neural networks," *Stability and Control: Theory and Applications*, vol. 2, no. 1-2, pp. 59-74, 1999.

[37] van den Bergh, F. and Engelbrecht, A. P., "Cooperative learning in neural networks using particle swarm optimizers," *South African Computer Journal*, vol. 26 pp. 84-90, 2000.

[38] Zhang, C., Shao, H., and Li, Y. Particle swarm optimisation for evolving artificial neural network. Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics 2000, pp. 2487-2490, 2000

[39] Lu, W. Z., Fan, H.-Y., and Lo, S. M., "Application of evolutionary neural network method in predicting pollutant levels in downtown area of Hong Kong," *Neurocomputing*, vol. 51 pp. 387-400, 2003.

[40] Settles, M., Rodebaugh, B., and Soule, T. Comparison of genetic algorithm and particle swarm optimizer when evolving a recurrent neural network. Lecture Notes in Computer Science (LNCS) No. 2723: Proceedings of the Genetic and Evolutionary Computation Conference 2003 (GECCO 2003), Chicago, IL, USA. pp. 151-152, 2003

[41] He, Z., Wei, C., Yang, L., Gao, X., Yao, S., Eberhart, R. C., and Shi, Y. Extracting rules from fuzzy neural network by particle swarm optimization. Proceedings of IEEE Congress on Evolutionary Computation (CEC 1998), Anchorage, Alaska, USA. 1998

[42] Shi, Y. and Eberhart, R. C. Particle swarm optimization with fuzzy adaptive inerita weight. Proceedings of the Workshop on Particle Swarm Optimization 2001, Indianapolis, IN. 2001

[43] Xiao, X., Dow, E. R., Eberhart, R. C., Ben Miled, Z., and Oppelt, R. J. Gene clustering using self-organizing maps and particle swarm optimization. Proceedings of Second IEEE International Workshop on High Performance Computational Biology, Nice, France. 2003

[44] Paquet, U. and Engelbrecht, A. P. Training support vector machines with particle swarms. Proceedings of the International Joint Conference on Neural Networks, 2003 (IJCNN 2003), pp. 1598, 2003

[45] Rasmussen, T. K. and Krink, T., "Improved Hidden Markov Model training for multiple sequence alignment by a particle swarm optimization--evolutionary algorithm hybrid," *Biosystems*, vol. 72, no. 1-2, pp. 5-17, Nov.2003.

[46] Abido, M. A. Particle swarm optimization for multimachine power system stabilizer design. Power Engineering Society Summer Meeting, pp. 1346-2001, 2001

[47] Abido, M. A., "Optimal design of power system stabilizers using particle swarm optimization," *IEEE Transactions on Energy Conversion*, vol. 17, no. 3, pp. 406-413, Sept.2002.

[48] Abido, M. A., "Optimal power flow using particle swarm optimization," *International Journal of Electrical Power and Energy Systems*, vol. 24, no. 7, pp. 563-571, 2002.

[49] Agrafiotis, D. K. and Cedeno, W., "Feature selection for structure-activity correlation using binary particle swarms," *Journal of Medicinal Chemistry*, vol. 45, no. 5, pp. 1098-1107, Feb.2002.