# Analysis on Diversity of Particle Swarm Optimization

Nayan Ranjan Samal
Department of Electrical Engineering, Orissa Engineering College, Bhubaneswar, India

***Abstract:*** ***Particle Swarm Optimization dynamics is one of the most important members of the swarm intelligence family. This algorithm based on population based search, which employs a number of agents, called particles, the position and velocity of which are adapted over time with an ultimate objective to search the optima in a given search landscape. The paper provides an extensive review of the diversity of the population in corporate with the Particle Swarm Optimization algorithm. It outlines the hybridization of genetic algorithm with basic algorithm, called GA-PSO, then the self adapting property introduced evolutionary hybrid EPSO. Next, the paper examines the strategy for adjustment of particle and size, the adaptive PSO. Finally it emphasises on the tuning of parameters, the tribes.***

***Keywords: Swarm intelligence, diversity, hybridization, self adapting, and adjustment.***

## 1 Introduction

Researchers are keen to hybridize PSO with other evolutionary technique. For instance, selection, crossover and mutation operations in GA have been introduced into the PSO by some researchers. By the selection operation, the particles with the best performance are copied into the next generation to keep the best performing particles. Crossover operation is used in PSO to exchange information between a pair of individual particles to have the ability to jump to the new search areas like other evolutionary algorithms. The mutation operation is borrowed from evolutionary algorithm with the idea that PSO will increase its ability to escape from local optima.

PSO has attracted a good number of researchers from diverse domains of science, engineering and humanities, particularly for its following characteristics:

***Simplicity:*** PSO is simple, and can be easily implemented in any high level programming language. Main body of a PSO program comprises a few lines of code. This particular feature of PSO attracts researchers from different disciplines with minimum programming skill.

***Good Performance:*** PSO outperforms binary coded GA, and has comparable performance with real coded GA. It is found to give to good accuracy in determining optima for uni-modal, multi-modal, and functions with very rough surface. PSO is better than the differential evolutionary (DE) algorithms on occasions. DE employs a greedy search as the parameter vectors in current iteration is either better or of similar quality in DE. In PSO, particles move away from its historical best iteration, and thus it is not a greedy algorithm.

***Few Control Parameters:*** PSO has few control parameters; the inertial co-efficient, the local and the global acceleration constants. Extensive research has already been undertaken to study the performance of PSO on the selection of parameters. There is however scope of further research on this issue particularly, selection of control parameters to have better exploration in the first phase, and faster convergence in the exploitation phase.

***Low Space Complexity:*** Because of its low space complexity, PSO is preferred to its competitive counterparts, such as CMA-ES [1]. The low space complexity is a useful feature for PSO for complex optimization of high dimensional search problems.

## 2. Hybrid PSO

There are so many evolutionary computation techniques that have already been tested and incorporated withthe particle swarm optimization algorithm. Many authors have considered incorporating selection, mutation and crossover, as well as the differential evolution (DE), into the PSO algorithm. The main goals are to increase the diversity of the population by:

1) either preventing the particles to move too close to each other and collide [2], [3] or
2) to self-adapt parameters such as the constriction factor, acceleration constants [4], or inertia weight [5] and
3) to obtain improved performance due to single constituent method alone.

So that, the hybrid versions of PSO have been developed and tested in different applications. The most common ones include hybridization of genetic algorithm and PSO (GA-PSO), evolutionary PSO (EPSO) and differential evolution PSO (DEPSO and C-PSO) which are discussed in this section.

***Hybrid of Genetic Algorithm and PSO (GA-PSO):*** The GA-PSO enhance the advantages of both swarm intelligence and a natural selection mechanism as in GA, to increase the population of fitter agents, while decreasing the population of poorly performing agents in each iteration. Therefore, it possible to successively changes the current search arena by considering

$p-best$ and $g-best$ values, in the same time it takes a leap from one arena to another by the selection mechanism, resulting in acceleration in the convergence speed of the whole algorithm.

Basically the GA-PSO synergism employs a major aspect of the classical GA approach, which is the capability of "breeding". However, some authors have also analyzed the inclusion of mutation or a simple replacement of the best fitted value, as a means of improvement to the standard PSO formulation [6], [7].

The application of a reproduction system that modifies both the position and velocity vectors of randomly selected particles in order to further improve the potential of PSO to reach an optimum is considered by El-Dib *et al.* [6] as follows:

$$child_1(x) = p.parent_1(x) + (1-p).parent_2(x)$$

$$child_1(v) = (parent_1(v) + parent_2(v)).\frac{parent_1(v)}{parent_1(v) + parent_2(v)}$$

$$child_2(x) = p.parent_2(x) + (1-p).parent_1(x)$$

$$child_2(v) = (parent_1(v) + parent_2(v)).\frac{parent_2(v)}{parent_1(v) + parent_2(v)} \quad (1)$$

where, $p$ is a uniformly distributed random number between $[0, 1]$ $parent_{1,2}(x)$ represents the position vectors of randomly chosen particles, $parent_{1,2}(v)$ are the corresponding velocity vectors of each parent and $child_{1,2}(x)$, $child_{1,2}(v)$ are the offspring of the breeding process.

According to the fitness value Naka *et al.*, [7] suggest, replacing agent positions with low fitness values, with those with high fitness, according to a selection, keeping the $p-best$ information of the replaced agent so that a dependence on the past high evaluation position is accomplished (HPSO).

***Hybrid of Evolutionary Programming and PSO (EPSO):*** Evolutionary PSO incorporates a selection strategy to the original PSO algorithm, as well as self-adapting properties for its parameters. Angeline [8] considered a tournament selection method for evolutionary programming (EP). This approach employs the same update formulas as in the original PSO algorithm; however, the particles are selected as follows.

- The fitness value of each particle is compared with $k$ other particles and a score point for each particle with a worse fitness value is recorded. The population is sorted based on this score.
- The current positions and velocities of the best half of the swarm replace the positions and velocities of the worst half.
- The individual best of each particle of the swarm (best and worst half) remains unmodified. Therefore, at each iteration step, half of the individuals are moved to positions of the search space that are closer to the optimal solution than their previous positions while keeping their personal best points.

The difference between this method and the original particle swarm is that the exploitative search mechanism is emphasized. This should help the optimum to be found more consistently than the original particle swarm. In addition to the selection mechanism, Miranda and Fonseca [4], [9], [10] introduced a self-adaptation capability to the swarm by modifying the concept of a particle to include, not only the objective parameters, but also a set of strategy parameters (inertia and acceleration constants, simply called weights).

The general EPSO scheme can be summarized as follows [4], [9], [10].

- Replication: Each particle is replicated $r$ times.
- Mutation: Each particle has its weights mutated.
- Reproduction: Each mutated particle generates an offspring according to the particle movement rule.
- Evaluation: Each offspring has a fitness value.
- Selection: Stochastic tournament is carried out in order to select the best particle, which survives to the next generation.

The particle movement is defined as

$$\vec{v}_i(t+1) = \omega_{i1}^*.\vec{v}_i(t) + \omega_{i2}^*.rand_1(0,1).(\vec{p}_i - \vec{x}_i(t)) + \omega_{i3}^*.rand_2(0,1).(\vec{p}_g^* - \vec{x}_i(t))$$

$$\vec{x}_i(t+1) = \vec{x}_i(t) + \vec{v}_i(t+1) \quad (2)$$

where

$$\omega_{ik}^* = \omega_{ik} + \tau.rand \quad (3)$$

and $rand$ is a random number with normal distribution (0, 1).

The global best is also mutated by

$$\vec{p}_g^{*} = \vec{p}_g + \tau^{'}.rand \quad (4)$$

where $\tau$ and $\tau^{'}$ are learning parameters that can be either fixed or dynamically changing as strategy parameters.

***Hybrid of Differential Evolution and PSO (DEPSO and C-PSO):*** A differential evolution operator has been proposed to improve the performance of the PSO algorithm in two different ways:

1) it can be applied to the particle's best position to eliminate the particles falling into local minima (DEPSO) [11], [12], [13] or

2) it can be used to find the optimal parameters (inertia and acceleration constants) for the canonical PSO (composite PSO) [14].

***The DEPSO:*** The DEPSO method proposed by Zang and Xie [11] alternates the original PSO algorithm and the Differential Evolution operator, i.e., (1) and (2) are performed at the odd iterations and (4) at the even iterations. The DE mutation operator is defined over the particle's best positions $\vec{p}_i$ with a trial point $T_i = \vec{p}_i$ which for the $d^{th}$ dimension is derived as

$$If\,(rand < CR \;or\; d = k)\;\text{then}\; T_{id} = \vec{p}_{gd} + \delta_{2d} \quad (5)$$

where $k$ is a random integer value within $[1, n]$ which ensures the mutation in at least one dimension, $CR$ is a crossover constant $(CR \le 1)$ and $\delta_2$ is the case of $N = 2$ for the general difference vector

$$\delta_N = \frac{1}{N}\sum_{1}^{N} \Delta \quad (6)$$

where $\Delta$ is the difference between two elements randomly chosen in the *pbest* set.

If the fitness value of $T_i$ is better than the one for $\vec{p}_i$, then $T_i$ will replace $\vec{p}_i$. After the DE operator is applied to all the particles individual best values, the *gbest* value is chosen among the *pbest* .

***The Composite PSO (C-PSO):*** In most of the previously presented algorithms, the selection of the PSO parameters is made basically by trial and error. The selection of PSO parameters by some other algorithms such as GA, EP, or DE may procedure more efficient result. Composite PSO algorithm is a method that employs DE for parameter selection of PSO. The C-PSO algorithm employing DE algorithm is explained below [14]:

- ***Step 1:*** Initialize $i$ to zero and set the maximum number of iterations as $I$ . Generate initial position of particles $(\vec{x}_i)$, initial velocity $(\vec{v}_i)$ and the initial PSO parameters $[X_i = (\omega, \alpha^l, \alpha^g)]$ randomly. The size of $\vec{x}$, $\vec{v}$ and $X$ is equal to $N_p$, the size of the population, and $i$ is the current iteration number.
- ***Step 2:*** For each $X_i$, calculate $\vec{v}_i(t)$ and $\vec{x}_i(t)$ using (1) and (2).
  Calculate the fitness function value for each particle.
- ***Step 3:*** For each parameter vector (here $X_i$) of DE, we select three companion vectors randomly from a given domain of $(\omega, \alpha^l, \alpha^g)$ and perform mutation, followed by recombination and selection using classical DE $|rand|_1$ algorithm and thus obtain $X_i^*$ ( the best individual) corresponding to $X_i$. Replace $X_i$ by $X^*$ and repeat step 2 and 3 until a terminal number of iterations of DE (selected *a priori*) is reached.
- ***Step 4:*** The process continues from Step 2: until the stopping criterion is met.

**3. The Adaptive PSO**

Many authors have suggested and adjustments the parameters of the PSO algorithm in different ways such as: adding a random component to the inertia weight [15], [16], [17], applying fuzzy logic [18], [19], using a secondary PSO to find the optimal parameters of a primary PSO [20], Q- learning [21], or adaptive critics [22], [23].

The prominent considered by Zhang *et al.* [24] for the adjustment of the number of particles and the neighborhood size. Authors modified the PSO algorithm by adding an improvement index for the particles of the swarm as follows:

$$\delta(\vec{x}_i) = \frac{f(\vec{x}_i(t_0)) - f(\vec{x}_i(t))}{f(\vec{x}_i(t_0))} \quad (7)$$

where $f(\vec{x}_i(t))$ is the fitness function value for particle $i$ at iteration $t$ .

An improvement threshold has to be defined as the limit for the minimum acceptable improvement. Then, the adaptive strategies are as follows [24].

- ***Adjust the swarm size:*** If the particle has enough improvement but it is the worst particle in its neighborhood, then remove the particle. On the other hand, if the particle does not have enough improvement but it is the best particle in its neighborhood, then generate a new particle.
- ***Adjust the inertia weight:*** The more a particle improves itself, the smaller the area this particle needs to explore. In contrast, if the particle has a deficient improvement then it is desirable to increase its search space. The adjustment of the inertia weight is done accordingly.
- ***Adjust the neighborhood size:*** If the particle is the best in its neighborhood but it has not improved itself enough, then the particle needs more information and the size of the neighborhood has to be increased. If the particle has improved itself satisfactorily, then it does not need to ask many neighbors and its neighborhood size can be reduced.

A species-based PSO (SPSO) has been proposed by Li [25] where, the swarm population is divided into species of subpopulations based on their similarity. Each species is grouped around a dominating particle called the species seed. At each iteration step, the species seeds are identified and adopted as neighborhood bests for the species groups. Over successive iterations, the adaptation of the species allows the algorithm to find multiple local optima, from which the global optimum can be identified.

**4. The TRIBES**

Like most optimization heuristics, PSO suffers from the drawback of the selection of its parameter values. The performance of a PSO algorithm is directly related to the tuning of such parameters. Usually, such tuning is a lengthy, time consuming and delicate process. A new adaptive PSO algorithm called TRIBES avoids manual tuning, and adapts rules which automatically change the particles' behaviors as well as the topology of the swarm. In TRIBES, according to the swarm behavior, the topology is changed and according to the performances of the particles the strategies of displacement are chosen. In 2002 [26], van den Bergh carried out a systematic study of the influence of the parameters on the behavior of PSO. A modification of the size of each particle neighborhood dynamically with time was proposed by Suganthan (1999) [27]. Ye et al. in 2002 [28] suggested a method, where it searches for inactive particles and replaces them by new ones, which are more capable of exploring new areas of the search space. By the use of an improvement threshold, Zhang et al. (2003) [24] proposed a modification of either the swarm size, or the constriction factor, or the neighborhood size. Then in 2004 [29], Yasuda and Iwasaki proposed an algorithm where parameters are defined according to the velocity information of the swarm. Finally, Clerc (2006) [30] proposed an adaptive algorithm, called TRIBES. In TRIBES, only the adaptation rules can be modified or added by the user, however, the parameters change according to the swarm behavior. TRIBES is a stochastic algorithm, and hence the results given by the algorithm are probabilistic in nature. In the same time TRIBES, like all other optimization algorithms, cannot guarantee finding an optimal solution to every given problem within a pre-defined time. Moreover, TRIBES is aimed at designing a competitive algorithm to the best known algorithms, so far, so as to reduce the algorithm setup time by eliminating the need of parameter tuning.

**Evolution of the tribes:**

To modify the swarm's topology, it is required to set up rules where, quality qualifiers are defined for each particle and each tribe. The evolution of the tribes is described as follows:

- ***Quality of a tribe:*** We judge the quality of a tribe, and often use adjectives: good or bad to qualify them. A particle is termed *good* if it has just improved its best performance, and termed *neutral* if it has not. A tribe is declared *bad* if none of its particles has improved its best location during the last iteration. If at least one of the particles of the tribe has improved its best location during the last iteration, the tribe is declared either *good* or *bad* with a probability of 0.5. The *best* and the *worst* particles are defined within the tribe framework.
- ***Removal of a particle:*** A good tribe will be able to eliminate one of its particles and only the worst of them. In the case of a mono particle tribe, the removal is only made if one of the informants has a better performance. In the case of a mono particle tribe, since the removal of the particle leads to the removal of the whole tribe, they are placed on the best informant of the particle to be removed.
- ***Generation of a particle:*** A bad tribe generates at least one new particle, while keeping contact with it. In fact, the number of particles generated by a single *bad* tribe is defined below, which has been empirically determined by Clerc (2005) [30]. In this version, two particles are generated, one which could be anywhere in the search space called *free* particle and the other in a much more restricted field called *confined* particle.

$$NB_{particles} = Max\left(2, \left[\frac{9.5 + 0.124.(D-1)}{tribeNb}\right]\right),$$

where $D$ is the dimension of the search space and $tribeNb$is the number of tribes in the swarm.

- ***Free Particle:*** It is generated randomly, according to a uniform distribution either in the whole or on a side or on a vertex of the search space for diversifying the population with the equation given below: In the whole search space: $X_{generated-j} = U(x_{\min j}, x_{\max j}),\ j = \{1, ....., D\},$

On a side of the search space: $X_{generated-j} = \begin{cases} U(x_{\min j}, x_{\max j}) & j \in I \subset \{1,.....,D\}, \\ x_{boundj} & j \in J \subset \{1,.....,D\}, \end{cases}$

On a vertex of the search space: $X_{generated-j} = x_{bound\ j},\ j = \{1,....., D\}$,

where $U(x_{\min j}, x_{\max j})$ is a real number chosen from the uniform distribution in the interval $[x_{\min j}, x_{\max j}]$ and $x_{bound\ j}$ is either $x_{\min j}$ or $x_{\max j}$ with a probability of 0.5. $I$ and $J$ are two sub-spaces of $\{1,....., D\}$ and are randomly generated for each new particle defined within aside of the search space.

- ***Confined particle:*** Let $\vec{X}_{best}$ is the best particle of the generating tribe and $\vec{I}_{best}$ be the best informer of $\vec{X}_{best}$. Also, let $\vec{p}_{x_{best}}$ and $\vec{p}_{l_{best}}$ denote the best locations of $\vec{X}_{best}$ and $\vec{I}_{best}$. Then the new particle will be generated in the $D$-sphere with center $\vec{p}_{l_{best}}$ and radius $\left\|\vec{p}_{x_{best}} - \vec{p}_{l_{best}}\right\|$ according to the following:

$$\vec{X}_{generated} = alea_{sphere}\left(\vec{p}_{l_{best}}, \left\|\vec{p}_{x_{best}} - \vec{p}_{l_{best}}\right\|\right),$$

where $alea_{sphere}\left(\vec{p}_{l_{best}}, \left\|\vec{p}_{x_{best}} - \vec{p}_{l_{best}}\right\|\right)$ is a point chosen randomly with uniform distribution in a hyper-sphere of center $\vec{p}_{l_{best}}$ and radius $\left\|\vec{p}_{x_{best}} - \vec{p}_{l_{best}}\right\|$.

- ***Frequency of the adaptation:*** In these structural adaptations, more time is required to transform the information between the particles. Theoretically, the time required between two adaptations must be equal to the diameter of the information graph. However, the computation time consuming. Thus, if NL is the number of information links at the time of the last adaptation, the next adaptation will occur after NL/2 iterations. NL is estimated as:

$$NL = \sum_{n=1}^{tribeNB} \exp loreNB[n]^2 + tribeNB.(tribeNB - 1),$$

where $tribeNB$ is the number of tribes in the swarm, and $\exp lorerNB[n]$ is the number of particles of the tribe n. Each particle of tribe n is linked to all other particles of the tribe and to itself. So, for tribe n, the number of communication information links within a tribe called *intra-tribe* is $\exp lorerNB[n]^2$. Each best particle of a tribe called shaman is linked to all other shamans; so called inter-tribes communication information links is $tribeNB - 1$.

.The algorithm for the structural adaptations is given below:

*test = 0*
***For*** *i = 1 to tribeNb*
  ***If*** $tribe[i].status = bad$
    ***Generate*** $NB_{particle} = Max\left(2, \left\lceil \frac{9.5 + 0.124.(D-1)}{tribeNB} \right\rceil\right) particles$
    *test = 1*
  ***End If***
  ***If*** $tribe[i].size \neq 1$
    ***Remove*** *the worst particle of* $tribe[i]$
    ***If*** $tribe[i].size \neq 1$
      ***Redirect*** *informers of the removed particle to the best informer of the removed particle*
    ***else***
      $tribeNb = tribeNB - 1$
      ***Redirect*** *informers to the best external informers*
    ***End If***
  ***End If***
***End for***
***If*** *test = 1*
  $tribeNb = tribeNB + 1$
  ***Aggregate*** *all the generated particles to the new tribe*
  ***Link*** *the shaman of the new tribe to the other shamans*
***End If***
***Compute*** *NL*

***Evolution of swarm:*** In the beginning, there is only one particle, representing a single tribe. After the first iteration, this particle does not improve its location and hence another particle will be generated, forming a second tribe. After the second iteration, if neither of the two particles improves its situation, two tribes generate two particles each and the same process is repeated to generate new particles. Thus, the swarm's exploratory capability will grow. Adaptations will be more and more spaced in time. Then, the swarm has more and more chances to find a good solution between two consecutive adaptations. However, once a promising area is found, each tribe will gradually remove its worst particles. Ideally, when convergence is confirmed, each tribe will be reduced to a single particle.

***The strategies of displacement:*** Each particle adopts a strategy of displacement according to the recent past of the particle. This will enable a particle with a good behavior to have a greater scope of exploration. A special strategy that can be compared to a local search is defined for very good particles. Accordingly, the algorithm will choose to call the most appropriate displacement strategy.

Serra et al., in 1997 [31] first introduced a method called pivot strategy. Let us denote the best location of the particle by $\vec{p}$, the best position of the informers of the particle by $\vec{g}$ and the objective function by $f$. Then, the movement is carried out as follows:

$$\vec{X} = c_1.alea_{sphere}(H_p) + c_2.alea_{sphere}(H_g)$$

with $c_1 = \frac{\tilde{f}(\vec{p})}{\tilde{f}(\vec{p}) + \tilde{f}(\vec{g})}$, $c_2 = \frac{\tilde{f}(\vec{g})}{\tilde{f}(\vec{p}) + \tilde{f}(\vec{g})}$, $alea_{sphere}(H_p)$ being a point chosen randomly with uniform distribution in the hyper-sphere with centre $\vec{p}$ and radius $\|\vec{p} - \vec{g}\|$, and $alea_{sphere}(H_g)$ being a point chosen randomly with uniform distribution in the hyper-sphere with centre $\vec{g}$ and radius $\|\vec{p} - \vec{g}\|$. The function $\tilde{f}$ is defined by:

$$\tilde{f}(\vec{x}) = \begin{cases} \left| f(\vec{x}) - f(\vec{x}_{opt}) \right| & \text{if } \vec{x}_{opt} \text{ is known,} \\ \left| f(\vec{x}) \right| & \text{if } \vec{x}_{opt} \text{ is not known.} \end{cases}$$

The algorithm that summarizes TRIBES is as following:

***Initialize*** *a population of particles with random positions*

*For each individual* $i$, $\vec{p}_i$ *is* ***initialized*** *to* $\vec{X}_i$

***Evaluate*** *the objective function for each particle and compute* $\vec{g}$

***Repeat***

  ***Determine*** *statuses of all particles*

  ***Choose*** *the displacement strategies*

  ***Update*** *the positions of the particles*

  ***Evaluate*** *the objective function for each particle*

  ***Compute*** *new* $\vec{p}_i$ *and* $\vec{g}$

  ***If*** $n < NL$

    ***Determine*** *the qualities of the tribes*

    ***Adapt*** *swarm*

    ***Compute*** *NL*

  ***End If***

***Until*** *the stopping criterion is met*

In the above algorithm $\vec{g}$ is the best location reached by the swarm and the $\vec{p}'s$ are the best locations for each particle. $NL$ is the number of information links during the adaptation of the last swarm and $n$ is the number of iterations since the last adaptation of the swarm.

TRIBES are an adaptive algorithm with no parameter to tune. It saves the trouble of defining the parameters, thus saving time. The tests and comparisons made on TRIBES give different indications. First, TRIBES gives competitive results for both multimodal and unimodal functions compared to Standard PSO 2006, even in high dimensions. However, in the case of multimodal functions it gives worse than other algorithms. Secondly, TRIBES is faster than other algorithms in the beginning and quickly reaches "acceptable" solutions. However, once these solutions are reached, TRIBES does not succeed in improving them further. Thus, it is concluded that TRIBES is an adaptive PSO inspired optimization technique, efficient in quickly finding a good region of the landscape, but less efficient for local refinement.

## Conclusion:

This paper surveys the research and development of hybrid PSO. During the last decade, it has gathered considerable interest from the natural computing research community and has been seen to offer rapid and effective optimization of complex

multidimensional search spaces, with adaptations to multiple objectives and constrained optimization. Hybridization, in particular, is a useful way of combining PSO with some related field of evolutionary computation for rapid optimization and improving the performance of PSO, such as stagnation. Considerable research has been invested in adapting and refining PSO algorithms to cope with multiple objectives optimization in the presence of constraints, both of which are important steps to facilitating engineering design optimization. The appreciable levels of success in these areas in recent years remain as the active research topics Challenges remain, in areas such as dynamic environments, avoiding stagnation, handling constraints and multiple objectives. Like evolutionary algorithms, PSO has become an important tool for optimization and other complex problem solving. The next decade will no doubt see further refinement of the approach and integration with other techniques, as well as applications moving out of the research laboratory and into industry and commerce. Further understanding of the relative strengths of PSO and other techniques, and the challenges in deploying a PSO based system are required. However, to the optimization toolbox PSO and its diversity certainly welcome as a better addition.

**References:**


[1] S. Das, P. N. Suganthan. "Differential Evolution: A Survey of the State-of-the-Art". IEEE Transaction on Evolutionary Copmutation, vol. pp, issue 99, pp. 1-28, Oct. 2010.

[2] T. Blackwell and P. Bentley, "Don't push me! Collision-avoiding swarms," in *Proc. IEEE Congr. Evol.Comput.*, May 2002, vol. 2, pp. 1691–1696.

[3] T. Krink, J. Vesterstrom, and J. Riget, "Particle swarm optimization with spatial particle extension," in *Proc. IEEE Congr. Evol.Comput.*, May 2002, vol. 2, pp. 1474–1479.

[4] V. Miranda and N. Fonseca, "New evolutionary particle swarm algorithm (EPSO) applied to voltage/VAR control," in *Proc. 14th Power Syst. Comput. Conf.*, Jun. 2002.

[5] M. Lovbjerg and T. Krink, "Extending particle swarms with self-organized criticality," in *Proc. IEEE Congr.Evol.Comput.*,May 2002, vol. 2, pp. 1588–1593.

[6] A. El-Dib, H. Youssef, M. El-Metwally, and Z. Osman, "Load flow solution using hybrid particle swarm optimization," in *Proc. Int. Conf. Elect., Electron., Comput. Eng.*, Sep. 2004, pp. 742–746.

[7] S. Naka, T. Genji, T.Yura, and Y. Fukuyama, "A hybrid particle swarm optimization for distribution state estimation," *IEEE Trans. Power Syst.* , pp. 60–68, Feb. 2003.

[8] P. Angeline, "Using selection to improve particle swarm optimization," in *Proc. IEEE Int. Conf. Evol. Comput.*, May 1998, pp. 84–89.

[9] V. Miranda and N. Fonseca, "EPSO best of two worlds meta-heuristic applied to power system problems," in *Proc. IEEE Congr. Evol.Comput.*, May 2002, vol. 2, pp. 1080–1085.

[10] V. Miranda and N. Fonseca, "EPSO evolutionary particle swarm optimization, a new algorithm with applications in power systems," in *IEEE/PES Transmission and Distribution Conf. Exhibition Asia Pacific*, Oct. 2002, vol. 2, pp. 745–750.

[11] W. Zhang and X. Xie, "DEPSO: Hybrid particle swarm with differential evolution operator," in *Proc. IEEE Int. Conf. Syst., Man, Cybern.*, Oct. 2003, vol. 4, pp. 3816–3821.

[12] H. Talbi and M. Batouche, "Hybrid particle swarm with differential evolution for multimodal image registration," in *Proc. IEEE Int. Conf. Ind. Technol.*, Dec. 2004, vol. 3, pp. 1567–1572.

[13] P. Moore and G.Venayagamoorthy, "Evolving combinational logic circuits using particle swarm, differential evolution and hybrid DEPSO," *Int. J. Neural Syst.*, vol. 16, no. 2, pp. 163– 77, 2006.

[14] S. Kannan, S. Slochanal, and N. Padhy, "Application of particle swarm optimization technique and its variants to generation expansion problem," *ELSERVIER Electric Power Syst. Res.*, vol. 70, no. 3, pp. 203–210, Aug. 2004.

[15] R. Eberhart and Y. Shi, "Particle swarm optimization: Developments, applications and resources," in *Proc. IEEE Congr. Evol.Comput.*, May 2001, vol. 1, pp. 81–86.

[16] S. Mohagheghi, Y. Del Valle, G. Venayagamoorthy, and R. Harley, "A comparison of PSO and backpropagation for training RBF neural networks for identification of a power system with STATCOM," in *Proc. IEEE Swarm Intell.Symp.*, Jun. 2005, pp. 381–384.

[17] Y. del Valle, S. Mohagheghi, G. Venayagamoorthy, and R. Harley, "Training MLP neural networks for identification of a small power system: Comparison of PSO and backpropagation," in *Proc. 6th Int. Conf. Power Syst. Operation and Planning*, May 2005, pp. 153–157.

[18] Y. Shi and R. Eberhart, "Fuzzy adaptive particle swarm optimization," in *Proc. IEEE Congr.Evol.Comput.*, May 2001, vol. 1, pp. 101–106.

[19] Y. Shi and R. Eberhart, "Particle swarm optimization with fuzzy adaptive inertia weight," in *Proc. Workshop on Particle Swarm Optimization, Purdue School of Engineering and Technology*, Indianapolis, IN, Apr. 2001.

[20] S. Doctor, G. Venayagamoorthy, and V. Gudise, "Optimal PSO for collective robotic search applications," in *Proc. IEEE Congr. Evol.Comput.*, Jun. 2004, vol. 2, pp. 1390–1395.

[21] M. Khajenejad, F. Afshinmanesh, A. Marandi, and B. N. Araabi, "Intelligent particle swarm optimization using Q-learning," in *Proc. IEEE Swarm Intell.Symp.*, May 2006, pp. 7–12.

[22] G. Venayagamoorthy, "Adaptive critics for dynamic particle swarm optimization," in *Proc. IEEE Int. Symp. Intell. Control*, Sep. 2004, pp. 380–384.

[23] S. Doctor and G. Venayagamoorthy, "Improving the performance of particle swarm ptimization using adaptive critics designs," in *Proc. IEEE Swarm Intell. Symp.*, Jun. 2005, pp. 393–396.

[24] W. Zhang, Y. Liu, and M. Clerc, "An adaptive PSO algorithm for reactive power optimization," in *Proc. 6th Int. Conf. Advances in Power System Control, Operation and Management*, Nov. 2003, pp. 302–307.

[25] X. Li, "Adaptively choosing neighborhood bests using species in a particle swarm optimizer for multimodal function optimization," in *Proc. Genetic Evol. Comput. Conf.*, Jun. 2004, pp. 105–116.

[26] Frans van den Bergh.*An analysis of particle swarm optimizers*.PhD thesis, University of Pretoria, South Africa, 2002.

[27] P. N. Suganthan,"Particle swarm optimizer with neighborhood operator". Proc. IEEE Int. Congr. Evolutionary Computation, vol. 3, 1999, pp. 1958–1962

[28] Ye, X. F., Zhang, W. J., & Yang, Z. L. (2002). Adaptive particle swarm optimization on individual level.In*Proceedings of the international conference on signal processing (ICSP)* (pp. 1215–1218). Piscataway:IEEE Press.

[29] Yasuda, K., & Iwasaki, N. (2004). Adaptive particle swarm optimization using velocity information ofswarm.In *Proceedings of the IEEE conference on system, man and cybernetics* (pp. 3475–3481). Piscataway:IEEE Press.

[30] Clerc, M. (2006). Particle swarm optimization.In International scientific and technical encyclopaedia. Hoboken:Wiley.

[31] Serra, P., Stanton, A. F., & Kais, S. (1997).Method for global optimization. *Physical Review, 55*, 1162–1165.