An Off-The-Shelf PSO

Anthony Carlisle
Department of Mathematical and Computer Sciences, Huntingdon College
antho@huntingdon.edu

Gerry Dozier
Department of Computer Science and Software Engineering, Auburn University
gvdozier@eng.auburn.edu

Abstract

What attributes and settings of the Particle Swarm Optimizer constants result in a good, off-the-shelf, PSO implementation? There are many parameters, both explicit and implicit, associated with the Particle Swarm Optimizer that may affect its performance. There are the social and cognitive learning rates and magnitudes, the population size, the neighborhood size (including global neighborhoods), synchronous or asynchronous updates, and various additional controls, such as inertia and constriction factors. For any given problem, the values and choices for some of these parameters may have significant impact on the efficiency and reliability of the PSO, and yet varying other parameters may have little or no effect. What set of values, then, constitutes a good, general purpose PSO? While some of these factors have been investigated in the literature, others have not. In this paper we use existing literature and a selection of benchmark problems to determine a set of starting values suitable for an "off the shelf" PSO.

Introduction

Since the introduction of the particle swarm optimizer by James Kennedy and Russ Eberhart in 1995 [8], numerous variations of the basic algorithm have been developed in the literature. Each researcher seems to have a favorite implementation - different population sizes, different neighborhood sizes, and so forth. In this paper we examine a variety of these choices with the goal of defining a canonical particle swarm optimizer, that is, an off-the-shelf algorithm to be used as a good starting point for applying PSO.

The original PSO formulae defined each particle as a potential solution to a problem in D-dimensional space, with particle i represented $X_i=(x_{i1},x_{i2},...,x_{iD})$. Each particle also maintains a memory of its previous best position, $P_i=(p_{i1},p_{i2},...,p_{iD})$, and a velocity along each dimension, represented as $V_i=(v_{i1},v_{i2},...,v_{iD})$. At each iteration, the P vector of the particle with the best fitness in the local neighborhood, designated *g*, and the P vector of the current particle are combined to adjust the velocity along each dimension, and that velocity is then used to compute a new position for the particle. The portion of the adjustment to the velocity influenced by the individual's previous best position (P) is considered the *cognition* component, and the portion influenced by the best in the neighborhood is the *social* component [3,5,8].

In Kennedy's early versions of the algorithm, these formulae are:

$$v_{id} = v_{id}+\varphi_1*\text{rand}()*(p_{id} - x_{id})+\varphi_2*\text{rand}()*(p_{gd} - x_{id})$$
$$x_{id} = x_{id}+v_{id}$$

Constants φ_1 and φ_2 determine the relative influence of the social and cognition components, and are often both set to the same value to give each component (the cognition and social learning rates) equal weight. Angeline, in [1], calls this the *learning rate*. A constant, *Vmax*, was used to arbitrarily limit the velocities of the particles and improve the resolution of the search.

In [9] Eberhart and Shi show that PSO searches wide areas effectively, but tends to lack local search precision. Their solution in that paper was to introduce ω, an inertia factor, that dynamically adjusted the velocity over time, gradually focusing the PSO into a local search:

$$v_{id} = \omega*v_{id}+\varphi_1*\text{rand}()*(p_{id} - x_{id})+\varphi_2*\text{rand}()*(p_{gd} - x_{id})$$

More recently, Maurice Clerc has introduced a *constriction factor* [2], *K*, that improves PSO's ability to constrain and control velocities. In [4], Shi and Eberhart found that *K*, combined with constraints on *Vmax*, significantly improved the PSO performance. *K* is computed as:

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$$

where $\varphi=\varphi_1+\varphi_2,\varphi>4$, and the PSO is then

$$v_{id} = K(v_{id}+\varphi_1*\text{rand}()*(p_{id} - x_{id})+\varphi_2*\text{rand}()*(p_{gd} - x_{id}))$$

To test the various parameter settings, we start with the PSO settings Shi and Eberhart used in [4]: 30 particles, φ_1 and φ_2 both set to 2.05, *Vmax* set equal to *Xmax,* and incorporating Clerc's constriction factor. We assume, in absence of evidence otherwise, that the neighborhood is *global*, and particles are updated synchronously (That is,

gbest is determined between iterations). We also make use of the their suite of functions (also used by Kennedy in [7]): the Sphere function (De Jong's F1), the Rosenbrock function, the generalized Rastrigrin function, the generalized Griewank function, and Schaffer's F6 function. Figure 1 shows the formula and parameters used for each function.

Function	Equation	Parameters
Sphere (De Jong F1)	$f(x) = \sum_{i=1}^{n} x_i^2$	Dimensions: 30 Xmax: 100 Acceptable error: <0.01
Rosenbrock	$f(x) = \sum_{i=1}^{n} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2)$	Dimensions: 30 Xmax: 30 Acceptable error: <100
Rastrigrin	$f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	Dimensions: 30 Xmax: 5.12 Acceptable error: <100
Griewank	$f(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	Dimensions: 30 Xmax: 600 Acceptable error: <0.1
Schaffer F6	$f(x) = 0.5 - \frac{\left(\sin\sqrt{x^2 + y^2}\right)^2 - 0.5}{\left(1.0 + 0.001(x^2 + y^2)\right)^2}$	Dimensions: 2 Xmax: 100 Acceptable error: <0.00001

Figure 1. Functions in the test suite

In every experiment, each function was run for 20 repetitions, with an upper limit of 100,000 iterations. For each set of repetitions we captured the success rate, the average and median iterations for successful runs, and the average and median counts of calls to the evaluation function. The latter is a measure of the actual work performed by the algorithm within the set of runs.

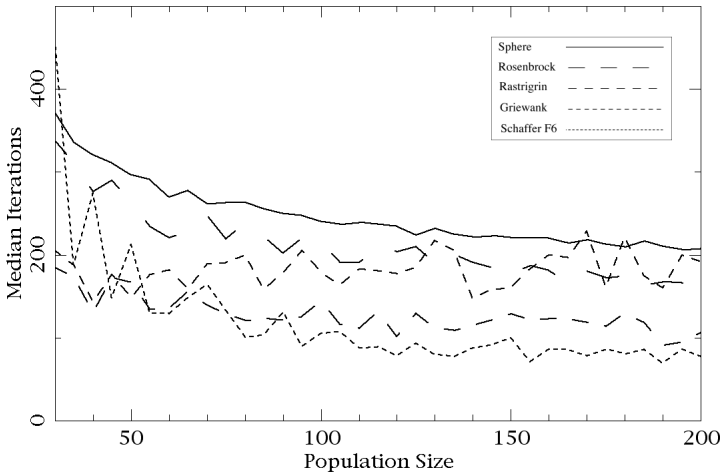
Experiment 1: Population Size

For the initial experiment, we elected to explore the effects of changing the population size. In [10] Shi and Eberhart reported that "PSO is not sensitive to the population size." We found this to be generally true in terms of performance, but not in terms of cost. We ran the PSO for all five functions using the best performance parameters from Shi and Eberhart's recent paper [4], that is, using Clerc's *constriction factor* K computed with φ set to 4.1 (social learning rate = cognitive learning rate = 2.05), and *Vmax* set equal to *Xmax*. We varied the population from 5 to 200 in steps of five and compared the success rates, median iterations, and median evaluations.

PSO was 100% successful in solving the Griewank function in all cases, even when the population was as small as five particles. PSO was 100% successful in solving the Rosenbrock function at populations of 10 or more, and reliably solved the Sphere function at populations above 25. PSO did not solve Schaffer's F6 with 100% consistently until the population reached 95, but was consistently at 85% or better starting at 30. The Rastrigrin function was never solved consistently 100%, but the success rate did stay at 75% or above beginning at a population of 30.

When we considered the median iterations required to find a solution (Figure 2), it is obvious that all the swarms had a reduction in the number of iterations required to solve all the functions (except the Griewank function) as the population increased. We would expect that, in general, more particles would search more space, and a solution would then be found sooner. However, as the population increases, each iteration represents a greater cost, as more particles call upon the evaluation function. In Figure 3 we graph the median evaluations (costs) to population and it becomes clear that the increase in cost more than offsets the reduction in iterations.

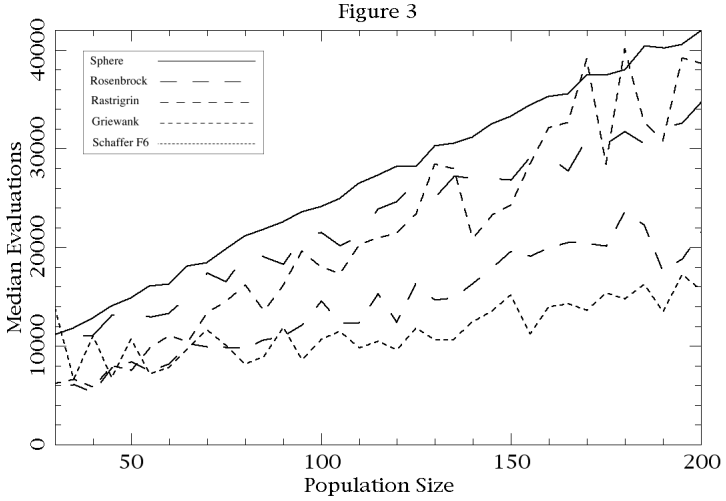
Figure 2



Conclusion: A population size of 30 appears to be a good choice. It is small enough to be efficient, yet large enough to produce reliable results.

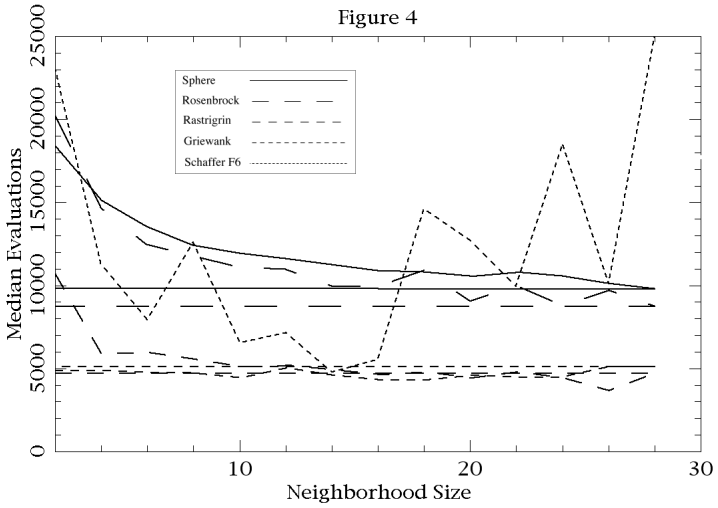
Experiment 2: Neighborhood Size

Figure 3



The neighborhood size for a swarm is the number of neighboring particles that influence a particular particle's movement. In [3], Eberhart and Kennedy conclude that a small local neighborhood is better at avoiding local minima, and that a global neighborhood converges faster. P. N. Suganthan, in [11], gradually increased the neighborhood size from small to global with encouraging, yet ultimately inconclusive, results. In this experiment we use a population of 30, as decided in our previous experiment, and vary the neighborhood size from 2 to global in steps of 2. We found that PSO was 100% successful in solving Sphere, Rosenbrock, and Griewank functions for all neighborhood sizes. The the success rate for solving both Rastrigrin and Schaffer F6 became inconsistent after the neighborhood size grew above 8, with solutions to Schaffer F6 showing more fluctuation than solutions to Rastrigrin. Both of these functions have many local minima, so their performance is in keeping with Eberhart and Kennedy's observation. For both functions the success rates stayed at or above 70%, with the sole exception of a 50% rate when solving Schaffer F6 at a neighborhood size of 28.

Figure 4



When we consider the median evaluation calls required for each success (Figure 4), with the exception of the wildly fluctuating success rates for solving Schaffer F6, increasing the neighborhood size does no harm, and significantly improves some performance (solving functions Sphere, Rosenbrock, and Rastrigrin).

Conclusion: The Global neighborhood appears to be a better general choice, as it seems to require less work for the same results.

Experiment 3: Synch or Async Updates

In the usual implementation of the PSO algorithm, particles can be thought of as being in a circular linked list, and a particle's local neighborhood is a sliding window stretching some distance to each side of the particle. It seems natural that the local best (*lbest*) is computed asynchronously, that is, as each particle is about to be moved, the best neighbor is determined and that influence is applied to the particle's motion. Conversely, it also seems natural that the evaluation of the global best (*gbest*) is done synchronously, between iterations, so that all particles are moved in parallel, then the best particle in the population is selected as the *gbest*, and the next iteration is run.

There is a difference in the two methods beyond the obvious. In the synchronous update, all particles have moved in parallel before the best selection is made, but in the asynchronous update, the neighbors on one side of the particle to be adjusted have already been updated, whereas the neighbors on the other side have not. The question, then, is, "Does synchronization matter?"

To address this question, we ran PSO at various neighborhood sizes and applied updates both synchronously and asynchronously. Interestingly, synchronous updates were almost always more costly than asynchronous updates.

Figure 5 shows the results of this experiment. We compared synchronously updating and asynchronously updating the particles of the swarm using three different neighborhood sizes: 2, 14, and global. We found that in almost every instance, asynchronous updating found solutions faster than synchronous updating. In the few examples where synchronous updates performed better, the advantage was very small.

Conclusion: Asynchronous updates are generally less costly.

	Median Iterations					
	Neighborhood of 2		Neighborhood of 14		Global Neighborhood	
	Async	Sync	Async	Sync	Async	Sync
Sphere	622.5	628.0	376.5	401.5	331.5	368.5
Rosenbrock	685.0	645.0	365.0	367.5	288.0	327.0
Rastrigrin	312.0	314.5	162.5	162.0	155.0	206.5
Griewank	152.0	181.5	154.5	148.5	173.0	171.0
Schaffer F6	624.0	993.5	310.0	499.5	255.5	522.5

Figure 5 Asynchronous vs. Synchronous updates

Experiment 4: Cognitive/Social Ratio

In [6], Kennedy asserts that the sum of the values of the cognitive and social components of the PSO (φ_1 and φ_2) should be about 4.0, and common usage is to set them each 2.05 each. However, in an earlier work [5], Kennedy also looked at models where the two components had varying values, specifically, zero for the social component (cognition-only model), zero for the cognitive component (social-only model), and setting the two equal (full model). In that work, he found a performance advantage to the social-only model.

To see what effect changing the relationship between the cognitive and social components had, we varied the cognitive rate φ_1 from 0.0 to 4.1, and computed the matching social rate as $\varphi_2=4.1-\varphi_1$.

	Best Performance	Median Evaluations	Acceptable Range for φ_1
Sphere	2.90	7,679.0	2.05-3.00
Rosenbrock	2.80	7,318.0	1.60-3.10
Rastrigrin	2.50	4,213.5	2.05-3.50
Griewank	2.90	4,026.0	1.00-3.60
Schaffer F6	2.90	5,470.0	2.05-3.60

Figure 6 Cognitive Component Values

The results are rather interesting, as shown in Figure 6. The usual settings of ($\varphi_1=2.05$,$\varphi_2=2.05$) did not result in the best performance for the swarm on any of the functions, and the swarm performed better on the trig-based functions (Rastrigrin, Griewank, and Schaffer F6) as the social component diminished. This implies that the social component tends to lead to more local minima trapping. The "acceptable range" shown in Figure 6 is an informal observation of the range of for φ_1 and φ_2 where PSO consistently finds solutions within a few percentage points of the best solution found, as shown in the "median evaluations" column.

Conclusion: A reasonable compromise for the cognitive and social component values appear to be 2.8 and 1.3, respectively.

Experiment 5: Magnitude of φ

The previous setting of 4.1 for φ was based on common usage established before the introduction of Clerc's constriction factor. We wondered if that value was still valid when applying the constriction factor, so we ran the test suite varying φ from 1.0 to 6.0, but maintaining the same cognitive/social ratio as above (2.8:1.3). Essentially, the swarm performed best for all functions at the settings found in the experiments above (that is, $\varphi=4.1$) except Schaffer F6. Interestingly, in solving F6, the swarm fared best in the area around $\varphi=1.1$, with settings of $\varphi_1=0.82$ and $\varphi_2=0.38$, producing a median evaluation count of 931.5. The reasons for this extraordinary improvement appear to be related to the shape of the function, and are thus very problem specific.

Conclusion: The settings from experiment 4 ($\varphi=4.1$) are best in the general case.

Experiment 6: Is *Vmax* Necessary?

Clerc's constriction factor *K* should act as a damper to escalating velocities, presumably negating the need for *Vmax*. However, in [4] Shi and Eberhart found that reducing *Vmax* from a very large value to *Vmax=Xmax* significantly improved PSO performance. It appears the influence of *Vmax* is related to the enforcement of *Xmax*. The functions in the test suite do not actually require setting an *Xmax*, as all of the functions are centered around a global minima. If we choose not to enforce *Xmax* or *Vmax*, so that the particles are free to fly as far and as fast a possible, the results are not significantly different from enforcing *Xmax* and setting *Vmax=Xmax*. The first two

columns of Figure 7 reflect this similarity. However, some problems obviously do require that *Xmax* be enforced in order to disregard solutions outside a specified, possibly asymmetric, search space, so the question arises, "If reducing *Vmax* to *Xmax* improved PSO performance, what effect would further reductions have?"

	Median Iterations				
	No *Xmax* enforced	*Vmax*= *Xmax*	*Vmax*= *Xmax*/2	*Vmax*= *Xmax*/4	V_{id}=0.0 at *Xmax*
Sphere	226.0	226.5	210.0	195.5	231.5
Rosenbrock	199.5	207.5	187.0	152.5	205.0
Rastrigrin	131.0	122.0	94.5	69.5	124.0
Griewank	147.0	146.0	113.0	120.0	178.0
Schaffer F6	272.0	285.0	280.0	761.0	261.5

Figure 7 *Xmax* and *Vmax*

The third and forth columns of Figure 7 show the results of setting *Vmax*=*Xmax*/2 and *Vmax*=*Xmax*/4, respectively. Note that, in general, performance improves as *Vmax* shrinks. Obviously, there must be a lower limit to this reduction, as *Vmax* is the *step size* of the swarm, the maximum distance a particle can travel in an iteration. Reducing it by too much impedes the ability of the swarm to search.

The advantage to reducing *Vmax* is greatest on particles "pinned" to the *Xmax* wall. As *Vmax* shrinks, so does the pressure exerted by the particle's velocity, allowing the particle's memory of its best location and that of the neighborhood best to pull the particle back into the search region.

As an alternative to trying to fine tune *Vmax*, we considered "unpinning" a particle at *Xmax* by resetting its velocity to zero. In that way, the pull of the particle's best location memory and that of the neighborhood best immediately begins drawing the particle back into the search space. The right-most column in Figure 7 show the outcome of that adjustment. The results, while not as good as fine tuning *Vmax*, are quite acceptable, given that it allows the removal of yet another parameter.

Conclusion: For problems where *Xmax* must be enforced, when a particle reaches *Xmax*, set its velocity to zero. If *Xmax* is not enforced, *Vmax* need not be enforced, either.

Summary: The Off-The-Shelf PSO

For a general purpose, off-the-shelf PSO we recommend the following settings: a swarm of 30 particles using Clerc's constriction formula, setting φ_1=2.8 and φ_2=1.3, with a global neighborhood updated asynchronously, and, if *Xmax* is enforced, setting the velocity of particles at *Xmax* to zero. No single set of parameters is perfect for every problem, and even within this limited test suite, these settings produce, in some cases, noticeably less than optimal performance, but overall these settings provide a good starting point for tailoring the PSO to a particular problem.

As a final test, we compare four sets of parameters run against our test suite. The first, labeled **Set A** in Figure 8, is a traditional PSO configuration: 30 particles, neighborhood of 2, asynchronous updates,*Vmax* set small (0.01), and φ=(2.05,2.05). The second set, **Set B**, is the same, but with a global neighborhood and synchronous updates. As a third set, **Set C**, we use the settings with which we started this paper: 30 particles, global neighborhood with synchronous, Clerc's *constriction factor*, *Vmax*=*Xmax*, and φ=(2.05,2.05). Finally, for **Set D**, we use the settings we derived during these experiments: 30 particles, global neighborhood, asynchronous updates, Clerc's *constriction factor*, no *Vmax*, *Xmax* enforced with the velocity of particles at *Xmax* set to zero, and φ=(2.8,1.3). All runs were repeated 20 times, and all limited to 100,000 iterations.

The results, shown in Figure 8, clearly indicate the considerable improvement gained by implementing Clerc's *constriction factor*, and the consistent, but somewhat lesser, improvement of our recommended settings over previously used combinations.

	Set A		**Set B**		**Set C**		**Set D**	
	Success Rate	Median Iterations	Success Rate	Median Iterations	Success Rate	Median Iterations	Success Rate	Median Iterations
Sphere	100.0%	2,233.0	100.0%	737.0	100.0%	361.5	100.0%	226.5
Rosenbrock	100.0%	1,884.5	95.0%	1,200.0	100.0%	322.0	100.0%	199.5
Rastrigrin	100.0%	5,237.5	70.0%	3,514.0	70.0%	175.5	100.0%	131.0
Griewank	100.0%	224.5	100.0%	162.5	100.0%	170.0	100.0%	147.0
Schaffer F6	100.0%	1,023.0	95.0%	669.5	85.0%	404.0	85.0%	272.0

Figure 8 Comparison of different parameter settings

Our recommendation, then, for the canonical particle swarm optimizer is:

$$v_{id} = \begin{cases} K(v_{id}+\varphi_1*\text{rand}()*(p_{id} - x_{id}) +\varphi_2*\text{rand}()*(p_{gd} - x_{id})), & Xmin<x_{id}<Xmax \\ 0 \text{ otherwise} \end{cases}$$

$$x_{id} = \begin{cases} x_{id}+v_{id}, & \text{Xmin}<(x_{id}+v_{id})<\text{Xmax} \\ Xmax, & (x_{id}+v_{id})>Xmax \\ Xmin, & Xmin<(x_{id}+v_{id}) \end{cases}$$

where

$$K = \frac{2}{|2 - \varphi - \sqrt{\varphi^2 - 4\varphi}|}$$

$\varphi=\varphi_1+\varphi_2$, and $\varphi_1=2.8$, $\varphi_2=1.3$.

and a populations size of 30, global neighborhood, with updates applied asynchronously.

In this paper we have tried to distill the best general PSO parameter settings from the pioneering work by James Kennedy, Russ Eberhart, Yuhui Shi, Maurice Clerc, and the other researchers who are furthering the evolution of the Particle Swarm Optimizer, as well as add our own small contributions. Our goal was to define a simple, general purpose PSO swarm, to be used as the base swarm description, so that only exceptions to this swarm definition would need to be stated. Picking out the particulars of a researcher's implementation of PSO can be difficult, and often details are omitted. We hope that declaring the canonical swarm would simplify discerning the particulars. Swarm descriptions could read, "I used the canonical swarm, but with a population size of 50," or "We implemented the canonical swarm, but reduced φ_1 and φ_2 to 1.95 each." Whether this terminology will become common among PSO devotees remains to be seen.

References

[1] Angeline, P. (1998). Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference, The 7th Annual Conference on Evolutionary Programming, San Diego, USA.

[2] Clerc, M. (1999) The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. Proceedings, 1999 ICEC, Washington, DC, pp 1951-1957.

[3] Eberhart, R. and Kennedy, J. (1995). A New Optimizer Using Particle Swarm Theory, Proc. Sixth International Symposium on Micro Machine and Human Science (Nagoya, Japan), IEEE Service Center, Piscataway, NJ, 39-43.

[4] Eberhart, R.C., and Shi, Y. (2000), Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization, 2000 Congress on Evolutionary Computing, vol. 1, pp. 84-88.

[5] Kennedy, J. (1997), The Particle Swarm: Social Adaptation of Knowledge, IEEE International Conference on Evolutionary Computation (Indianapolis, Indiana), IEEE Service Center, Piscataway, NJ, 303-308.

[6] Kennedy, J. (1998). The Behavior of Particles, 7th Annual Conference on Evolutionary Programming, San Diego, USA.

[7] Kennedy, J. (2000), Stereotyping: Improving Particle Swarm Performance With Cluster Analysis. 2000 Congress on Evolutionary Computing, Vol. II, pp 1507-1512.

[8] Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization, IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, IV: 1942-1948.

[9] Shi, Y. H., Eberhart, R. C., (1998). Parameter Selection in Particle Swarm Optimization, The 7th Annual Conference on Evolutionary Programming, San Diego, USA.

[10] Shi, Y. and Eberhart, R.C. (1999), Empirical Study of Particle Swarm Optimization. 1999 Congress on Evolutionary Computing, Vol. III, pp 1945-1950.

[11] Sugarthan, P.N. (1999), Particle Swarm Optimiser with Neighbourhood Operator. 1999 Congress on Evolutionary Computing, Vol. III, pp 1958-1964.