Mean particle swarm optimisation for function optimisation

Kusum Deep* and Jagdish Chand Bansal

Department of Mathematics,
Indian Institute of Technology Roorkee,
Roorkee – 247667, India
E-mail: kusumfma@iitr.ernet.in
E-mail: jcbansal@gmail.com
*Corresponding author

Abstract: In this paper, a new particle swarm optimisation algorithm, called MeanPSO, is presented, based on a novel philosophy by modifying the velocity update equation. This is done by replacing two terms of original velocity update equation by two new terms based on the linear combination of pbest and gbest. Its performance is compared with the standard PSO (SPSO) by testing it on a set of 15 scalable and 15 nonscalable test problems. Based on the numerical and graphical analyses of results it is shown that the MeanPSO outperforms the SPSO, in terms of efficiency, reliability, accuracy and stability.

Keywords: particle swarm optimisation; PSO; MeanPSO; global optimisation; velocity update equation.

Reference to this paper should be made as follows: Deep, K. and Bansal, J.C. (2009) 'Mean particle swarm optimisation for function optimisation', *Int. J. Computational Intelligence Studies*, Vol. 1, No. 1, pp.72–92.

Biographical notes: Kusum Deep is Associate Professor in the Department of Mathematics, Indian Institute of Technology Roorkee, India. Her research interests are in numerical optimisation including evolutionary computations: genetic algorithms, memetic algorithms, particle swarm optimisation, etc. She has more than 50 research papers in journals/conferences. She holds a number of national and international awards.

Jagdish Chand Bansal, is currently a PhD student at the Indian Institute of Technology Roorkee, India. His research interests are in particle swarm optimisation and genetic algorithm.

1 Introduction

Mathematical models of many real life problems turn out to be nonlinear in nature, having local as well as global optimal solutions. Usually, it is more difficult to obtain global optimal solution(s), as compared to local optimal solutions of nonlinear optimisation problems, but in many cases it is advantageous and sometimes even necessary, to search for the global optimal solution(s).

In order to determine the global optimal solution of a nonlinear optimisation problem, usually two approaches are followed:

1 the deterministic approach

2 the probabilistic approach.

Deterministic methods extensively use analytical properties such as continuity, convexity, differentiability etc. of the objective function and the constraints to locate a neighbourhood of the global optimum. It is now established that deterministic methods are best suited for a restricted class of functions, namely convex functions, one dimensional rational or Lipschitz functions and polynomials etc. whose mathematical properties can be easily determined and utilised at specified points or in specified intervals. On the other hand, probabilistic approaches rely on computational power and try to search the global optima in a probabilistic sense. These methods utilise randomness in an efficient way to explore the search space over which the objective function is to be optimised. Contrary to general expectation, stochastic methods perform well in most of the realistic problems. Although probabilistic methods do not give an absolute guarantee of determining the global minima, these methods are sometimes preferred over deterministic methods, because they are applicable to a wider class of functions as they depend on function evaluations alone and do not assume any mathematical properties of the functions involved.

Particle swarm optimisation (PSO) is a population-based optimisation technique, which is an alternative tool to genetic algorithm (GAs) and other evolutionary algorithms (EAs) and gained lots of attention in recent years. PSO is a stochastic search technique with reduced memory requirement, computationally effective and easier to implement as compared to EAs. In 1995 Kennedy and Eberhart (1995) introduced the PSO as a new heuristic method. The idea is based on the simulation of the social behaviour of bird flock and fish schools. Initially PSO was designed for continuous optimisation problems, but later a wide variety of challenging engineering and science applications came into being.

The rest of the paper is organised as follows: In Section 2, the standard PSO (SPSO) is described. In Section 3, a literature review on PSO is carried out. The proposed MeanPSO is described in Section 4. In Section 5, the set of benchmark test problems is given. In Section 6, analyses of results of computational experiments are discussed and in the end conclusions are derived in Section 7.

2 The SPSO

The beauty of PSO lies in its simplicity and ease of applicability. The coordinates of each particle represent a possible solution associated with two vectors – the position vector and the velocity vector.

Consider the n-dimensional optimisation problem

$$\text{Minimise } f(x), \text{ where } f : R^n \to R$$

Corresponding to each feasible solution, the position vector is represented by $x_i = (x_{i1}, x_{i2}, x_{i3}, \ldots, x_{in})$ and the velocity vector is represented by $v_i = (v_{i1}, v_{i2}, v_{i3}, \ldots, v_{in})$. A swarm consists of a number of particles (feasible solutions) that proceed (fly) through the search space towards the optimal solution. Each particle

updates its position based on its own best exploration, overall best swarm exploration and its previous velocity vector according to the following equations:

$$v_i^{k+1} = v_i^k + c_1 r_1 (pbest_i^k - x_i^k) + c_2 r_2 (gbest^k - x_i^k) \quad (1)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (2)$$

where

c_1 and c_2 are two positive constants called acceleration coefficients.

r_1 and r_2 are random numbers, uniformly distributed in *[0, 1]*

$x_i = \left(x_{i1}, x_{i2}, ..., x_{in}\right)$ is the current position of the ith particle

$pbest_i = \left(x_{i1}^{pbest}, x_{i2}^{pbest}, ..., x_{in}^{pbest}\right)$ is the best position of the ith particle achieved based on its own experience

$gbest = \left(x_1^{gbest}, x_2^{gbest}, ..., x_n^{gbest}\right)$ is the position of the best particle based on the overall swarm's experience

k is the iteration counter

A constant, maximum velocity (*Vmax*) is used to arbitrarily limit the velocities of the particles and improve the resolution of the search. (Shi and Eberhart, 1998a, 1998b) introduced an inertia factor w, in order to overcome the problem of premature convergence of PSO. The resulting velocity update equation becomes:

$$v_i^{k+1} = w v_i^k + c_1 r_1 (pbest_i^k - x_i^k) + c_2 r_2 (gbest^k - x_i^k)$$

In this paper, the inertia weight version of PSO is regarded as the SPSO.

This paper proposes, a new PSO, called MeanPSO, in which the basic velocity update equation is modified. The comparison is made with SPSO using thirty benchmark test problems of varied difficulty levels. The first problem set, contains 15 problems which are scalable in nature whereas second problem set contains 15 problems which are non-scalable. The results of the proposed MeanPSO are shown for problem size up to 500.

3 Literature review on PSO

A host of variations of improved versions of PSO is available in literature. They can be categorised as follows:

The first category consists of modifications based on new coefficients in velocity and position update equations. Angeline (1998a) showed that the original PSO has poor local search capability. The concept of an inertia weight was developed to better control exploration and exploitation. The inclusion of an inertia weight in the particle swarm optimisation algorithm (MeanPSO) was first reported in the literature in 1998 (Shi and Eberhart, 1998a, 1998b). Eberhart and Shi (2000) indicates that the optimal strategy is to initially set w to 0.9 and reduce it linearly to 0.4, allowing initial exploration followed by

acceleration toward an improved global optimum. Clerc and Kennedy (2002) introduced a constriction factor, χ, which improves PSO's ability to constrain and control velocities. Eberhart and Shi (2000) found that χ, combined with constraints on *Vmax*, significantly improved the PSO performance. Although constriction factor version of PSO is faster in convergence than linearly decreasing inertia weight version of PSO, but may get stuck at the local optima in multi modal functions.

The second category involves information regarding social sharing. Kennedy and Mendes (2002) concluded that the van Neumann neighbourhood topology provides the most superior performance. In the variable neighbourhood methodology of Suganthan (1999), initially an individual particle constitutes the neighbourhood, but during the later generations all the particles are included in the neighbourhood. Mohais et al. (2004) proposed dynamically adjusted neighbourhoods in which directed structures were used for the topology of the initial population and during the subsequent generations edges of the structures were randomly migrated from one source to another. Liang et al. (2004) demonstrated the use of a new learning methodology to make particles have different learning exemplars for different dimensions. Van den Bergh and Engelbrecht (2004) showed how to split the decision vector into several sub vectors, each being allotted to their own swarms. Peram et al. (2003) utilised the additional information of the nearby higher fitness particle that was selected according to fitness-distance ratio (FDR) indicating the ratio of the fitness improvement over the respective distance. Baskar and Suganthan (2004) simulated the modified PSO and FDR-PSO concurrently with frequent message passing between them. Janson and Middendorf (2005) used dynamic hierarchy to define the structure of the neighbourhood. He et al. (2004) introduced passive congregation into the velocity update equation.

Hybridisation with other EAs falls within the third category. PSO was combined with the selection operator of GA (Angeline, 1998b). Lovbjerg et al. (2001) combined PSO with the concept of breeding and subpopulation. Poli et al. (2005) extended PSO by using genetic programming. Zhang and Xie (2003) merged differential evolution operator into PSO. Krink and Lovbjerg (2002) combined PSO, GAs and hill climbing. Esquivel and Coello Coello (2003), Higasbi and Iba (2003) and Stacey et al. (2003) attempted to employ mutation in PSO. Hendtlass (2003) provided a study where the memory of particles is extended, in analogy with the pheromone trails of ant colony optimisation.

Diversity increasing mechanism to prevent convergence to local minima is covered in category four. Silva et al. (2002) presented a predator-pray model to maintain population diversity. Zhang et al. (2002) re-initialise the velocities of all the particles at a predefined extension interval. Krink et al. (2002) suggested several collision strategies to avoid crowding of the swarm. Lovbjerg and Krink (2002) suggested self-organised criticality. Riget and Vesterstrem (2002) introduced the use of the attractive phase and repulsive phase alternately as per a diversity measure. Velocity update equation is modified in (Baskar and Suganthan, 2004; Kennedy and Mendes, 2003; Pasupuleti and Battiti, 2006; Liu et al., 2004; Wei et al., 2004). A good review of PSO can be found in (Hu et al., 2004; Banks et al., 2007, 2008).

4 The proposed MeanPSO

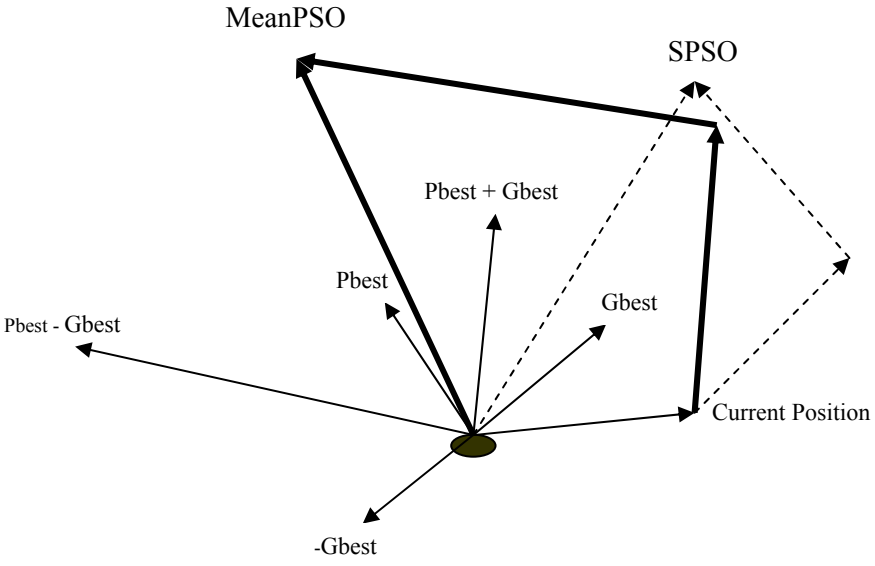
The motivation behind introducing MeanPSO is that in the velocity update equation instead of comparing the particle's current position with gbest and pbest, it is compared

with the linear combinations $\frac{pbest_i^k + gbest^k}{2}$ and $\frac{pbest_i^k - gbest^k}{2}$ of pbest and gbest. Thus, we introduce a new velocity update equation as follows:

$$v_i^{k+1} = \omega v_i^k + c_1 r_1 \left(\frac{pbest_i^k + gbest^k}{2} - x_i^k \right) + c_2 r_2 \left(\frac{pbest_i^k - gbest^k}{2} - x_i^k \right) \tag{3}$$

In the velocity update equation of this new PSO the first term represents the current velocity of the particle and can be thought of as a momentum term. The second term is proportional to the vector $\left(\frac{pbest_i^k + gbest^k}{2} - x_i^k \right)$ and is responsible for the attraction of particle's current position towards the mean of the positive direction of global best position (gbest) and positive direction of its own best position (pbest). The third term is proportional to the vector $\left(\frac{pbest_i^k - gbest^k}{2} - x_i^k \right)$ and is responsible for the attraction of particle's current position towards the mean of the positive direction of its own best position (pbest) and the negative direction of the global best position (–gbest). Clearly, MeanPSO seems to be suitable name for this modified PSO. The relative position of the new positions generated by SPSO and MeanPSO can be visualised in Figure 1.

Figure 1 Comparative movement of a particle in SPSO and MeanPSO



Notes: _ _ _ _ _ _ _ Movement of SPSO
Movement of MeanPSO

The pseudo code of MeanPSO is shown below:

Algorithm MeanPSO:

For t= 1 to the max. bound of the number on iterations,

For i=1 to the swarm size,

For j=1 to the problem dimensionality,

Apply the velocity update equation (3);

Update Position using equation (2);

End- for-j;

Compute fitness of updated position;

If needed, update historical information for pbest and gbest;

End-for-i;

Terminate if gbest meets problem requirements;

End-for-t;

End algorithm.

5 The test bed

Many times it is found that the evaluation of a proposed algorithm is evaluated only on a few benchmark problems. However, in this paper we consider a test bed of thirty benchmark problems with varying difficulty levels and problem size. The relative performance of SPSO and MeanPSO is evaluated on two kinds of problem sets. Problem Set 1 consists of 15 scalable problems, i.e., those problems in which the dimension of the problems can be increased/decreased at will. In general, the complexity of the problem increases as the problem size is increased. Problem Set 2 consists of those problems in which the problem size is fixed, but the problems have many local as well as global optima. The Problem Set 1 is shown in Table 1 and Problem Set 2 is shown in Table 2.

Table 1 Details of Problem Set 1

Serial no.	*Function name*	*Expression*	*Search space*	*Objective function value*
1	Ackley	$\text{Min } f(x) = -20\exp\left(-0.02\sqrt{n^{-1}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(n^{-1}\sum_{i=1}^{n}\cos\left(2\pi x_i\right)\right) + 20 + e$	$-30 \le x_i \le 30$	0
2	Cosine mixture	$\text{Min } f(x) = -0.1\sum_{i=1}^{n} cos(5\pi x_i) + \sum_{i=1}^{n} x_i^2$	$-1 \le x_i \le 1$	$-0.1 \times (n)$

Table 1 Details of Problem Set 1 (continued)

Serial no.	*Function name*	*Expression*	*Search space*	*Objective function value*
3	Exponential	$\text{Min } f(x) = -\exp\left(-0.5\sum_{i=1}^{n} x_i^2\right)$	$-1 \le x_i \le 1$	–1
4	Griewank	$\text{M inf}\left(x\right) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n} \cos\left(\frac{x_i}{\sqrt{i}}\right)$	$-600 \le x_i \le 600$	0
5	Rastrigin	$\text{Min } f(x) = 10n + \sum_{i=1}^{n}\left[x_i^2 - 10\cos\left(2\pi x_i\right)\right]$	$-5.12 \le x_i \le 5.12$	0
6	Function 6	$\text{Min } f(x) = \sum_{i=1}^{n-1}\left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	$-30 \le x_i \le 30$	0
7	Zakharov's	$\text{Min } f(x) = \sum_{i=1}^{n} x_i^2 + \left[\sum_{i=1}^{n}\left(\frac{i}{2}\right)x_i\right]^2 + \left[\sum_{i=1}^{n}\left(\frac{i}{2}\right)x_i\right]^4$	$-5.12 \le x_i \le 5.12$	0
8	Sphere	$\text{Min } f(x) = \sum_{i=1}^{n} x_i^2$	$-5.12 \le x_i \le 5.12$	0
9	Axis parallel hyper ellipsoid	$\text{Min } f(x) = \sum_{i=1}^{n} i x_i^2$	$-5.12 \le x_i \le 5.12$	0
10	Schwefel 3	$\text{Min } f(x) = \sum_{i=1}^{n}\left\|x_i\right\| + \prod_{i=1}^{n}\left\|x_i\right\|$	$-10 \le x_i \le 10$	0
11	Dejong noise	$\text{Min } f(x) = \sum_{i=1}^{n}\left(x_i^4 + rand\left(0,1\right)\right)$	$-10 \le x_i \le 10$	0
12	Schwefel 4	$\text{Min } f(x) = Max\left\{\left\|x_i\right\|, 1 \le i \le n\right\}$	$-100 \le x_i \le 100$	0
13	Cigar	$\text{Min } f(x) = x_i^2 + 100000\sum_{i=2}^{n} x_i^2$	$-10 \le x_i \le 10$	0

Table 1 Details of Problem Set 1 (continued)

Serial no.	*Function name*	*Expression*	*Search space*	*Objective function value*
14	Brown 3	$\text{Min} f(x) = \sum_{i=1}^{n-1} \left[\left(x_i^2\right)^{\left(x_{i+1}^2+1\right)} + \left(x_{i+1}^2\right)^{\left(x_i^2+1\right)} \right]$	$-1 \le x_i \le 4$	0
15	Function 15	$\text{Min} f(x) = \sum_{i=1}^{n} \left(0.2x_i^2 + 0.1x_i^2 \sin 2x_i\right)$	$-10 \le x_i \le 10$	0

Table 2 Details of Problem Set 2

Serial no.	*Function name*	*Expression*	*Search space*	*Objective function value*
1	Becker and Lago	$\text{Min} f(x) = \left(\lvert x_1\rvert - 5\right)^2 + \left(\lvert x_2\rvert - 5\right)^2$	$-10 \le x_1, x_2 \le 10$	0
2	Bohachevsky 1	$\text{Min} f(x) = x_1^2 + 2x_2^2 - 0.3\cos\left(3\pi x_1\right) - 0.4\cos\left(4\pi x_2\right) + 0.7$	$-50 \le x_1, x_2 \le 50$	0
3	Bohachevsky 2	$\text{Min} f(x) = x_1^2 + 2x_2^2 - 0.3\cos\left(3\pi x_1\right)\cos\left(4\pi x_2\right) + 0.3$	$-50 \le x_1, x_2 \le 50$	0
4	Branin	$\text{Min} f(x) = a(x_2 - bx_1^2 + cx_1 - d)^2 + g(1-h)\cos(x_1) + g$, $a = 1, b = \frac{5.1}{4\pi^2}$, $c = \frac{5}{\pi}$, $d = 6$, $g = 10$, $h = \frac{1}{8\pi}$	$-5 \le x_1 \le 100$ $\le x_2 \le 15$	$\frac{5}{4\pi}$
5	Eggcrate	$\text{Min} f(x) = x_1^2 + x_2^2 + 25\left(\sin^2 x_1 + \sin^2 x_2\right)$	$-2\pi \le x_1 \le 2\pi$	0
6	Helical Valley	$\text{Min} f(x) = 100\left[(x_2 - 10\theta)^2 + \left(\sqrt{(x_1^2 + x_2^2)} - 1\right)^2\right] + x_3^2$ $\text{Where} \quad \theta = \begin{cases} \frac{1}{2\pi}\tan^{-1}\frac{x_2}{x_1} & \text{if } x_1 \ge 0 \\ \frac{1}{2\pi}\tan^{-1}\frac{x_2}{x_1} + \frac{1}{2} & \text{if } x_2 < 0 \end{cases}$	$-10 \le x_1, x_2, x_3 \le 10$	0
7	Kowalik	$\text{Min} f(x) = \sum_{i=1}^{11}\left(a_i - \frac{x_1(1 + x_2 b_i)}{(1 + x_3 b_i + x_4 b_i^2)}\right)^2$	$0 \le x_i \le 0.42$	3.0748×10^{-4}

Table 2 Details of Problem Set 2 (continued)

Serial no.	*Function name*	*Expression*	*Search space*	*Objective function value*
8	Miele and Cantrell	$\text{Min} f(x) = (\exp(x_1) - x_2)^4 + 100(x_2 - x_3)^6 + (\tan(x_3 - x_4))^4 + x_1^8$	$-1 \le x_i \le 1$	0
9	Modified Rosenbrock	$\text{Min} f(x) = 100(x_2 - x_1^2)^2 + \left[6.4(x_2 - 0.5)^2 - x_1 - 0.6\right]^2$	$-5 \le x_1, x_2 \le 5$	0
10	Easom	$\text{Min} f(x) = -\cos(x_1)\cos(x_2) \exp\left(-(x_1 - \pi)^2 - (x_2 - \pi)^2\right)$	$-10 \le x_1, x_2 \le 10$	–1
11	Periodic	$\text{Min} f(x) = 1 + \sin^2 x_1 + \sin^2 x_2 - 0.1\exp(-x_1^2 - x_2^2)$	$-10 \le x_1, x_2 \le 10$	0.9
12	Powell's	$\text{Min} f(x) = (x_1 + 10x_1)^2 + 5(x_3 - x_4)^2 + (x_2 - 2x_3)^4 + 10(x_1 - x_4)^4$	$-10 \le x_i \le 10$	0
13	Camel back-3	$\text{Min} f(x) = 2x_1^2 + 1.05x_1^4 + \frac{1}{6}x_1^6 + x_1x_2 + x_2^2$	$-5 \le x_1, x_2 \le 5$	0
14	Camel back-6	$\text{Min} f(x) = 4x_1^2 + 2.1x_1^4 + \frac{1}{3}x_1^6 + x_1x_2 - 4x_2^2 + 4x_2^4$	$-5 \le x_1, x_2 \le 5$	–1.0316
15	Aluffi-Pentini's	$\text{Min} f(x) = 0.25x_1^4 - 0.5x_1^2 + 0.1x_1 + 0.5x_2^2$	$-10 \le x_1, x_2 \le 10$	–0.3523

6 Analyses of results

The SPSO and the MeanPSO are coded in C++ and implemented on Pentium IV 2.4 GHz machine with 512 MB RAM under WINXP platform. Thirty independent runs with different seed for the generation of random numbers are taken. However, the same seed is used for generating the initial swarm for SPSO and MeanPSO for the ith run, where i = 1, 2, . . . , 30. A run is said to be a successful run if the best objective function value found in that run lies within 1% accuracy of the best known objective function value of that problem. The maximum number of function evaluations is fixed to be 30,000. The swarm size is fixed to 20. The inertia weight w is 0.7 and the acceleration coefficients for SPSO and MeanPSO are set to be $c_1 = c_2 = 2$.

A number of criterions are used to evaluate the performance of SPSO with MeanPSO. The percentage of success is used to evaluate the reliability. The average number of

function evaluations of successful runs and the average computational time of the successful runs, are used to evaluate the cost. For problem Set 1, by fixing the problem size ten, this information is recorded in Table 3. The quality of the solution obtained is measured by the minimum, maximum, mean and standard deviation of the objective function values out of thirty runs. This is shown in Table 4. The corresponding information for Problem Set 2 is shown in Tables 5 and 6, respectively.

Table 3 Comparative results of SPSO and MeanPSO for Problem Set 1

Problem no.	*No. of successful runs out of 30 runs*		*Average function evaluations of successful runs*		*Average computational time in seconds of successful runs*	
	SPSO	*MeanPSO*	*SPSO*	*MeanPSO*	*SPSO*	*MeanPSO*
1	0	30	-	973	-	0.1697
2	30	30	3,788	478	0.5822	0.0776
3	30	30	1,515	268	0.2974	0.0469
4	0	30	-	1,889	-	0.3531
5	0	30	-	1,281	-	0.2530
6	5	30	17,126	1,003	1.3094	0.1609
7	27	30	11,020	685	0.9620	0.1359
8	30	30	3,989	519	0.5982	0.0838
9	30	30	3,928	557	0.5924	0.0874
10	26	30	7,907	931	0.8077	0.1327
11	30	30	3,551	397	0.5120	0.0687
12	11	30	26,418	1,058	1.5012	0.1734
13	30	30	10,437	1,217	0.8609	0.1815
14	12	30	3,241	492	0.5496	0.0806
15	30	30	4,378	483	0.6366	0.0776

From Table 3, it is observed that SPSO could not solve three problems at all, whereas MeanPSO solved all the problems with 100 % success rate. In five problems SPSO could not give 100% success rate, but MeanPSO solved them with 100% success rate. In the remaining seven problems both showed 100% success rate. In all the 15 problems, MeanPSO required lesser number of function calls and lesser computational time as compared to SPSO. In observing Table 4, it can be seen that MeanPSO gives a better quality of solutions as compared to SPSO. Thus, for the scalable problems MeanPSO outperforms SPSO with respect to efficiency, reliability, cost and robustness.

From Table 5, it is observed that SPSO could not solve three problems with 100 % success, whereas MeanPSO solved all the problems with 100 % success. In five problems SPSO requires lesser function evaluations, whereas in 10 problems MeanPSO requires lesser function evaluations. SPSO requires lesser computational time in five problems whereas MeanPSO requires lesser computational time in 10 problems. Also Table 6 shows that the quality of the solution obtained by MeanPSO is superior to SPSO. Thus for non scalable problems as well, MeanPSO outperforms SPSO.

Table 4 Comparative objective function value obtained in 30 runs by SPSO and MeanPSO for Problem Set 1

Problem no.	*Minimum function value*		*Maximum function value*		*Mean function value*		*Standard deviation*	
	SPSO	*MeanPSO*	*SPSO*	*MeanPSO*	*SPSO*	*MeanPSO*	*SPSO*	*MeanPSO*
1	-	0.00158	-	0.00993	-	0.00722	-	0.00212
2	–0.99514	–0.99857	–0.99001	–0.99020	–0.99159	–0.99336	0.00163	0.00236
3	–0.99486	–0.99000	–0.99486	–0.99680	–0.99158	–0.99296	0.00137	0.00172
4	-	0.00163	-	0.00988	-	0.00684	-	0.00214
5	-	0.00223	-	0.00977	-	0.00623	-	0.00210
6	0.00905	0.00287	0.00997	0.00996	0.00959	0.00731	0.00037	0.00216
7	0.00497	0.00162	0.00998	0.00989	0.00895	0.00621	0.00121	0.00225
8	0.00539	0.00195	0.00991	0.00983	0.00853	0.00639	0.00137	0.00226
9	0.00387	0.00123	0.00998	0.00995	0.00819	0.00647	0.00160	0.00215
10	0.00838	0.00389	0.00998	0.00987	0.00947	0.00806	0.00045	0.00142
11	0.00324	0.00079	0.00986	0.00983	0.00766	0.00482	0.00184	0.00276
12	0.00839	0.00493	0.00999	0.00998	0.00943	0.00853	0.00057	0.00139
13	0.00478	0.00270	0.00995	0.00967	0.00816	0.00620	0.00149	0.00195
14	0.00821	0.00150	0.00995	0.00986	0.00887	0.00670	0.00052	0.00235
15	0.002946	0.00248	0.00987	0.00990	0.00887	0.00725	0.00101	0.00209

Table 5 Comparative results of SPSO and MeanPSO for Problem Set 2

Problem no.	*No. of successful runs out of 30*		*Average function evaluations of successful runs*		*Average computational time in seconds of successful runs*	
	SPSO	*MeanPSO*	*SPSO*	*MeanPSO*	*SPSO*	*MeanPSO*
1	30	30	287	2,290	0.0302	0.2411
2	30	30	1,040	494	0.1067	0.0506
3	30	30	1,049	496	0.1124	0.0483
4	28	30	397	2,922	0.0424	0.3098
5	30	30	601	334	0.0656	0.0391
6	30	30	1,386	1,228	0.1818	0.1599
7	30	30	497	396	0.0568	0.0492
8	30	30	136	166	0.0251	0.0291
9	30	30	811	777	0.0854	0.0844
10	30	30	410	1,724	0.0457	0.1828
11	17	30	3,937	1,625	0.3942	0.0796
12	27	30	1,906	463	0.3021	0.0683
13	30	30	312	177	0.0349	0.0197
14	30	30	370	414	0.0390	0.0437
15	30	30	358	284	0.0405	0.0286

Table 6 Comparative objective function value obtained in 30 runs by SPSO and MeanPSO for Problem Set 2

Problem No.	*Minimum function value*		*Maximum function value*		*Mean function value*		*Mean standard deviation*	
	SPSO	*MeanPSO*	*SPSO*	*MeanPSO*	*SPSO*	*MeanPSO*	*SPSO*	*MeanPSO*
1	0.00090367	0.00068764	0.009313	0.009451	0.004839	0.004566	0.00265713	0.0026018
2	0.00065094	0.00019463	0.009640	0.009801	0.005787	0.005663	0.00258542	0.0030817
3	0.00000057	0.00022558	0.009839	0.009209	0.004799	0.004081	0.00301775	0.00271624
4	0.39823200	0.39803636	0.407859	0.407867	0.402836	0.402929	0.00315662	0.00234766
5	0.00019469	0.00004860	0.009617	0.009215	0.005621	0.004145	0.00262824	0.00308623
6	0.00006860	0.00032030	0.009918	0.009989	0.006015	0.007207	0.00305621	0.00230317
7	0.00042848	0.00039900	0.005611	0.008284	0.002128	0.001816	0.00139186	0.00163721
8	0.00000854	0.00043010	0.009296	0.009465	0.003445	0.00544	0.00295115	0.00314992
9	0.00060393	0.00037036	0.009986	0.009757	0.006626	0.006089	0.00321908	0.00288361
10	–1.0000000	–0.9995980	–0.99014	–0.990708	–0.995106	–0.99564	0.00307568	0.00244675
11	0.9001060	0.90074100	0.909147	0.909949	0.905785	0.904563	0.00250054	0.00292832
12	0.0005413	0.00212752	0.009829	0.009981	0.005897	0.0055	0.00259474	0.00273822
13	0.0004526	0.00034736	0.009806	0.009775	0.005452	0.005517	0.00283807	0.00257551
14	–1.031301	–1.0307300	–1.02189	–1.022140	–1.026985	–1.026755	0.0028258	0.00268092
15	–0.352106	–0.3522663	–0.34230	–0.342541	–0.347324	–0.3475712	0.0029373	0.00300151

To observe the consolidated effect of percentage of success, average number of function evaluations and average computational time on SPSO and MeanPSO a performance index (PI) is used as given in (Deep and Thakur, 2007).

The relative performance of an algorithm using this PI is calculated as:

$$PI = \frac{1}{N_p}\sum_{i=1}^{N_p}\left(k_1\alpha_1^i + k_2\alpha_2^i + k_3\alpha_3^i\right)$$

where

$$\alpha_1^i = \frac{Sr^i}{Tr^i},$$

$$\alpha_2^i = \begin{cases} \dfrac{Mf^i}{Af^i}, & if\ Sr^i > 0 \\ 0, & if\ Sr^i = 0 \end{cases} \text{ and}$$

$$\alpha_3^i = \begin{cases} \dfrac{Mt^i}{At^i}, & if\ Sr^i > 0 \\ 0, & if\ Sr^i = 0 \end{cases}$$

$i = 1, 2, ..., N_p$

Sr^i Number of successful runs of ith problem

Tr^i Total number of runs of ith problem

Mf^i Minimum of average number of function evaluations of successful runs used by all algorithms in obtaining the solution of ith problem

Af^i Average number of function evaluations of successful runs used by an algorithm in obtaining the solution of ith problem

Mt^i Minimum of average time used by all the algorithms in obtaining the solution of ith problem

At^i Average computational time used by an algorithm in obtaining the solution of ith problem

Np^i Total number of problems analysed.

k_1, k_2 and k_3 ($k_1 + k_2 + k_3 = 1$ and $0 \le k_1, k_2, k_3 \le 1$) are the weights assigned to percentage of success, average number of function evaluations and execution time of successful runs, respectively. From above definition it is clear that PI is a function of k_1, k_2 and k_3. Since, $k_1 + k_2 + k_3 = 1$, one of k_i, $i = 1, 2, 3$ could be eliminated to reduce the number of dependent variables from the expression of PI. But it is still difficult to analyse the behaviour of PI, because the surface plots of PI for SPSO and MeanPSO are overlapping and it is difficult to visualise them. So, we adopt the same methodology as given in (Deep and Thakur, 2007) i.e., equal weights are assigned to two terms at a time in the PI expression. This way PI becomes a function of one variable. The resultant cases are as follows

$$k_1 = W,\ k_2 = k_3 = \frac{1-W}{2},\ 0 \le W \le 1 \tag{1}$$

$$k_2 = W,\ k_1 = k_3 = \frac{1-W}{2},\ 0 \le W \le 1 \tag{2}$$

$$k_3 = W,\ k_1 = k_2 = \frac{1-W}{2},\ 0 \le W \le 1 \tag{3}$$

The PI is obtained for SPSO and MeanPSO for all the thirty problems and is shown in Figure 2. It is clear that the proposed MeanPSO outperforms the SPSO.

Figure 2a Performance index when $k_1 = W,\ k_2 = k_3 = \frac{1-W}{2}$

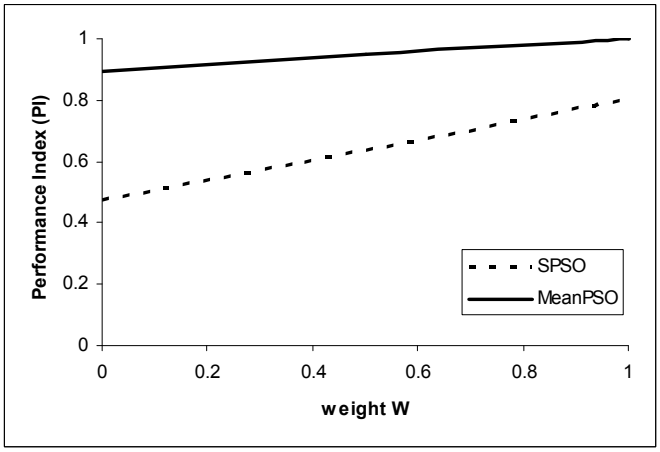


Figure 2b Performance index when $k_2 = W,\ k_1 = k_3 = \frac{1-W}{2}$

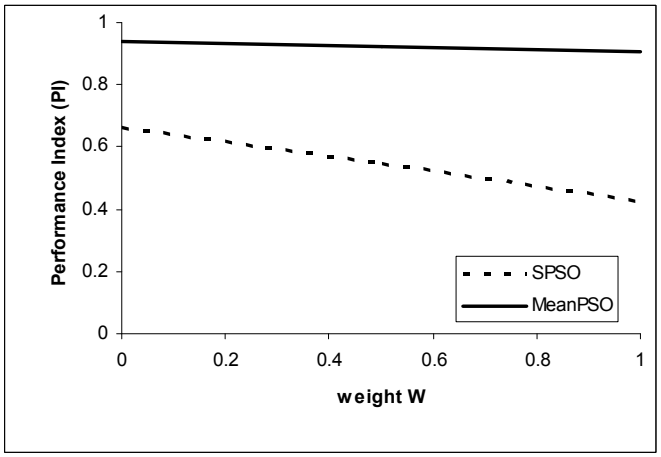
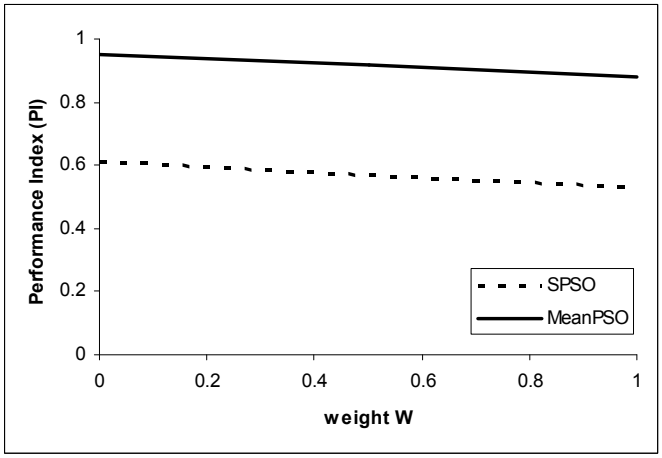


Figure 2c Performance index when $k_3 = W$, $k_1 = k_2 = \dfrac{1-W}{2}$



The next task is to investigate the stability of the MeanPSO by observing its effect when the size of the problems is increased. For this purpose, all 15 scalable problems are selected and their size is varied from 100 to 500. The average number of function evaluations when problem size is increased from 100 to 500 is shown in Figure 5 for all problems of Problem Set 1. The swarm size is 10, 20, 30, 40 and 50 for problem size 100, 200, 300, 400 and 500, respectively.

From Figure 5, it is observed that MeanPSO requires comparatively very less number of function evaluations even for large scale problems.

Figure 3 Convergence graphs of Problems Set 1

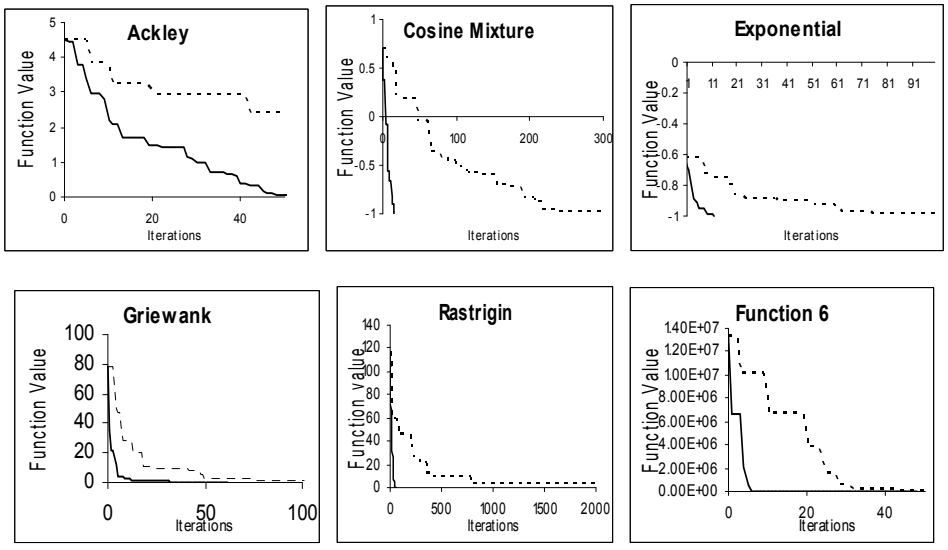
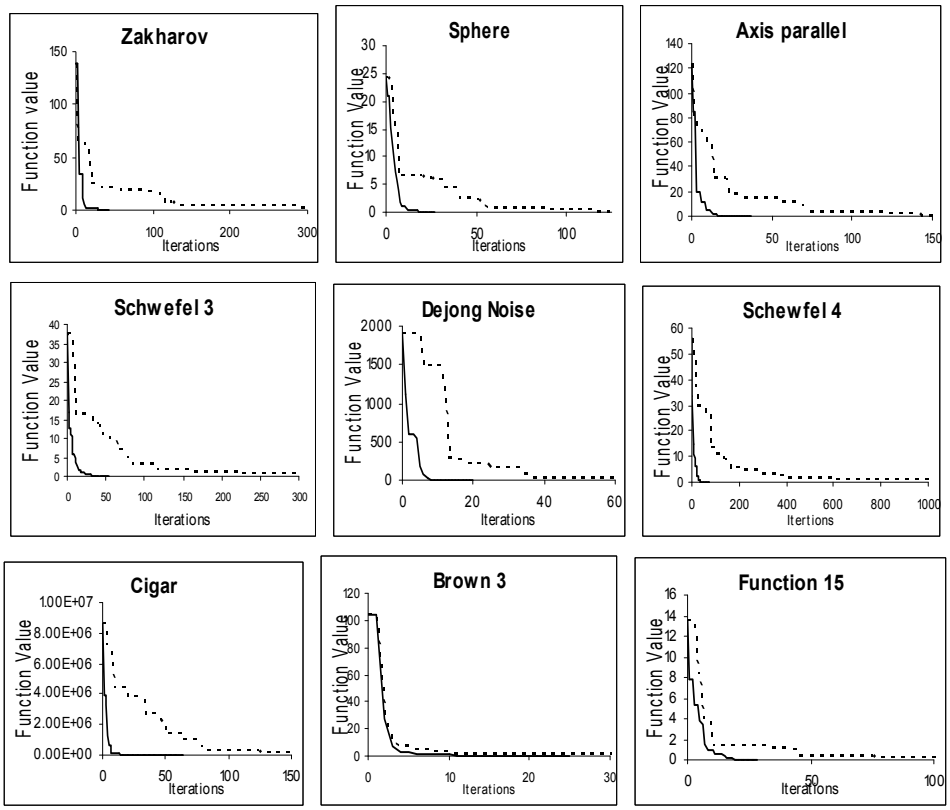
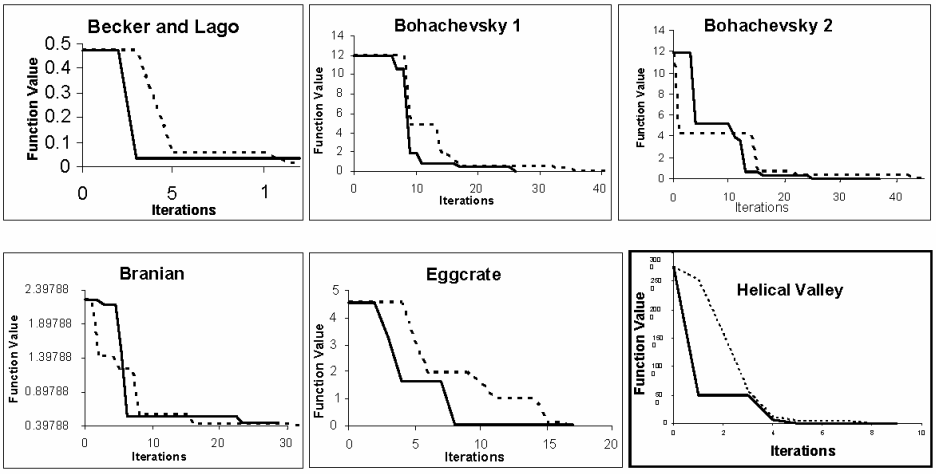


Figure 3 Convergence graphs of Problems Set 1 (continued)



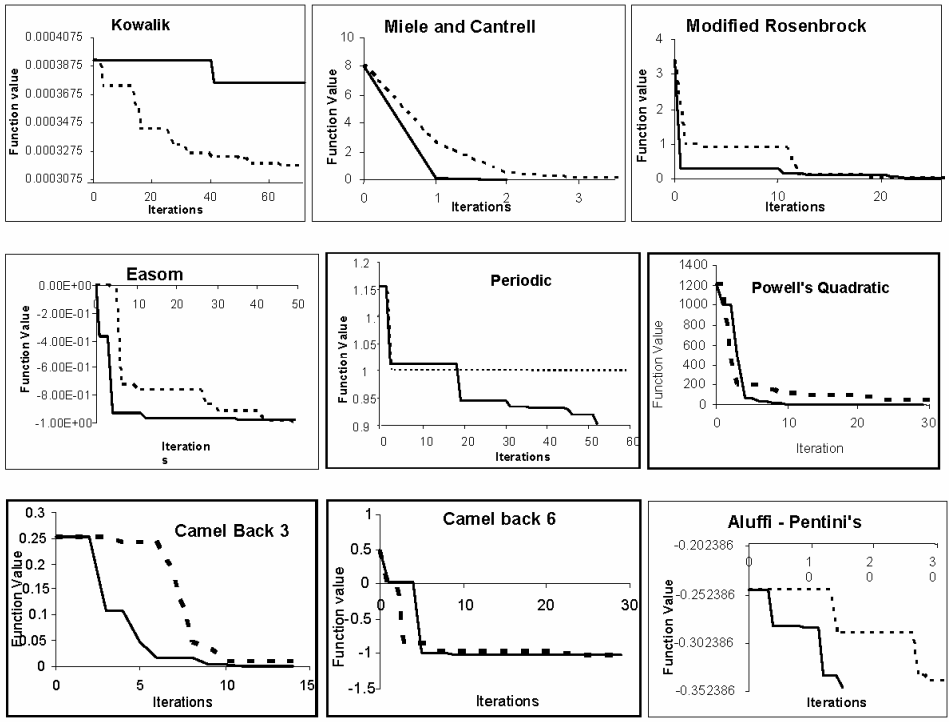
Notes: _____ SPSO _____ MeanPSO

Figure 4 Convergence graphs of Problems Set 2



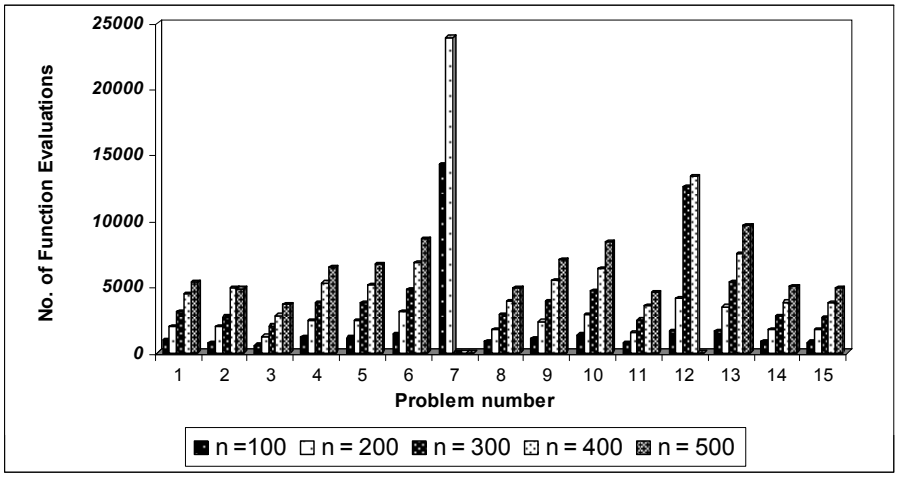
Notes: _____ SPSO _____ MeanPSO

Figure 4 Convergence graphs of Problems Set 2 (continued)



Notes: _____ SPSO _____ MeanPSO

Figure 5 Required number of function evaluations by MeanPSO when problem size is increased from 100 to 500 in problem Set 1



7 Conclusions

In this paper, a new PSO approach for function optimisation is presented. It is based on a basic change in the velocity update equation. Two terms in the original velocity update equation of PSO are replaced by the two new terms containing the linear combination of pbest and gbest. It is tested on 15 scalable problems and 15 nonscalable problems. It is shown that the new MeanPSO outperforms SPSO in terms of efficiency, accuracy, reliability and robustness. Particularly for large size problems MeanPSO outperforms SPSO. As a future work and as suggested by the anonymous reviewer one can also take the median of corresponding directions of all the particles in the current swarm in place of mean. In this paper the effect of change of parameters in MeanPSO is not explored. In a future study parameters fine tuning may be carried out for better performance. Also the application of MeanPSO to the real world problems would be interesting as a future research.

Acknowledgements

Authors acknowledge the editor and two anonymous reviewers for their valuable comments and suggestions. The second author gratefully acknowledges funding from University Grant Commission (UGC), India under Grant Number 6405-11-61.

References

Angeline, P.J. (1998a) 'Evolutionary optimization versus particle swarm optimization philosophy and performance differences', *Lecture Notes in Computer Science*, Vol. 1447, pp.601–610, Springer, Berlin.

Angeline, P.J. (1998b) 'Using selection to improve particle swarm optimization', *Proceedings of the IEEE Conference on Evolutionary Computations*, pp.84–89.

Banks, A., Vincent, J. and Anyakoha, C. (2007) 'A review of particle swarm optimization, Part I: Background and development', *Natural Computing: an International Journal*, Vol. 6, No. 4, pp.467–484.

Banks, A., Vincent, J. and Anyakoha, C. (2008) 'A review of particle swarm optimization, Part II: Hybridisation, combinatorial, multicriteria and constrained optimization and indicative applications', *Natural Computing: an International Journal*, Vol. 7, No. 1, pp.109–124.

Baskar, S. and Suganthan, P.M. (2004) 'A novel concurrent particle swarm optimization', *Proceedings of the Congress on Evolutionary Computations*, pp.792–796.

Clerc, M. and Kennedy, J. (2002) 'The particle swarm – explosion, stability and convergence in a multi dimensional complex space', *IEEE Transaction Evolutionary Computation*, Vol. 6, pp.58–73.

Deep, K. and Thakur, M. (2007) 'A new crossover operator for real coded genetic algorithms', *Applied Mathematics and Computation*, Vol. 188, No. 1, pp.895–911.

Eberhart, R.C. and Shi, Y. (2000) 'Comparing inertia weights and constriction factors in particle swarm optimization', *Proc. Congress on Evolutionary Computation*, San Diego, CA, pp.84–88,.

Esquivel, S.C. and Coello Coello, C.A. (2003) 'On the use of particle swarm optimization with multi modal functions', *Proceedings of the Congress on Evolutionary Computations*, pp.1130–1136.

He, S., Wu, Q.H., Wen, J.Y., Saunders, J.R. and Paton, R.C. (2004) 'A particle swarm optimizer with passive congregation', *Biosystems*, Vol. 78, pp.135–147.

Hendtlass, T. (2003) 'Preserving diversity in particle swarm optimization', *Lecture Notes in Computer Science*, Vol. 2718, pp.31–40.

Higasbi, N. and Iba, H. (2003) 'Particle swarm optimization with Gaussian mutation', *Proceedings of the 2003 IEEE Swarm Intelligence Symposium*, pp.72–79.

Hu, X., Shi, Y. and Eberhart, R.C. (2004) 'Recent advances in particle swarm', *Proceedings of Congress Evolutionary Computation*, Vol. 1, pp.90–97.

Janson, S. and Middendorf, M. (2005) 'A hierarchical particle swarm optimizer and its adaptive variant', *IEEE Transaction on System, Man and Cybernetics*, Part B, Vol. 38, pp.1272–1282.

Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimization', *Proceedings IEEE International Conference Neural Networks*, Vol. 4, pp.1942–1948.

Kennedy, J. and Mendes, R. (2002) 'Population structure and particle swarm performance', *Proceedings of the Congress on Evolutionary Computation*, pp.1671–1676.

Kennedy, J. and Mendes, R. (2003) 'Neighborhood topologies in fully – informed and best-of-neighborhood particle swarms', *Proceedings of the IEEE International Workshop on Soft Computing in industrial Applications*, pp.45–50.

Krink, T. and Lovbjerg, M. (2002) 'The lifecycle model: combining particle swarm optimization, genetic algorithms and hill climbing', *Proceedings of Parallel Problem solving from Nature*, Vol. 7, pp.621–630.

Krink, T., Vesterstrem, J.S. and Riget, J. (2002) 'Particle swarm optimization with spatial particle extension', *Proceedings of the Congress on Evolutionary Computation*, pp.1474–1479.

Liang, J.J., Qin, A.K., Suganthan, P.N. and Baskar, S. (2004) 'Particle swarm optimization algorithms with novel learning strategies', *Proceedings of the IEEE Conference on Systems, Man and Cybernetics*, pp.3659–3664.

Liu, H., Li, B., Wang, X., Ji, Y. and Tang, Y. (2004) 'Survival density particle swarm optimization for neural network training', ISNN (1), LNCS 3173, pp.332–337, Springer-Verlag.

Lovbjerg, M. and Krink, T. (2002) 'Extending particle swarm optimizers with self-organized criticality', *Proceedings of the Congress on Evolutionary Computation*, pp. 1588–1593.

Lovbjerg, M., Rasmussen, T.K. and Krink, T. (2001) 'Hybrid particle swarm optimizer with breeding and subpopulation', *Proceedings of the Third Genetic and Evolutionary Computation Conference*, pp.469–476.

Mohais A., Ward, C. and Posthoff, C. (2004) 'Randomized directed neighborhood with edge migration in particle swarm optimization', *Proceedings of the IEEE Conference on Evolutionary Computational*, pp.548–555.

Pasupuleti, S. and Battiti, R. (2006) 'The gregarious particle swarm optimizer (G-PSO)', *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation (Seattle, Washington, USA). GECCO '06*, pp.67–74, ACM, New York, NY.

Peram, T., Veeramachaneni, K. and Mohan, C. K. (2003) 'Fitness-distance-ratio based particle swarm optimization', *Proceedings of the IEEE Swarm Intelligence Symposium*, pp.174–181.

Poli, R., Laugdon, W.B. and Holland, O. (2005) 'Extending particle swarm optimization via genetic programming', *Proceedings of the Eighth European Conference on Genetic Programming*, pp.291–300.

Riget, J. and Vesterstrem, J.S. (2002) 'A diversity-guided particle swarm optimizer – the ARPSO', Technical Report 2002-02, EVALife, Department of Computer Science, University of Aarbus.

Shi, Y. and Eberhart, R.C. (1998a) 'A modified particle swarm optimizer', *Proceedings of the IEEE International Conference on Evolutionary Computation*, pp.69–73.

Shi, Y. and Eberhart, R.C. (1998b) 'Parameter selection in particle swarm optimization,' *7th Annual Conference on Evolutionary Programming*, San Diego, USA.

Silva, A., Neves, A. and Costa, E. (2002) 'An empirical comparison of particle swarm and predator – pray optimization', *Lecture Notes in Computer Science*, Vol. 2464, pp.103–110, Springer, Berlin.

Stacey, A., Jancic, M. and Grundy, I. (2003) 'Particle swarm optimization with mutation', *Proceedings of the Congress on Evolutionary Computation*, pp.1425–1430.

Suganthan, P.N. (1999) 'Particle swarm optimizer with neighborhood operator', *Proceedings of the Congress on Evolutionary Computation*, pp.1958–1962.

Van den Bergh, F. and Engelbrecht, A.P. (2004) 'A cooperative approach to particle swarm optimization', *IEEE Trans. On Evolutionary Computation*, Vol. 8, No.3, pp.225–239.

Wei, J., Yuncan, X. and Jixin, Q. (2004) 'An improved particle swarm optimization algorithm with disturbance', *IEEE International Conference on Systems, Man and Cybernetics*, pp.5900–5904.

Zhang, W.J. and Xie, X.F. (2003) 'DEPSO: hybrid particle swarm with differential evolution operator', *Proceedings of the IEEE International Conference on System, Man and Cybernetics*, pp.3816–3821.

Zhang, W.J., Xie, X.F. and Yang, Z.L. (2002) 'Hybrid particle swarm optimizer with mass extinction', *International Conference on Communication, Circuits and Systems*, pp.1170–1173.