Particle Swarm Optimization: Performance Tuning and Empirical Analysis

Millie Pant, Radha Thangaraj, and Ajith Abraham

Abstract. This chapter presents some of the recent modified variants of Particle Swarm Optimization (PSO). The main focus is on the design and implementation of the modified PSO based on diversity, Mutation, Crossover and efficient Initialization using different distributions and Low-discrepancy sequences. These algorithms are applied to various benchmark problems including unimodal, multimodal, noisy functions and real life applications in engineering fields. The effectiveness of the algorithms is discussed.

1 Introduction

The concept of PSO was first suggested by Kennedy and Eberhart [1]. Since its development is 1995, PSO has emerged as one of the most promising optimizing technique for solving global optimization problems. Its mechanism is inspired by the social and cooperative behavior displayed by various species like birds, fish etc including human beings. The PSO system consists of a population (swarm) of potential solutions called particles. These particles move through the search domain with a specified velocity in search of optimal solution. Each particle maintains a memory which helps it in keeping the track of its previous best position. The positions of the particles are distinguished as personal best and global best. PSO has been applied to solve a variety of optimization problems and its performance is compared with other popular stochastic search techniques like Genetic algorithms, Differential Evolution, Simulated Annealing etc. [2], [3], [4]. Although PSO has shown a very good performance in solving many test as well as real life optimization problems, it suffers from the problem of premature convergence like most of the stochastic search techniques, particularly in case of multimodal optimization problems. The *curse* of premature convergence greatly affects the performance of algorithm and many times lead to a sub optimal solution [5]. Aiming at this shortcoming of PSO algorithms, many variations have been

Millie Pant and Radha Thangaraj
Department of Paper Technology, IIT Roorkee, India
email: millifpt@iitr.ernet.in, t.radha@ieee.org

Ajith Abraham
Q2S, Norwegian University of Science and Technology, Norway
email: ajith.abraham@ieee.org

A. Abraham et al. (Eds.): Foundations of Comput. Intel. Vol. 3, SCI 203, pp. 101–128.
springerlink.com

developed to improve its performance. Some of the interesting modifications that helped in improving the performance of PSO include introduction of inertia weight and its adjustment for better control of exploration and exploitation capacities of the swarm [6] [7], introduction of constriction factor to control the magnitudes of velocities [8], impacts of various neighborhood topologies on the swarm [9], extension of PSO via genetic programming [10], use of various mutation operators into PSO [11] – [13]. In the present study ten recent versions of PSO are considered. Out of the ten chosen versions, five versions are based on the efficient initialization of swam, three versions are diversity guided and the remaining versions makes use of cross-over operator to improve the performance of PSO.

The present article has seven sections including the introduction. In the next section, a brief description of the basic PSO is given. Section 3 is divided into three subsections; in 3.1, PSO versions with different initialization schemes are described; in section 3.2 three diversity guided PSO are given and in Section 3.3 PSO with crossover operator is described. Section 4 is devoted to numerical problems consisting of ten popular bench mark problems and two real life problems. In Section 5 and Section 6, describe the experimental settings and numerical results respectively. The chapter finally concludes with Section 7.

2 Particle Swarm Optimization

The working of the Basic Particle Swarm Optimization (BPSO) may be described as: For a D-dimensional search space the position of the i^{th} particle is represented as $X_i = (x_{i1}, x_{i2}, \ldots x_{iD})$. Each particle maintains a memory of its previous best position $P_{besti} = (p_{i1}, p_{i2} \ldots p_{iD})$. The best one among all the particles in the population is represented as $P_{gbest} = (p_{g1}, p_{g2} \ldots p_{gD})$. The velocity of each particle is represented as $V_i = (v_{i1}, v_{i2}, \ldots v_{iD})$. In each iteration, the P vector of the particle with 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 a new position of the particle is determined using that velocity. The two basic equations which govern the working of PSO are that of velocity vector and position vector given by:

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (1)$$

$$x_{id} = x_{id} + v_{id} \quad (2)$$

The first part of equation (1) represents the inertia of the previous velocity, the second part is the cognition part and it tells us about the personal experience of the particle, the third part represents the cooperation among particles and is therefore named as the social component. Acceleration constants c_1, c_2 and inertia weight w are the predefined by the user and r_1, r_2 are the uniformly generated random numbers in the range of [0, 1].

3 Modified Version of Particle Swarm Optimization

Empirical studies have shown that the basic PSO has a tendency of premature convergence [518], [559], [602], [606], [649] and the main reason for this

behavior is due to the loss of diversity in successive iterations. It has been observed that the presence of a suitable operator may help in improving the performance of PSO quite significantly. This chapter concentrates on two things; first is on the efficient generation of population using different initialization schemes and second is the use of diversity to guide the swarm using different operations like repulsion, mutation and crossover.

3.1 Efficient Initialization Particle Swarm Optimization

PSO (and other search techniques, which depend on the generation of random numbers) works very well for problems having a small search area (i.e. a search area having low dimension), but as the dimension of search space is increased, the performance deteriorates and many times converge prematurely giving a suboptimal result [5]. This problem becomes more persistent in case of multimodal functions having several local and global optima. One of the reasons for the poor performance of a PSO may be attributed to the dispersion of initial population points in the search space i.e. to say, if the swarm population does not cover the search area efficiently, it may not be able to locate the potent solution points, thereby missing the global optimum [14]. This difficulty may be minimized to a great extent by selecting a well-organized distribution of random numbers.

This section analyzes the behavior of some simple variations of PSO where only the initial distribution of random numbers is changed. Initially in the algorithms the initial uniform distribution is replaced by other probability distributions like exponential, lognormal and Gaussian distributions. It is interesting to see that even a small change in the initial distribution produces a visible change in the numerical results. After that more specialized algorithms are designed which use low discrepancy sequences for the generation of random numbers. A brief description of the algorithms is given in the subsequent sections.

The most common practice of generating random numbers is the one using an inbuilt subroutine (available in most of the programming languages), which uses a uniform probability distribution to generate random numbers. It has been shown that uniformly distributed particles may not always be good for empirical studies of different algorithms. The uniform distribution sometimes gives a wrong impression of the relative performance of algorithms as shown by Gehlhaar and Fogel [15].

3.1.1 Initializing the Swarm Using Different Probability Distributions [16]

Different Probability Distributions like Exponential and Gaussian have already been used for the fine tuning of PSO parameters [17], [18]. But for initializing the swarm most of the approaches use uniformly distributed random numbers. Pant et al. [16] investigated the possibility of having a different probability distribution (Gaussian, Exponential, Lognormal) for the generation of random number other than the uniform distribution. Empirical results showed that distributions other than uniform distribution are equally competent and in most of the cases are better than uniform distribution. The algorithms GPSO, EPSO and LNPSO use

Gaussian, exponential and lognormal distributions respectively. The algorithms follow the steps of BPSO given in Section 2 except for the fact that they use mentioned distributions in place of uniform distributions.

3.1.2 Initializing the Swarm Using Low-Discrepancy Sequences [19]

Theoretically, it has been proved that low discrepancy sequences are much better than the pseudo random sequences because they are able to cover the search space more evenly in comparison to pseudo random sequences (please see Figures 1(a) and 1(b)). Some previous instances where low discrepancy sequences have been used to improve the performance of optimization algorithms include [20] – [24]. In [22] – [24] authors have made use of Sobol and Faure sequences. Similarly, Nguyen et al. [21] have shown a detailed comparison of Halton Faure and Sobol sequences for initializing the swarm. In the previous studies, it has already been shown that the performance of Sobol sequence dominates the performance of Halton and Faure sequences. The performance of PSO using Van der Corput sequence called VCPSO along with PSO with Sobol sequence called SOPSO (which is said be superior than other low discrepancy sequences according to the previous studies) for swarm initialization is scrutinized and tested them for solving global optimization problems in large dimension search spaces by Pant et al. [19].

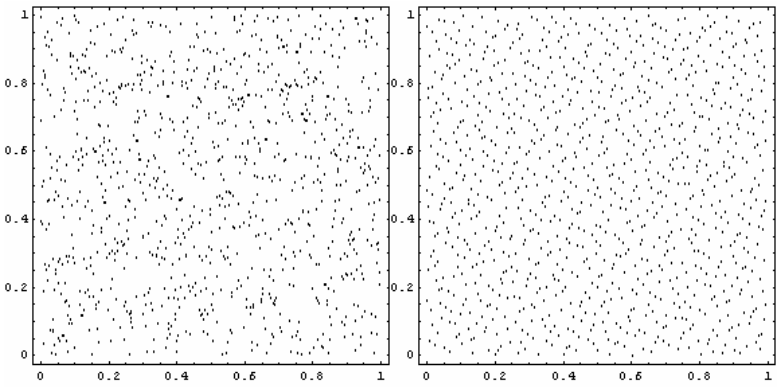


Fig. 1(a) Sample points generated using a pseudo random sequence. 1(b) Sample points generated using a quasi random sequence

Brief description of the sequences used in VCPSO and SOPSO:

Van der Corput Sequence

A Van der Corput sequence is a low-discrepancy sequence over the unit interval first published in 1935 by the Dutch mathematician J. G. Van der Corput. It is a digital (0, 1)-sequence, which exists for all bases $b \geq 2$. It is defined by the *radical inverse function* $\varphi_b : \mathrm{N}_0 \rightarrow [0, 1)$. If $n \in \mathrm{N}_0$ has the b-adic expansion

$$n = \sum_{j=0}^{T} a_j b^{j-1} \tag{3}$$

with $a_j \in \{0,\ldots, b-1\}$, and $T = \lfloor \log_b n \rfloor$ then φ_b is defined as

$$\varphi_b(n) = \sum_{j=0}^{T} \frac{a_j}{b^j} \tag{4}$$

In other words, the *j*th *b*-adic digit of *n* becomes the *j*th *b*-adic digit of $\varphi_b(n)$ behind the decimal point. The Van der Corput sequence in base *b* is then defined as $(\varphi_b(n))_{n \geq 0}$.

The elements of the Van der Corput sequence (in any base) form a dense set in the unit interval: for any real number in [0, 1] there exists a sub sequence of the Van der Corput sequence that converges towards that number. They are also uniformly distributed over the unit interval.

Sobol Sequence

The construction of the Sobol sequence [25] uses linear recurrence relations over the finite field, F2, where F2 = {0, 1}. Let the binary expansion of the non-negative integer n be given by $n = n_1 2^0 + n_2 2^1 + \ldots + n_w 2^{w-1}$. Then the nth element of the jth dimension of the Sobol Sequence, $X_n^{(j)}$, can be generated by:

$$X_n^{(j)} = n_1 v_1^{(j)} \oplus n_2 v_2^{(j)} \oplus \ldots \oplus n_w v_w^{(j)}$$

where $v_i^{(j)}$ is a binary fraction called the ith direction number in the jth dimension. These direction numbers are generated by the following q-term recurrence relation:

$v_i^{(j)} = a_1 v_{i-1}^{(j)} \oplus a_2 v_{i-2}^{(j)} \oplus \ldots \oplus a_q v_{i-q+1}^{(j)} \oplus v_{i-q}^{(j)} \oplus (v_{i-q}^{(j)} / 2^q)$ i > q, and the bit, a_i, comes from the coefficients of a degree-q primitive polynomial over F2.

VC-PSO and SO-PSO Algorithm

It has been shown that uniformly distributed particles may not always be good for empirical studies of different algorithms. The uniform distribution sometimes gives a wrong impression of the relative performance of algorithms as shown by Gehlhaar and Fogel [15].

The quasi random sequences on the other hand generates a different set of random numbers in each iteration, thus providing a better diversified population of solutions and thereby increasing the probability of getting a better solution.

Keeping this fact in mind we decided to use the Vander Corput sequence and Sobol sequence for generating the swarm. The swarm population follows equation (1) and (2) for updating the velocity and position of the swarm. However for the generation of the initial swarm Van der Corput Sequence and Sobol Sequences have been used for VC-PSO and SO-PSO respectively.

3.2 Diversity Guided Particle Swarm Optimization

Diversity may be defined as the dispersion of potential candidate solutions in the search space. Interested readers may please refer to [26] for different formulae used for calculating diversity. One of the drawbacks of most of the population based search techniques is that they work on the principle of contracting the search domain towards the global optima. Due to this reason after a certain number of iterations all the points get accumulated to a region which may not even be a region of local optima, thereby giving suboptimal solutions [5]. Thus without a suitable diversity enhancing mechanism it is very difficult for an optimization algorithm to reach towards the true solution. The problem of premature convergence becomes more persistent in case of highly multimodal functions like Rastringin and Griewank having several local minima. This section presents three algorithms Attraction Repulsion PSO (ATREPSO), Gaussian Mutation PSO (GMPSO) and Quadratic Interpolation PSO (QIPSO) which use different diversity enhancing mechanisms to improve the performance of the swarm. All the algorithms described in the given sub sections use diversity threshold values d_{low} and d_{high} to guide the movement of the swarm. The threshold values are predefined by the user. In ATREPSO, the swarm particles follow the mechanism of repulsion so that instead of converging towards a particular location the particles are diverged from that location. In case of GMPSO and QIPSO evolutionary operators like mutation and crossover are induced in the swarm to perturb the population. These algorithms are described in the following subsections.

3.2.1 Attraction Repulsion Particle Swarm Optimization Algorithm [27]

The Attraction Repulsion Particle Swarm Optimization Algorithm (ATREPSO) of Pant et al. [27] is a simple extension of the Attractive and Repulsive PSO (ARPSO) proposed by Vesterstorm [28], where a third phase called *in between* phase or the phase of *positive conflict* is added In ATREPSO, the swarm particles switches alternately between the three phases of attraction, repulsion and an 'in between' phase which consists of a combination of attraction and repulsion. The three phases are defined as:

Attraction phase (when the particles are attracted towards the global optimal)

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{5}$$

Repulsion phase (particles are repelled from the optimal position)

$$v_{id} = wv_{id} - c_1 r_1 (p_{id} - x_{id}) - c_2 r_2 (p_{gd} - x_{id}) \tag{6}$$

In-between phase (neither total attraction nor total repulsion)

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) - c_2 r_2 (p_{gd} - x_{id}) \tag{7}$$

In the in-between phase, the individual particle is attracted by its own previous best position p_{id} and is repelled by the best known particle position p_{gd}. In this way there is neither total attraction nor total repulsion but a balance between the two.

The swarm particles are guided by the following rule

$$v_{id} = \begin{cases} wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}), div > d_{high} \\ wv_{id} + c_1 r_1 (p_{id} - x_{id}) - c_2 r_2 (p_{gd} - x_{id}), d_{low} < div < d_{high} \\ wv_{id} - c_1 r_1 (p_{id} - x_{id}) - c_2 r_2 (p_{gd} - x_{id}), div < d_{low} \end{cases} \quad (8)$$

3.2.2 Gaussian Mutation Particle Swarm Optimization Algorithm [29]

The concept of mutation is quite common to Evolutionary Programming and Genetic Algorithms. The idea behind mutation is to increase of diversity of the population. There are several instances in PSO also where mutation is introduced in the swarm. Some mutation operators that have been applied to PSO include Gaussian [244], Cauchy [655], [656], Nonlinear [589], Linear [589] etc. The Gaussian Mutation Particle Swarm Optimization (GMPSO) algorithm given in this section is different from the above mentioned algorithms as it uses the threshold values to decide the activation of mutation operator. The concept is similar to that of ATREPSO i.e. to use diversity to decide the movement of the swarm. The algorithm uses the general equations (1) and (2) for updating the velocity and position vectors. At the same time a track of diversity is also kept which starts decreasing slowly and gradually after a few iterations because of the fast information flow between the swam particles leading to clustering of particles. It is at this stage that the Gaussian mutation operator given as $X_{t+1}[i] = X_t[i] + \eta^*$ *Rand()*, where *Rand* is a random number generated by Gaussian distribution, is activated with the hope to increase the diversity of the swarm population. Here η is a scaling parameter.

3.2.3 Quadratic Interpolation Particle Swarm Optimization Algorithm [30]

As mentioned in the previous section, there are several instances available in literature on the use of mutation operator however there are not much references on the use of reproduction operator. One of the earlier references on the use of reproduction operator can be found in Clerc [101]. The Quadratic Interpolation Particle Swarm Optimization (QIPSO) algorithm described in this chapter uses concept of reproduction. It uses diversity as a measure to guide the swarm. When the diversity becomes less than d_{low}, then the quadratic crossover operator is activated to generate a new potential candidate solution. The process is repeated iteratively till the diversity reaches the specified threshold d_{high}. The quadratic crossover operator used in this paper is a nonlinear crossover operator which makes use of three particles of the swarm to produce a particle which lies at the point of minima of the quadratic curve passing through the three selected particles.

It uses a = X_{min}, (best particle with minimum function value) and two other randomly selected particles {b, c} (a, b and c are different particles) from the swarm to determine the coordinates of the new particle $\tilde{x}^i = (\tilde{x}^1, \tilde{x}^2, \ldots\ldots \tilde{x}^n)$, where

$$\tilde{x}^i = \frac{1}{2} \frac{(b^{i^2} - c^{i^2}) * f(a) + (c^{i^2} - a^{i^2}) * f(b) + (a^{i^2} - b^{i^2}) * f(c)}{(b^i - c^i) * f(a) + (c^i - a^i) * f(b) + (a^i - b^i) * f(c)} \quad (9)$$

The nonlinear nature of the quadratic crossover operator used in this work helps in finding a better solution in the search space.

3.3 Crossover Based Particle Swarm Optimization

In this section two more modifications applied to the QIPSO given in Section 3.2 are described.

3.3.1 QIPSO-1 [31] and QIPSO-2 [32] Algorithms

The basic idea of QIPSO-1 and QIPSO-2 are modified versions of QIPSO algorithm given in section 3.2, which differ from each other in selection criterion of the individual. In QIPSO-1, the new point generated by the quadratic interpolation given by equation (9) is accepted in the swarm only if it is better than the worst particle of the swarm, where as in QIPSO-2, the particle is accepted if it is better than the global best particle.

4 Numerical Problems

One of the shortcomings of population based search techniques is that there are not many concrete proofs available to establish their authority for solving a wide range of problems. Therefore the researchers often depend on empirical studies to scrutinize the behavior of an algorithm. The numerical problems may be divided into two classes; benchmark problems and real life problems. For the present article ten standard benchmark functions and two real life problems described in the following subsections are taken.

4.1 Benchmark Problems

A collection of ten benchmark problems given in Table 1 is taken for the present study to analyze the behavior of algorithms taken in this study. These problems may not be called exhaustive but they provide a good launch pad for testing the credibility of an optimization algorithm. The first eight problems are scalable i.e. the problems can be tested for any number of variables. However for the present study medium sized problems of dimension 20 are taken. The three dimensional graphs of the test functions are depicted in Figures 2(a) to (i).

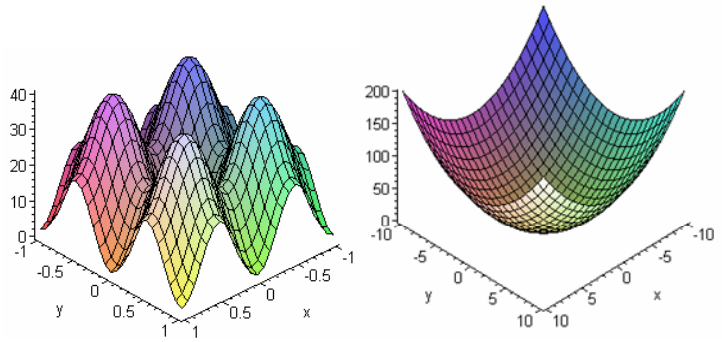
The special properties of the benchmark functions taken in this study may be described as:

- The first function f_1, commonly known as Rastringin function, is a highly multimodal function where the degree of multimodality increases with the increase in the dimension of the problem.
- The second function f_2, also known as spherical function is a continuous, strictly convex and unimodal function and usually do not pose much difficulty for an optimization algorithm.

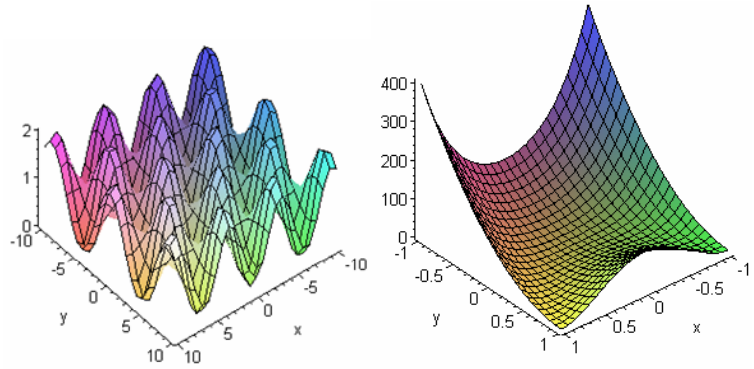
- Griewank function is the third function. It is highly multimodal function having several local minima.
- The search space of the fourth function is dominated by a large gradual slope. Despite the apparent simplicity of the problem it is considered difficult for search algorithms because of its extremely large search space combined with relatively small global optima.
- f_5 is a noisy function where a uniformly distributed random noise is added to the objective function. Due to the presence of noise the objective function keeps changing from time to time and it becomes a challenge for an optimization algorithm to locate the optimum.
- Functions f_6 to f_8 are again multimodal functions having several optima. Such functions provide a suitable platform for testing the credibility of an optimization algorithm.
- Function f_9 and f_{10} are two dimensional functions. Function f_{10} is although simple in appearance but it an interesting and challenging function having 786 local minima and 18 global minima.

Table 1 Numerical Benchmark Problems [3]

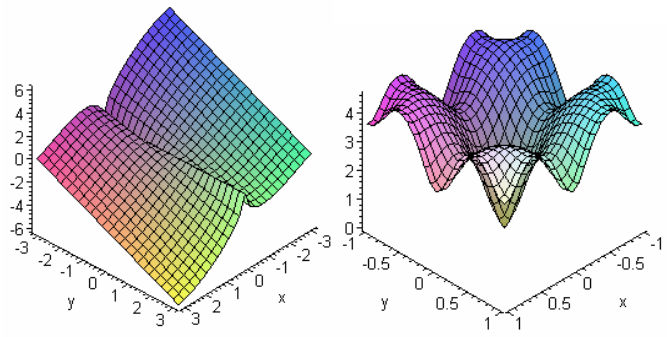
Name of function	**Function Definition**	**Range**	**Minimum Value**
Rastrigin Function	$f_1(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12,5.12]	0
Spherical Function	$f_2(x) = \sum_{i=1}^{n} x_i^2$	[-5.12,5.12]	0
Griewank Function	$f_3(x) = \frac{1}{4000} \sum_{i=0}^{n-1} x_i^2 + \sum_{i=0}^{n-1} \cos(\frac{x_i}{\sqrt{i+1}}) + 1$	[-600,600]	0
Rosenbrock Function	$f_4(x) = \sum_{i=0}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2$	[-30,30]	0
Noisy Function	$f_5(x) = (\sum_{i=0}^{n-1} (i+1)x_i^4) + rand[0,1]$	[-1.28,1.28]	0
Schewefel Function	$f_6(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{\lvert x_i \rvert})$	[-500,500]	-8329.658
Ackley Function	$f_7(x) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))$	[-32,32]	0
Sine Function	$f_8(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(i\frac{x_i^2}{\pi}))^{2m}, \quad m = 10$	[-π,π]	---
Himmelblau Function	$f_9(x) = (x_2 + x_1^2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + x_1$	[-5,5]	-3.78396
Shubert Function	$f_{10}(x) = \sum_{j=1}^{5} j\cos((j+1)x_1 + j) \sum_{j=1}^{5} j\cos((j+1)x_2 + j)$	[-10,10]	-186.7309



(a) Function f_1 (b) Function f_2

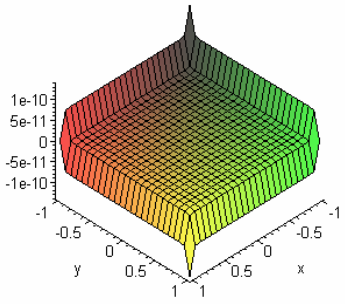


(c) Function f_3 (d) Function f_4

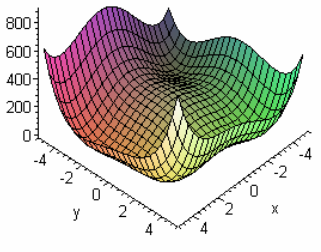


(e) Function f_6 (f) Function f_7

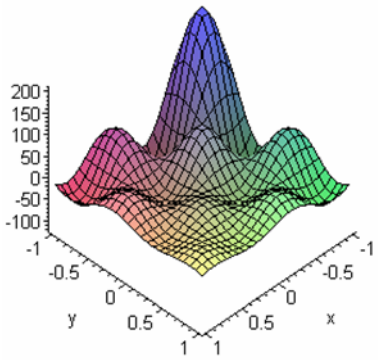
Fig. 2 Three Dimensional graphs of benchmark problems



(g) Function f_8



(h) Function f_9



(i) Function f_{10}

Fig. 2 (*continued*)

4.2 Real Life Problems

Two real life engineering design problems are considered to depict the effectiveness of the algorithms discussed in the present article. These are nonlinear problems and both the problems are common in the field of electrical engineering. Mathematical model of the problems are given as:

4.2.1 Design of a Gear Train [33]

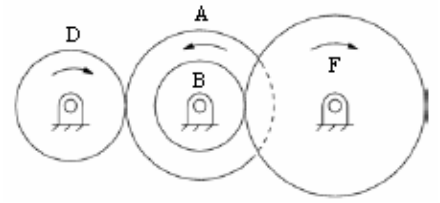
The first problem is to optimize the gear ratio for the compound gear train. This problem shown in Figure 3 was introduced by Sandgren [34]. It is to be designed such that the gear ratio is as close as possible to 1/6.931. For each gear the number of teeth must be between 12 and 60. Since the number of teeth is to be an integer, the variables must be integers. The mathematical model of gear train design is given by,

$$\text{Min} \quad f = \left\{ \frac{1}{6.931} - \frac{T_d T_b}{T_a T_f} \right\}^2 = \left\{ \frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4} \right\}^2$$

Subject to: $12 \le x_i \le 60 \quad i = 1,2,3,4$

$[x_1, x_2, x_3, x_4] = [T_d, T_b, T_a, T_f]$, x_i's should be integers. T_a, T_b, T_d, and T_f are the number of teeth on gears A, B, D and F respectively.

Fig. 3 Compound Gear Train



4.2.2 Transistor Modeling [35]

The second problem is a transistor modeling problem. The mathematical model of the transistor design is given by,

Minimize $f(x) = \gamma^2 + \sum_{k=1}^{4} (\alpha_k{}^2 + \beta_k{}^2)$

Where

$$\alpha_k = (1 - x_1 x_2) x_3 \{\exp[x_5 (g_{1k} - g_{3k} x_7 \times 10^{-3} - g_{5k} x_8 \times 10^{-3})] - 1\} g_{5k} + g_{4k} x_2$$

$$\beta_k = (1 - x_1 x_2) x_4 \{\exp[x_6 (g_{1k} - g_{2k} - g_{3k} x_7 \times 10^{-3} + g_{4k} x_9 \times 10^{-3})] - 1\} g_{5k} x_1 + g_{4k} .$$

$$\gamma = x_1 x_3 - x_2 x_4$$

Subject to: $x_i \ge 0$

The numerical constants g_{ik} are given by the matrix

$$\begin{bmatrix} 0.485 & 0.752 & 0.869 & 0.982 \\ 0.369 & 1.254 & 0.703 & 1.455 \\ 5.2095 & 10.0677 & 22.9274 & 20.2153 \\ 23.3037 & 101.779 & 111.461 & 191.267 \\ 28.5132 & 111.8467 & 134.3884 & 211.4823 \end{bmatrix}$$

This objective function provides a least-sum-of-squares approach to the solution of a set of nine simultaneous nonlinear equations, which arise in the context of transistor modeling.

5 Experimental Settings

Like all Evolutionary Algorithms, PSO has a set of parameters which is to be defined by the user. These parameters are population size, inertia weight, acceleration constants etc. these parameters may be varied as per the complexity

of the problem. For example the population size in PSO related literature has been suggested as 2n to 5n, where n is the number of decision variables or a fixed population size. In the present study a fixed population size of thirty is taken for all the problems, which gave reasonably good results. Similarly various examples are available on the variations done in inertia weight and acceleration constants. For the present study, which consist of small moderate size problems of dimension 2 and 20, the list parameters which gave sufficiently good results is summarized below.

Common Parameters:

Population Size (NP):	Number of variables 30
Inertia weight:	Linearly decreasing (0.9 – 0.4)
Acceleration Constants:	c1 = c2 = 2.0
Stopping Criterion:	Maximum number of generations = 10000

Probability Distributions for initializing the swarm:
Gaussian distribution:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \tag{10}$$

with mean 0 and standard deviation 1, i.e. N (0,1).
Exponential distribution:

$$f(x) = \frac{1}{2b}\exp(-|x-a|/b), \quad -\infty \le x < \infty, \tag{11}$$

with a, b > 0.It is evident that one can control the variance by changing the parameters a and b.
Log-normal distribution:

$$f(x) = \frac{e^{-(\ln x)^2/2}}{x\sqrt{2\pi}} \tag{12}$$

with mean 0 and standard deviation 1.

Diversity Measurement

$$Diversity(S(t)) = \frac{1}{n_s}\sum_{i=1}^{n_s}\sqrt{\sum_{j=1}^{n_x}(x_{ij}(t) - \overline{x_j(t)})^2}$$

Threshold values: $d_{high} = 0.25$, $d_{low} = 5.0*10^{-6}$

Repair method for points violating the boundary conditions

Hardware Settings
All algorithms are executed on a P-IV PC. Programming language used is DEV C++

Table 2 Comparison results of PSO, GPSO, EPSO, and LNPSO (Mean/diversity/standard deviation)

Function	PSO	GPSO	EPSO	LNPSO
f_1	22.339158 0.000115 15.932042	**9.750543** 0.364310 5.433786	12.173973 5.380822e-05 9.274301	23.507127 0.264117 15.304573
f_2	1.167749e-45 2.426825e-23 5.222331e-46	**1.114459e-45** 3.168112e-23 4.763499e-46	1.167749e-45 3.909771e-23 5.222331e-46	**1.114459e-45** 2.778935e-23 4.763499e-46
f_3	0.031646 0.000710 0.025322	**0.004748** 1.631303e-08 0.012666	0.011611 0.001509 0.019728	0.011009 0.000877 0.019186
f_4	22.191725 2.551408 1.615544e+04	9.992837 2.527997 3.168912	8.995165 1.8737 3.959364	**4.405738** 2.904427 4.121244
f_5	8.681602 0.340871 9.001534	0.636016 0.210458 0.296579	**0.380297** 0.237913 0.281234	0.537461 0.254176 0.285361
f_6	-6178.559896 0.072325 489.3329	**-6354.119792** 0.059937 483.654032	-6306.353646 0.026106 575.876696	-6341.4000 0.034486 568.655436
f_7	3.483903e-18 3.651635e-18 8.359535e-19	**3.136958e-18** 5.736429e-13 8.596173e-19	3.368255e-18 3.903653e-18 8.596173e-19	3.368255e-18 3.846865e-18 8.596173e-19
f_8	-18.1594 1.17699 1.05105	-18.5162 0.603652 0.907089	**-18.675** 2.63785 1.06468	-18.3944 0.221685 1.02706
f_9	-3.331488 2.747822e-05 1.24329	**-3.63828** 1.71916e-009 0.346782	**-3.63828** 1.65462e-009 0.346782	-3.49261 1.4056e-009 0.445052
f_{10}	-186.730941 0.362224 1.424154e-05	**-186.731** 1.0056 3.3629e-014	**-186.731** 0.19356 3.11344e-014	**-186.731** 0.82143 2.00972e-014

6 Numerical Results

A comparative analysis of the algorithms described is given Tables 2 to 9. Each algorithm was executed 100 times and the average fitness function value, diversity and standard deviation are reported. Tables 2 to 5 give the numerical results for benchmark problems whereas, the numerical results of real life problems are given in Tables 6 to 9.

Table 2, gives the numerical results of PSO versions initialized with Gaussian, exponential and lognormal probability distributions. From the numerical results it can be seen that the PSO using Gaussian mutation, GPSO, gave the best

Table 3 Comparison results of PSO, VC-PSO and SO-PSO (Mean/diversity/standard deviation)

Function	PSO	VC-PSO	SO-PSO	Function	PSO	VC-PSO	SO-PSO
f_1	22.3391 0.00011 15.9320	9.99929 1.00441 4.08386	**8.95459** 0.319194 2.65114	f_6	-6178.559 0.072325 4.893e+02	**-6503.05** 3.469e-06 477.252	-6252.51 2.478e-06 472.683
f_2	1.16e-45 2.42e-23 5.22e-46	**1.17e-108** 7.15e-054 4.36e-108	1.51e-108 6.36e-055 4.46e-108	f_7	3.483e-18 3.651e-18 8.359e-19	5.473e-19 5.039e-19 1.776e-18	**4.585e-19** 6.506e-17 1.538e-18
f_3	0.03164 0.00071 0.02532	**0.00147** 1.233e-08 0.00469	0.001847 9.940e-09 0.004855	f_8	-18.1594 1.17699 1.05105	-18.2979 0.0306 0.8902	**-18.70665** 0.0316574 1.028749
f_4	22.1917 2.55140 1.61e+04	**6.30326** 2.01591 3.99428	6.81079 2.61624 3.76973	f_9	-3.331488 2.747e-05 1.24329	-3.58972 1.439e-09 0.388473	**-3.78396** 3.946e-09 1.47699
f_5	8.68160 0.34087 9.00153	**0.410042** 0.230096 0.294763	0.806175 0.191133 0.868211	f_{10}	-186.730 0.36222 1.424e-05	**-186.731** 1.10502 2.770e-14	**-186.731** 0.32435 3.595e-14

performance in comparison to other versions, followed by EPSO and LNPSO. For the first function, f_1, GPSO gave the best function value of approximately 10.00 which is much better than the values obtained by the other algorithms. For f_2, which is a simple spherical function all the algorithms gave more or less similar results. However GPSO and LNPSO gave a slightly better performance. For f_3, once again the average fitness function value obtained by GMPSO is much better than the average fitness function value obtained by EPSO and LNPSO. For f_6 and f_7 once again GMPSO outperformed the other algorithms given in Table 2. For f9, both GMPSO and EPSO gave same result, which is better than the other two algorithms. Whereas for f_{10}, GMPSO, EPSO and LNPSO gave same result which is marginally better than the result obtained by basic PSO. In all, out of the 10 test functions GPSO outperformed others in 7 test cases. EPSO gave better results in 4 cases and LNPSO performed better in 3 cases. In all the cases the results were better than Basic PSO using uniform distribution.

Table 3, gives the comparison of PSO versions initialized with low discrepancy sequences with the basic PSO. It can be observed that PSO initialized with Sobol sequence (SOPSO) gave slightly better results than PSO initialized with Van der corput sequence. But a notable thing is that although SOPSO outperformed VCPSO in most of the test cases, the percentage of improvement is only marginal. Whereas, if we compare these results with basic PSO then the quality of solutions obtained by SOPSO and VCPSO is significantly better than the solutions obtained by basic PSO. For example, in f_1, which is a highly multimodal function the optimum function value obtained by VCPSO and SOPSO is approximately 10.00 and 9.00 respectively where as the optimum function value obtained by basic PSO is approximately 22.00. Likewise, there is a significant improvement in the

Table 4 Comparison results of PSO, QIPSO, ATREPSO and GMPSO (Mean/diversity/ standard deviation)

Function	PSO	QIPSO	ATREPSO	GMPSO
f_1	22.339158 0.000115 15.932042	**11.946888** 0.015744 9.161526	19.425979 7.353246 14.349046	20.079185 7.143211e-05 13.700202
f_2	1.167749e-45 2.426825e-23 5.222331e-46	**0.000000** 0.000000 0.000000	4.000289 e-17 8.51205 0.000246	7.263579e-17 0.00026 6.188854e-17
f_3	0.031646 0.000710 0.025322	**0.01158** 3.391647e-05 0.01285	0.025158 0.000563 0.02814	0.024462 0.000843 0.039304
f_4	22.191725 2.551408 1.615544e+04	**8.939011** 1.983866 3.106359	19.49082 1.586547 3.964335e+04	14.159547 6.099418e-05 4.335439e+04
f_5	8.681602 0.340871 9.001534	**0.451109** 0.0509 0.328623	8.046617 2.809409 8.862385	7.160675 0.29157 7.665802
f_6	-6178.559896 0.072325 489.3329	**-6355.58664** 0.00881 477.532584	-6183.6776 199.95052 469.611104	-6047.670898 0.062176 482.926738
f_7	3.483903e-18 3.651635e-18 8.359535e-19	**2.461811e-24** 0.000127 0.014425	0.018493 42.596802 0.014747	1.474933e-18 0.061308 1.153709e-08
f_8	-18.1594 1.17699 1.05105	-18.4696 1.2345 0.092966	**-18.9829** 0.39057 0.272579	-18.3998 1.63242 0.403722
f_9	-3.331488 2.747822e-05 1.24329	**-3.783961** 0.637823 0.190394	-3.751458 3.214462 0.174460	-3.460233 9.066805e-06 0.45782
f_{10}	-186.730941 0.362224 1.424154e-05	**-186.730942** 2.169003 3.480934e-14	-186.730941 5.410105 1.424154e-05	**-186.730942** 0.239789 1.525879e-05

function value for functions f_4 and f_5. For f_4, VCPSO and SOPSO gave function values as 6.00 and 7.00 approximately and basic PSO gave an average fitness function value of 22.00. In f_5, VCPSO and SOPSO converged to 0.4 and 0.8 respectively while basic PSO converged to an optimum function value of 8.00.

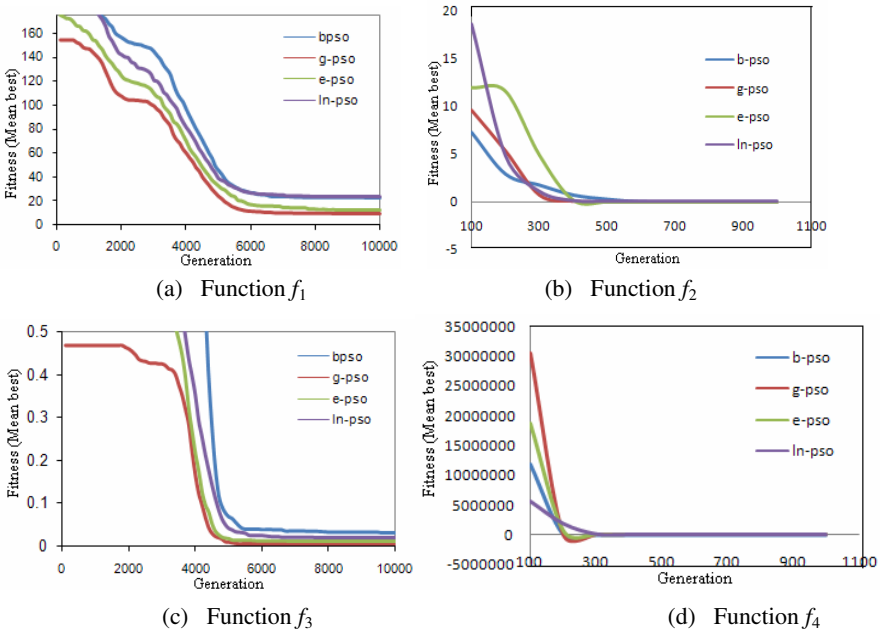
In Table 4, the results of diversity guided PSO algorithms are given. From the numerical results it is evident that the PSO assisted with quadratic crossover operator, QIPSO is a clear winner. QIPSO gave significantly better performance than ATREPSO, GMPSO and basic PSO in 9 out of 10 test cases taken for the present study. Second place goes to ATREPSO and third to GMPSO.

Table 5, gives the comparison of modified versions of QIPSO with basic PSO. If a comparison is done between QIPSO1 and QIPSO2, than from the numerical

Table 5 Comparison results of PSO, QIPSO-1 and QIPSO-2 (Mean best fitness)

Fun-ction	**BPSO**	**QIPSO-1**	**QIPSO-2**
	Mean Best Fitness	Mean Best Fitness	Mean Best Fitness
*f*1	22.339158	0.994954	**5.97167e-01**
*f*2	1.167749e-45	**2.523604e-45**	8.517991e-43
*f*3	0.031646	0.015979	**2.940000e-02**
*f*4	22.191725	77.916591	**51.0779**
*f*5	8.681602	0.454374	**4.540630e-01**
*f*6	-6178.559896	**-9185.074692**	-9.185054e+03
f7	3.483903e-18	**5.89622e-10**	6.300262e-09
*f*8	-18.1594	**-27.546**	**-27.546**
*f*9	-3.331488	-3.58972	**-3.78396**
*f*10	-186.730941	**-186.731**	**-186.731**

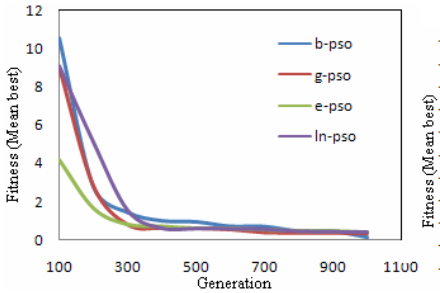
results it can be seen that QIPSO-2, in which the new particle is accepted in the swarm only if it is better than the global best particle is better than QIPSO-1 in terms of average fitness function value. However once again it can be observed that the improvement of QISPO-2 over QIPSO-1 is only marginal whereas both the algorithms performed much better than the basic PSO. Empirical results are graphically illustrated in Figures 4-7.



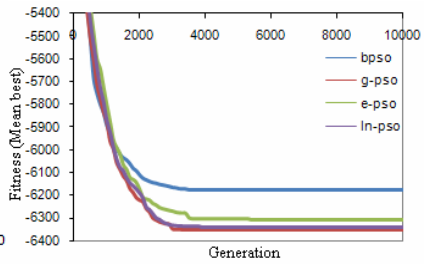
(a) Function f_1 (b) Function f_2

(c) Function f_3 (d) Function f_4

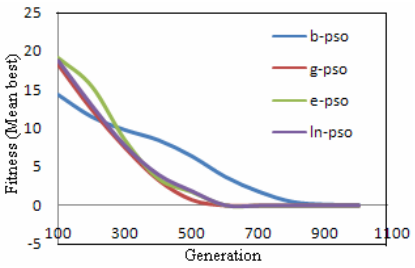
Fig. 4 Performance for BPSO, GPSO, EPSO and LNPSO



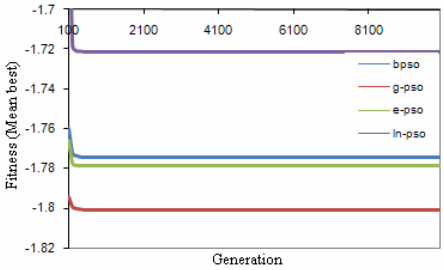
(e) Function f_5



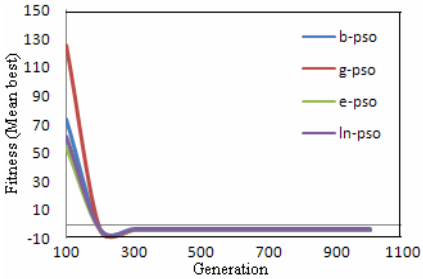
(f) Function f_6



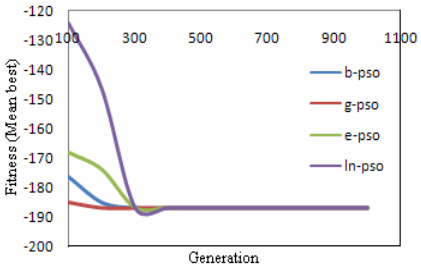
(g) Function f_7



(h) Function f_8

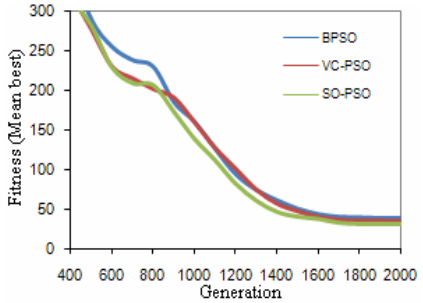


(i) Function f_9

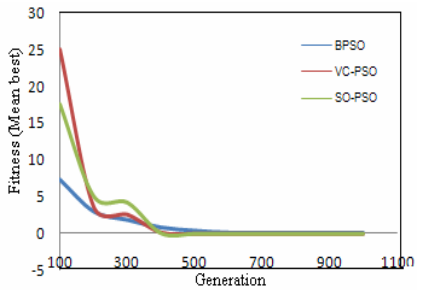


(j) Function f_{10}

Fig. 4 (*continued*)

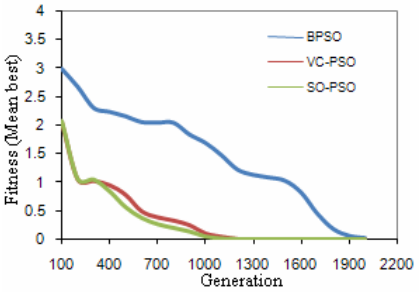


(a) Function f_1

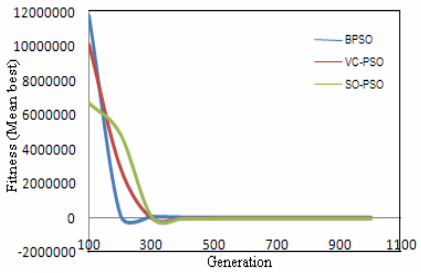


(b) Function f_2

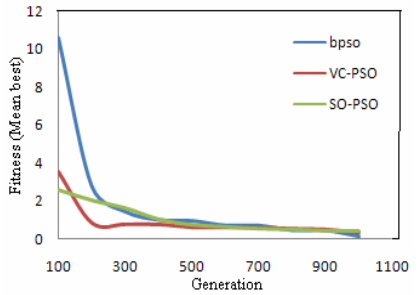
Fig. 5 Performance curves for BPSO, VC-PSO and SO-PSO



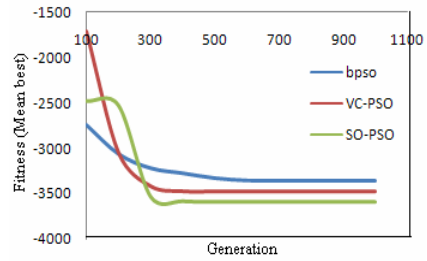
(c) Function f_3



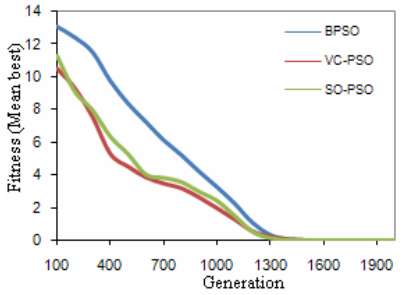
(d) Function f_4



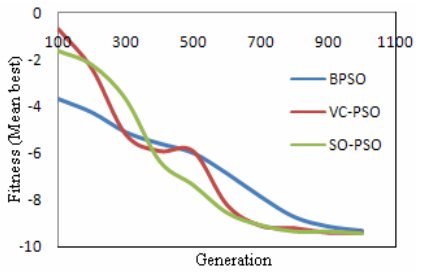
(e) Function f_5



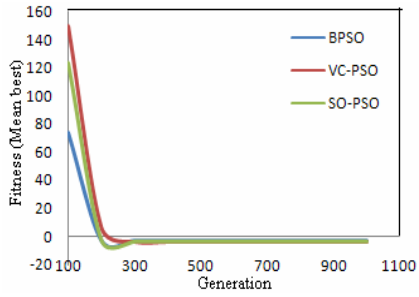
(f) Function f_6



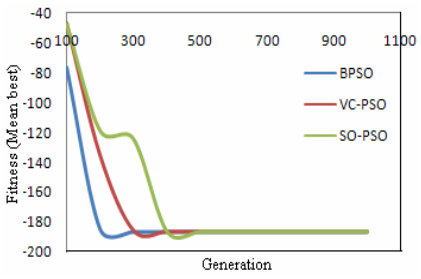
(g) Function f_7



(h) Function f_8

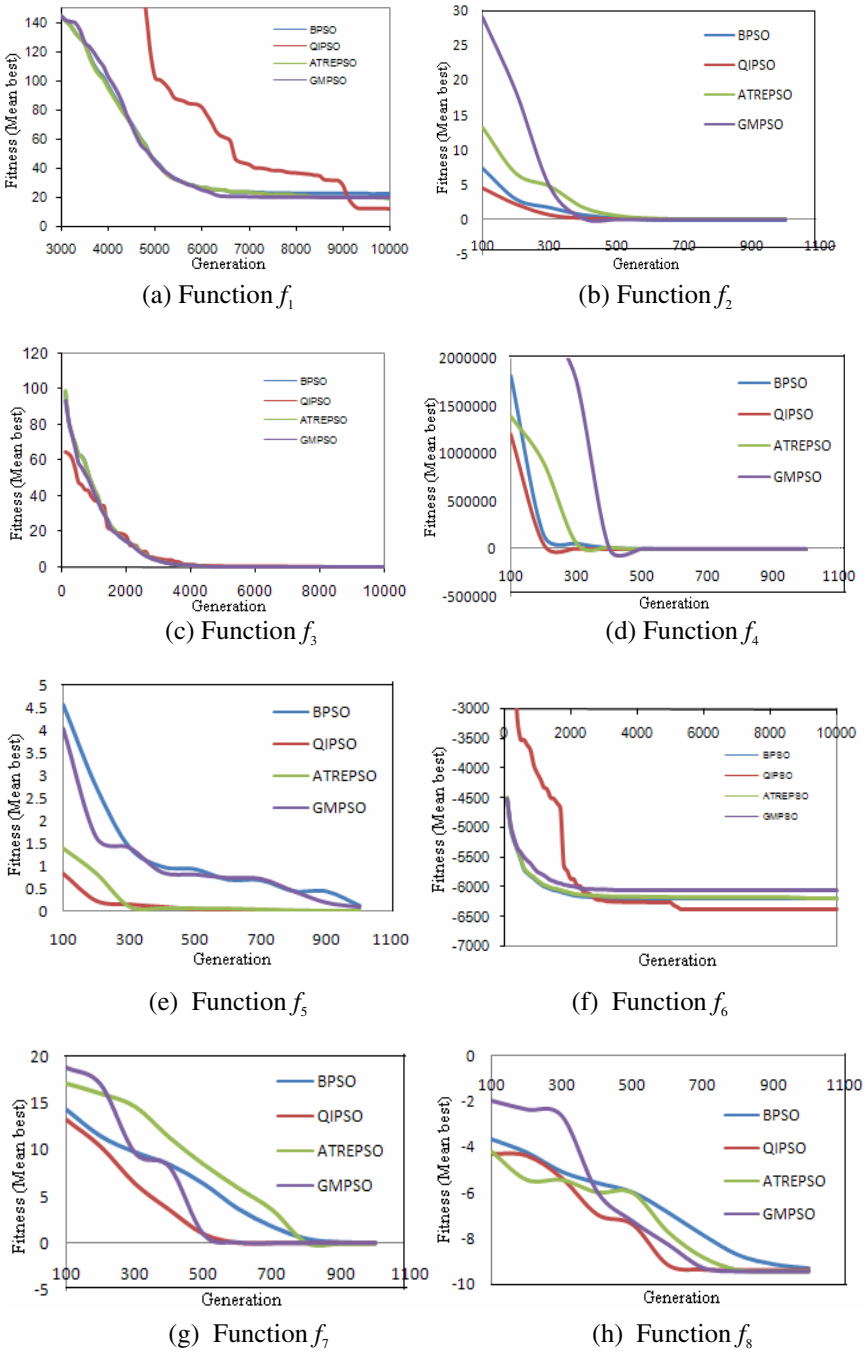


(i) Function f_9



(j) Function f_{10}

Fig. 5 (*continued*)



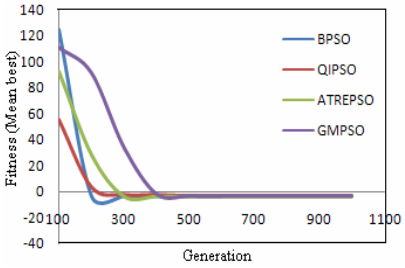
(a) Function f_1 (b) Function f_2

(c) Function f_3 (d) Function f_4

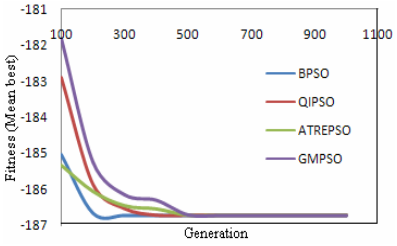
(e) Function f_5 (f) Function f_6

(g) Function f_7 (h) Function f_8

Fig. 6 Performance curves for BPSO, QIPSO, ATREPSO and GMPSO

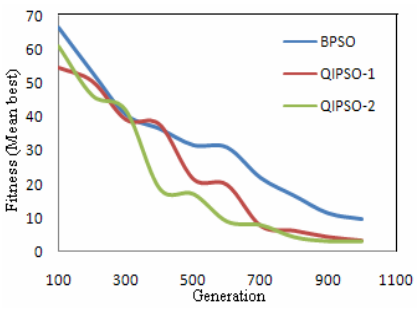


(i) Function f_9

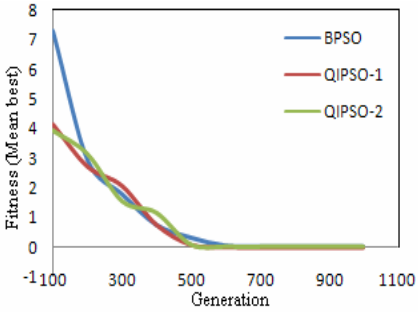


(j) Function f_{10}

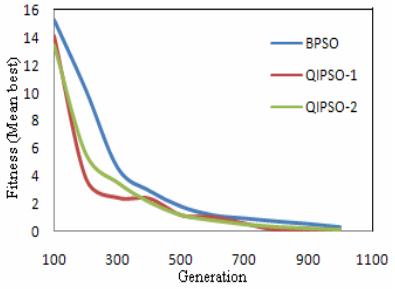
Fig. 6 (*continued*)



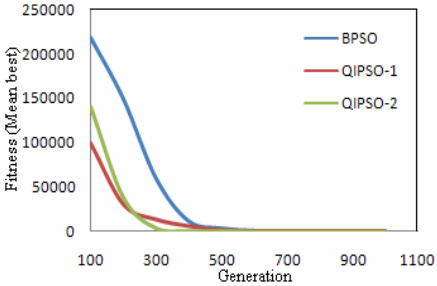
(a) Function f_1



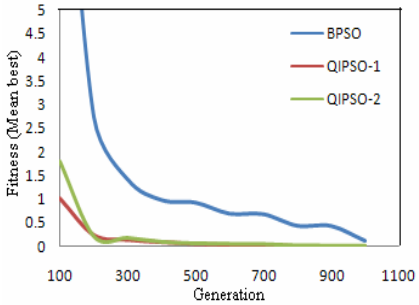
(b) Function f_2



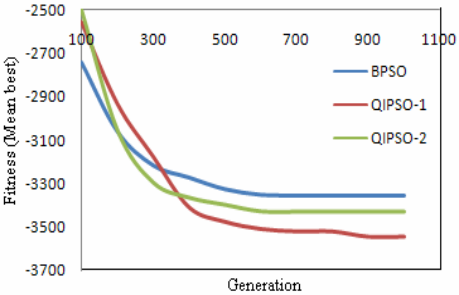
(c) Function f_3



(d) Function f_4

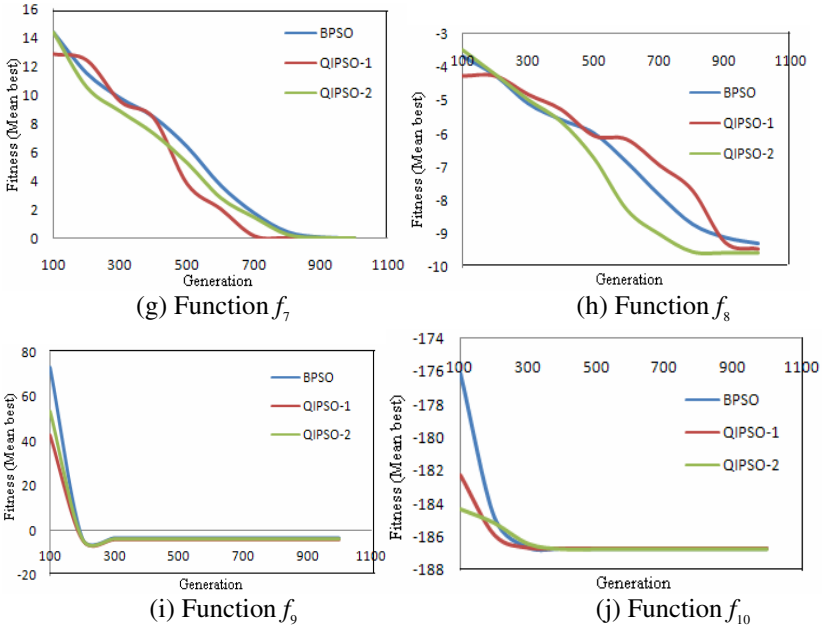


(e) Function f_5



(f) Function f_6

Fig. 7 Performance curves for BPSO, QIPSO-1and QIPSO-2



(g) Function f_7 (h) Function f_8

(i) Function f_9 (j) Function f_{10}

Fig. 7 (*continued*)

Table 6 Comparison results of real life problems (BPSO, GPSO, EPSO and LNPSO)

Gear Train Design				
Item	BPSO	GPSO	EPSO	LNPSO
x1	13	20	20	20
x2	31	13	13	13
x3	57	53	53	53
x4	49	34	34	34
f (x)	9.989333e-11	**2.331679e-11**	**2.331679e-11**	2.33168e-011
Gear Ratio	0.14429	0.14428	0.14428	0.14428
Error (%)	0.007398	0.000467	0.000467	0.000467
Transistor Modeling				
Item	BPSO	GPSO	EPSO	LNPSO
x1	0.901019	0.901241	0.901279	0.90097
x2	0.88419	0.883919	0.888237	0.880522
x3	4.038604	3.756517	3.854668	3.94582
x4	4.148831	3.861717	3.986954	4.081
x5	5.243638	5.387461	5.338548	5.28292
x6	9.932639	10.551659	10.410919	9.95503
x7	0.100944	0.26037	0.091619	0.221577
x8	1.05991	1.077294	1.083181	1.05418
x9	0.80668	0.764622	0.752615	0.825799
f(x)	0.069569	**0.058406**	0.05974	0.06292

Table 7 Comparison results of real life problems (BPSO, VC-PSO and SO-PSO)

Gear Train Design			
Item	BPSO	VC-PSO	SO-PSO
x1	13	16	16
x2	31	19	19
x3	57	49	49
x4	49	43	43
f (x)	9.989333e-11	**2.782895e-12**	2.7829e-012
Gear Ratio	0.14429	0.14428	0.14428
Error (%)	0.007398	0.000467	0.000467
Transistor Modeling			
Item	BPSO	VC-PSO	SO-PSO
x1	0.901019	0.900433	0.901031
x2	0.88419	0.52244	0.885679
x3	4.038604	1.07644	4.05936
x4	4.148831	1.949464	4.17284
x5	5.243638	7.853698	5.23002
x6	9.932639	8.836444	9.88428
x7	0.100944	4.771224	0.025906
x8	1.05991	1.007446	1.06251
x9	0.80668	1.854541	0.802467
f(x)	0.069569	**0.011314**	0.067349

Table 8 Comparison results of real life problems (BPSO, QIPSO, ATREPSO and GMPSO)

Gear Train Design				
Item	**BPSO**	**QIPSO**	**ATREPSO**	**GMPSO**
x1	13	15	19	19
x2	31	26	16	16
x3	57	51	43	43
x4	49	53	49	49
f (x)	9.98e-11	**2.33e-11**	2.78e-12	2.78e-12
Gear Ratio	0.14429	0.14428	0.14428	0.14428
Error (%)	0.007398	0.000467	0.000467	0.000467
Transistor Modeling				
Item	**BPSO**	**QIPSO**	**ATREPSO**	**GMPSO**
x1	0.901019	0.90104	0.900984	0.90167
x2	0.88419	0.884447	0.886509	0.877089
x3	4.038604	4.004119	4.09284	3.532352
x4	4.148831	4.123703	4.201832	3.672409
x5	5.243638	5.257661	5.214615	5.512315
x6	9.932639	9.997876	9.981726	10.80285
x7	0.100944	0.096078	5.69e-06	0.56264
x8	1.05991	1.062317	1.061709	1.074696
x9	0.80668	0.796956	0.772014	0.796591
f(x)	0.069569	0.066326	0.066282	**0.065762**

Table 9 Comparison results of real life problems (BPSO, QIPSO-1 and QIPSO-2)

Gear Train Design			
Item	BPSO	QIPSO-1	QIPSO-2
x1	13	19	13
x2	31	16	20
x3	57	43	34
x4	49	49	53
f (x)	9.989333e-11	**2.7829e-012**	2.33168e-011
Gear Ratio	0.14429	0.14428	0.14428
Error (%)	0.007398	0.000467	0.000467
Transistor Modeling			
Item	BPSO	QIPSO-1	QIPSO-2
x1	0.901019	0.901952	0.90107
x2	0.88419	0.895188	0.653572
x3	4.038604	3.66753	1.42074
x4	4.148831	3.67355	2.0913
x5	5.243638	5.44219	7.29961
x6	9.932639	11.2697	10.00
x7	0.100944	0.097903	4.09852
x8	1.05991	1.10537	1.00974
x9	0.80668	0.679967	1.59885
f(x)	0.069569	0.061881	**0.0514062**

Numerical results of real life problems are given in Tables 6 – 9. From these Tables, it is very difficult to claim the superiority of a particular algorithm over the others because the optimum function value obtained by all the algorithms is more or less similar. Although in some cases modified algorithms gave slightly better results than the basic PSO. This is probably due to the fact that both the real life problems, though nonlinear in nature, are small in size and do not pose any severe challenge for an optimization algorithm.

7 Conclusions

This article presents some recent simple and modified versions PSO. The algorithms considered may be divided into two classes; (1) algorithms without having any special operator but simply changing the initial configuration of the swarm and (2) algorithms having some special operator .

In all nine modified versions of PSO are presented in this chapter. These are:

- Gaussian Particle Swarm Optimization (GPSO)
- Exponential Particle Swarm Optimization (EPSO)

- Lognormal Particle Swarm Optimization (LNPSO)
- Sobol Particle Swarm Optimization (SOPSO)
- Van der Corput Particle Swarm Optimization (VCPSO)
- Attraction and Repulsion Particle Swarm Optimization (ATREPSO)
- Gaussian Mutation Particle Swarm Optimization (GMPSO)
- Quadratic Interpolation Particle Swarm Optimization (QIPSO)

The first five algorithms namely GPSO, EPSO, LNPSO, SOPSO and VCPSO described in the chapter use different initialization schemes for generating the swarm population. These schemes include Gaussian, exponential and lognormal probability distributions and quasi random sequences Sobol and Vander Corput to initialize the swarm. As expected, PSO algorithms initialized with quasi random sequences performed much better than the PSO initiated with the usual computer generated random numbers having uniform distribution (Please also see Table 3). However the interesting part of the study is that PSO initiated with Gaussian, exponential and lognormal distribution improved its performance quite significantly (Please also see Table 2).

The second part of the research consisted of modified PSO versions assisted with special operators like repulsion, mutation and crossover. In this part three algorithms called ATREPSO, GMPSO and QIPSO are given. The QIPSO is further modified into two versions QIPSO1 and QIPSO2. The common feature of these operator assisted PSO algorithms is that they all use diversity as guide to implement the operators. All the nine algorithms are applied on ten standard benchmark problems and two real life problems. The results obtained by these algorithms on the ten benchmark problems and two real life problems were either superior or at par with the basic PSO having uniform probability distribution to initialize the swarm. We did not compare the algorithms without any special operator with the ones having special operator because it will not be a fair comparison. However, among other algorithms PSO assisted with crossover operator QIPSO and its versions gave the best results. The present study may further be extended to solve the constrained optimization problems. Another interesting thing will be to combine PSO algorithms having different initialization scheme with PSO assisted with some special operator.

References

1. Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. In: IEEE International Conference on Neural Networks (Perth, Australia), IEEE Service Center, Piscataway, NJ, pg. IV, pp. 1942–1948 (1995)
2. Angeline, P.J.: Evolutionary Optimization versus Particle Swarm Optimization: Philosophy and Performance Difference. In: The 7th Annual Conference on Evolutionary Programming, San Diego, USA (1998)
3. Vesterstrom, J., Thomsen, R.: A Comparative study of Differential Evolution, Particle Swarm optimization, and Evolutionary Algorithms on Numerical Benchmark Problems. In: Proc. IEEE Congr. Evolutionary Computation, Portland, OR, June 20-23, pp. 1980–1987 (2004)

4. Vesterstrøm, J.S., Riget, J., Krink, T.: Division of Labor in Particle Swarm Optimisation. In: Proceedings of the Fourth Congress on Evolutionary Computation (CEC 2002), vol. 2, pp. 1570–1575 (2002)
5. Liu, H., Abraham, A., Zhang, W.: A Fuzzy Adaptive Turbulent Particle Swarm Optimization. International Journal of Innovative Computing and Applications 1(1), 39–47 (2007)
6. Shi, Y., Eberhart, R.C.: A Modified Particle Swarm Optimizer. In: Proc. IEEE Congr. Evolutionary Computation, pp. 69–73 (1998)
7. Eberhart, R.C., Shi, Y.: Particle Swarm Optimization: Developments, Applications and Resources. In: Proc. IEEE Congr. Evolutionary Computation, vol. 1, pp. 27–30 (2001)
8. Clerc, M.: The Swarm and the Queen: Towards a Deterministic and adaptive Particle Swarm Optimization. In: Proc. of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 1951–1957 (1999)
9. Kennedy, J.: Small Worlds and Mega-Minds: Effects of Neighborhood Topology on Particle Swarm Performance. In: Proc. of the IEEE Congress on Evolutionary Computation, vol. 3, pp. 1931–1938 (1999)
10. Poli, R., Langdon, W.B., Holland, O.: Extending Particle Swarm Optimization via Genetic Programming. In: Keijzer, M., Tettamanzi, A.G.B., Collet, P., van Hemert, J., Tomassini, M. (eds.) EuroGP 2005. LNCS, vol. 3447, pp. 291–300. Springer, Heidelberg (2005)
11. Ting, T.-O., Rao, M.V.C., Loo, C.K., Ngu, S.-S.: A New Class of Operators to Accelerate Particle Swarm Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. (4), pp. 2406–2410 (2003)
12. Paquet, U., Engelbrecht, A.P.: A New Particle Swarm Optimizer for Linearly Constrained Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, vol. (1), pp. 227–233 (2003)
13. Parsopoulos, K.E., Plagianakos, V.P., Magoulus, G.D., Vrahatis, M.N.: Objective Function "Strectching" to Alleviate Convergence to Local Minima. Nonlinear Analysis, Theory, Methods and Applications 47(5), 3419–3424 (2001)
14. Grosan, C., Abraham, A., Nicoara, M.: Search Optimization Using Hybrid Particle Sub-Swarms and Evolutionary Algorithms. International Journal of Simulation Systems, Science & Technology, UK 6(10&11), 60–79 (2005)
15. Gehlhaar, Fogel: Tuning Evolutionary programming for conformationally flexible molecular docking. In: Proceedings of the fifth Annual Conference on Evolutionary Programming, pp. 419–429 (1996)
16. Pant, M., Radha, T., Singh, V.P.: Particle Swarm Optimization: Experimenting the Distributions of Random Numbers. In: 3rd Indian Int. Conf. on Artificial Intelligence (IICAI 2007), India, pp. 412–420 (2007)
17. Krohling, R.A., Coelho, L.S.: PSO-E: Particle Swarm with Exponential Distribution. In: IEEE Congress on Evolutionary Computation, Canada, pp. 1428–1433 (2006)
18. Krohling, R.A., Swarm, G.: A Novel Particle Swarm Optimization Algorithm. In: Proc. of the 2004 IEEE Conference on Cybernetics and Intelligent Systems, Singapore, pp. 372–376 (2004)
19. Pant, M., Thangaraj, R., Abraham, A.: Improved Particle Swarm Optimization with Low-discrepancy Sequences. In: IEEE Cong. on Evolutionary Computation (CEC 2008), Hong Kong (accepted, 2008)
20. Kimura, S., Matsumura, K.: Genetic Algorithms using low discrepancy sequences. In: Proc of GEECO 2005, pp. 1341–1346 (2005)

21. Nguyen, X.H., Nguyen, Q.U., Mckay, R.I., Tuan, P.M.: Initializing PSO with Randomized Low-Discrepancy Sequences: The Comparative Results. In: Proc. of IEEE Congress on Evolutionary Algorithms, pp. 1985–1992 (2007)
22. Parsopoulos, K.E., Vrahatis, M.N.: Particle Swarm Optimization in noisy and continuously changing environments. In: Proceedings of International Conference on Artificial Intelligence and soft computing, pp. 289–294 (2002)
23. Brits, R., Engelbrecht, A.P., van den Bergh, F.: A niching Particle Swarm Optimizater. In: Proceedings of the fourth Asia Pacific Conference on Simulated Evolution and learning, pp. 692–696 (2002)
24. Brits, R., Engelbrecht, A.P., van den Bergh, F.: Solving systems of unconstrained Equations using Particle Swarm Optimization. In: Proceedings of the IEEE Conference on Systems, Man and Cybernetics, vol. 3, pp. 102–107 (2002)
25. Chi, H.M., Beerli, P., Evans, D.W., Mascagni, M.: On the Scrambled Sobol Sequence. In: Sunderam, V.S., van Albada, G.D., Sloot, P.M.A., Dongarra, J. (eds.) ICCS 2005. LNCS, vol. 3516, pp. 775–782. Springer, Heidelberg (2005)
26. Engelbrecht, A.P.: Fundamentals of Computational Swarm Intelligence. John Wiley & Sons Ltd., Chichester (2005)
27. Pant, M., Radha, T., Singh, V.P.: A Simple Diversity Guided Particle Swarm Optimization. In: IEEE Cong. on Evolutionary Computation (CEC 2007), Singapore, pp. 3294–3299 (2007)
28. Riget, J., Vesterstrom, J.S.: A diversity-guided particle swarm optimizer – the arPSO. Technical report, EVAlife, Dept. of Computer Science, University of Aarhus, Denmark (2002)
29. Pant, M., Radha, T., Singh, V.P.: A New Diversity Based Particle Swarm Optimization using Gaussian Mutation. Int. J. of Mathematical Modeling, Simulation and Applications (accepted)
30. Pant, M., Thangaraj, R.: A New Particle Swarm Optimization with Quadratic Crossover. In: Int. Conf. on Advanced Computing and Communications (ADCOM 2007), India, pp. 81–86. IEEE Computer Society Press, Los Alamitos (2007)
31. Pant, M., Thangaraj, R., Abraham, A.: A New Particle Swarm Optimization Algorithm Incorporating Reproduction Operator for Solving Global Optimization Problems. In: 7th International Conference on Hybrid Intelligent Systems, Kaiserslautern, Germany, pp. 144–149. IEEE Computer Society press, USA (2007)
32. Millie Pant, T., Pant, M., Radha, T., Singh, V.P.: A New Particle Swarm Optimization with Quadratic Interpolation. In: Int. Conf. on Computational Intelligence and Multimedia Applications (ICCIMA 2007), India, vol. 1, pp. 55–60. IEEE Computer Society Press, Los Alamitos (2007)
33. Kannan, B.K., Kramer, S.N.: An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design. J. of Mechanical Design, 116/405 (1994)
34. Sandgren, E.: Nonlinear Integer and Discrete Programming in Mechanical Design. In: Proc. of the ASME Design Technology Conference, Kissimme, Fl, pp. 95–105 (1988)
35. Price, W.L.: A Controlled Random Search Procedure for Global Optimization. In: Dixon, L.C.W., Szego, G.P. (eds.) Towards Global Optimization 2, vol. X, pp. 71–84. North Holland Publishing Company, Amsterdam (1978)
36. Secrest, B.R., lamont, G.B.: Visualizing Particle Swarm Optimization – Gaussian Particle Swarm Optimization. In: Proc. of IEEE Swarm Intelligence Symposium, pp. 198–204 (2003)

37. Stacey, A., Jancic, M., Grundy, I.: Particle Swarm Optimization with Mutation. In: Proc. of the IEEE Congress on Evolutionary Computation, vol. 2, pp. 1425–1430 (2003)
38. van der Bergh, F.: An Analysis of Particle Swarm Optimizers. PhD thesis, Department of Computer Science, University of Pretoria, Pretoria, South Africa (2002)
39. van der Bergh, F., Engelbrecht, A.P.: A New Locally Convergent Particle Swarm Optimizer. In: Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics, pp. 96–101 (2002)
40. Xie, X., Zhang, W., Yang, Z.: A Dissipative Particle Swarm Optimization. In: Proc. of the IEEE Congress on Evolutionary Computation, vol. 2, pp. 1456–1461 (2002)
41. Higashi, H., Iba, H.: Particle Swarm Optimization with Gaussian Mutation. In: Proc. of IEEE Swarm Intelligence Symposium, pp. 72–79 (2003)
42. Yao, X., Liu, Y.: Fast Evolutionary Programming. In: Fogel, L.J., Angeline, P.J., Back, T.B. (eds.) Proc. of the 5th Annual Conf. Evolutionary Programming, pp. 451–460 (1996)
43. Yao, X., Liu, Y., Lin, G.: Evolutionary Programming made faster. IEEE Trans. On Evolutionary Computation 3(2), 82–102 (1999)
44. Ting, T.-O., Rao, M.V.C., Loo, C.K., Ngu, S.-S.: A new Class of Operators to accelerate Particle Swarm optimization. In: Proceedings of IEEE Congress on Evolutionary Computation, vol. 4(656), pp. 2406–2410 (2003)
45. Clerc, M.: Think Locally, Act Locally: The way of Life of Cheap-PSO, an Adaptive PSO. Technical report (2001), http://clerc.maurice.free.fr/PSO/
46. Rigit, J., Vesterstorm, J.S.: Controlling Diversity in Particle Swarm Optimization. Master's thesis, University of Aahrus, Denmark (487) (2002)
47. Rigit, J., Vesterstorm, J.S.: Particle Swarms: Extensions for improved local, multi modal, and dynamic search in Numerical optimization. Masters thesis, department of Computer Science, University of Aahrus (620) (2002)
48. Brits, R.: Niching Strategies for Particle swarm optimization. Masters thesis, Department of Computer Science, university of Pretoria (67) (2002)
49. Brits, R.E., Van den Bergh, F.: Solving unconstrained equations using Particle Swarm Optimization. In: Proceedings of the IEEE congress on systems, man and cybernetics, vol. 3(70), pp. 102–107 (2002)