NATURE INSPIRED ALGORITHMS AND THEIR APPLICATIONS

A THESIS

Submitted in partial fulfilment of the equirements for the award of the degree

of

DOCTOR OF PHILOSOPHY

in

PAPER TECHNOLOGY

by

RADHA T

DEPARTMENT OF PAPER TECHNOLOGY
INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE - 247 667 (INDIA)

SEPTEMBER, 2009

INDIAN INSTITUTE OF TECHNOLOGY ROORKEE
ROORKEE

CANDIDATE'S DECLARATION

I hereby certify that the work which is being presented in this thesis entitled **NATURE INSPIRED ALGORITHMS AND THEIR APPLICATIONS** in partial fulfilment of the requirements for the award of *the Degree of Doctor of Philosophy* and submitted in the Department of Paper Technology of Indian Institute of Technology Roorkee, Roorkee is an authentic record of my own work carried out during a period from January 2007 to September 2009 under the supervision of Dr. Millie Pant, Asst. Professor and Dr. V. P. Singh, Professor, Department of Paper Technology, Indian Institute of Technology Roorkee, Roorkee.

The matter presented in this thesis has not been submitted by me for the award of any other degree of this or any other Institution.

(RADHA T)

This is to certify that the above statement made by the candidate is correct to the best of our knowledge.

Date: September 24, 2009

(V. P. Singh)
Supervisor

(Millie Pant)
Supervisor

The Ph.D. Viva-Voce Examination of ***Mrs. Radha T***, Research Scholar, has been held on................

Signature of Supervisors

Signature of External Examiner

ABSTRACT

Optimization problems arise in various disciplines such as Engineering Designs, Agricultural Sciences, Manufacturing Systems, Economics, Physical Sciences, Pattern Recognition etc. In fact newly developed computational techniques are being used in every sphere of human activity where decisions are to be taken in some complex situations that can be represented by a mathematical model. In view of their practical utility there is a need for efficient and robust computational algorithms, which can numerically solve on computers mathematical models of medium as well as large size optimization problems arising in different fields.

Mathematical models of real life problems often turn out to be nonlinear in nature. Such problems may have local as well as global optimal solutions. Global optimal solution is usually more difficult to obtain, as compared to local optimal solution, but in many cases it is advantageous and sometimes even necessary, to search for the global optimal solution.

In order to determine the global optimal solution of a nonlinear optimization problem, usually two approaches are followed: (1) The Deterministic Approach and (2) The Probabilistic Approach. Deterministic methods extensively use analytical properties such as continuity, convexity, differentiability etc. of the objective and the constraints to locate a neighborhood of the global optimum. It is now established that deterministic methods are best suited for restricted classes of functions, namely convex functions, one dimensional rational or Lipchitz functions and Polynomials etc. whose mathematical properties can be easily determined and utilized at specified points or in specified intervals. The stochastic methods, on the other hand, utilize randomness in an efficient way to explore the set over which the objective function is to be optimized contrary to general expectation. Stochastic methods perform well in the case of the most of the realistic problems over which these have been tried. In stochastic or probabilistic methods, two phases are generally employed. In the first phase, also called global phase, the function is evaluated at a number of randomly sampled points. In the second phase, also called local phase, these points are manipulated by local searches to yield a possible candidate for a global optima. 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 of function evaluations alone and do not assume any mathematical properties of the functions being used.

Nature Inspired Algorithms (NIA) are relatively a newer addition to class of population based stochastic search techniques. These algorithms are based on the evolutionary, self organising and collective processes in nature for example the concepts of natural evolution like selection, reproduction and mutation form the key ingredients of certain NIA whereas the socio-cooperative behaviour displayed by natural species like birds, ants, termites and bees (and also human behaviour) form basis of some other NIA. These algorithms seem promising because of their social – cooperative approach and because of their ability to adapt themselves in the continuously changing environment.

Some popular NIA include Genetic Algorithms (GA), Ant Colony Optimization (ACO), Evolutionary Programming (EP), Evolutionary Strategies (ES), Particle Swarm Optimization (PSO), Differential Evolution (DE) etc. These algorithms have been successfully applied to several bench mark and real life problems arising in various fields of Science and Engineering. Although, these algorithms give a better performance than the classical optimization techniques in most of the cases, it has been observed that most of the stochastic algorithms have certain drawbacks like slow or premature convergence (resulting in inferior solution), inability to locate a global optima or getting stuck in a local optima. These problems become more persistent in case of multimodal problems i.e. problems with several local and global optima or noisy functions with shifting optima (dynamic optimization problems), i.e. to say problems in which global optima is not static but keeps on changing with time.

Keeping in mind the shortcomings of the existing techniques, algorithms are developed/ modified in this Thesis which not only keep a balance between the two antagonist factors; exploration and exploitation (thereby maintaining diversity of the population and preventing premature convergence), but are also be efficient in terms of computational time.

In the present study the two stochastic population based search algorithms namely, PSO and DE are considered because of their popularity and wide applicability. In the past few years, these two techniques have emerged as powerful optimization tools for solving complex optimization problems, which are otherwise difficult to solve by the usual classical methods.

Both PSO and DE have been successfully applied to a wide range of problems including benchmark and real life problems.

The objectives of this thesis are:

(1) To design efficient and reliable computational techniques based on DE and PSO algorithms for obtaining the global optimal solution of constrained / unconstrained nonlinear optimization problems.

(2) To test the proposed algorithms on test problems appearing in literature.

(3) To apply the algorithms for solving real life problems arising in various fields of Science and Engineering.

Different aspects of PSO and DE algorithms like initialization of population, effect of diversity and inclusion of evolutionary operators (for PSO), modifications in the existing operators (for DE) and hybridization of algorithms are considered in this Thesis and several modifications are suggested for the improvement of these two algorithms in terms of convergence rate without compromising with the solution quality. Finally the algorithms are validated on a set of several test and real life problems.

The Thesis is divided into ten chapters.

The first chapter is introductory in nature in which definitions and literature review are presented.

In the second chapter, various analyses are done on the generation of initial population for the PSO and DE algorithms. Uniformly distributed random numbers are generally used for the initialization of population in population based search algorithms. However in the second chapter, the initial population for DE and PSO algorithms is initiated using various probability distributions and low discrepancy sequences. For this investigation, the probability distributions namely: Gaussian (G), Exponential (E), Beta (BT) and Gamma (GA) distributions and the low discrepancy sequences namely: Van der Corput (VC) sequence and Sobol (S) sequence used. Based on the above mentioned distributions and low discrepancy sequences, the following 12 modifications are proposed viz. VC-PSO, SO-PSO, GPSO, EPSO, BTPSO, GAPSO, VC-DE, SO-DE, GDE, EDE, BTDE and GADE. The proposed algorithms are tested on standard benchmark problems and the results are compared with the basic versions of PSO and DE which follows the uniform distribution for initializing the swarm. The simulation results show that a significant improvement can be made in the performance of PSO and DE, by simply changing the distribution of random numbers to other than uniform distribution as the proposed algorithms outperform the basic versions by a noticeable percentage. In an overall comparison, the algorithms which follow the Beta distribution and Van der Corput sequence are superior to other algorithms.

Chapter 3 deals exclusively with the possible modifications in the PSO algorithm in order to improve its performance. For this purpose other evolutionary operators like mutation and crossover are added to the basic PSO algorithm and the results are recorded. A quantum behaved PSO is also designed in this chapter. In all, chapter 3 consists of several modified versions of basic PSO algorithm. Some of these versions are: ATREPSO, GMPSO, BMPSO, GAMPSO, BGMPSO, QIPSO1, QIPSO2, QIPSO3, QIPSO4, SMPSO1, SMPSO2, GWPSO+UD, MPSO, Q-QPSO1, Q-QPSO2, SMQPSO1 and SMQPSO2. The proposed algorithms are tested on standard benchmark problems. The results obtained by these algorithms on all benchmark problems are either superior or at par with the basic PSO algorithm. In an overall comparison, the improved PSO algorithms assisted with Quadratic Interpolation operator (QIPSOs and Q-QPSOs) algorithms gave the best results.

Chapter 4 is devoted to the modifications for the DE algorithm. Two new mutant vectors based on the Laplace probability distribution (LDE) and the using the concept of Quadratic Interpolation (DE-QI) are proposed. Five versions of LDE are suggested namely LDE1, LDE2, LDE3, LDE4 and LDE5. Also, an improved version of DE with adaptive control parameters (ACDE) is proposed. The performance of all the proposed algorithms is validated on a set of test problems and the numerical results are compared with basic DE and with two other versions of DE. The numerical results show that the proposed algorithms help in improving the convergence rate up to 50% in comparison to the basic DE and at the same time maintained a good success rate as well. In comparison among all the proposed algorithms, LDE4 and DE-QI are superior with others.

Chapter 5 deals with the concept of hybridization of algorithms which is a class of modified algorithms consisting of the integration of two or more algorithms. Three hybrid global optimization algorithms namely DE-PSO, MDE and AMPSO algorithms are proposed. The performance of proposed algorithms are validated on a set of benchmark problems and the numerical results are compared with classical DE, PSO, EP and 5 other variants of DE, PSO and EP available in the literature. The numerical results show that the proposed algorithms are either superior or at par with all the compared algorithms in terms of convergence rate and solution quality.

In chapter 6, focus is laid on the solution of constrained optimization problems. A new constraint handling mechanism for solving constrained optimization problems is proposed. It is a simple approach for handing constraints which do not require any additional parameters. Based on the new constraint handling mechanism, two algorithms are proposed namely ICPSO and ICDE. The performance of ICPSO and ICDE algorithms are validated on 20 constrained benchmark problems and compared with two other variants (constraint) of PSO and DE in the literature. The numerical results show that the proposed algorithms are quite promising algorithms for solving constraint optimization problems.

Chapters 7, 8 and 9 are devoted to real life optimization problems. In chapter 7, the problem is to determine the In-Situ efficiency of Induction Motor without performing no-load test, which is not easily possible for the motors working in process industries where continuous operation is required. This problem is modeled as an unconstrained optimization problem and is framed by four different methods. The differences in the method are based on the number of input parameters used to the optimization algorithms and modifications in the equivalent circuit of the motor. Basic versions of PSO, DE and their 6 variants namely QPSO, ATREPSO, GMPSO, SMPSO1, LDE1 and DE-QI are used to solve this problem. The results obtained by the above mentioned family of PSO and DE algorithms are compared with Genetic Algorithm (GA) and a physical efficiency measurement method, called torque-gauge method. The performances in terms of objective function and convergence time prove the effectiveness of the proposed variants of PSO and DE algorithms used for comparison.

In chapter 8, another problem that is quite common in the field of Electrical Engineering is considered. For this problem, the objective is to compute the values of the decision variables called relays, which control the act of isolation of faulty lines from the system without disturbing the healthy lines. This problem is modeled as a nonlinear constrained optimization problem, in which the objective function to be minimized is the sum of the operating times of all the relays, which are expected to operate in order to clear the faults of their corresponding zones. Three cases of the IEEE Bus system are considered viz. IEEE 3-bus, IEEE 4-bus and IEEE 6-bus system. It is a very complex problem and many PSO versions proposed in this thesis were not able to solve it. The problem was finally solved by using the family of DE algorithms namely LDE1, LDE2, LDE3, LDE4, LDE5 and DE-QI. The results obtained by the aforesaid algorithms are compared with the earlier published results. From the numerical and graphical results, it is shown that the proposed DE algorithms used in this study are either best or comparable with the other algorithms both in terms of solution quality and convergence rate.

Chapter 9 consists of 12 small real life problems taken from different fields.

The Thesis finally completes with Chapter 10, where conclusions based on the present work are drawn and suggestions for future work are made.

ACKNOWLEDGEMENTS

I wish to acknowledge my deep sense of gratitude and indebtedness to Dr. Millie Pant and Prof. V. P. Singh, Department of Paper Technology, IIT Roorkee, their invaluable guidance, sincere advice and encouragement throughout the completion of this research work. I feel highly privileged to have worked under them during the course of this work.

Their encouraging support and thorough involvement in formulation of the research work and preparation of manuscript are gratefully acknowledged. I sincerely appreciate their pronounced individualities, humanistic and warm personal approach, which has given me strength to carry out this research work smoothly. I humbly acknowledge a lifetime's gratitude to them.

My heartily gratitude to Prof. J. S. Upadhyaya, Head of Department and Prof. A. K. Ray, Department of Paper Technology and Dr. Kusum Deep, Department of Mathematics, IIT Roorkee, whose humanistic and warm personal approach always helped me from beginning to end of my work.

I express my deep sense of gratitude to Prof. Ajith Abraham, Norwegian University of Science and Technology, Norway for his valuable discussions, suggestions and encouragements during the work.

I thank my friends Abitha, Geetha, Subbu, Kavitha, Bibin and Mussrat Ali for their positive attitude and cooperation always encouraged me to accomplish my research work.

My special, sincere, heartfelt gratitude and indebtedness are due to my parents for their sincere prayers, and constant encouragement. I appreciate and feel proud to always have helping hands and patience from my husband Thanga Raj and my daughter Harshini. I am proud to humbly dedicate this research work to my family.

Acknowledge Him in all Thy ways and He shall direct Thy paths.

(RADHA T)

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

ACK	Ackley's path function
ALP	Alphine function
APH	Axis parallel hyperellipsoid
BR	Branin function
CB6	Six hump camel back function
CLV	Colvillie function
DE	Differential Evolution
DeJ	Dejong's function
DeJ1	Dejong's function1 (no noise)
DeJ-N	Dejong's function with noise
EP	Evolutionary Programming
g01	Constrained problem 1
g02	Constrained problem 2
g03	Constrained problem 3
g04	Constrained problem 4
g05	Constrained problem 5
g06	Constrained problem 6
g07	Constrained problem 7
g08	Constrained problem 8
g09	Constrained problem 9
g10	Constrained problem 10
g11	Constrained problem 11
g12	Constrained problem 12
g13	Constrained problem 13
g14	Constrained problem 14
g15	Constrained problem 15
g16	Constrained problem 16
g17	Constrained problem 17

g18	Constrained problem 18
g19	Constrained problem 19
g20	Constrained problem 20
GP	Goldstein and price problem
GP1	Generalized penalized function 1
GP2	Generalized penalized function 2
GR	Griewank function
HM1	Hartmann function 1
HM2	Hartmann function 2
LM	Levy and Mantalvo function
MC	Mccormic function
MH	Modified Himmelblau function
Mic	Michalewicz function
MT	Matyas function
PSO/BPSO	Basic Particle Swarm Optimization
RB	Rosenbrock function
RS	Ratringin function
SB1	Shubert function 1
SB2	Shubert function 2
SDP	Sum of different power
SH6	Shaffer's function 6
SH7	Shaffer's function 7
SK	Shekel's Problem
ST	Step function
SWF	Schwefel function
SWF1.2	Schwefel's function 1.2
SWF2.21	Schwefel's function 2.21
SWF2.22	Schwefel's function 2.22
T2N	Test2N function
ZK	Zhakarov function

Chapter 1

Introduction

[The present chapter is introductory in nature. It gives the basic definitions relevant to the present study. It also provides a brief literature review concerning the Particle Swarm Optimization and Differential Evolution techniques, which forms the basis of the present work. The chapter ends with an outline of the work done in this thesis]

1.1 Optimization

Optimization is ubiquitous and spontaneous process that forms an integral part of our day-to-day life. In the most basic sense, it can be defined as an art of selecting the best alternative among a given set of options. Optimization problems arise in various disciplines such as engineering designs, agricultural sciences, manufacturing systems, economics, physical sciences, pattern recognition etc. In fact optimization techniques are being extensively used in various spheres of human activities, where decisions have to be taken in some complex situation which can be represented by mathematical models. Optimization can thus be viewed as a kind of decision making, or more specifically, as one of the major quantitative tools in network of decision making, in which decisions have to be taken to optimize one or more objectives in some prescribed set of circumstances. In view of their practical utility there is a need for efficient and robust computational algorithms, which can numerically solve on computers mathematical models of medium as well as large size optimization problem arising in different fields.

1.2 Definition of an Optimization Problem

An optimization problem consists of three main components; (i) an objective function, (ii) decision variables and (iii) a set of constraints. The function to be optimized could be linear or non-linear, fractional or geometric etc. Sometimes, even the explicit mathematical formulation of the function may not be available. Often the function is to be optimized in a prescribed domain which is specified by a number of constraints in the form of inequalities and equalities. The process of optimization addresses the problem of determining those values of the independent variables which do not violate the constraints and at the same time give an optimal value of the function being optimized.

The general non-linear optimization problem is defined as:

Minimize / Maximize $f(\bar{x})$, where $f: R^n \to R$

Subject to: $x \in S \subset R^n$

where S is defined by:

$$g_j(\bar{x}) \le 0,\ j = 1,2,...,p$$

$$h_k(\bar{x}) = 0,\ k = 1,2,...,q$$

$$a_i \le x_i \le b_i\ (i = 1,......,n).$$

p and q are the number of inequality and equality constraints respectively, a_i and b_i are lower and upper bounds of the decision variable x_i.

Any vector x satisfying all the above constraints is called feasible solution. The best of the feasible solution is called an optimal solution. If the objective function and all constraints are linear then the model is called Linear Programming Problem (LPP). If the solution has an additional requirement that the decision variables are integers then the model is called Integer Programming Problem (IPP). If some variables are integers and other variables are real then the problem is called Mixed Integer Programming Problem (MIPP). If the objective function and/or constraints are nonlinear then the problem is called Non-Linear Programming Problem (NLPP) (Kapur, 1996; Bector, 2005).

1.3 Local and Global Optimal Solutions

For a minimization problem, a feasible solution x^* is said to global minima of the problem if $f(x^*) \le f(x)$ for all $x \in S$. If $f(x^*) \le f(x)$ for all $x \in S \cap N_z(x^*)$, where $N_z(x^*)$ is called a ε neighborhood of x^*, then x^* is called local minima. A point x^* is a stationary point if the derivative of the function $f(x)$ is zero at x^*.

An optimization problem may have no optimal solution, only one optimal solution or more than one optimal solution. If the problem has a unique local optimal solution, then it is also the global optimal solution. However, if the problem has more than one local optimal solution, then one or more of these may be global optimal solutions. In a LPP, every local optimal solution is also the global optimal solution, on the other hand in a NLPP, if the objective function (for minimization case) is convex and its domain of definition defined by the set of constraints is also convex, then the local optimal solution is generated to be global optimal solution.

In most of the NLPP, a global optimal solution rather than a local optimal solution is desired. Determining the global optimal solution of a NLPP is much more difficult than determining the local optimal solution. However, because of the practical necessity, the search for the global optima is often necessary.

For a twice-differentiable function conditions exists which can be used to find a local optimal solution. If the test fails then due to the continuous differentiability of the function a point with a lower function value can be found in its neighborhood. In this way a sequence of points converging to the local optimal can be conducted. However, such tests are not sufficient in solving global optimization problems. In a way we can say that the global optimization problem is unsolvable in a finite number of steps. It is so because for any given point it cannot be guaranteed that it is not the global minima without evaluating the function in atleast one point of every neighborhood of that point. Since the neighborhoods of a point can be unbounded so infinite numbers of steps are needed to reach the global minima.

1.4 Methods for Global Optimization

Global Optimization refers to finding the extreme value of a given nonconvex function in a certain feasible region and such problems are classified in two classes; unconstrained and constrained problems. Solving global optimization problems has made great gain from the interest in the interface between computer science and operations research.

In general, the classical optimization techniques have difficulties in dealing with global optimization problems. One of the main reasons of their failure is that they can easily be entrapped in local minima. Moreover, these techniques cannot generate or even use the global information needed to find the global minimum for a function with multiple local minima.

The interaction between computer science and optimization has yielded new practical solvers for global optimization problems, called *meta-heuristics*. The term "meta-heuristics" was first coined by Glover in 1986 (Glover, 1986). The word "meta-heuristics" contains all heuristics methods that show evidence of achieving good quality solutions for the problem of interest within an acceptable time. The structures of meta-heuristics are mainly based on simulating nature and artificial intelligence tools. Meta-heuristics mainly invoke exploration and exploitation of search procedures in order to diversify the search all over the search space and intensify the search in some promising areas. Therefore, meta-heuristics cannot easily be entrapped in local minima.

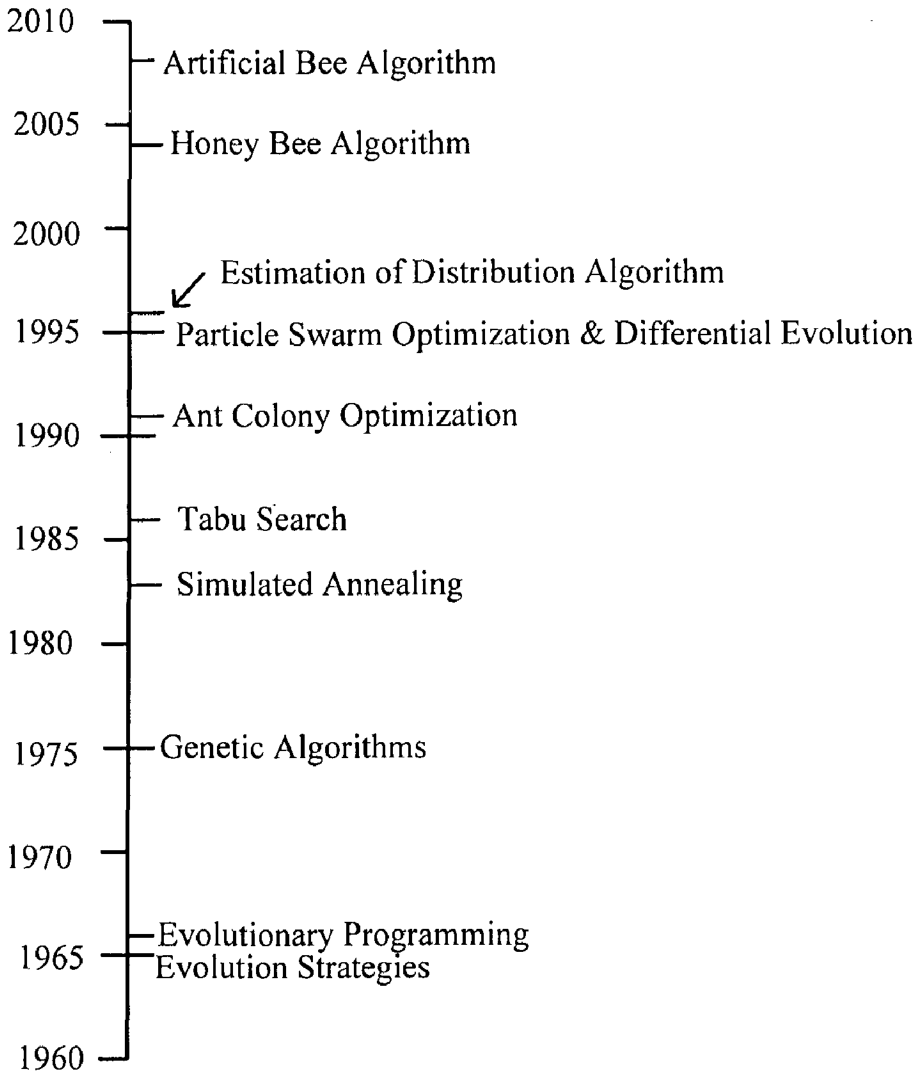

Figure 1.1 Main meta-heuristics in chronological order

Meta-heuristics can be classified into two classes; population-based methods and point-to-point methods. In the latter methods, the search invokes only one solution at the end of each iteration from which the search will start in the next iteration. On the other hand, the population-based methods invoke a set of many solutions at the end of each iteration. Simulated Annealing (Kirkpatrick, 1983) and Tabu Search (Fred, 1986) are examples of point-to-point methods. Some of the population-based methods are: Genetic Algorithm (Goldberg, 1986), Evolutionary Programming (Fogel et al, 1965), Evolution Strategies (Rochenberg, 1973), Ant Colony Optimization, Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) and Differential Evolution (DE) (Storn and Price, 1995). The present study is concentrated on two meta-heuristics namely PSO and DE.

1.5 Particle Swarm Optimization for Global Optimization

The concept of classical/basic PSO (PSO/BPSO) was first suggested by Kennedy and Eberhart (1995). Since its development, PSO has become one of the most promising optimizing techniques for solving global optimization problems. Its mechanism is inspired by the social and cooperative behavior displayed by various species like birds, fish, termites, ants and even 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. In the past several years, PSO has been successfully applied in many research and application areas. It has been demonstrated that PSO gets better results in a faster, cheaper way in comparison to other methods like Genetic algorithms, Simulated Annealing etc.

Another reason for the popularity of PSO is that there are few parameters to adjust. The main parameters of the classical PSO are Swarm Size (say S), Inertia Weight w and Acceleration Constants c_1, c_2. These control parameters are user defined and have to be carefully selected in order to make the algorithm efficient and robust. One version, with slight variations, works well in a wide variety of applications. Particle Swarm Optimization has been used for approaches that can be used across a wide range of applications, as well as for specific applications focused on a specific requirement.

It can be said that PSO algorithm is not only a tool for optimization, but also it is a tool for representing socio-cognition of human and artificial agents, based on principles of social psychology. Some scientists suggest that knowledge is optimized by social interaction and thinking is not only private but also interpersonal. PSO as an optimization tool provides a population-based search procedure in which individuals called particles change their position (state) with time. In a PSO system, particles fly around in a multidimensional search space. During flight, each particle adjusts its position according to its own experience, and according to the experience of a neighboring particle, making use of the best position encountered by itself and its neighbor. Thus, as in modern GAs and memetic algorithms, a PSO system combines local search methods with global search methods, attempting to balance exploration and exploitation.

1.6 Working of Particle Swarm Optimization

The particles or members of the swarm fly through a multidimensional search space looking for a potential solution. Each particle adjusts its position in the search space from time to time according to the flying experience of its own and of its neighbors (or colleagues).

For a D-dimensional search space the position of the i^{th} particle is represented as $X_i = (x_{i1}, x_{i2}, ..., x_{id}, ..., x_{iD})$. Each particle maintains a memory of its previous best position $P_i = (p_{i1}, p_{i2}, ..., p_{id}, ..., p_{iD})$. The best one among all the particles in the population is represented as $P_g = (p_{g1}, p_{g2}, ..., p_{gd}, ..., p_{gD})$. The velocity of each particle is represented as $V_i = (v_{i1}, v_{i2}, ..., v_{id}, ..., v_{iD})$, is clamped to a maximum velocity $V_{\max} = (v_{\max,1}, v_{\max,2}, ..., v_{\max,d}, ..., v_{\max,D})$ which is specified by the user. $V_{\max}$ determines the resolution with which regions between the present position and the target position are searched. Large values of $V_{\max}$ facilitate global exploration, while smaller values encourage local exploitation. If $V_{\max}$ is too small, the swarm may not explore sufficiently beyond locally good regions. On the other hand, too large values of $V_{\max}$ risk the possibility of missing a good region (Engelbrecht, 2005).

During each generation each particle is accelerated towards the particles previous best position and the global best position. At each iteration a new velocity value for each particle is calculated based on its current velocity, the distance from the global best position. The new velocity value is then used to calculate the next position of the particle in the search space. This process is then iterated a number of times or until a minimum error is achieved. The two basic equations which govern the working of PSO are that of velocity vector and position vector given by:

$$v_{id} = v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{1.1}$$

$$x_{id} = x_{id} + v_{id} \tag{1.2}$$

Here c_1 and c_2 are acceleration constants. They represent the weighting of the stochastic acceleration terms that pull each particle toward personal best and global best positions. Therefore, adjustment of these constants changes the amount of tension in the system. Low of these constants allow particles to roam far from the target regions before tugged back, while high values result in abrupt movement toward, or past, target regions (Eberhart and Shi, 2001). The constants r_1, r_2 are the uniformly generated random numbers in the range of [0, 1].

The first part of Eqn. (1.1) represents particle's previous velocity, which serves as a memory of the previous flight direction. This memory term can be seen as a momentum, which prevents the particle from drastically changing direction, and to bias it towards the current direction. The second part (i.e. $c_1 r_1 (p_{id} - x_{id})$) is the cognition part and it tells us about the personal experience of the particle. In a sense, this cognition part resembles individual memory of the position that was best for the particle. The effect of this term is that particles are drawn back to their own best positions, resembling the tendency of individuals to return to situations or places that most satisfied in the fast. The third part (i.e. $c_2 r_2 (p_{gd} - x_{id})$) represents the cooperation among particles and is therefore named as the social component (Kennedy, 1997). This term resembles a group norm or standard which individuals seek to attain. The effect of this term is that each particle is also drawn towards the best position found by the particle's neighborhood. Shi and Eberhart (1998) have introduced a new parameter called inertia weight in to the velocity vector equation (1.1). Then the velocity vector becomes:

$$v_{id} = \omega * v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \tag{1.3}$$

The inertia weight ω is employed to control the impact of the previous history of velocities on the current velocity, thereby influencing the trade-off between global and local exploration abilities of the particles. It can be a positive constant or even a positive linear or nonlinear function of time. A larger inertia weight facilitates global exploration while a smaller inertia weight tends to facilitate local exploration to fine-tune the current search area. Suitable selection of the inertia weight provides a balance between global and local exploration abilities and thus requires less iteration on average to find the optimum (Eberhart and Shi, 1998). Initially the inertia weight was kept static during the entire search duration for every particle and dimension. With the due course of time inertia weights with dynamic weights were introduced.

Maurice Clerc (Clerc, 1999; Clerc and Kennedy, 2002) has introduced a constriction factor K that improves PSO's ability to constrain and control velocities and Shi and Eberhart (2000) found that K, combined with constraints on V_{max}, significantly improved the PSO performance. The value of the constriction factor is calculated as a function of the cognitive and social parameters c_1 and c_2 :

$$K = \frac{2}{|2-\varphi-\sqrt{\varphi^2-4\varphi}\,|}\ , \ \varphi = c_1 + c_2\ , \ \varphi > 4 \qquad (1.4)$$

With the constriction factor K, the PSO formula for computing the velocity is:

$$v_{id} = K(v_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id})) \qquad (1.5)$$

Usually, when the constriction factor is used, φ is set to 4.1 ($c_1 = c_2 = 2.05$), and the constriction factor K is 0.729. Carlisle and Dozier (2001) show that cognitive and social values of c_1=2.8 and c_2=1.3 yield good results for their test set.

The constriction approach is effectively equivalent to the inertia weight approach. Both approaches have the objective of balancing exploration and exploitation, and in doing so of improving convergence time and the quality of solution found. Low values of ω and K result in exploitation with little exploration, while large values result in exploration with difficulties in reefing solutions (Engelbrecht, 2005). The working of PSO in space is given in Figure 1.2.

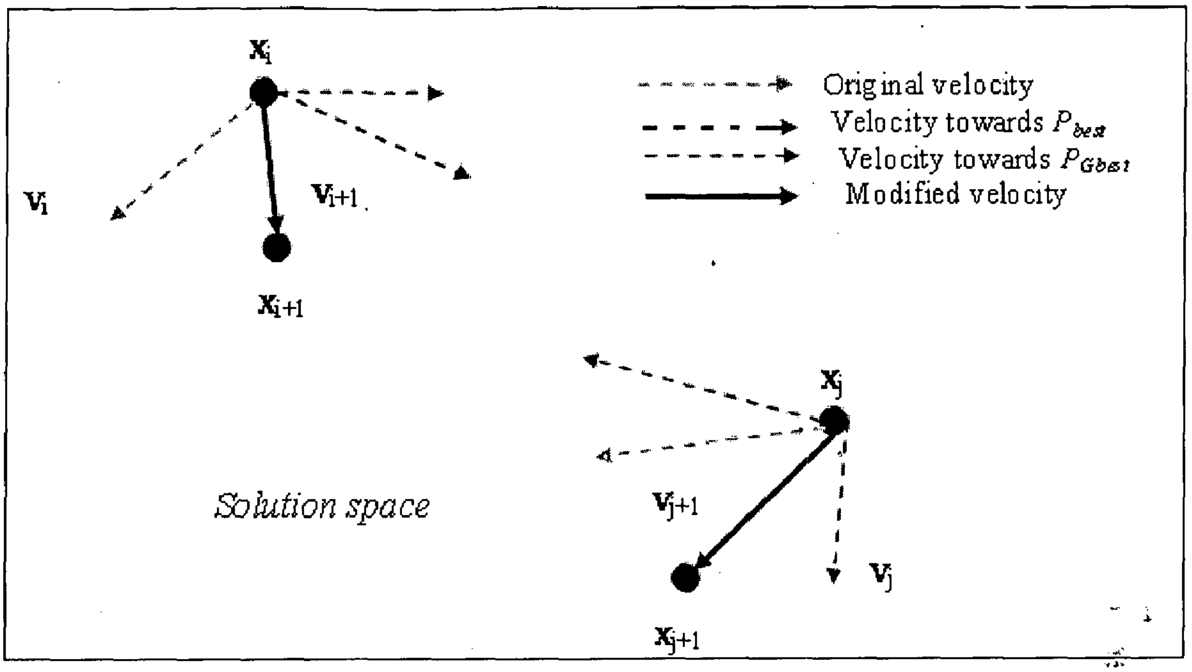

Figure 1.2 Searching mechanism of PSO

1.7 Differences between PSO and other Meta-Heuristics

The most striking difference that separates PSO from other meta-heuristics is that PSO chooses a philosophy of cooperation over competition. The other algorithms commonly use some form of decimation, survival of the fittest. In contrast, the PSO population is stable and individuals are not destroyed or created. Individuals are influenced by the best performance of their neighbors. Individuals eventually converge on optimal points in the problem domain. In addition, the PSO traditionally does not have genetic operators like crossover between individuals and mutation, and other individuals never substitute particles during the run. Instead, the PSO refines its search by attracting the particles to positions with good solutions. Moreover, compared with Genetic Algorithms (GAs), the information sharing mechanism in PSO is significantly different. In GAs, chromosomes share information with each other. So the whole population moves like a one group towards an optimal area. In PSO, only gbest (or Pbest) gives out the information to others. It is a one way information sharing mechanism. The evolution only looks for the best solution. In PSO, all the particles tend to converge to the best solution quickly, comparing with GA, even in the local version in most cases.

1.8 Types of Neighborhood Topologies of PSO

The feature that drives PSO is the social interaction. Particles within the swarm learn from each other and on the basis of the knowledge acquired moves towards better neighbors. Within the PSO, particles in the same neighborhood communicate with one another by exchanging information about the success of each particle in that neighborhood. All particles move towards some quantification of what is believed to be a better position. With a highly connected social network, most of the individuals can connect with one another, with the consequence that the information about the previous best member quickly filters through the social network. Different social network structures have been studied for PSO and empirically studied. In this section we give an overview of some of the structures that have been studied in the past.

- Star topology
- Wheel topology
- Ring or Circle topology
- Von Neumann or Square topology
- Hybrid topology

1.8.1 Star topology

Star Topology is also known as gbest, is a fully connected neighborhood relation. In star topology, one particle is selected as a hub, which is connected to all other particles in the swarm. However, all the other particles are only connected to the hub. Using the gbest model the propagation is very fast (i.e. all the particles in the swarm will be affected by the best solution found in iteration t, immediately in iteration t+1). However, this fast propagation may result in the premature convergence problem. This occurs when some poor individuals attract the population- due to a local optimum or bad initialization - preventing further exploration of the search space. The Star or gbest topology links every individual with every other, so that the social source of influence is in fact the best-performing member of the population.

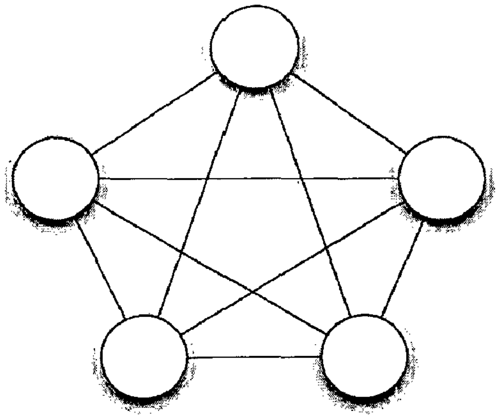

Figure 1.3 Star topology

1.8.2 Wheel topology

The Wheel topology effectively isolates individuals from one another, as all information has to be communicated through the focal individual. This focal individual compares performances of all individuals in the neighborhood, and adjusts its trajectory toward the very best neighbor. If the new position of the focal particle results in better performance, then the improvement is communicated to all the members of the neighborhood. The wheel network slows down the propagation of good solutions through the swarm.

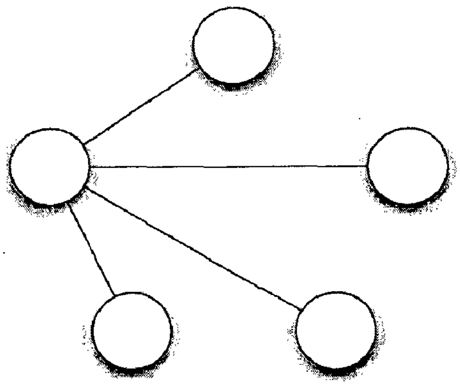

Figure 1.4 Wheel topology

1.8.3 Ring or Circle topology

A Ring topology is also known as lbest, connects each particle to its K immediate neighbors (e.g. K = 2 (left and right particles). The "flow of information" in ring topology is heavily reduced compared to the star topology. In the ring topology, which is a regular ring lattice as studied by Watts and Strogetz (1998), parts of the population that are distant from one another are also independent of one another. However, using the ring topology will slow down the convergence rate because the best solution found has to propagate through several neighborhoods before affecting all particles in the swarm. This slow propagation will enable the particles to explore more areas in the search space and thus decreases the chance of premature convergence.

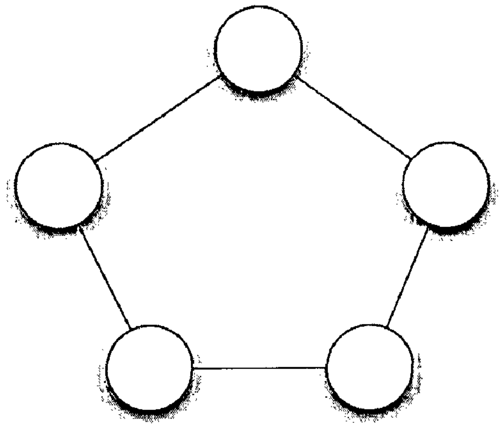

Figure 1.5 Circle topology

1.8.4 Von Neumann or Square topology

Von Neumann is also a type of lbest model. Von Neumann topology was proposed by Kennedy and Mendes. In Von Neumann topology, particles are connected using a grid network (2-dimensional lattice) where each particle is connected to its four neighbor particles (above, below, right, and left particles). However, not like Ring topology, lbest here represent the best value obtained so far by any particle of the neighbors (above, below, right, and left particles). Like Ring topology, using Von Neumann topology will slow down the convergence rate. Slow propagation will enable the particles to explore more areas in the search space and thus decreases the chance of premature convergence.

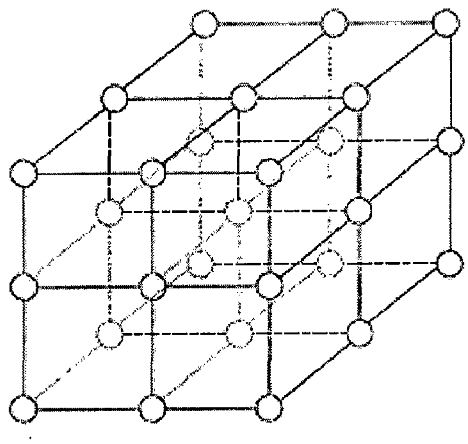

Figure 1.6 Square topology

1.8.5 Hybrid topology

Hybrid topology (or model) is a combination of star, ring and Von Neumann topologies. For each generation, the particle will analyze its next position using all different topologies. Particle will select the topology with the smallest fitness value and will update its velocity and position according to it.

1.9 Literature Survey on Particle Swarm Optimization

In PSO, many variations have been developed to improve its performance. Parsopoulos and Vrahatis (2002) presented a modified PSO algorithm called "stretching" (SPSO) that is oriented towards solving the problem of finding all global minima. In this algorithm, the so-called deflection and stretching techniques, as well as a repulsion technique are incorporated into the original PSO. Miranda and Fonseca (2002; 2002a) introduced self adaptation capabilities to the swarm by modifying the concept of a particle to include, not only the objective parameters, but also a set of strategic parameters. Xie et al (2002) proposed a dissipative Particle Swarm Optimization according to the self-organization of the dissipative structure. In the dissipative PSO, the authors reinitialize velocities and positions based on chaos factors which serve as probabilities of introducing chaos in the system.

Zhang et al (2003) have considered the adjustment of the number of particles, and the neighborhood size. Li (2004) has proposed a species-based PSO (SPSO). According to this method, the swarm population is divided into species of subpopulations based on their similarity. Each species is grouped around a dominating particle called the species seed. At each iteration step, the species seeds are identified and adopted as neighborhood bests for the species groups. Over successive iterations, the adaptation of the species allows the algorithm to find multiple local optima, from which the global optimum can be identified.

The cooperative PSO (CPSO), as a variant of the original PSO algorithm, is presented by van den Bergh and Engelbrecht (2004). CPSO employs cooperative behavior in order to significantly improve the performance of the original PSO algorithm. It uses multiple swarms to optimize different components of the solution vector cooperatively. Passive congregation, a mechanism that allows animals to aggregate into groups, has been proposed by He et al (2004) as a possible alternative to prevent the PSO algorithm from being trapped in local optima and to improve its accuracy and convergence speed. Baskar and Suganthan (2004) have proposed a cooperative scheme, referred to as concurrent PSO (CONPSO), where the problem hyperspace is implicitly partitioned by having two swarms searching concurrently for a solution with frequent message passing of information.

Xen et al (2007) proposed an Adaptive Dissipative Particle Swarm (ADPSO) with mutation operation that combines the idea of the particle swarm optimization with concepts of mutation from Evolutionary Algorithm. The ADPSO algorithm uses Cauchy mutation operation to escape from the attraction of local minimum and also uses an adaptive inertia weight strategy. Wei et al (2008) have proposed an improved Particle Swarm Optimization algorithm named Elite Particle Swarm Optimization with mutation (EPSOM). In EPSOM, the elite particles and the bad particles are distinguished from the swarm after some initial iteration steps, bad particles are replaced with the same number of elite particles, and a new swarm is generated. Also they introduced a new mutation operator in order to avoid the loss of diversity. Many of the improved versions of PSO algorithm can be found in (Banks et al, 2008; Das et al, 2008). A detailed literature survey on diversity based improved versions of PSO, mutation based PSO, Crossover based PSO etc. are given in Chapter 3.

1.10 Differential Evolution for Global Optimization

Differential Evolution (DE), a vector population based stochastic optimization method was introduced by Price and Storn (1995). It is capable of handling nondifferentiable, nonlinear and multimodal objective functions.

DE is easy to implement, requires few, easily chosen control parameters and exhibits fast convergence. The three control parameters of DE are the scale factor F, the crossover rate Cr and the population size. The scale factor F is a positive real number that controls the rate at which the population evolves. While there is no upper limit on F, effective values are seldom greater than 1.0. The crossover rate, $Cr \in [0, 1]$, is a user-defined value that controls the fraction of parameter values that are copied from the mutant. Experimental results have shown that performance of DE is better than many other well known Evolutionary Algorithms (EAs) (Storn and Price, 1997; Storn, 1999). While DE shares similarities with other EAs, it differs significantly in the sense that in DE, distance and direction information is used to guide the search process (Engelbrecht, 2005).

In a population of potential solutions within an n-dimensional search space, a fixed number of vectors are randomly initialized, then evolved over time to explore the search space and to locate the minima of the objective function. At each iteration, called generation, new vectors are generated by the combination of vectors randomly chosen from the current population (mutation). The outcoming vectors are then mixed with a predetermined target vector. This operation is called recombination and produces the trial vector. Finally, the trial vector is accepted for the next generation if and only if it yields a reduction in the value of the objective function. This last operator is referred to as a selection. The working of DE is discussed in more detail in section 1.11.

DE is similar to GAs in that a population of individuals is used to search for an optimal solution. The main difference between Gas and DE is that, in GAs, mutation is the result of small perturbations to the genes of an individual while in DE mutation is the result of arithmetic combinations of individuals. At the beginning of the evolution process, the mutation operator of DE favors exploration. As evolution progresses, the mutation operator favors exploitation. Hence, DE automatically adapts the mutation increments (i.e. search step) to the best value based on the stage of the evolutionary process. Mutation in DE is therefore not based on a predefined probability density function.

1.11 Working of Differential Evolution

A general DE variant may be denoted as DE/X/Y/Z, where X denotes the vector to be mutated, Y specifies the number of difference vectors used and Z specifies the crossover scheme which may be binomial or exponential. For the more details the interested reader may refer to http://www.icsi.Berkley.edu/~storn/code.html.

The working of DE is as follows: First, all individuals are initialized with uniformly distributed random numbers and evaluated using the fitness function provided. Then the following will be executed until maximum number of generation has been reached or an optimum solution is found.

***Mutation*:**

For a D-dimensional search space, for each target vector $X_{i,g}$ at the generation g, its associated mutant vector is generated via certain mutation strategy. The most often used mutation strategies implemented in the DE codes are listed below.

DE/rand/1: $V_{i,g} = X_{r_1,g} + F*(X_{r_2,g} - X_{r_3,g})$ (1.6)

DE/rand/2: $V_{i,g} = X_{r_1,g} + F*(X_{r_2,g} - X_{r_3,g}) + F*(X_{r_4,g} - X_{r_5,g})$ (1.7)

DE/best/1: $V_{i,g} = X_{best,g} + F*(X_{r_1,g} - X_{r_2,g})$ (1.8)

DE/best/2: $V_{i,g} = X_{best,g} + F*(X_{r_1,g} - X_{r_2,g}) + F*(X_{r_3,g} - X_{r_4,g})$ (1.9)

DE/rand-to-best/1: $V_{i,g} = X_{r_1,g} + F*(X_{best,g} - X_{r_2,g}) + F*(X_{r_3,g} - X_{r_4,g})$ (1.10)

where $r_1, r_2, r_3, r_4, r_5 \in \{1,2,....,NP\}$ are randomly chosen integers, different from each other and also different from the running index i. F (>0) is a scaling factor which controls the amplification of the difference vector. $X_{best,g}$ is the best individual vector with the best fitness value in the population at generation g.

***Crossover*:**

Once the mutation phase is over, crossover is performed between the target vector and the mutated vector to generate a trial point for the next generation. Crossover is introduced to increase the diversity of the population (Storn and Price, 1997).

The mutated individual, $V_{i,G+1} = (v_{1,i,G+1}, \ldots, v_{D,i,G+1})$, and the current population member, $X_{i,G} = (x_{1,i,G}, \ldots, x_{D,i,G})$, are then subject to the crossover operation, that finally generates the population of candidates, or "trial" vectors, $U_{i,G+1} = (u_{1,i,G+1}, \ldots, u_{D,i,G+1})$, as follows:

$$u_{j,i,G+1} = \begin{cases} v_{j,i,G+1} & if \quad rand_j \le Cr \vee j = k \\ x_{j,i,G} & otherwise \end{cases} \tag{1.11}$$

Where j, k ∈ {1,..., D} k is a random parameter index, chosen once for each *i*, C_r is the crossover probability parameter whose value is generally taken as $C_r \in [0, 1]$.

Selection:

The final step in the DE algorithm is the selection process. Each individual of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value survives the tournament selection and go to the next generation.

As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. A notable point in DE's selection scheme is that a trial vector is not compared against all the individuals in the current generation, but only against one individual, its counterpart, in the current generation. The population for the next generation is thus selected from the individuals in current population and its corresponding trial vector according to the following rule:

$$X_{i,G+1} = \begin{cases} U_{i,G+1} & \text{if} \quad f(U_{i,G+1}) \le f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \tag{1.12}$$

Thus, each individual of the temporary (trial) population is compared with its counterpart in the current population. The one with the lower objective function value will survive from the tournament selection to the population of the next generation. As a result, all the individuals of the next generation are as good as or better than their counterparts in the current generation. In DE trial vector is not compared against all the individuals in the current generation, but only against one individual, its counterpart, in the current generation. The working of DE in space is given in Figure 1.7.

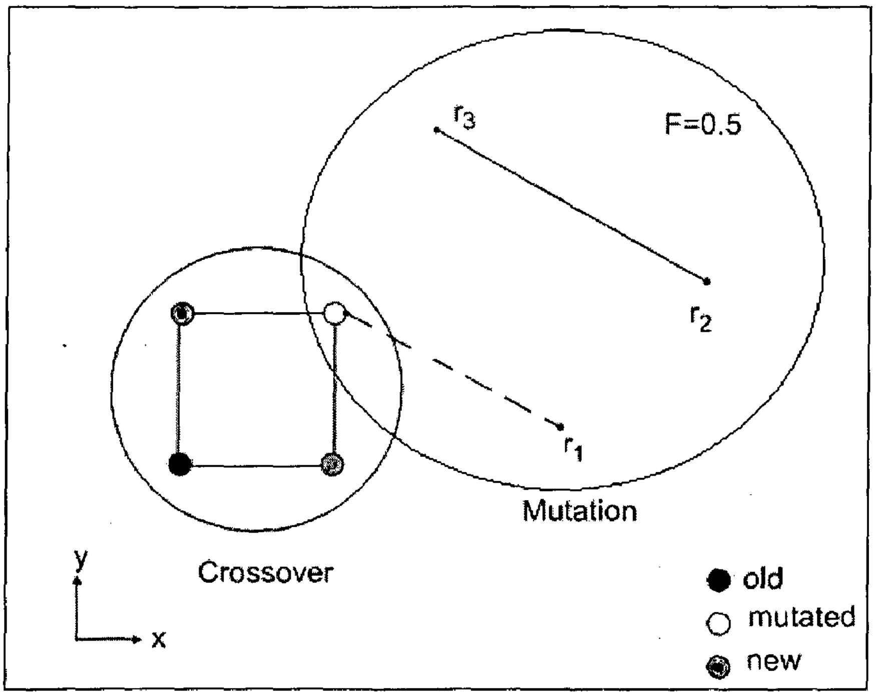

Figure 1.7 Working of DE

1.12 Literature Survey on Differential Evolution

This section presents a brief review of different variants of DE available in literature. Vesterstroem and Thomsan (2004) compared the DE algorithm with particle swarm optimization (PSO) and EAs on numerical benchmark problems. DE outperformed PSO and EAs in terms of the solution's quality on most benchmark problems. Ali and Torn (2004) proposed a new version of DE algorithm, and also suggested some modifications to the classical DE in order to improve its efficiency and robustness. They introduced an auxiliary population of NP individuals alongside the original population and proposed a rule for calculating the control parameter F automatically. Sun, et al. (2004) proposed a combination of the DE algorithm and the estimation of distribution algorithm, which tries to guide its search towards a promising area by sampling new solutions from a probability model. Liu and Lampinen introduced Fuzzy Adaptive Differential Evolution (FADE) (Liu and Lampinen, 2005) using fuzzy logic controllers, whose inputs incorporate the relative function values and individuals of successive generations to adapt the parameters for the mutation and crossover operation. Based on the experimental results over a set of benchmark functions, the FADE algorithm outperformed the conventional DE algorithm.

Yang et al. (2007) proposed a hybridization of DE with the Neighborhood Search (NS), which appears as a main strategy underpinning Evolutionary Programming. The resulting algorithm, known as NSDE, performs mutation by adding a normally distributed random value to each target-vector. Rahnamayan et al (2008) have proposed an Opposition-based DE (ODE) that is specially suited for noisy optimization problems. The conventional DE algorithm was enhanced by utilizing the opposition number-based optimization concept in three levels, namely, population initialization, generation jumping, and local improvement of the population's best member. Yang et al. (2008) used a Self-adaptive NSDE in the cooperative coevolution framework that is capable of optimizing large scale non-separable problems (up to 1000 dimensions). They proposed a random grouping scheme and adaptive weighting for problem decomposition and coevolution. Noman and Iba (2005; 2008) proposed the Fittest Individual Refinement (FIR); a crossover-based local search method for DE. The FIR scheme accelerates DE by enhancing its search capability through exploration of the neighborhood of the best solution in successive generations.

There are quite different conclusions about the rules for choosing the ***control parameters*** of DE. Price and Storn (1995) stated that the control parameters of DE are not difficult to choose. On the other hand, Gämperle et al. (2002) reported that choosing the proper control parameters for DE is more difficult than expected. Liu and Lampinen (2002) reported that effectiveness, efficiency, and robustness of the DE algorithm are sensitive to the settings of the control parameters. The best settings for the control parameters can be different for different functions and the same function with different requirements for consumption time and accuracy. However, there still exists a lack of knowledge on how to find reasonably good values for the control parameters of DE for a given function (Liu and Lampinen, 2005).

Das et al. (2005) introduced two schemes for adapting the scale factor F in DE. In the first scheme they varied F randomly between 0.5 and 1.0 in successive iterations. They suggested decreasing F linearly from 1.0 to 0.5 in their second scheme. This encourages the individuals to sample diverse zones of the search space during the early stages of the search. During the later stages, a decaying scale factor helps to adjust the movements of trial solutions finely so that they can explore the interior of a relatively small space in which the suspected global optimum lies. Teo (2006) proposed an attempt at self-adapting the population size parameter in addition to self-adapting crossover and mutation rates. Brest et al. (2006, 2006a) encoded control parameters F and Cr into the individual and evolved their values by using two new probabilities $\tau_1$and τ_2. In their algorithm, a set of F values was assigned to each individual in the population. With probability τ_1, F is reinitialized to a new random value in the range of [0.1, 1.0], otherwise it is kept unchanged. The control parameter Cr, assigned to each individual, is adapted in an identical fashion, but with a different re-initialization range of [0, 1] and with the probability τ_2. With probability τ_2, Cr takes a random value in [0, 1], otherwise it retains its earlier value in the next generation. Differential Evolution with Preferential Crossover (DEPC) was suggested by M. M. Ali in 2007 (Ali, 2007). In his work he suggested three changes in the basic DE structure. The DEPC algorithm uses F_i as a random variable in $[-1, -0.4] \cup [0.4, 1]$ for each targeted point. Secondly DEPC used two population sets S_1 and S_2 containing N points. The function of the auxiliary set S_2 in DEPC is to keep record of the trial points that are discarded in DE. Potential trial points in S_2 are then used for further explorations. Finally DEPC used a new crossover rule, namely the preferential crossover, which always generates feasible trial points. Ali tested his algorithm on comprehensive set of benchmark problems and showed that DEPC outperforms the basic DE in most of the test cases.

Yang et al. (2008a) proposed a self adaptive differential evolution algorithm with neighborhood search (SaNSDE). SaNSDE proposes three self-adaptive strategies: self adaptive choice of the mutation strategy between two alternatives, self-adaptation of the scale factor F, and self-adaptation of the crossover rate Cr. Qin et al (2009) proposed a Self-adaptive DE algorithm (SaDE), where the choice of learning strategy and the two control parameters F and CR are not required to be pre-defined. During evolution, the suitable learning strategy and parameter settings are gradually self-adapted according to the learning experience.

1.13 Applications of PSO and DE Algorithms

PSO and DE algorithms have been applied successfully to a wide variety of problems occurring in different fields of science and engineering. It is very difficult to summarize all the applications of PSO and DE algorithms, therefore considering the brevity of space, in this section a brief review of some of the applications of these algorithms is given.

PSO has been applied to solve a number of interesting test problems including evolving weights and structures of neural networks (He et al, 1998; van den Bergh, 1999; Salerno, 1997), analyzing human tremor (Eberthart and Hu, 1999), registering 3D- 3D biomedical image (Wachowiak et al, 2004), controlling reactive power and voltage (Yoshida, 2000; Abido, 2002), pattern recognition (Paterilni and Krink, 2006), quadratic assignment problems (Liu and Abraham, 2007), job scheduling (Liu et al, 2009), multimedia processing (Hassanien et al, 2008), bioinformatics problems (Hassanien et al, 2008a).

DE algorithm has also been successfully applied to diverse domains of science and engineering, such as mechanical engineering design (Rogalski et al, 1999; Joshi and Sanderson, 1999), signal processing (Das and Konar, 2006) chemical engineering (Wang and Jang, 2000; Lampinen, 1999; Onwubolu and Babu, 2004), machine intelligence, and pattern recognition (Omran et al, 2005), (Das et. al, 2008). Many of the most recent developments in DE algorithm design and applications can be found in (Chakraborty, 2008). A comparison of DE and PSO algorithms is made in the following papers (Pant et al, 2008; 2008a; Mishra, 2006).

1.14 Computational Steps of Basic PSO and DE Algorithms

This section gives the computational steps of basic PSO and DE algorithms which have been followed throughout the thesis.

Computational Steps of basic PSO algorithm:

Step 1	*Initialize PSO parameters*
Step 2	*Randomly initialize the positions and velocities of all particles*
Step 3	*Evaluate the fitness function values of all particles in the swarm*
Step 4	*Update particles personal best position and global best position (i.e. P_i and P_g)*
Step 5	*Set $t = 1$, t refers the iteration number*
Step 6	*Update the velocity vector using Eqn. (1.3)*
Step 7	*Update particle's position using Eqn. (1.2)*
Step 8	*Evaluate particle's fitness values*
Step 9	*Update P_i and P_g*
Step 10	*Set $t = t+1$*
Step 11	*If (Stopping criteria is reached) then go to step 12*
	Else go to step 6
Step 12	*Print the global best particle and the corresponding fitness function value*

Computational Steps of Basic DE algorithm:

Step 1	*Initialize DE parameters*
Step 2	*Randomly initialize the positions of all particles*
Step 3	*Evaluate the fitness function values of all particles (X_i) in the population*
Step 3	*Set $g = 1$*
Step 4	*// Mutation*
	Generate mutant vectors $V_{i,g+1}$ corresponding to each target vector $X_{i,g}$ via one of the Eqns. (1.6) to (1.10)
Step 5	*// Crossover*
	Generate trial vector $U_{i,g+1}$ for each particle using Eqn. (1.11)
Step 6	*//Selection*
	Update particles position using Eqn. (1.12)

Step 7	*Set g = g + 1*
Step 8	*If (Stopping criteria is reached) then go to step 9*
	Else go to step 4
Step 9	*Print the global best particle and the corresponding fitness function value*

1.15 Objective of the Present Work

Though PSO and DE have been successfully applied to a wide range of test and real life problems experimental analysis shows that sometimes these algorithms do not perform up to the expectations. Like all other population based search techniques there are certain drawbacks associated with these algorithms. For example premature convergence is a problem common to both PSO and DE algorithms. In such a case the population converges to some local optima of a multimodal objective function, losing its diversity. The situation when the algorithm does not show any improvement though it accepts new individuals in the population is known as stagnation. The probability of stagnation depends on how many different potential trial solutions are available and also on their capability to enter into the population of the subsequent generations. Further, like other evolutionary computing algorithms, the performance of PSO and DE deteriorates with the growth of the dimensionality of the search space as well. The main objective of the present work is to suggest simple and efficient modifications in the basic structure of PSO and DE algorithms to improve their performance by overcoming their drawbacks, so that they can effectively solve the test as well as real life application problems.

1.16 Outline of the Thesis

This thesis is divided into ten chapters which are organized as follows,

Chapter 1 gives the brief introduction to Particle Swarm Optimization and Differential Evolution algorithms.

Chapter 2 investigates the effect of initiating the population of PSO and DE with different probability distributions like Gaussian, Exponential, Gamma and Beta and classical low discrepancy sequences like Vander Corput sequence and Sobol sequence for solving global optimization problems in large dimension search spaces. The proposed algorithms are tested on standard benchmark problems and the results are compared with the basic versions of PSO and

DE which follows the uniform distribution for initializing the swarm. The simulation results show that a significant improvement can be made in the performance of PSO and DE, by simply changing the distribution of random numbers to other than uniform distribution and quasi random sequence as the proposed algorithms outperform the basic versions by noticeable percentage.

Chapter 3 describes the improved versions of the Particle Swarm Optimization algorithm. The main focus is on the design and implementation of the improved PSO algorithms based on diversity, Crossover and Mutation using different probability distributions and Low-discrepancy sequences. Also this chapter introduces a new velocity vector and inertia weight in classical PSO. The proposed algorithms are applied to various benchmark problems including unimodal, multimodal, noisy functions and comparisons made with some other variants of PSO in the literature.

Chapter 3A discusses about Quantum Particle Swarm Optimization (QPSO) and describes the improved versions of QPSO; it is an extension of chapter 3.

Chapter 4 describes the improved versions of the classical Differential Evolution algorithm. The improved algorithms are based on the mutant vector, the scale factor F and the crossover rate Cr of DE. This chapter proposes two new mutant vectors based on the concept of Quadratic Interpolation (DE-QI) and the Laplace probability distribution (LDE). Five versions of LDE are proposed namely LDE1, LDE2, LDE3, LDE4 and LDE5. This chapter also introduces a dynamic scaling factor and Crossover rate. The proposed algorithms are examined with several standard benchmark problems and the results are compared with the classical DE and some other variants of DE in the literature.

Chapter 5 presents three hybrid two phase global optimization algorithms namely DE-PSO, MDE, AMPSO for solving global optimization problems. DE-PSO consists of alternating phases of Differential Evolution and Particle Swarm Optimization. MDE consists of alternating phases of Differential Evolution and Evolutionary Programming. AMPSO algorithm is a hybrid version of Particle Swarm Optimization and Evolutionary Programming. Two versions of AMPSO called AMPSO1 and AMPSO2 are proposed. Both the algorithms use EP based adaptive mutation using Beta distribution. AMPSO1 mutates the personal best position of the swarm and AMPSO2 mutates the global best swarm position. The performance of proposed algorithms is evaluated on standard unconstrained test problems.

Chapter 6 proposes a new constraint handling mechanism for solving constrained optimization problems. Based on the new constraint handling mechanism, two algorithms are proposed namely ICPSO and ICDE. The Improved Constraint Particle Swarm Optimization (ICPSO) algorithm is initialized using quasi random Vander Corput sequence and differs from unconstrained PSO algorithm in the phase of updating the position vectors and sorting every generation solutions. Also, the Improved Constraint DE (ICDE) algorithm differs from unconstrained DE algorithm only in the place of initialization, selection of particles to the next generation and sorting the final results. The performance of ICPSO and ICDE algorithms are validated on twenty constrained benchmark problems. The numerical results show that the proposed algorithms are quite promising algorithms for solving constraint optimization problems.

Chapter 7 investigates the performance of PSO, DE and their proposed variants with the real life problem namely In-situ efficiency determination of Induction Motor (5hp). By the application of PSO and DE algorithms in this problem, the motor efficiency can be obtained without performing no-load test, which is not easily possible for the motors working in process industries where continuous operation is required. Results are compared with Genetic Algorithm (GA) and a physical efficiency measurement method, called torque-gauge method. The performances in term of objective function (error in the efficiency) and convergence time prove the effectiveness of the optimization algorithms used for comparison in this chapter.

Chapter 8 presents the model of Directional Overcurrent Relay settings in Electrical Power Systems, which is modeled as a constrained nonlinear optimization problem. The optimization problem corresponding to IEEE 3-bus, IEEE 4-bus and IEEE 6-bus system is considered. The five DE algorithms namely: LDE1, LDE2, LDE3, LDE4, LDE5 and DE-QI discussed in chapter 4 are used to solve the resulting optimization problem.

Chapter 9 gives eleven real life problems, which are collected from various fields. They are: Static Power Scheduling problem, Dynamic Power Scheduling problem, Cost Optimization of Transformer Design, Weight Minimization of a Speed Reducer, Heat Exchanger Network Design, Gas Transmission Compressor Design, Optimal Design of a Industrial Refrigeration

System, Optimization of Transistor Modeling, Optimal Capacity of Gas Production Facilities, Optimal Thermohydralic Performance of an Artificially Roughened Air Heater and Design of a Gear Train. The proposed algorithms discussed in chapter 2, 3, 4 and 5 are used to solve the above real life problems. Empirical results show that the proposed algorithms are quite competent for solving the considered real life problems.

Chapter 10 finally concludes the thesis and gives the scope for the future work.

Appendices (List of unconstrained test problems, List of constrained test problems) are given in the end.

2

Efficient Initialization Methods in PSO and DE

[This chapter investigates the effect of initiating the population with various probability distributions and low discrepancy sequences on the behavior of the basic Particle Swarm Optimization and Differential Evolution algorithms. The probability distributions: Gaussian, Exponential, Beta and Gamma distribution and the low discrepancy sequences: Van der Corput and Sobol are considered in this study for generating the initial population. The proposed algorithms are tested on standard benchmark problems and the results are compared with the basic versions of PSO and DE which follows the uniform distribution for initializing the swarm. The simulation results show that a significant improvement can be made in the performance of PSO and DE, by simply changing the distribution of random numbers to other than uniform distribution and quasi random sequence as the proposed algorithms outperform the basic versions by a noticeable percentage.]

2.1 Introduction

Many of the population based stochastic search techniques which depend on the generation of random numbers work very well for problems having a small search area (i.e. a search area having low dimension), but as we go on increasing the dimension of search space the performance deteriorates and many times converge prematurely giving a suboptimal result (Liu et al, 2007). 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 these algorithms may be attributed to the dispersion of initial population points in the search space i.e. to say, in case of PSO, 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 (Grosan et al, 2005). This difficulty may be minimized to a great extent by selecting a well-organized distribution of random numbers which constitute the initial population.

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. This method is not very proficient as it has been shown that uniform pseudo-random number sequences have discrepancy of order (log (log N))$^{1/2}$ and thus do not achieve the lowest possible discrepancy. Also, it has been shown that uniformly distributed particles may not always be good for empirical studies of different algorithms. Moreover, the uniform distribution sometimes gives a wrong impression of the relative performance of algorithms as shown by Gehlhaar and Fogel (1996).

Subsequently, researchers have proposed an alternative way of generating 'quasirandom' numbers through the use of low discrepancy sequences. Their discrepancies have been shown to be optimal, of order (log N)^s/N (Gentle, 1998; Knuth, 1998). These sequences are useful for global optimization, because of the variation of random numbers that are produced in each iteration, thus providing a better diversified population of solutions and thereby increasing the probability of getting a better solution.

Some instances on the use of different initialization methods (other than uniform distribution) available in literature are as follows; Kimura and Matsumura (2005) used Halton sequence for initializing the Genetic Algorithms (GA) population and showed that a real coded GA performs much better when initialized with a quasi random sequence in comparison to a GA which initialized with a population having uniform probability distribution. Instances where quasi random sequences have been used for initializing the swarm in PSO can be found in (Parsopoulos and Vrahatis, 2002; Brits, 2002; Brits, 2002a; Nguyen, 2007). In (Parsopoulos and Vrahatis, 2002; Brits, 2002; Brits, 2002a) authors have made use of Sobol and Faure sequences. Similarly, Nguyen et al (2007) 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.

However to the best of our knowledge, no results are available on the performance of Van der Corput sequence which is a well known sequence and forms the basis of many other sequences and also no one has used different Probability distributions for generating the initial population. Keeping this fact in mind, the present study is to scrutinize the performance of PSO and DE using Van der Corput sequence along with Sobol sequence and different probability distributions (Gaussian, Exponential, Beta and Gamma) for population initialization and tested them for solving global optimization problems in large dimension search spaces.

The organization of this chapter is as follows: Section 2.2 and 2.3 briefly describes the low discrepancy sequences and the probability distributions used in this study respectively. In section 2.4, the twelve proposed algorithms of PSO and DE with different initialization methods are given. Section 2.5 and 2.6 give the parameter settings of PSO and DE algorithms and the numerical results respectively. Finally this chapter concludes with section 2.7

2.2 Low-Discrepancy Sequences

Mathematically, the discrepancy of a sequence is the measure of its uniformity which may be defined as follows:

For a given set of points $x^1, x^2, \ldots, x^N \in I^S$ and a subset $G \subset I^S$, define a counting function $S_N(G)$ as the number of points $x^i \in G$. For each $x = (x_1, x_2, \ldots x_S) \in I^S$, let G_x be the rectangular s-dimensional region.

$G_x = [0,x_1) \times [0,x_2) \times \ldots \times [0,x_S)$ with volume $x_1x_2\ldots x_N$.

Then the discrepancy of points is given by $D^*_N(x^1, x^2, x^3 \ldots x^N) = \text{Sup} \mid S_N(G_x) - Nx_1x_2\ldots x_S \mid, x \in I^S$.

The discrepancy is therefore computed by comparing the actual number of sample points in a given volume of a multi-dimensional space with the number of sample points that should be there assuming a uniform distribution.

A Low-discrepancy sequence is a sequence with the property that for all values of N, its subsequence $x_1, \ldots, x_N$ has a low discrepancy. Low-discrepancy sequences are also called quasi-random or sub-random sequences, due to their use as a replacement of uniformly distributed random numbers.

2.2.1 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 (1935). It is a digital (0, 1)-sequence, which exists for all bases $b \geq 2$. Many of the relevant low discrepancy sequences are linked to the Van der Corput sequence introduced initially for dimension s = 1 and base b = 2. The Van der Corput discovery inspired other quasi random sequences like Halton (1960), Faure, Sobol (Sobol, 1967; Sobol, 1976) etc.

It is defined by the *radical inverse function* φ_b: $N_0 \rightarrow [0, 1)$. If $n \in N_0$ has the b-adic expansion

$$n = \sum_{j=0}^{T} a_j b^{j-1} \tag{2.1}$$

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

$$\varphi_b(n) = \sum_{j=0}^{T} \frac{a_j}{b^j} \tag{2.2}$$

In other words, the jth b-adic digit of n becomes the jth 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. Figure 2.1 shows the random numbers distributed by Van der Corput sequence.

2.2.2 Sobol Sequence

The Sobol sequence is the most widely deployed low-discrepancy sequence, and is used for calculating multi-dimensional integrals and in quasi-Monte Carlo simulation. The construction of the Sobol sequence (Chi et al, 1999) 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:

$$X_n^{(j)} = n_1 v_1^{(j)} \oplus n_2 v_2^{(j)} \oplus \ldots\ldots \oplus n_w v_w^{(j)} \tag{2.3}$$

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:

$$v_i^{(j)} = a_1 v_{i-1}^{(j)} \oplus a_2 v_{i-2}^{(j)} \oplus \ldots \oplus a_q v_{i-q+1}^{(j)} \oplus v_{i-q}^{(j)} \oplus (v_{i-q}^{(j)} / 2^q) \tag{2.4}$$

We have i > q, and the bit, a_i, comes from the coefficients of a degree-q primitive polynomial over F2. Figure 2.2 shows the random numbers distributed by Sobol sequence.

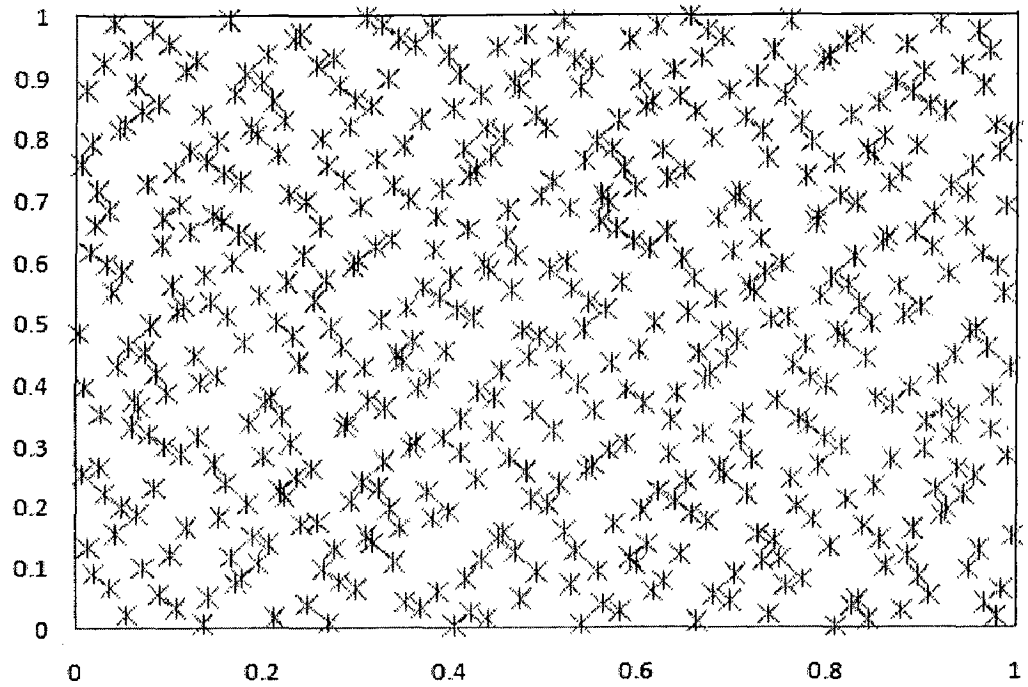

Figure 2.1 Sample points generated using Van der Corput sequence

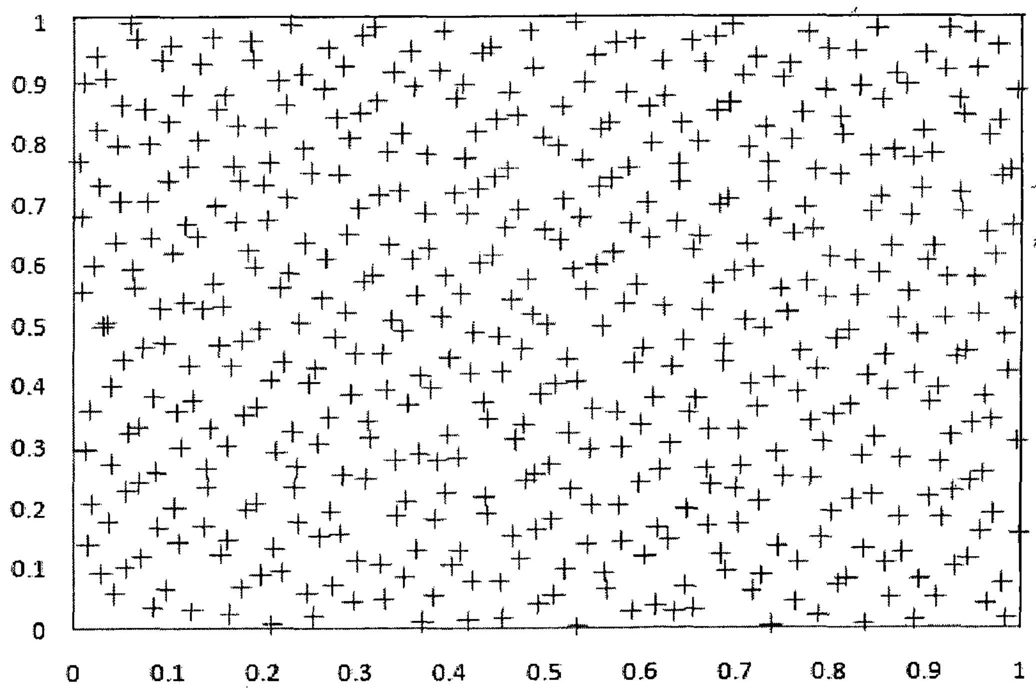

Figure 2.2 Sample points generated using Sobol sequence

2.3 Probability Distributions

In probability theory and statistics, a probability distribution identifies either the probability of each value of an unidentified random variable (when the variable is discrete), or the probability of the value falling within a particular interval (when the variable is continuous). The probability distribution describes the range of possible values that a random variable can attain and the probability that the value of the random variable is within any (measurable) subset of that range. When the random variable takes values in the set of real numbers, the probability distribution is completely described by the cumulative distribution function, whose value at each real x is the probability that the random variable is smaller than or equal to x.

There are two types of probability distributions: discrete probability functions and continuous probability functions. A discrete probability function is a function that can take a discrete number of values (not necessarily finite). This is most often the non-negative integers or some subset of the non-negative integers. Continuous probability functions are defined for an infinite number of points over a continuous interval. Discrete probability functions are referred to as probability mass functions and continuous probability functions are referred to as probability density functions. The term probability function covers both discrete and continuous distributions. When we are referring to probability functions in generic terms, we may use the term probability density functions to mean both discrete and continuous probability functions.

Some practical uses of probability distributions are:

- To calculate confidence intervals for parameters and to calculate critical regions for hypothesis tests.
- For univariate data, it is often useful to determine a reasonable distributional model for the data.
- Statistical intervals and hypothesis tests are often based on specific distributional assumptions. Before computing an interval or test based on a distributional assumption, we need to verify that the assumption is justified for the given data set. In this case, the distribution does not need to be the best-fitting distribution for the data, but an adequate enough model so that the statistical technique yields valid conclusions.
- ***Simulation studies with random numbers generated from using a specific probability distribution are often needed.***

There are various probability distributions that show up in various different applications. Among all the probability distributions, the present study investigates the characteristics of PSO and DE with four probability distributions namely: Gaussian distribution, Exponential distribution, Beta distribution and Gamma distribution.

2.3.1 Gaussian Distribution

The normal distribution or Gaussian distribution is a continuous probability distribution that describes data that clusters around a mean or average. The graph of the associated probability density function is bell-shaped, with a peak at the mean, and is known as the Gaussian function or bell curve. The probability density function of a Gaussian probability distribution is given by:

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}, \ -\infty \le x \le \infty \qquad (2.5)$$

where μ is the mean and σ is the standard deviation. For a mean 0 and standard deviation1, this formula simplifies to:

$$f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{x^2}{2}} \qquad (2.6)$$

In the present study, Gaussian distributed random numbers with mean zero and standard deviation 1 is used.

2.3.2 Exponential Distribution

The Exponential distribution is a only continuous memoryless probability distribution that describes the times between events in a Poisson process, i.e. a process in which events occur continuously and independently at a constant average rate. It is a continuous analog of the geometric distribution. Also, it is a commonly used distribution in reliability engineering. Mathematically, Exponential distribution is a fairly simple distribution, which many times lead to its use in inappropriate situations. It is, in fact, a special case of the Weibull distribution.

The probability density function (pdf) of an exponential distribution is,

$$f(x,\lambda)=\begin{cases}\lambda e^{-\lambda x}, & x\geq 0\\ 0, & x<0\end{cases} \tag{2.7}$$

Here $\lambda > 0$ is the parameter of the distribution, often called the rate parameter. The distribution is supported on the interval $[0, \infty)$.

The generalized exponential probability function is defined by,

$$f(x)=\frac{1}{2b}e^{-(x-a)/b}\ , -\infty\leq x\leq\infty,\ a,b>0 \tag{2.8}$$

It is evident that one can control the variance by changing the parameters a and b. This study uses the Eqn. (2.8) to generate the exponentially distributed random numbers.

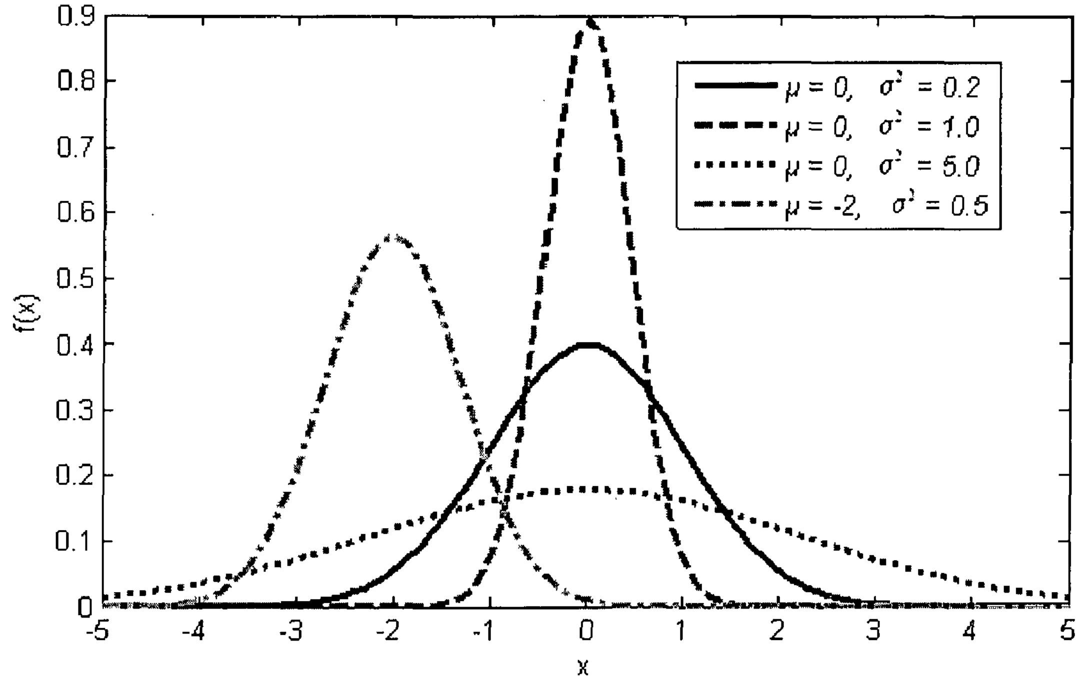

Figure 2.3 Probability density function of Gaussian distribution

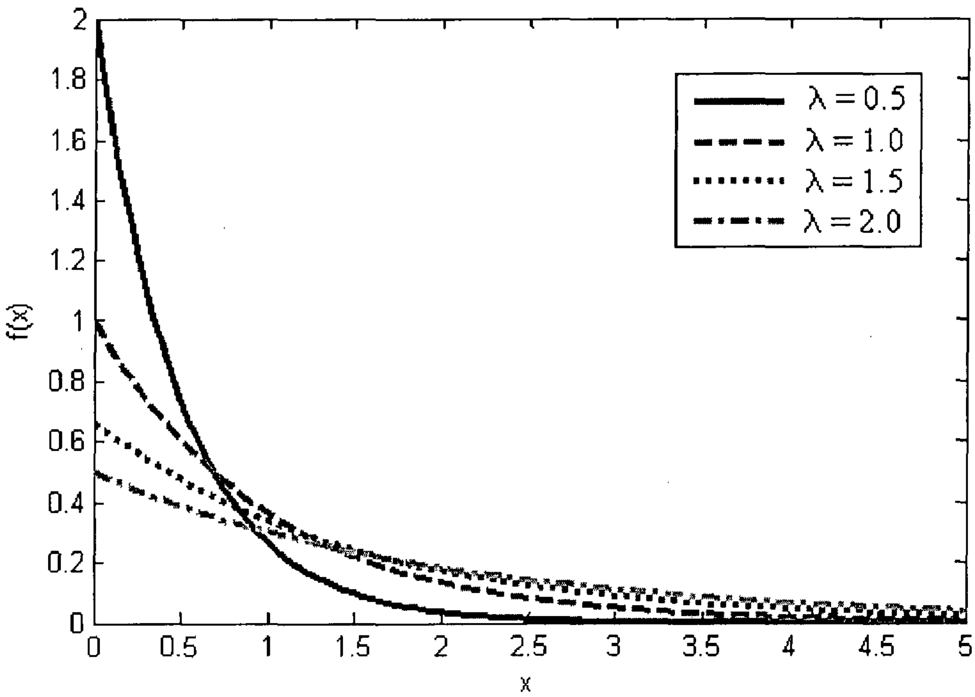

Figure 2.4 Probability density function of Exponential distribution

2.3.3 Beta Distribution

The Beta distribution is a family of continuous probability distributions parameterized by two positive shape parameters, typically denoted by α and β. It is a general type of statistical distribution and also related to the gamma distribution. In Bayesian statistics, it can be seen as the posterior distribution of the parameter p of a binomial distribution after observing $\alpha - 1$ independent events with probability p and $\beta - 1$ with probability $1 - p$, if the prior distribution of p was uniform. The beta distribution is a more flexible probability density function compared to the normal distribution because it can accommodate different ranges and different shapes from left skew. The Probability density function of Beta distributions is given by:

$$f(x) = \frac{(x-a)^{\alpha-1}(b-x)^{\beta-1}}{B(\alpha,\beta)(b-a)^{\alpha+\beta-1}},\ a \le x \le b;\ \alpha,\beta > 0 \qquad (2.9)$$

where α and β are the shape parameters, a and b are the lower and upper bounds, respectively, of the distribution, and $B(\alpha$, β) is the beta function.

$$B(\alpha,\beta) = \int_0^1 x^{\alpha-1}(1-x)^{\beta-1}dx \qquad (2.10)$$

The case where $a = 0$ and $b = 1$ is called the standard beta distribution. The equation for the standard beta distribution is:

$$f(x) = \frac{x^{\alpha-1} * (1-x)^{\beta-1}}{B(\alpha,\beta)} \qquad (2.11)$$

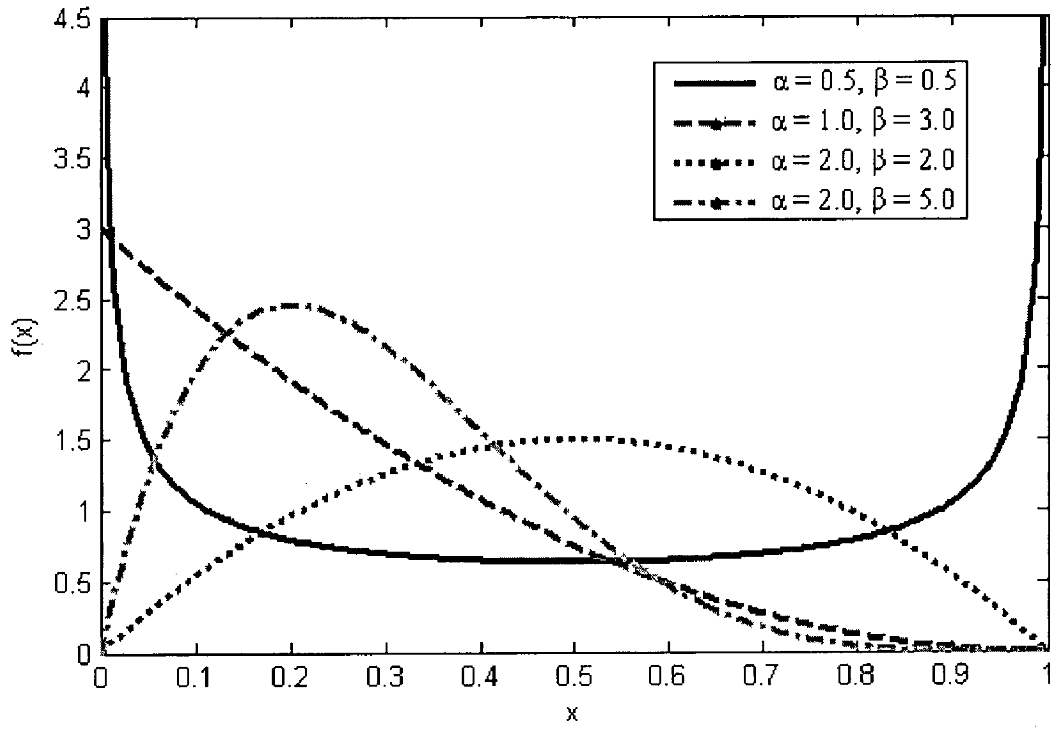

Figure 2.5 Probability density function of Beta distribution

2.3.4 Gamma Distribution

A gamma distribution is a general type of statistical distribution that is related to the beta distribution and arises naturally in processes for which the waiting times between Poisson distributed events are relevant. It is widely used in engineering, science, and business, to model continuous variables that are always positive and have skewed distributions. Gamma distributions have two free parameters, shape parameter and scale parameter labeled as α and θ. If α is an integer then the distribution represents the sum of α independent exponentially distributed random variables, each of which has a mean of θ.

The general formula for the probability density function of the gamma distribution is:

$$f(x)=\frac{\left(\frac{x-\mu}{\theta}\right)^{\alpha-1} e^{-\left(\frac{x-\mu}{\theta}\right)}}{\theta*\Gamma(\alpha)},\ x \geq \mu;\ \alpha,\theta>0 \tag{2.12}$$

where $\Gamma(\alpha)=\int_0^\infty t^{\alpha-1}e^{-t}dt$ (2.13)

The case where $\mu = 0$ and $\theta = 1$ is called the standard gamma distribution. The probability density function for the standard gamma distribution is:

$$f(x)=\frac{x^{\alpha-1}e^{-x}}{\Gamma(\alpha)},\ x \geq 0;\ \alpha>0 \tag{2.14}$$

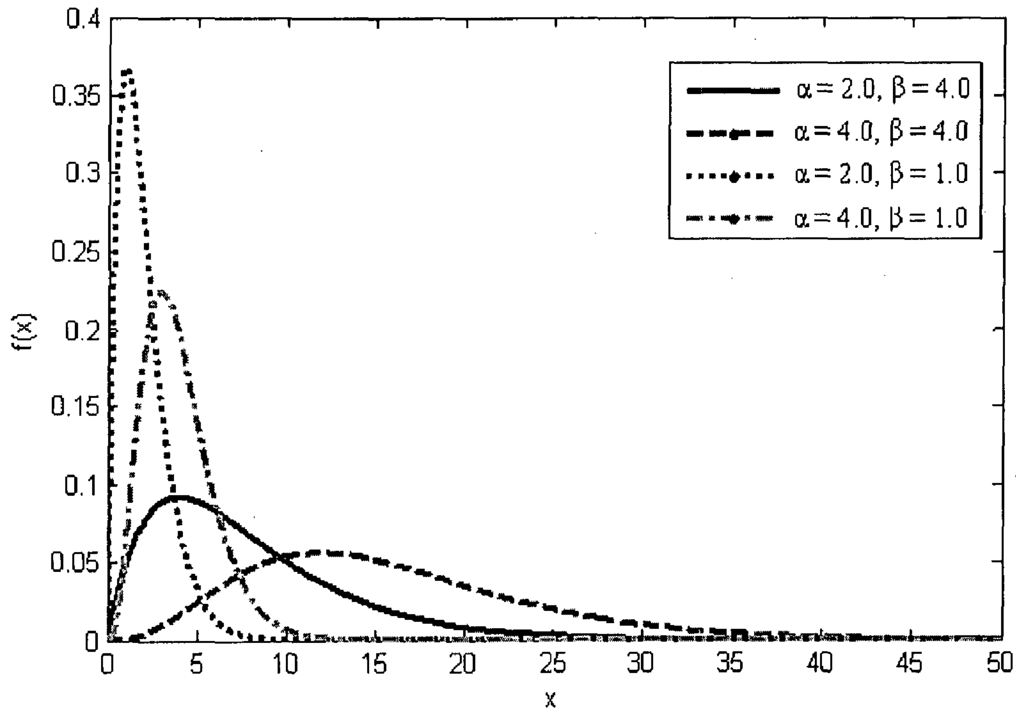

Figure 2.6 Probability density function of Gamma function

2.4 Proposed PSO and DE Algorithms

Based on the low discrepancy sequences and probability distributions mentioned in the previous sections, 6 versions of PSO and 6 versions of DE are proposed in this chapter. The computational steps of the proposed versions are as same as of basic PSO and DE algorithms and they differ only in the phase of initializing the population.

The six modified versions of PSO are:

- GPSO: PSO algorithm, initializing the population with Gaussian distributed random numbers
- EPSO: PSO algorithm, initializing the population with Exponential distributed random numbers
- BTPSO: PSO algorithm, initializing the population with Beta distributed random numbers
- GAPSO: PSO algorithm, initializing the population with Gamma distributed random numbers
- VC-PSO: PSO algorithm, initializing the population with the random numbers generated by Van der Corput sequence
- SO-PSO: PSO algorithm, initializing the population with the random numbers generated by Sobol sequence

The six modified versions of DE are:

- GDE: DE algorithm, initializing the population with Gaussian distributed random numbers
- EDE: DE algorithm, initializing the population with Exponential distributed random numbers
- BTDE: DE algorithm, initializing the population with Beta distributed random numbers
- GADE: DE algorithm, initializing the population with Gamma distributed random numbers
- VC-DE: DE algorithm, initializing the population with the random numbers generated by Van der Corput sequence
- SO-DE: DE algorithm, initializing the population with the random numbers generated by Sobol sequence

2.5 Parameter Settings

In order to check the compatibility of the proposed algorithms a test suite of five unconstrained, classical bench mark functions are considered, given in Appendix I, that are often used for deciding the credibility of an optimization algorithm. For each function, four different dimension sizes of 10, 20, 30 and 50 are taken. The maximum number of generations is set as 1000, 1500, 2000 and 3000 with population sizes of 20, 40 corresponding to the dimensions 10, 20, 30 and 50 respectively. Stopping criteria for all the algorithms is taken as the maximum number of generations. A total of 30 runs for each experimental setting are conducted and the average fitness of the best solutions throughout the run is recorded. For all the PSO algorithms, a linearly decreasing inertia weight (0.9 - 0.4) is used along with the user defined parameters $c_1 = c_2 = 2.0$. For all the DE versions, the scaling factor F is taken as 0.5 and the crossover rate Cr is taken as 0.9. The mean best fitness value and the percentage of improvement for all the given functions are recorded in Tables 2.1 – 2.6, in which P represents the swarm population, D represents the dimension and Gne represents the maximum number of permissible generations. Figure 2.7 (a) – 2.7 (e) shows the performance curves of PSO, GPSO, EPSO, BTPSO, GAPSO, VC-PSO and SO-PSO algorithms. The performance curves of DE, GDE, EDE, BTDE, GADE, VC-DE and SO-DE algorithms are shown in Figure 2.8 (a) – (c).

2.6 Numerical Results and Discussion

***Comparison of Proposed PSO algorithms with basic PSO*:**

Performance comparisons of proposed PSO algorithms with basic PSO in terms of mean best fitness values are given in Tables 2.1 and 2.2. Table 2.3 gives the improvement (%) of proposed versions of PSO with basic PSO. The numerical results of Table 2.1 and 2.2 show that in all the test cases (40 cases for each algorithm) except two cases in EPSO algorithm, the proposed versions of PSO perform much better than the basic PSO algorithm. If the comparison is made with the performance of proposed algorithms with each other then it can be seen that BTPSO gave better results than the other compared algorithms in 32 test cases out of 40 cases tried. In 4 test cases, the performance of VCPSO is better than others. EPSO algorithm performs well in 2 test cases, GPSO and GAPSO algorithms gave better result than the other algorithms in one case for each. Thus the numerical results show that the modified initialization methods improved the performance of PSO with a noticeable percentage, particularly the beta distribution is working well in the initialization process of PSO.

Comparison of Proposed DE algorithms with basic DE:

Performance comparisons of proposed DE algorithms with basic DE in terms of mean best fitness values are given in Tables 2.4 and 2.5. Table 2.6 gives the improvement (%) of proposed versions of DE with basic DE. From the numerical results it can be seen that all the proposed DE algorithms are either better or at par with basic DE algorithm. For each algorithm, a total of 40 test cases tried. If the comparison is made with the proposed algorithms with each other, then in most of the test cases VCDE algorithm, which follows the Vander Corput sequence for Initial population, gave better performance than other algorithms. VCDE algorithm is working well in 23 cases out of 40 cases. In 10 test cases BTDE, the DE algorithm which follows the beta distribution for initializing the population, is better than other compared algorithms. BTDE and VCDE algorithms are performed the same in 2 test cases and in 3 test cases; all the proposed algorithms are perform the same. Thus from the numerical results it is concluded that the modified initialization methods improved the performance of DE algorithm with a noticeable percentage, particularly the low discrepancy Van der Corput sequence is working well in the initialization process of DE.

Table 2.1 Comparison results of PSO, GPSO, EPSO, BTPSO and GAPSO

Function	P	D	Gne	PSO	GPSO	EPSO	BTPSO	GAPSO
RS	20	10	1000	5.5572	4.279237	5.194298	0.149243	3.930455
		20	1500	22.8892	19.450845	19.211362	1.542594	20.103193
		30	2000	47.2941	39.200201	45.251921	4.863791	44.89109
		50	3000	105.46585	89.870645	86.615777	6.288665	94.627789
	40	10	1000	3.5623	3.234385	2.785873	0.198991	3.427863
		20	1500	16.3504	14.228897	16.416922	2.159893	15.778632
		30	2000	38.5250	33.481344	32.68904	3.840700	31.840566
		50	3000	92.840048	74.711812	83.488755	5.689304	84.665796
GR	20	10	1000	0.0919	0.028665	0.034969	1.626e-20	0.003328
		20	1500	0.0313	0.009839	0.025832	3.385e-17	1.125e-16
		30	2000	0.0182	0.008239	0.012904	3.329e-13	0.001849
		50	3000	0.014701	0.002463	0.004312	6.519e-10	0.004557
	40	10	1000	0.0862	0.029166	0.034237	2.711e-21	0.006584
		20	1500	0.0286	0.013535	0.029871	6.776e-20	5.421e-20
		30	2000	0.0127	0.005053	0.012316	8.298e-16	3.027e-15
		50	3000	0.009857	0.003696	0.007507	2.365e-11	0.007010
RB	20	10	1000	96.1715	4.540158	3.497455	4.775826	6.864273
		20	1500	214.6764	22.654988	23.16014	15.132349	19.179048
		30	2000	316.4468	36.824102	52.497253	25.579255	41.526082
		50	3000	533.64808	86.800428	100.78990	45.529225	83.657666
	40	10	1000	70.2139	3.619349	4.022827	4.306128	3.624577
		20	1500	180.9671	19.152783	14.194387	14.483019	15.039252
		30	2000	299.7061	34.613255	49.837002	24.958234	29.332349
		50	3000	482.13533	68.619791	76.646435	44.33589	78.453189
ACK	20	10	1000	6.965e-12	4.830e-12	6.863e-13	2.521e-12	2.067e-12
		20	1500	3.560e-07	5.239e-08	6.186e-08	2.987e-08	6.168e-08
		30	2000	3.618e-05	8.495e-06	8.791e-06	1.197e-06	5.263e-06
		50	3000	3.43866	0.528656	0.861133	7.767e-05	0.569936
	40	10	1000	7.897e-13	3.660e-13	2.301e-13	1.477e-14	1.435e-13
		20	1500	5.045e-08	1.567e-08	1.662e-08	1.002e-09	9.988e-09
		30	2000	7.269e-06	4.119e-06	3.495e-06	1.093e-07	4.040e-07
		50	3000	1.966554	0.080034	0.101208	8.179e-06	0.001329
SWF2.21	20	10	1000	4.831e-05	8.572e-06	9.252e-06	1.055e-06	9.064e-06
		20	1500	1.501872	0.163695	0.174117	0.011875	0.187012
		30	2000	10.918735	0.781431	0.96442	0.137883	0.891136
		50	3000	33.567141	1.646942	2.527187	0.300847	2.177328
	40	10	1000	7.421e-07	2.193e-07	1.508e-07	2.967e-08	1.571e-07
		20	1500	0.343671	0.038344	0.038119	0.004469	0.039788
		30	2000	7.481037	0.53186	0.563958	0.076697	0.500093
		50	3000	29.928682	1.59299	2.393682	0.324051	2.079004

Table 2.2 Comparison results of PSO, VC-PSO and SO-PSO

Function	P	D	Gne	PSO	VC-PSO	SO-PSO
RS	20	10	1000	5.5572	3.880492	4.647559
		20	1500	22.8892	19.61551	22.2139
		30	2000	47.2941	40.79308	41.61307
		50	3000	105.4659	81.50977	88.21129
	40	10	1000	3.5623	3.351035	2.984863
		20	1500	16.3504	15.91926	16.0216
		30	2000	38.525	31.75592	34.68411
		50	3000	92.84005	62.149	66.01864
GR	20	10	1000	0.0919	0.003326	0.001229
		20	1500	0.0313	0.002583	0.000986
		30	2000	0.0182	0.000986	7.65e-12
		50	3000	0.014701	1.85e-09	2.16e-09
	40	10	1000	0.0862	0.005169	0.001725
		20	1500	0.0286	0.003194	8.67e-20
		30	2000	0.0127	3.06e-14	3.28e-14
		50	3000	0.009857	3.77e-11	5.84e-11
RB	20	10	1000	96.1715	3.650252	4.233931
		20	1500	214.6764	13.44568	16.28811
		30	2000	316.4468	32.39934	33.61518
		50	3000	533.6481	84.53622	86.26835
	40	10	1000	70.2139	4.041561	3.908925
		20	1500	180.9671	12.40308	15.8566
		30	2000	299.7061	31.89796	36.66065
		50	3000	482.1353	81.06562	83.58392
ACK	20	10	1000	6.97e-12	3.11e-12	3.00e-12
		20	1500	3.56e-07	8.57e-08	6.68e-08
		30	2000	3.62e-05	2.65e-06	8.69e-06
		50	3000	3.43866	2.02e-03	5.69e-04
	40	10	1000	7.90e-13	5.61e-14	3.50e-14
		20	1500	5.05e-08	2.65e-09	2.51e-09
		30	2000	7.27e-06	4.42e-07	6.69e-07
		50	3000	1.966554	1.37e-04	1.09e-05
SWF2.21	20	10	1000	4.83e-05	2.21e-11	4.42e-06
		20	1500	1.501872	0.069028	0.086019
		30	2000	10.91874	0.443966	0.437305
		50	3000	33.56714	0.828851	0.802782
	40	10	1000	7.42e-07	4.65e-15	5.89e-08
		20	1500	0.343671	0.016538	0.020741
		30	2000	7.481037	0.32125	0.280413
		50	3000	29.92868	0.706595	0.508336

Table 2.3 Improvement (%) of proposed PSO algorithms in comparison with basic PSO

Function	P	D	Gne	GPSO	EPSO	BTPSO	GAPSO	VC-PSO	SO-PSO
RS	20	10	1000	22.9	6.5	97.3	29.3	30.2	16.4
		20	1500	15.0	16.1	93.3	12.2	14.3	2.95
		30	2000	17.1	4.3	89.7	5.1	13.7	12.0
		50	3000	14.8	17.9	94.0	10.3	22.7	16.4
	40	10	1000	9.2	21.8	94.4	3.8	5.93	16.2
		20	1500	12.9	-	86.8	3.5	2.64	2.01
		30	2000	13.1	15.1	95.2	17.4	17.6	9.97
		50	3000	19.5	10.1	93.9	8.8	33.1	28.9
GR	20	10	1000	68.8	61.9	100	96.4	96.4	98.7
		20	1500	68.6	17.5	100	100	91.7	96.8
		30	2000	54.7	29.1	100	89.8	94.6	100
		50	3000	83.2	70.7	100	69.0	100	100
	40	10	1000	66.2	60.3	100	92.4	94.0	98.0
		20	1500	52.7	-	100	100	88.8	100
		30	2000	60.2	3.0	100	100	100	100
		50	3000	62.5	23.8	100	28.9	100	100
RB	20	10	1000	95.3	96.7	95.0	92.9	96.2	95.6
		20	1500	89.4	89.2	92.9	91.1	93.7	92.4
		30	2000	88.4	83.4	91.9	86.9	89.8	89.4
		50	3000	83.7	81.1	91.5	84.3	84.2	83.8
	40	10	1000	94.8	94.3	93.9	94.8	94.2	94.4
		20	1500	89.4	92.2	91.9	91.7	93.1	91.2
		30	2000	88.5	83.4	91.7	90.2	89.4	87.8
		50	3000	85.8	84.1	90.8	83.7	83.2	82.7
ACK	20	10	1000	30.7	90.1	63.8	70.3	55.3	57.0
		20	1500	85.3	82.6	91.6	82.7	75.9	81.2
		30	2000	76.5	75.7	96.7	85.5	92.7	76.0
		50	3000	84.6	75.0	100	83.4	99.9	100
	40	10	1000	53.7	70.9	98.1	81.8	92.9	95.6
		20	1500	68.9	67.1	98.0	80.2	94.7	95.0
		30	2000	43.3	51.9	98.5	94.4	93.9	90.8
		50	3000	95.9	94.9	100	99.9	100	100
SWF2.21	20	10	1000	82.3	80.8	97.8	81.2	100	90.9
		20	1500	89.1	88.4	99.2	87.5	95.4	94.3
		30	2000	92.8	91.2	98.7	91.8	95.9	96.0
		50	3000	95.1	92.5	99.1	93.5	97.5	97.6
	40	10	1000	70.4	79.7	96.0	78.8	100	92.1
		20	1500	88.8	88.9	98.7	88.4	95.2	94.0
		30	2000	92.9	92.5	99.0	93.3	95.7	96.3
		50	3000	94.7	92.0	98.9	93.1	97.6	98.3

Table 2.4 Comparison results of DE, GDE, EDE, BTDE and GADE

Function	P	D	Gne	DE	GDE	EDE	BTDE	GADE
RS	20	10	1000	4.14561	2.87204	4.06888	0.000177	4.1401
		20	1500	15.2779	10.6392	13.3219	0.239455	13.262
		30	2000	29.73	19.3196	29.1693	0.174208	27.2259
		50	3000	74.4649	46.0006	44.3187	1.82141	61.7617
	40	10	1000	5.53601	3.41977	5.17709	0.00000	2.84298
		20	1500	23.0242	19.8479	17.0667	0.00000	13.7773
		30	2000	27.5721	24.4541	24.305	0.497477	17.5868
		50	3000	35.8071	31.8632	30.7283	0.299003	35.7962
GR	20	10	1000	0.067271	1.45e-06	0.000227	1.95e-07	1.27e-05
		20	1500	0.615169	0.004686	0.00436	7.13e-06	0.007953
		30	2000	2.20295	0.001803	0.008911	5.35e-05	0.012796
		50	3000	6.95086	0.005007	0.025115	0.000291	0.030986
	40	10	1000	0.020922	0.00000	0.00000	0.00000	0.00000
		20	1500	0.002464	0.00000	0.00000	0.00000	0.00000
		30	2000	0.008624	0.00000	0.00000	0.00000	0.00000
		50	3000	0.016103	1.10e-08	4.57e-17	2.17e-20	7.18e-09
RB	20	10	1000	25.0847	7.93441	7.0691	8.11989	4.66401
		20	1500	67.711	21.653	38.7838	18.276	41.3566
		30	2000	71.6169	45.539	73.7554	28.1323	49.939
		50	3000	148.814	88.337	123.72	48.3172	114.997
	40	10	1000	5.82618	5.53632	5.10122	6.99747	4.37098
		20	1500	15.1055	15.923	15.931	16.7332	14.9939
		30	2000	27.6945	26.0006	26.5588	27.1433	44.7165
		50	3000	40.6621	47.6049	50.4318	47.2391	78.9742
ACK	20	10	1000	0.234128	0.119102	0.002729	0.000129	0.003586
		20	1500	2.93381	0.036436	0.497017	0.004674	0.975587
		30	2000	3.91141	0.395057	0.694482	0.014302	1.34968
		50	3000	7.95975	0.809041	1.41343	0.052592	2.2413
	40	10	1000	5.00e-16	1.45e-16	8.55e-16	1.45e-16	1.45e-16
		20	1500	3.70e-15	3.70e-15	3.70e-15	3.70e-15	3.70e-15
		30	2000	2.04e-14	1.09e-05	6.54e-15	3.70e-15	1.32e-07
		50	3000	1.13794	0.000451	0.015031	0.000158	0.102721
SWF2.21	20	10	1000	5.60763	0.111252	0.123874	0.00402	0.302069
		20	1500	19.2351	0.502519	0.558345	0.047622	0.873595
		30	2000	29.5582	0.541165	0.881994	0.099099	1.09369
		50	3000	35.1416	0.813952	1.17845	0.183615	1.31309
	40	10	1000	0.011005	0.002528	3.77e-05	8.83e-06	0.004511
		20	1500	3.07155	0.084686	0.116829	0.009763	0.246686
		30	2000	12.9832	0.286121	0.405295	0.040504	0.669035
		50	3000	25.4099	0.523261	0.777208	0.112097	0.954586

Table 2.5 Comparison results of DE, VC-DE and SO-DE

Function	P	D	Gne	DE	VC-DE	SO-DE
RS	20	10	1000	4.14561	0.198991	3.98875
		20	1500	15.2779	0.497477	9.24878
		30	2000	29.73	11.442	16.4139
		50	3000	74.4649	2.68638	30.233
	40	10	1000	5.53601	0.00000	1.69142
		20	1500	23.0242	0.00000	7.66115
		30	2000	27.5721	10.049	13.5318
		50	3000	35.8071	0.198991	22.486
GR	20	10	1000	0.067271	0.00000	0.001461
		20	1500	0.615169	0.00000	0.011825
		30	2000	2.20295	0.00000	0.005593
		50	3000	6.95086	5.42e-20	0.007802
	40	10	1000	0.020922	0.00000	0.00000
		20	1500	0.002464	0.00000	0.00000
		30	2000	0.008624	0.00000	0.00000
		50	3000	0.016103	5.42e-20	5.53e-07
RB	20	10	1000	25.0847	3.11355	4.1753
		20	1500	67.711	13.4495	28.1569
		30	2000	71.6169	37.4939	50.0921
		50	3000	148.814	45.2356	122.317
	40	10	1000	5.82618	2.64441	2.18146
		20	1500	15.1055	13.6994	11.3377
		30	2000	27.6945	23.6749	24.0394
		50	3000	40.6621	46.8276	56.9271
ACK	20	10	1000	0.234128	2.28e-15	0.117151
		20	1500	2.93381	5.12e-15	1.03826
		30	2000	3.91141	1.98623	1.01891
		50	3000	7.95975	1.51e-14	1.09739
	40	10	1000	5.00e-16	1.21e-16	1.45e-16
		20	1500	3.70e-15	4.41e-16	3.70e-15
		30	2000	2.04e-14	1.97e-15	5.47e-15
		50	3000	1.13794	1.79e-14	9.83e-07
SWF2.21	20	10	1000	5.60763	1.05e-08	0.319372
		20	1500	19.2351	5.40e-05	0.579449
		30	2000	29.5582	0.296617	0.618857
		50	3000	35.1416	0.019521	0.618857
	40	10	1000	0.011005	5.90e-09	0.004488
		20	1500	3.07155	1.20e-05	0.381279
		30	2000	12.9832	0.000362	0.407118
		50	3000	25.4099	0.001418	0.51291

Table 2.6 Improvement (%) in terms of average fitness function value of proposed DE algorithms in comparison with basic DE

Function	P	D	Gne	GDE	EDE	BTDE	GADE	VC-DE	SO-DE
RS	20	10	1000	30.7	1.85	100	0.13	95.2	3.78
		20	1500	30.4	12.8	98.43	13.2	96.7	39.5
		30	2000	35	1.89	99.41	8.42	61.5	44.8
		50	3000	38.2	40.5	97.55	17.1	96.4	59.4
	40	10	1000	38.2	6.48	100	48.6	100	69.4
		20	1500	13.8	25.9	100	40.2	100	66.7
		30	2000	11.3	11.8	98.2	36.2	63.6	50.9
		50	3000	11	14.2	99.16	0.03	99.4	37.2
GR	20	10	1000	100	99.7	100	100	100	97.8
		20	1500	99.2	99.3	100	98.7	100	98.1
		30	2000	99.9	99.6	100	99.4	100	99.7
		50	3000	99.9	99.6	100	99.6	100	99.9
	40	10	1000	100	100	100	100	100	100
		20	1500	100	100	100	100	100	100
		30	2000	100	100	100	100	100	100
		50	3000	100	100	100	100	100	100
RB	20	10	1000	68.4	71.8	67.63	81.4	87.6	83.4
		20	1500	68	42.7	73.01	38.9	80.1	58.4
		30	2000	36.4	-	60.72	30.3	47.6	30.1
		50	3000	40.6	16.9	67.53	22.7	69.6	17.8
	40	10	1000	4.98	12.4	-	25	54.6	62.6
		20	1500	-	-	-	0.74	9.31	24.9
		30	2000	6.12	4.1	1.99	-	14.5	13.2
		50	3000	-	-	-	-	-	-
ACK	20	10	1000	49.1	98.8	99.95	98.5	100	50
		20	1500	98.8	83.1	99.84	66.7	100	64.6
		30	2000	89.9	82.2	99.63	65.5	49.2	74
		50	3000	89.8	82.2	99.34	71.8	100	86.2
	40	10	1000	71.1	-	71.07	71.1	75.8	71.1
		20	1500	0	0	0	0	88.1	0
		30	2000	-	67.9	81.87	-	90.3	73.2
		50	3000	100	98.7	99.99	91	100	100
SWF2.21	20	10	1000	98	97.8	99.93	94.6	100	94.3
		20	1500	97.4	97.1	99.75	95.5	100	97
		30	2000	98.2	97	99.66	96.3	99	97.9
		50	3000	97.7	96.6	99.48	96.3	99.9	98.2
	40	10	1000	77	99.7	99.92	59	100	59.2
		20	1500	97.2	96.2	99.68	92	100	87.6
		30	2000	97.8	96.9	99.69	94.8	100	96.9
		50	3000	97.9	96.9	99.56	96.2	100	98

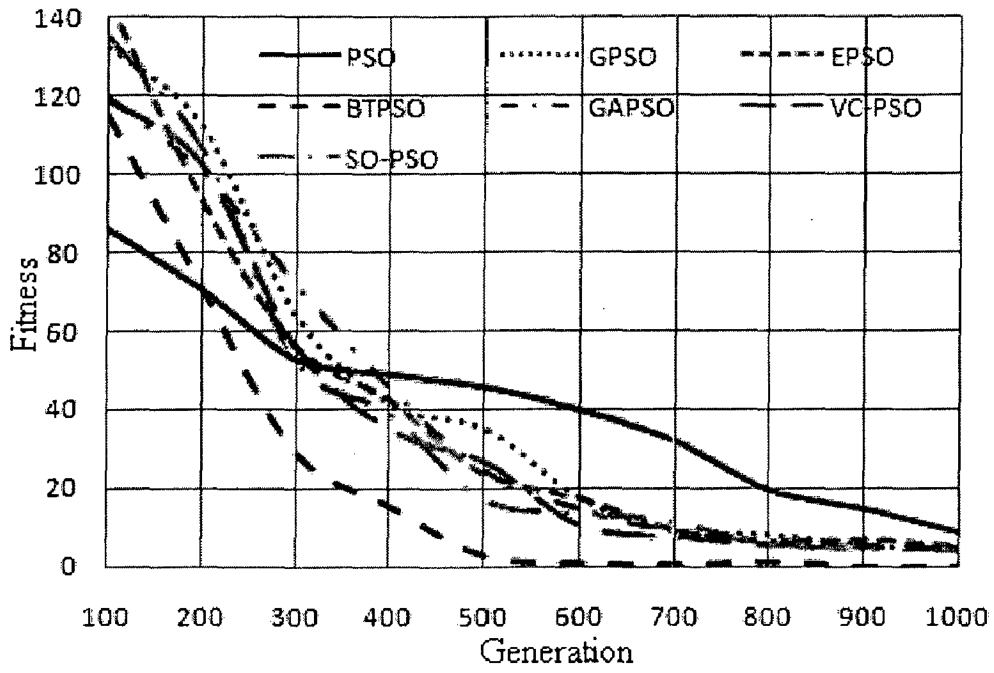

Figure 2.7 (a) Function RS

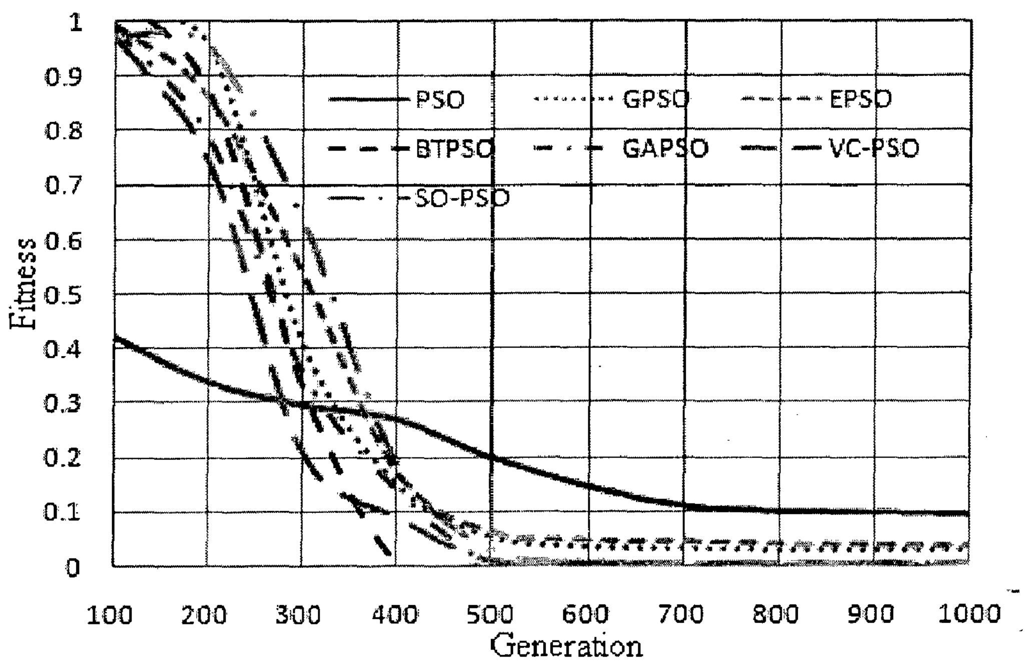

Figure 2.7 (b) Function GR

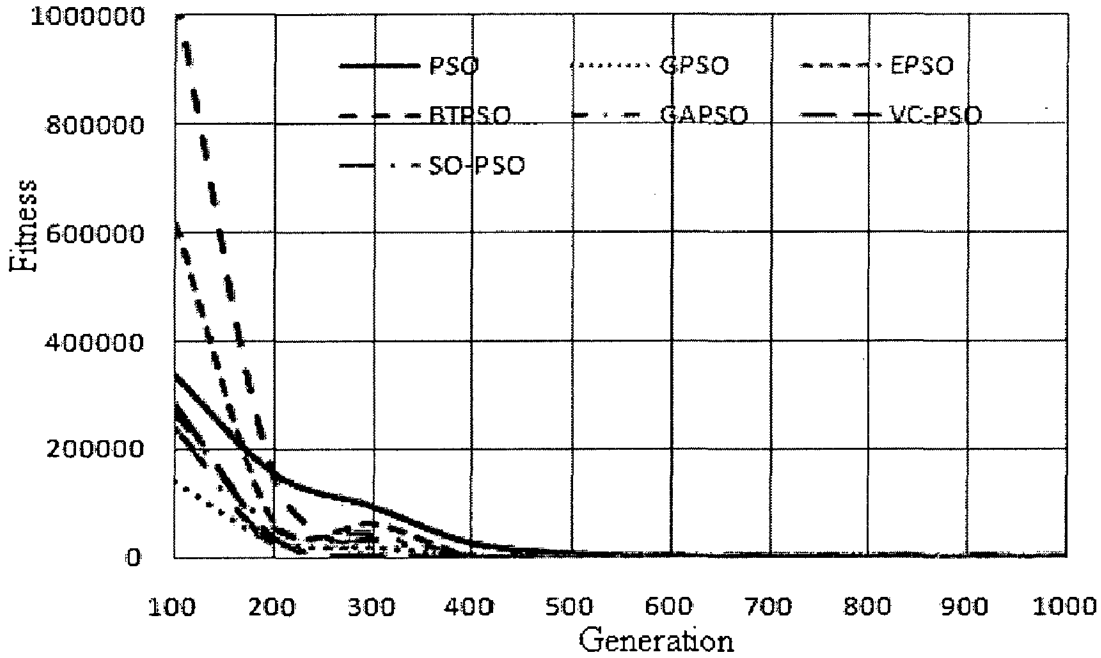

Figure 2.7 (c) Function RB

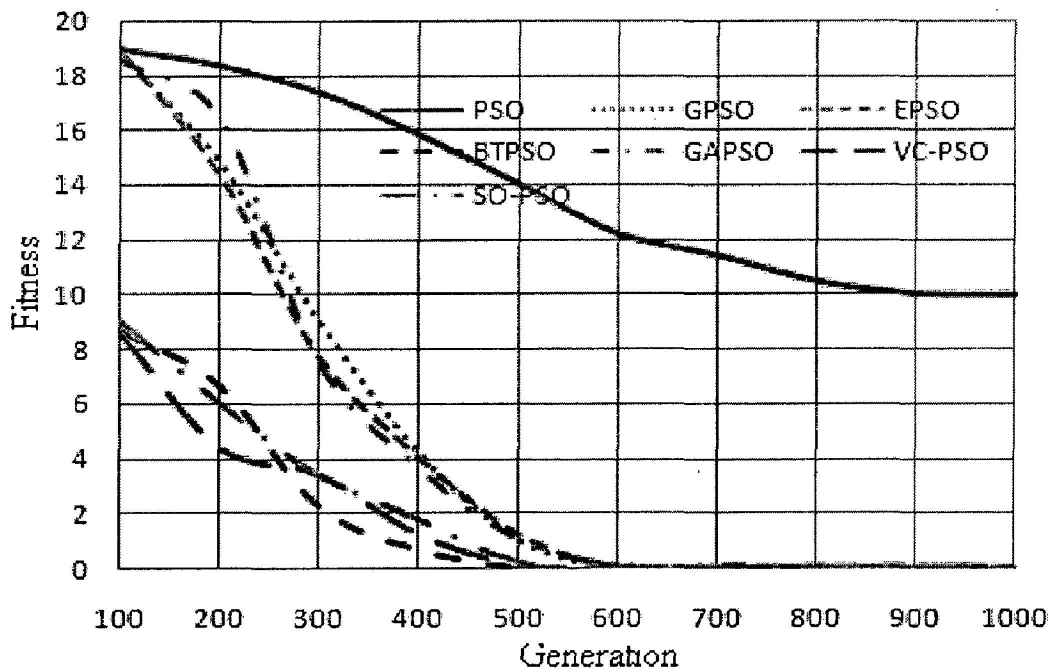

Figure 2.7 (d) Function ACK

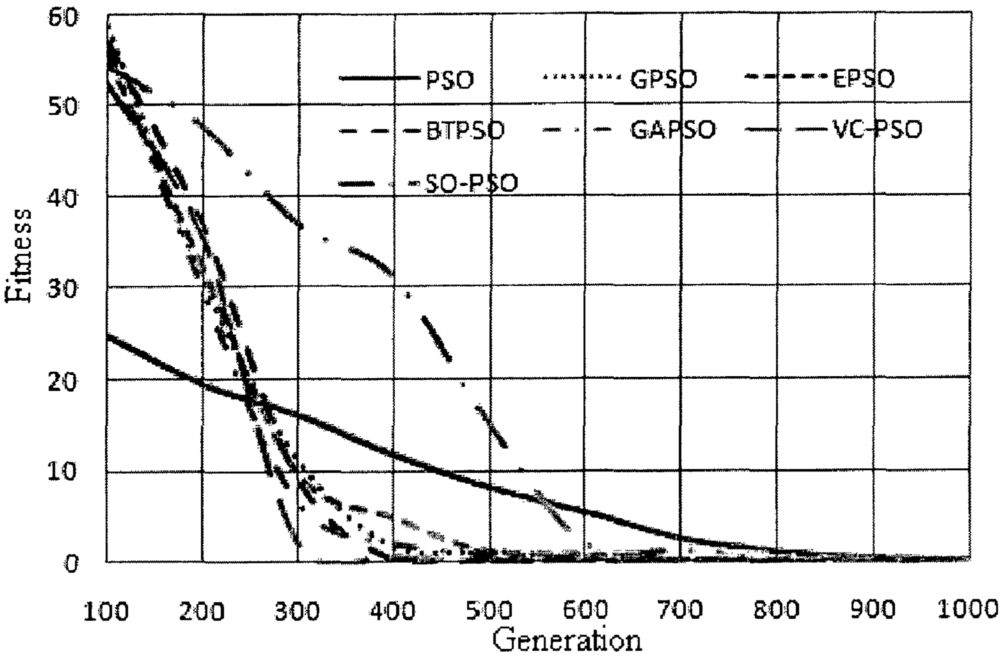

Figure 2.7 (e) Function SWF2.21

Figure 2.7 Performance curves of PSO, GPSO, EPSO, BTPSO, GAPSO, VC-PSO and SO-PSO algorithms

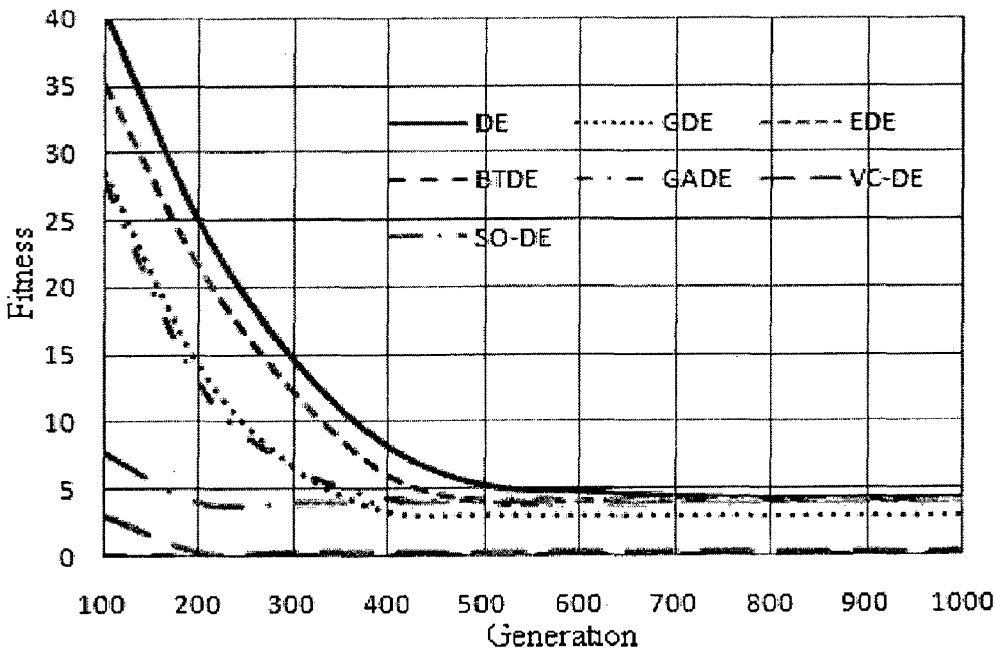

Figure 2.8 (a) Function RS

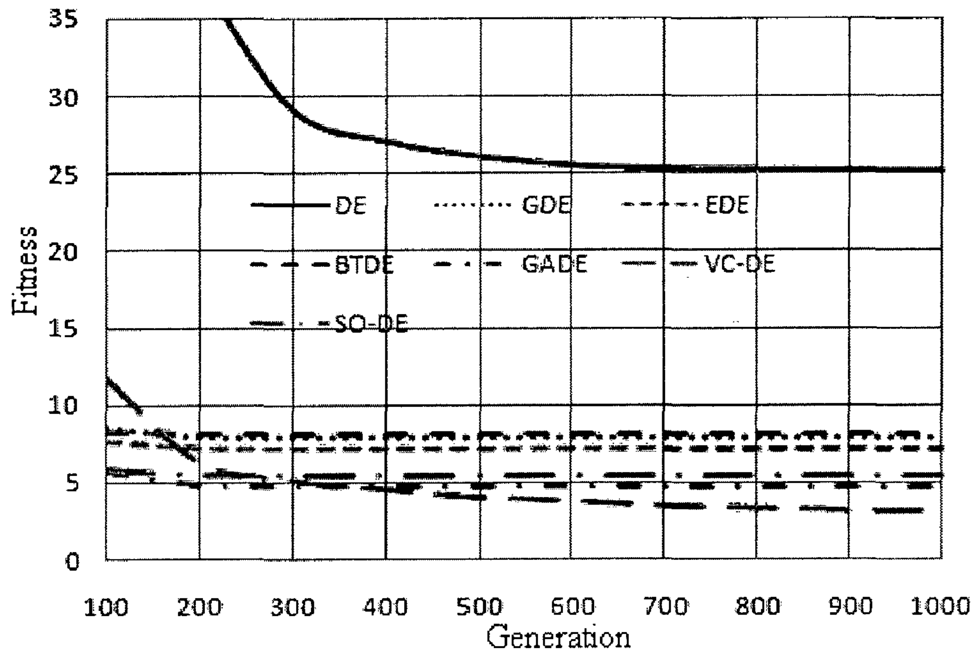

Figure 2.8 (b) Function RB

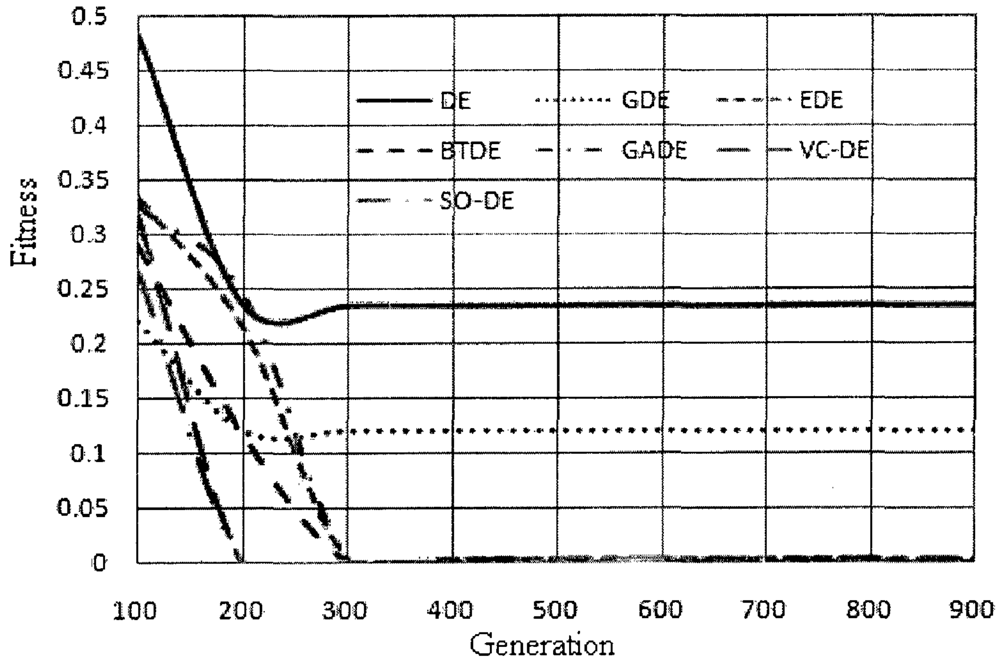

Figure 2.8 (c) Function ACK

Figure 2.8 Performance curves of DE, GDE, EDE, BTDE, GADE, VC-DE and SO-DE algorithms

2.7 Conclusion

This chapter presented some modified versions of PSO and DE algorithms. These algorithms are differing from the basic versions only in the place of initializing the population. The probability distributions: Gaussian, Exponential, Beta and Gamma distribution and the low discrepancy sequences: Van der Corput and Sobol were considered in this study for initializing the population of PSO and DE. With respect to the above mentioned probability distributions and low discrepancy sequences, a total of 12 modified versions of PSO and DE (6 versions for each algorithm) were reported. The presented algorithms were tested with five standard benchmark problems with different dimensions (10, 20, 30 and 50) and different population sizes (20 and 40) and the results are compared with the basic versions of PSO and DE which follows the uniform distribution for initializing the swarm. The numerical results show that only with the change in the initial distribution of random numbers the improvement in average fitness function value is as high as 100% in many of the test cases. In overall comparison, the algorithms which follow the Beta distribution and Van der Corput sequence were superior to other algorithms.

Chapter 3

Improved Particle Swarm Optimization Algorithms

[This chapter describes the improved versions of Particle Swarm Optimization algorithm. The main focus is on the design and implementation of the improved PSO algorithms based on diversity, Crossover and Mutation using different distributions and Low-discrepancy sequences. Also this chapter introduces a new velocity vector and an inertia weight in classical PSO.]

3.1 Introduction

Many variants of PSO have been developed in the past to improve its performance. Some of the interesting modifications that helped in enhancing the performance of PSO include introduction of inertia weight and its adjustment for better control of exploration and exploitation capacities of the swarm (Shi and Eberhart, 1998; Eberhart and Shi 2001), introduction of constriction factor to control the magnitudes of velocities (Clerc, 1999), impacts of various neighborhood topologies on the swarm (Kennedy, 1999), extension of PSO via genetic programming (Poli et al, 2005), use of various mutation operators into PSO (Ting et al, 2003; Paquet and Engelbrecht, 2003; Parsopoulos et al, 2001). This chapter proposes some variants of PSO based on diversity, mutation and crossover. Also a new inertia weight and a velocity vector are introduced.

This chapter has ten sections including the introduction. In section 3.2, two simple diversity guided PSO are given and in Section 3.3, four diversity based mutation versions of PSO are given. In section 3.4, four modified versions PSO with crossover operator is described. Section 3.5 gives two mutation based variants of PSO; Section 3.6 and 3.7 introduces a new inertia weight and a new velocity vector in classical PSO respectively. Parameter settings are given in section 3.8 and the result analysis are given in section 3.9. The chapter finally concludes with section 3.10.

3.2 Diversity Based Simple Variants of PSO

Diversity is a very important aspect in population-based optimization algorithms. Large diversity directly implies that a large area of the search space can be explored. It may be defined as the dispersion of potential candidate solutions in the search space.

The diversity measure of the swarm can be calculated as (Krink et al, 2002; Vesterstrom et al, 2002):

$$Diversity(S(t)) = \frac{1}{n_s}\sum_{i=1}^{n_s}\sqrt{\sum_{j=1}^{n_x}(x_{ij}(t) - \overline{x_j(t)})^2} \tag{3.1}$$

where S is the swarm, $n_s = |S|$ is the swarm size, n_x is the dimensionality of the problem, x_{ij} is the j'th value of the i'th particle and $\overline{x_j(t)}$ is the average of the j-th dimension over all particles, i.e.

$$\overline{x_j(t)} = \frac{\sum_{i=1}^{n_s} x_{ij}(t)}{n_s} \tag{3.2}$$

Riget et al (2002) gives an alternate formula for calculating the swarm diversity, it is based on the diameter of the swarm.

$$Diversity(S(t)) = \frac{1}{diameter(S(t))}\frac{1}{n_s}\sum_{i=1}^{n_s}\sqrt{\sum_{j=1}^{n_x}(x_{ij}(t) - \overline{x_j(t)})^2} \tag{3.3}$$

Where $diameter(S(t))$ is the diameter of the swarm, i.e. the distance between the two furthest apart particles. Interested readers may refer to (Engelbrecht, 2005; Olurunda and Engelbrecht, 2008; Shi and Eberhart, 2008) for different formulae used for calculating diversity.

Most of the population based search techniques 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 (Liu et al, 2007). Loss of diversity becomes more prominent for multimodal functions having several optima or noisy functions where the optimum keeps shifting from one position to other. Loss of diversity generally takes place when the balance between the two antagonists processes exploration (searching of the search space) and exploitation (convergence towards the optimum) is disturbed. In case of Evolutionary

Algorithms (EA) the population diversity is generally lost during the process of evolution (crossover and mutation), whereas in case of PSO the diversity loss is generally attributed to the fast information flow between the swarm particles. Thus in absence of a good diversity enhancing mechanism the optimization algorithms are unable to explore the search space effectively.

Previously, Riget et al (2002) proposed a diversity guided PSO called ARPSO, in which, when diversity of population drops below a lower bound, d_{low}, it will be switched to the repulsion phase, in which the diversity will increase. Finally, when a diversity of d_{high} is reached, it will be switched back to the attraction phase. PSOBC algorithm is also a diversity guided algorithm proposed by Niu et al (2006). In PSOBC, the individual particle was repelled by the worst known particle position and its own previous worst position and it was proved that it much more like the nature works than ARPSO. In DRPSO (Jiang et al, 2008),the individual particle was repelled by the worst known particle position and its own previous worst position, at the same time particles do diffusion movement in the process of repulsion.

This section presents two algorithms Attraction – Repulsion PSO (ATREPSO) and Quadratic Interpolation PSO1 (QIPSO1) which uses different diversity enhancing mechanisms to improve the performance of the swarm. These two algorithms 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 QIPSO1 evolutionary operator crossover is induced in the swarm to perturb the population. These algorithms are described in the following subsections.

3.2.1 Attraction – Repulsion Particle Swarm Optimization (ATREPSO)

The Attraction – Repulsion Particle Swarm Optimization Algorithm (ATREPSO) is a simple extension of the Attractive and Repulsive PSO (ARPSO) proposed by Riget et al (2002), where a third phase called *in between* phase or the phase of *positive conflict* is added. It is quite natural to think that (diversity <) d_{low} and (diversity >) d_{high} may not be the only two possibilities for deciding the movement of the swarm, but many times the diversity may lie in between the two threshold values. For this reason a third phase is proposed, which is activated when the diversity is greater than d_{low} but less then d_{high}. 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)

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}) \quad (3.4)$$

Repulsion phase (particles are repelled from the optimal position)

$$v_{id} = wv_{id} - c_1 r_1 (p_{id} - x_{id}) - c_2 r_2 (p_{gd} - x_{id}) \quad (3.5)$$

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

$$v_{id} = wv_{id} + c_1 r_1 (p_{id} - x_{id}) - c_2 r_2 (p_{gd} - x_{id}) \quad (3.6)$$

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

$$v_{id} = \begin{cases} wv_{id} + c_1 r_1 (p_{id} - x_{id}) + c_2 r_2 (p_{gd} - x_{id}), & div > d_{high} \\ wv_{id} + c_1 r_1 (p_{id} - x_{id}) - c_2 r_2 (p_{gd} - x_{id}), & d_{low} < div < d_{high} \\ wv_{id} - c_1 r_1 (p_{id} - x_{id}) - c_2 r_2 (p_{gd} - x_{id}), & div < d_{low} \end{cases} \quad (3.7)$$

Here *div* represents the diversity of the swarm. The idea behind the introduction of third phase is to improve the exploring and exploiting capabilities of ARPSO.

3.2.2 Quadratic Interpolation Particle Swarm Optimization (QIPSO1)

The Quadratic Interpolation Particle Swarm Optimization (QIPSO1) algorithm described in this section uses the concept of reproduction. Not many references are available in literature on the use of reproduction operator. One of the earlier references on the use of reproduction operator can be found in Clerc (2001). The reproduction operator applied in QIPSO1 is a quadratic interpolation (QI) operator. The quadratic interpolation operator is a nonlinear reproduction operator which makes use of three particles of the swarm to produce a new particle. QI is a gradient free direct search technique used for solving nonlinear optimization problems. Mathematically, the point generated by QI lies at the point of minima of the quadratic curve passing through three points. This concept has been used earlier in controlled random search technique (Mohan and Shanker, 1994; Ali and Torn, 2003). The method of quadratic interpolation takes advantages of the fact that a second-order polynomial often provides a good approximation to the shape of the function near optimum.

In QIPSO1 algorithm in order to provide more randomness to explore the search space the particle having the best fitness value (global best particle, P_g) is always selected whereas the other two points are randomly chosen from the remaining population. Mathematically, the working of QI operator may be defined as:

If $a = P_g$ represents the global best position of the swarm and let b, c represent randomly chosen swarm particles such that a $\neq$ b $\neq$c, then the coordinates of the new point generated by using ***QI operator*** is given as $\bar{x} = (x_1, x_2, ..., x_n)$, where

$$x_i = \frac{1}{2}\frac{(b_i^2 - c_i^2) * f(a) + (c_i^2 - a_i^2) * f(b) + (a_i^2 - b_i^2) * f(c)}{(b_i - c_i) * f(a) + (c_i - a_i) * f(b) + (a_i - b_i) * f(c)} \quad (3.8)$$

QIPSO1 is a simple and modified version of PSO with an added reproduction operator to enhance the performance of PSO without disturbing the inherent features of PSO. Like ARPSO (Riget et al, 2002) and ATREPSO, QIPSO1 algorithm uses diversity as a measure to guide the swarm, but instead of repulsing the population points, it makes use of reproduction operator to explore the promising areas of the search domain. When the diversity becomes less than d_{low}, then the QI operator is activated to generate a new potential candidate solution. The process is repeated iteratively till the diversity reaches the specified threshold d_{high}.

3.3 Diversity Based Mutation Versions of PSO

One of the simplest methods to overcome the problem of diversity loss is to capitalize the strengths of EA and PSO together in an algorithm. A variety of methods combining the aspects of EA and PSO are available in literature (Robinson et al, 2002; Shi and Krohling, 2002; Shi et al, 2003; Zhang and Xie, 2003; Hao et al, 2007; Yang et al, 2007a) etc. Out of the EA operators, mutation is the most widely used EA tool applied in PSO (Hu et al, 2003; Juang, 2003). The concept of mutation is quite common to Evolutionary Programming and Genetic

Algorithms. Mutation has been introduced into the PSO as a mechanism to increase the diversity of PSO, and by doing so improving the exploration abilities of the algorithm. Mutation can be applied to different elements of a particle swarm. The effect of mutation depends on which elements of the swarm are mutated. If only the neighborhood best position vectors are mutated, then effect is minimal, compared to mutation of particle position vectors. Velocity vector mutation is equivalent to particle's position vector mutation, under the condition that the same mutation operator is considered.

There are several instances in PSO where mutation is introduced in the swarm. Some mutation operators that have been applied to mutate the position vector in PSO include Gaussian (Wei at al, 2002; Higashi and Iba, 2003; Secrest and Lamont, 2003; Krohling, 2005; Sriyanyong, 2008), Cauchy (Stacy et al, 2003; Krohling, 2005), Chaos mutation (Dong et al, 2008; Yang at al, 2009; Yue-lin et al, 2008) etc.

3.3.1 Proposed Diversity Based Mutation Algorithms

Based on diversity based mutation, four modified versions of PSO are proposed in this section. They are: Gaussian Mutation PSO (GMPSO), PSO with Beta Mutation (BMPSO), PSO with Gamma Mutation (GAMPSO), and Beta & Gamma mutation PSO (BGMPSO). The GMPSO algorithm uses ***Gaussian mutation operator*** with the help of Gaussian distribution to mutate the particle; Likewise BMPSO and GAMPSO algorithms use ***Beta mutation operator*** and ***Gamma mutation operator*** respectively. The average of beta and gamma distributed random numbers are used to mutate the particles in BGMPSO algorithm and the mutation operator is called as ***Beta Gamma mutation operator***.

The four mutation operators are defined as:

Gaussian mutation operator:

$$X_i^{t+1} = X_i^t + \eta * N(0,1) \tag{3.9}$$

Where $N(0,1)$ is a random number generated by Gaussian distribution with mean zero and standard deviation one and η is a scaling parameter.

Beta mutation operator:

The Beta mutation operator is Evolutionary Programming based mutation operator.

$$X_i^{t+1} = X_i^t + \sigma_i' * Betarand_j() \tag{3.10}$$

where, $\sigma_i' = \sigma_i * \exp(\tau\ N(0,1) + \tau'\ N_j(0,1))$ (3.11)

$N(0,1)$ denotes a normally distributed random number with mean zero and standard deviation one. $N_j(0,1)$ indicates that a different random number is generated for each value of j. τ and τ' are set as $1/\sqrt{2n}$ and $1/\sqrt{2\sqrt{n}}$ respectively. *Betarand$_j$* () is a random number generated by beta distribution with parameters less than 1.

Gamma mutation operator:

The mutation operator is similar to beta mutation operator; but instead of using beta distribution, it used gamma distribution.

$$X_i^{t+1} = X_i^t + \sigma_i' * Gammarand_j() \tag{3.12}$$

Here σ_i' is same as of Eqn. (3.11).

Beta Gamma mutation operator:

$$X_i^{t+1} = X_i^t + \sigma_i' * 0.5 * (Betarand_j() + Gammarand_j()) \tag{3.13}$$

Here σ_i' is same as of Eqn. (3.11).

The four diversity based mutation algorithms have two phases namely attractive phase and mutation phase. The attractive phase is also define as the classical PSO, while in the mutation phase the swarm particles position vectors are mutated by using one of the above mentioned mutation operator. Also, the proposed algorithms use diversity threshold values d_{low} and d_{high} to guide the movement of the swarm. The threshold values are predefined by the user. The general c++ style code for applying diversity based mutation operator is given below:

```
Initialize the population
Do
    If (diversity < d_low)
        {Apply mutation operator}
    Else
        {Apply the usual position and velocity update equations of PSO}
    End if
    Update personal and global best positions of the particles
```

Until stopping criteria is reached

The proposed algorithms start with classical PSO i.e. it uses attractive phase (Eqn. (1.3)) for updating velocity vector and uses (1.2) for updating position vector. In the attractive phase the swarm is contracting, and consequently the diversity decreases. When the diversity of population drops below a lower bound, d_{low}, it will be switched to the mutation phase, with the hope to increase the diversity of the swarm population. This process is repeated until a maximum number iteration is reached or the stopping criterion is reached.

3.4 Crossover Based Variants of PSO

In PSO, the particles or members of the swarm fly through a multidimensional search space looking for a potential solution. When particles are exploring the search space, if some particle finds the current best position, the others will fly toward it. If the best position is a local optimum, particles cannot explore over again in the search space. Consequently, the algorithm will be trapped into the local optimum, results a premature convergence. This problem becomes more persistent in case of highly multimodal problems having several global and local optima. This drawback of PSO is due to the lack of diversity, which forces the swarm particles to converge to the global optimum found so far (after a certain number of iterations), which may not even be a local optimum. Thus without an effective diversity enhancing mechanism the PSO algorithm/ technique is not able to efficiently explore the search space. Inorder to improve the diversity of the swarm crossover is introduced. Crossover is the process of creating one or more new individuals through the combination of genetic material randomly selected from two or more parents. If selection focuses on the most-fit individuals, the selection pressure may cause premature convergence due to reduced diversity of the new populations. The crossover can help the particles jump out of the local optimization by sharing the others' information.

One of the earlier references on the use of reproduction operator can be found in Clerc (2001). Hao et al (2007a) proposed a crossover operator in classical PSO; the crossover is taken between each particle's individual best positions. After the crossover, the fitness of the individual best position is compared with that of the two offspring, and the best one is taken as the new individual best position. Niu and Gu (2006) proposed a modified PSO algorithm with genetic mutation and crossover operators and compared the results with GA and proved the modified PSO is better than PSO and GA with several benchmark problems. A multi-parent crossover operator is introduced by Wang et al (2008).

This section presents three QIPSO algorithms namely QIPSO2, QIPSO3 and QIPSO4, which are modified versions of QIPSO1 algorithm in section 3.2.2. These algorithms differ from each other in selection criterion of the individual.

3.4.1 Proposed Crossover Based Algorithms (QIPSO2, QIPSO3 and QIPSO3)

The new crossover operator is based on Quadratic Interpolation and is called as QI operator. For details about the proposed QI operator refer section 3.2.2. Based on the QI crossover operator, three algorithms are proposed. They are: QIPSO2, QIPSO3 and QIPSO4. These algorithms are differing from each other in selection criterion of the individual. The difference between the algorithms given in this section and QIPSO1 algorithm given in section 3.2.2 is that these algorithms do not use diversity to apply the crossover operator.

The crossover based algorithms start like the usual PSO algorithm using Eqns. (1.3) and (1.2). In QIPSO2, the new particle is accepted in the swarm irrespective of the fact whether it is better or worse than the worst particle present in the swarm. In this way the search is not limited to the region around the current best location but is in fact more diversified in nature. The process is repeated iteratively until a better solution is obtained.

QIPSO3 and QIPSO4 differ from each other and from QIPSO1 only in the selection criteria. In QIPSO3, if the new particle is better than the worst particle in the swarm then the worst particle is replaced by the new particle. Also in QIPSO4, if the new particle is better than the global best (Pg) particle in the swarm then the global best particle is replaced by the new particle.

3.5 Mutation Based Variants of PSO

Mutation is a popular phenomenon in the field of evolutionary algorithms like GA and EP. The work of mutation operator is to induce diversity in the population. Ratnaweera et al. (2004) state that lack of population diversity in PSO algorithms is understood to be a factor in their convergence on local minima. Therefore, the addition of a mutation operator to PSO should enhance its global search capacity and thus improve its performance. Most of the modern mutation operators defined in literature makes use of random probability distribution. Higashi et al. (2003) use a mutation operator that changes a particle dimension value using a random number drawn from a Gaussian distribution. Stacey et al. (2003) implement a mutation operator similar to that of Higashi et al. (2003), but a Cauchy probability distribution is used instead. Wang et al (2007) also used Cauchy probability distribution for mutation. Esquivel et al. (2003) incorporate a mutation operator into PSO that was developed by Michalewicz for use in real-valued Genetic Algorithms in (Michalewicz, 1996). This is called the Michalewicz's non-uniform mutation operator as the random numbers used to mutate values depends on the current algorithm iteration, with the probability of a value being mutated by a large amount being higher at the start of an optimization run. Secrest and Lamont (2003) also used Gaussian mutation operator to adjust the particle's position. Some of the recent researchers used chaos mutation in their study (Dong et al, 2008; Yang et al, 2009; Yue-lin et al, 2008). Wang et al (2007a) used the Cauchy distribution for mutation in the opposition based Particle Swarm Optimization algorithm. The mutation operator based on Cauchy probability distribution also used by Zhang et al (2007).

3.5.1 Sobol Mutated PSO Algorithms (SMPSO1 and SMPSO2)

In this Section, the effect of mutation operator is analyzed to preserve the diversity of the swarm. A new mutation operator based on Sobol sequence called ***Sobol Mutation (SM) operator*** is introduced to improve the performance of PSO. The SM operator unlike most of its contemporary mutation operators do not use the random probability distribution for perturbing the swarm population, but uses a quasi random Sobol sequence to find new solution vectors in the search domain. The reason behind using quasi random sequence is that quasi random sequences cover the search domain more evenly in comparison to the random probability distributions, thereby increasing the chances of finding a better solution. The SM operator is applied to two versions of PSO called SMPSO1 and SMPSO2. In SMPSO1, mutation is applied to the global best (gbest) particle, where as in SMPSO2, the worst particle of the swarm is mutated. The presence of SM operator makes the mutated particles to move systematically in the search space.

The proposed ***SM operator*** is defined as

$$SM = S_1 + (S_2 / \ln S_1) \tag{3.14}$$

Where S_1 and S_2 are random numbers in a Sobol sequence.

The proposed SMPSO algorithms start like the usual PSO algorithm up to the point of evaluating the position and velocity of the particles after which the systematic mutation is applied to the global best/worst particle to produce a perturbation in the population. If after mutation, the performance of the global best/worst position is improved, then the global best/worst particle is replaced with the mutated version. The quasi random numbers used in the SM operator allows the worst particle to move forward systemically and helps in exploring the search space more efficiently. As a result the probability of getting a better solution increases.

3.6 New Inertia Weight in PSO (GWPSO)

In case of PSO algorithms the concept of inertia weight w was introduced by Shi and Eberhart (1998) as a mechanism to control the exploration and exploitation skills of the swarm. The inertia weight controls the momentum of the swarm by weighing the contribution of the previous velocity. The value of inertia weight is very significant in order to ensure an optimal tradeoff between exploration and exploitation mechanisms of the swarm population. For $\omega \geq 1$, velocities increase over time, accelerating towards the maximum velocity and the swarm diverges. For $\omega < 1$, particles decelerate until their velocities reach zero (Engelbrecht, 2005).

Larger values of w enhance the exploration by locating promising regions in the search space whereas a smaller value helps to endorse the local exploitation. Initially the inertia weight was kept static during the entire search duration for every particle and dimension. With the due course of time inertia weights with dynamic weights were introduced. These approaches start with large inertia weight values, which decrease over time to smaller values. The choice of value for ω has to be made in conjunction with the selection of the acceleration constants c_1 and c_2. Some of the PSO algorithms using dynamic inertia weight available in literature include linearly decreasing inertia weight (Yoshida et al, 1999; Fan and Chiu, 2007) (used most frequently), Non linear decreasing inertia weight (Peram et al, 2003; Naka et al, 2001), Fuzzy adaptive inertia (Shi and Eberhart, 2001), dynamic inertia weight (Yang et al, 2007; Fan and Chang, 2007; Wang and Qian, 2008; Jiao et al, 2008), logarithm inertia weight (Yue-lin et al, 2008) etc. Besides these methods an approach which is worth mentioning is the inclusion of constriction factor introduced by Clerc (Clerc, 1999; Clerc and Kennedy, 2002). This approach is very much similar to the concept of inertia weight, where, the velocities are constricted by a constant K known as constriction coefficient. Thus suitable selection of the inertia weight provides a balance between global and local exploration and exploitation and results in less iteration on average to find a good optimum.

Keeping this in mind, a new inertia weight based on Gaussian distribution is introduced. Although Gaussian inertia weight has already been used (Engelbrecht, 2005), the present approach is completely different from their approach; in the present study the absolute value of Gaussian random numbers are used. Moreover most of the PSO algorithms use uniformly distributed random numbers for the generation of the swarm but besides using the uniform distribution (GWPSO+UD) the probability of using Gaussian (GWPSO+GD) and exponential (GWPSO+ED) distributions are also discussed for generating the initial swarm.

The new inertia weight suggested in this chapter uses half of the value of the Gaussian random number. The two factors responsible for the uniqueness of the three proposed algorithms introduced in this section are:

- Development of a new dynamic inertia weight using Gaussian distribution.
- Using of different probability distributions other than the uniform distribution for the generation of the initial swarm.

The definition of the proposed inertia weight is given as:

$$w = abs(N(0,1)/2 \qquad (3.15)$$

where $N(0,1)$ is a random number having Gaussian distribution with mean zero and standard deviation one.

3.7 Modified PSO with New Velocity Vector (MPSO)

Many researchers have studied the performance of PSO, mostly about the basic control parameters, such as the acceleration coefficients, inertia weight, velocity clamping, and swarm size (Kennedy and Eberhart, 2001; Zhang et al., 2005; Lee and Chen, 2007; Fan and Zahara, 2007; Ho et al, 2007). From these empirical studies, it can be concluded that PSO is sensitive to control parameters, but few studies are involved in the basic mechanism. In the classical PSO, velocity is an important parameter and is dynamically adjusted according to the historical behaviors of the particle and its companions. The position of the particle is changed by adding a velocity to the current position of the particle. The velocity vector drives the optimization process and reflects both the experimental knowledge of the particle and socially exchanged information from the particle's neighborhood. The experimental knowledge of a particle is generally referred to as the cognitive component, which is proportional to the distance of the particle from its own best position found since the first time step. The socially exchanged information is referred to as the social component of the velocity equation.

Based on the velocity update Eqn. (1.1), each particle's new position is influenced by the particle itself through its personal best position and the best position in its neighborhood. Kennedy and Mendes (2003) introduced a new velocity equation in which each particle is influenced by the success of all its neighbors, and not on the performance of only one individual. Thompson et al (2003) implemented an alternative approach where different velocity update equations are used for cluster centers and particles within clusters. In their method, each particle is adjusted on the basis of the distance from its own personal best position, the corresponding cluster's best position, and the current position of the cluster center. Blackwell and Bentley (2002) developed the charged PSO based on an analogy of electrostatic energy with charged particles. The charged PSO changes the velocity equation by adding a particle acceleration to standard velocity update equation (1.3). The Fitness-Distance-Ratio PSO (FDR PSO) is introduced by Peram et al (2003), in which a new term is added to the velocity update equation; each particle learns from the experience of the neighboring particles that have a better fitness than itself. Wei et al (2004) introduced a disturbance in velocity or position to prevent the premature phenomenon in basic PSO algorithm. Krohling (2005) proposed a velocity update vector with the use of absolute value of the Gaussian probability distribution.

Yang and Simon (2005) proposed NPSO algorithm, in which each particle adjusts its position according to its own previous worst solution and its group's previous worst solution to find the optimum value. That is the velocity update equation of NPSO depends upon the particle's personal worst position and the global worst position whereas in classical PSO the velocity update equation depends on the particle's personal best position and the global best position. Also PSO-E, a new PSO algorithm proposed by Krohling (2006), in which the exponential distribution is used for generating the weighting coefficients of velocity update equation of classical PSO. θ-PSO algorithm is a recently proposed PSO algorithm by Zhong et al (2008), which is based on the phase angle vector but not the velocity vector. In θ-PSO, an increment of phase angle vector $\Delta\theta$ replaces velocity vector υ and the positions are adjusted by the mapping of phase angles.

This section proposes a Modified Particle Swarm Optimization Algorithm (MPSO) with new velocity vector, which is based on the maximum distance between any two points in the solution space, distance between the global best particle and the personal best particle, objective function value of global best particle, objective function value of current particle and the current iteration number.

In the proposed PSO version, a probability P_v is fixed and is having a certain threshold value provided by the user. In every iteration, if the uniformly distributed random number U(0, 1) is less than P_v then the velocity vector is generated by using Eqn. (3.16) otherwise the velocity vector follows the standard PSO algorithm i.e. the velocity vector is generated by using Eqn. (1.3).

The proposed velocity vector is defined as:

$$V_{id} = \alpha * \alpha_1 * \alpha_2 * \alpha_3 * (P_{gd} - P_{id}) \tag{3.16}$$

Where

α is an adjustable coefficient.

$\alpha_1 = (MAXITE - ITE) / MAXITE$, $MAXITE$ represents the maximum number of iterations and ITE represents the current iteration number.

$\alpha_2 = (d_{\max} - d_{gi}) / d_{\max}$, d_{max} represents the maximum distance between two points in the solution space and d_{gi} represents the distance between the global best particle and the ith particle.

$\alpha_3 = f(P_g) / f(X_i)$, Where $f(P_g)$ is a fitness function value of the global best particle P_g

$f(X_i)$ is a fitness function value of the ith particle X_i.

The maximum distance d_{max} between two points in the solution space (a, b) is computed as:

$d_{max} = \sqrt{\sum_{i=1}^{D} (b_i - a_i)^2}$ Where $a = (a_1, a_2, ..., a_D)$, $b = (b_1, b_2, ..., b_D)$

The distance between two particles x_p and x_q can be calculated as follows:

$d_{pq} = \sqrt{\sum_{i=1}^{D} (x_{pi} - x_{qi})^2}$, D represents the dimension of swarm particle.

A C++ style computational code for the proposed algorithm may be given as:

Initialize the population

Do

  Linearly decrease w from 0.9 to 0.4 and set $c_1 = c_2 = 2.0$

  For i=1 to population size M

    For d=1 to dimension D

      Set P_v *and Generate U(0,1)*

      If (U(0,1) < P_v*) then*

        $V_{id} = \alpha * \alpha_1 * \alpha_2 * \alpha_3 * (P_{gd} - P_{id})$

      Else

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

      End if

      $x_{id} = x_{id} + v_{id}$

    End for

    If (f(X_i) < f(P_i)) $P_i = X_i$

      If (f(P_i) < f(P_g)) $P_g = P_i$

      End if

    End if

  End for

Until stopping criteria is reached

The four parameters α, α_1, α_2, α_3 helps in controlling the velocity of the swarm particles. Unlike the usual velocity equation of the basic PSO (given by Eqn. (1.3)) the proposed velocity vector do not make use of inertia weight and acceleration constants and is more or less adaptive in nature. From the velocity Eqn. (3.16), it can be easily seen that in the beginning the velocity is large therefore the particles move rapidly but during the subsequent generations the velocity decreases and the particles slows down as they reach towards the optimum solution. The presence of the parameter P_v, which helps in stochastic application of the basic velocity vector and the proposed velocity vector, helps in preventing the algorithm in becoming greedy in nature, thereby helping in preventing the premature convergence of the swarm.

3.8 Parameter Settings

Like all Evolutionary Algorithms, PSO has a set of parameters which are 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 2*n to 5*n, 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.

In order to make a fair comparison of PSO and proposed PSO algorithms, a same seed of random numbers is fixed so that the initial population is same for all the algorithms. A linearly decreasing inertia weight is used which starts at 0.9 and ends at 0.4, with the user defined parameters c_1=2.0 and c_2=2.0. The diversity measure of the swarm is calculated by using Eqn. (3.1). For MPSO, the velocity probability $\mathbf{P}_v$ are varied for different values and observed that the best results are obtained for 0.6 and the adjustable coefficient α is set as 0.5. A total of 30 runs for each experimental setting were conducted and the average fitness of the best solutions throughout the run was recorded. For comparison of proposed PSO algorithms with basic PSO, a collection of standard benchmark problems are considered. Mathematical models of the problems along with the true optimum value are given in Appendix I.

3.9 Results and Discussions

In order to compare the proposed PSO algorithms with basic PSO various performance metrics like average fitness function value, standard deviation (STD), NFE, CPU time, t-test values and percentage of improvement (%) are considered to check the efficiency and reliability of the algorithm.

3.9.1 Performance Analysis I: Comparison of Diversity Based Algorithms with Basic PSO

For comparison of PSO, ATREPSO, and QIPSO1 algorithms, a test suit of ten benchmark problems (RS, DeJ, GR, RB, DeJ-N, SWF, ACK, Mic, MH and SB1) with box constraints are considered to analyze the behavior of the algorithms. 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 2 to 20 are taken. The population size and maximum number of generations are taken as 30 and 10000 respectively for all the test problems.

The results of the given benchmark problems are shown in Table 3.1 in terms of mean best fitness and standard deviation. In Table 3.2, the improvement (%) of proposed algorithms in comparison with classical PSO and the t-values are given. Figure 3.1 shows the performance curves of PSO, QIPSO1 and ATREPSO algorithms. From the numerical results it can be seen that both the proposed versions (ATREPSO and QIPSO1) outperform the PSO algorithm in all the test cases by a significant difference. From the comparison of proposed algorithms with each other, it can be seen that QIPSO1 algorithm is better than ATREPSO algorithm in 8 test cases out of 10 test cases. ATREPSO gave better solution than QIPSO1 in only one test case. Remaining one test case both the algorithms perform the same.

3.9.2 Performance Analysis II: Comparison of Diversity Based Mutation Algorithms with Basic PSO

For comparison of PSO, GMPSO, BMPSO, GAMPSO and BGMPSO algorithms, a collection of 10 benchmark problems (RS, GR, RB, SWF, DeJ-N, ACK, SWF1.2, SWF2.22, SWF2.21 and SF7) with box constraints are considered to analyze the behavior of the algorithms. The entire given test problems are scalable i.e. the problems can be tested for any number of variables. However for the present study the dimension is taken as 30 for all the test problems. For each algorithm, the maximum number of generations is set as 2000 generations and the population size is set as 20. The results of the given benchmark problems are shown in Table 3.3 in terms of mean best fitness and standard deviation. In Table 3.4, the improvement

(%) of proposed algorithms in comparison with classical PSO and the t-values are given. Figure 3.2 shows the performance curves of PSO, GMPSO, BMPSO, GAMPSO and BGMPSO algorithms.

From the numerical results in Table 3.3, it can be seen that all the proposed algorithms perform better than the classical PSO algorithm by a significant difference. From the comparison of proposed algorithms with each other, it can be seen that GMPSO algorithm is better than the other three algorithms in 7 cases out of 10 cases; BMPSO is better than the other three in 2 cases; remaining one test case BGMPSO gave the better solution than the other three algorithms. The first test function is rastringin function, which is a multimodal function. For this function GMPSO, BMPSO, GAMPSO and BGMPSO gave a remarkable percentage of improvement of approximately 87%, 72%, 77% and 72% respectively in comparison with PSO. For the function SWF, BMPSO, GAMPSO and BGMPSO algorithms perform little better than PSO, but in this test case GMPSO algorithm gave approximately 40% improvement in comparison to classical PSO. Likewise all the other test cases also it can be seen that there is a noticeable percentage of improvement in average mean value by using the proposed diversity based mutation algorithms.

3.9.3 Performance Analysis III: Comparison of Crossover Based Algorithms with Basic PSO

For comparison of PSO and QIPSO (QIPSO2, QIPSO3 and QIPSO4) algorithms, a collection of 15 benchmark problems (RS, DeJ, GR, SWF, GP1, GP2, SWF2.22, SWF2.21, DEJ-N, SWF1.2, RB, LM, SF7, T2N and SB2) are considered. The number of particles in the swarm and the dimension are set as 30. For each algorithm, the maximum number of iterations allowed is set to 30,000. The results of the given benchmark problems are shown in Table 3.5 in terms of mean best fitness and standard deviation. In Table 3.6, the improvement (%) of proposed algorithms in comparison with classical PSO and the t-values are given. Figure 3.3 shows the performance curves of PSO and the proposed QIPSO algorithms.

From the numerical results in Table 3.5, it can be seen that all the proposed QIPSO algorithms perform better than the classical PSO algorithm by a significant difference. If the comparisons are made with the proposed algorithms with each other then it can be seen that

QIPSO2 algorithm is better than the other two algorithms in 4 cases out of 15 cases; QIPSO3 is better than the other two in 2 cases; QIPSO4 is better than the other two in 7 cases; in one test case QIPSO3 and QIPSO4 algorithms perform the same; remaining one test case all the algorithms including classical PSO also gave the same performance. The first test function is Rastringin function, which is a multimodal function. For this function QIPSO2 performs better than the other algorithms, followed by QIPSO4 and QIPSO3. For the second and third test problems, that are Dejong's function and Griewank function, QIPSO4 outperforms the other two, followed by QIPSO3 and QIPSO2. But for the SWF function QIPSO3 gave better performance than QIPSO2 and QIPSO4. The function GP1 is a multimodal function; for this function all the tested algorithms perform the same. For the functions GP2, SWF2.22, SWF1.2, RB, LM and T2N also QIPSO4 gave better performance than the other two algorithms. From the numerical results it is concluded that the quadratic interpolation based crossover operator improved the performance of classical PSO with a noticeable percentage.

3.9.4 Performance Analysis IV: Comparison of Mutation Based Algorithms with Basic PSO

For comparison, 15 benchmark problems (RS, GR, RB, DeJ, ACK, DeJ-N, SWF 2.22, SWF 1.2, ST, GP1, GP2, SWF, LM, SF7 and T2N) are considered. Each problem is evaluated for three different dimensions 10, 20 and 30 and size of the swarm is varied as 20, 40 and 80 for each of these population sizes. The stopping criteria is taken as the maximum numbers of generations reached which are 1000, 1500 and 2000 for dimensions 10, 20 and 30 respectively. The corresponding numerical results are given in Tables 3.7 - 3.10.

From the numerical results, it can be seen that the proposed SMPSO1 and SMSPO2 algorithms perform better than the classical PSO algorithm. From the comparison of SMPSO proposed algorithms with each other, it can be seen that SMPSO2 in which the worst particle is mutated is marginally better than SMPSO1. From the numerical results reported in Table 3.10, it can be seen easily judge that the proposed algorithms give a better performance in comparison to the PSO for almost all the cases. The superior performance is more evident when the dimension of the problems is increased up to 30. The convergence graphs of the proposed algorithms for selected benchmark problems are illustrated in Figure 3.4.

3.9.5 Performance Analysis V: Comparison of GWPSO Algorithms with Basic PSO

For all the four algorithms (PSO, GWPSO+UD, GWPSO+GD and GWPSO+ED), the number of particles in the swarm (swarm size) is taken to be 30. A test suite of 20 standard benchmark functions (RS, DeJ, GR, RB, DeJ-N, SWF, ACK, GP2, SF7, SB2, GP1, T2N, LM, DeJ-1, ST, SWF 2.21, SWF 2.22, MC, MH and Mic are considered. The test suite consists of a diverse set of problems of different dimensions including unimodal and multimodal functions, a noisy test function and a function with plateaus. The dimensions of the problems vary from 2 to 20. For each algorithm, the maximum number of iterations allowed was set to 10,000. The numerical results of the benchmark problems are shown in Tables 3.11 and 3.12 in terms of mean best fitness and standard deviation. In Table 3.13, the percentage of improvement for the three proposed algorithms in comparison with PSO is shown. Figure 3.5 shows the performance of PSO and GWPSO algorithms. From the numerical results it is quite evident that the proposed algorithms performed better than the PSO for almost all the test problems.

For the function RS, which is a highly multimodal Rastringin function, GWPSO+ED, GWPSO+GD and GWDPSO+UD gave a remarkable percentage of improvement of approximately 50%, 37% and 5% respectively in comparison to PSO. For function DeJ, a simple sphere function all the algorithms gave more or less similar results and converged to optimum. However the GWPSO algorithms showed some improvement in the average mean value in comparison to the PSO. For functions RB and SWF there is an improvement in the average mean value for any of the proposed GWPSO algorithms in comparison to PSO. Once more for function LM, a highly multimodal function, there is a huge improvement in comparison to PSO. Likewise for other functions also one can see that in most of the cases there is an improvement in average mean value by using the three proposed GWPSO algorithms.

3.9.6 Performance Analysis VI: Comparison Results of MPSO Algorithm

To check the efficiency of the proposed MPSO algorithm, a suit of thirty six benchmark problems are considered, which are given in Appendix I. For each algorithm, the stopping criteria is to terminate the search process when one of the following conditions is satisfied: (1) the maximum number of generations is reached (assumed 1000 generations), (2) $|f_{max} - f_{min}| < 10^{-4}$ where f is the value of objective function. A total of 30 runs for each experimental setting were conducted. Also the MPSO algorithm is compared with another variant of classical PSO called *θ-PSO*.

The results of all the benchmark problems are shown in Table 3.14 in terms of mean best fitness, standard deviation and SR (success rate). Table 3.15 gives the results of all benchmark problems in terms of diversity, NFE (number of function evaluations) and time. The percentage of improvement and t-values of proposed MPSO algorithm in comparison to classical PSO are given in Table 3.16. Performance curves of selected benchmark problems are given in Figure 3.6. Comparison results of MPSO algorithm with BPSO and *θ-PSO* algorithms are given in Tables 3.17 and 3.18. In order to make a fair comparison of MPSO and *θ-PSO,* the same error goal is fixed as stated in (Zhong et al, 2008) as: for *DeJ*, *GR* and *RB* are 0.01, 0.1 and 100 respectively.

The MPSO algorithm is compared with the classical PSO in terms of Average fitness function value, number of function evaluations (NFE), Success rate in % (SR) and run time. As expected the proposed MPSO algorithm performed much better than the classical PSO algorithm. From Table 3.14 it can be seen that when MPSO is used to solve the given benchmark problems the improvement in terms of average fitness function values is more than 99% in comparison to the PSO for about 9 out of 36 test cases. Also MPSO gave more than 75%, 50% and 30% improvement in 3 test cases for each in comparison with PSO in terms of fitness value. For all the remaining test cases both the algorithms gave the same performance in terms of fitness value. In comparison of PSO and MPSO in terms of success rate, MPSO gave better performance than PSO in most of the test cases. Some of the test cases (DeJ, GR, ACK, SWF1.2, SWF2.21, GP1, GP2, LM, CB6 and T2N) PSO gave 0% SR whereas MPSO gave more than 30% SR (including 100% SR). In terms of number of function evaluation also MPSO gave much better performance than PSO. But in terms of convergence time taken by PSO and MPSO then MPSO has taken more time for convergence than PSO in 22 test cases, this is because of the inclusion of added velocity part in the algorithm. Even though MPSO has the new added velocity part, it has taken less time than PSO in 14 test cases out of 36 test cases.

Thus from the numerical results, it is concluded that the incorporation of the proposed velocity vector helps in improving the performance of classical PSO in terms of final objective function value, NFE and convergence rate. Also the performance of proposed MPSO algorithm is compared with *θ-PSO,* a variance of classical PSO. From the numerical results of Table 3.17 and Table 3.18, it is clear that the performance of proposed MPSO is better than the *θ-PSO* algorithm also.

3.9.7 Performance Analysis VII: Comparison of Proposed Algorithms with each other

The numerical results for comparison of all proposed algorithms with each other (in terms of fitness, standard deviation and success rate) are given in Table 3.19. In Table 3.20, the comparison results in terms of NFE and convergence (in seconds) are given. A test suit of five benchmark problems (RS, DeJ, GR, RB, ACK) are considered for this comparison. The dimension of the each problem is set as 10 and the population size is taken as 50. For each algorithm, the stopping criteria is to terminate the search process when one of the following conditions is satisfied: (1) the maximum number of generations is reached (assumed 1000 generations), (2) $|f_{max} - f_{min}| < 10^{-4}$ where f is the value of objective function. A total of 30 runs for each experimental setting were conducted. For the first test problem, GWPSO+UD which is Gaussian inertia weight PSO initializing with uniform distribution, is superior with all the other compared algorithms. QIPSO3 algorithm gave better results in two test cases (for DeJ and RB). For the remaining two test cases, BMPSO and GWPSO+GD algorithms gave better results than the other algorithms.

Table 3.1 Comparison results of PSO, ATREPSO and QIPSO1 (Mean /standard deviation)

Function	PSO	ATREPSO	QIPSO1
RS	22.3391 (15.932042)	19.4259 (14.3490)	11.9468 (9.1615)
DeJ	1.16e-45 (5.22e-46)	4.00e-17 (0.00024)	0.0000 (0.0000)
GR	0.0316 (0.0253)	0.0251 (0.02814)	0.0115 (0.0128)
RB	22.1917 (1.61e+04)	19.4908 (3.96e+04)	8.9390 (3.1063)
DeJ-N	8.6816 (9.0015)	8.0466 (8.8623)	0.4511 (0.3286)
SWF	-6178.55 (4.89e+02)	-6183.6776 (469.61)	-6355.5866 (477.53)
ACK	3.48e-18 (8.35e-19)	0.0184 (0.0147)	2.46e-24 (0.0144)
Mic	-18.1594 (1.0510)	-18.9829 (0.2725)	-18.4696 (0.0929)
MH	-3.3314 (1.24329)	-3.7514 (0.17446)	-3.7839 (0.1903)
SB1	-186.7309 (0.00001)	-186.7309 (0.00001)	-186.7309 (3.48e-14)

Table 3.2 Improvement (%) and t-value of ATREPSO and QIPSO1 in comparison with PSO

Function	ATREPSO		QIPSO1	
	Improvement (%)	t-value	Improvement (%)	t-value
RS	13.04081185	0.744187639	46.52067451	3.572735456
DeJ	-	-9.12871e-13	100	12.17161239
GR	20.56962025	0.940827748	63.60759494	4.351471702
RB	12.17076655	0.000346064	59.71917429	0.004508573
DeJ-N	7.314319941	0.275335568	94.80395319	5.008088107
SWF	0.082831601	0.041345491	2.865179034	1.982854226
ACK	-	-6.855846978	99.99992931	22.82722339
Mic	4.53484146	4.154259498	1.708206218	1.616589318
MH	12.60731224	1.832328709	13.58287807	1.993456533
SB1	0.00000	0.00000	0.00000	0.00000

Table 3.3 Comparison results of PSO, GMSPO, BMPSO, GAMPSO and BGMPSO in terms of average fitness values and standard deviation

Function	PSO	GMPSO	BMPSO	GAMPSO	BGMPSO
RS	47.29223 (11.06489)	5.98732 (7.59559)	13.34445 (4.690)	10.90207 (6.456038)	13.46373 (6.904967)
GR	0.0182 (0.244025)	0.006451 (0.011695)	0.002525 (0.001589)	0.002669 (0.001841)	0.002562 (0.001473)
RB	316.4468 (80.001)	54.6533 (25.7658)	74.76106 (24.37858)	81.10299 (30.47879)	88.04853 (0.002478)
SWF	-6466.19 (643.4821)	-8968.49 (684.388)	-6718.49 (666.7723)	-6951.59 (369.7281)	-6587.17 (497.698)
DeJ-N	0.617222 (0.492993)	0.113899 (0.034068)	0.072433 (0.21498)	0.21431 (0.281486)	0.43356 (0.250547)
ACK	1.70 (4.53e-01)	0.239636 (0.263923)	0.077461 (0.06742)	0.078481 (0.05728)	0.011117 (0.012603)
SWF1.2	271.793 (208.325)	1.81142 (0.681979)	28.938 (98.7083)	63.4677 (59.2094)	156.43 (98.7083)
SWF2.21	15.2228 (3.652739)	0.74461 (0.085948)	9.96414 (2.00684)	1.23348 (0.297061)	1.82377 (0.509892)
SWF2.22	0.209776 (0.072407)	0.095135 (0.074405)	0.169551 (0.625909)	0.176346 (0.616302)	0.16719 (0.623112)
SH7	4.22028 (0.416)	1.8681 (0.235079)	4.10447 (0.460416)	3.41021 (0.3585)	2.24576 (0.460416)

Table 3.4 Improvement (%) and t-value of GMSPO, BMPSO, GAMPSO and BGMPSO in comparison with PSO

Function	GMPSO		BMPSO		GAMPSO		BGMPSO	
	IMP	t-value	IMP	t-value	IMP	t-value	IMP	t-value
RS	87.33	16.85	71.78	15.47	76.94	15.55	71.53	14.2
GR	64.55	0.26	86.12	0.35	85.33	0.34	85.92	0.35
RB	82.72	17.06	76.37	15.82	74.37	15.05	72.17	15.63
SWF	38.69	14.58	3.9	1.49	7.5	3.58	1.87	0.81
Dej-N	81.54	5.57	88.26	5.54	65.27	3.88	29.75	1.81
ACK	85.86	15.21	95.43	19.36	95.36	19.4	99.34	20.37
SWF1.2	99.33	7.09	89.35	5.77	76.64	5.26	42.44	2.74
SWF2.21	95.1	21.7	34.54	6.91	91.89	20.9	88.01	19.89
SWF2.22	54.64	6.04	19.17	0.34	15.93	0.29	20.3	0.37
SH7	55.73	26.96	2.74	1.02	19.19	8.07	46.78	17.42

Table 3.5 Comparison of proposed QIPSO2, QIPSO3 and QIPSO4 algorithms with PSO in terms of average fitness function value and standard deviation

Function	PSO	QIPSO2	QIPSO3	QIPSO4
RS	81.58668 (35.57092)	0.597167 (0.659803)	0.994954 (2.354492)	5.173762 (5.069386)
DeJ	2.62144 (7.86432)	8.51799e-43 (1.67879e-42)	2.5236e-45 (6.50519e-45)	1.0865e-52 (2.32221e-52)
GR	0.035265 (0.029109)	0.0294 (0.023866)	0.015979 (0.013563)	0.012296 (0.015833)
SWF	-8406.742 (595.7797)	-9185.054 (589.2412)	-9185.074692 (760.633113)	-8909.10755 (474.904681)
GP1	5.50585e-13 (2.75701e-25)	5.50585e-13 (3.02157e-26)	5.50585e-13 (1.91386e-25)	5.50585e-13 (9.81166e-26)
GP2	-1.147328 (0.003296)	-1.148241 (0.004395)	-1.149339 (0.003296)	-1.150438 (9.93014e-17)
SWF2.22	4.0000 (4.89897)	1.15457e-20 (3.24544e-20)	5.02094e-30 (1.27187e-29)	1.10655e-34 (1.45317e-34)
SWF2.21	0.000244 (0.000187)	0.000351 (0.00023)	0.000148 (8.12894e-05)	0.000162 (0.000491)
DeJ-N	24.53298 (14.64057)	0.454063 (0.354884)	0.454374 (0.353778)	0.454653 (0.354973)
SWF1.2	8.1039e-06 (3.62322e-06)	4.07231e-37 (6.0998e-37)	2.61421e-40 (5.84872e-40)	5.45193e-50 (1.46719e-49)
RB	99.79576 (438.4913)	31.27418 (24.32459)	77.916591 (166.009829)	24.79044 (30.2989)
LM	-13.01387 (9.36361)	-21.50231 (3.55271e-15)	-21.502311 (3.55271e-15)	-21.502311 (3.55271e-15)
SF7	3.531709 (2.484447)	0.858533 (0.263027)	0.974427 (0.283325)	0.904524 (0.344629)
T2N	-77.0129 (1.049466)	-77.95535 (0.461703)	-77.201394 (0.565469)	-77.955352 (0.461703)
SB2	-155.6138 (17.66488)	-325.8289 (10.99762)	-179.040627 (30.28809)	-185.807307 (23.57774)

Table 3.6 Comparison of proposed QIPSO2, QIPSO3 and QIPSO4 algorithms with PSO in terms of Improvement (%) and t-test values

Function	QIPSO2		QIPSO3		QIPSO4	
	Improvement	t-value	Improvement	t-value	Improvement	t-value
RS	99.26	12.46	98.78	12.38	93.65	11.64
DeJ	100	1.82	100	1.82	100	1.82
GR	16.63	0.85	54.68	3.28	65.13	3.79
SWF	9.25	5.08	9.25	4.41	5.97	3.61
GP1	0	0	0	0	0	0
GP2	0.07	0.91	0.17	2.36	0.27	5.16
SWF2.22	100	4.47	100	4.47	100	4.47
SWF2.21	-	-1.97	39.34	2.57	33.6	0.85
DeJ-N	98.14	9.00	98.14	9.00	98.14	9.00
SWF1.2	100	12.25	100	12.25	100	12.25
RB	68.66	0.85	21.92	0.25	75.15	0.93
LM	65.22	4.96	65.22	4.96	65.22	4.96
SF7	75.69	5.86	72.4	5.6	74.38	5.73
T2N	1.22	4.5	0.24	0.86	1.22	4.5
SB2	109.38	44.8	15.05	3.65	19.4	5.61

Table 3.7 Comparison of proposed SMPSO1 and SMPSO2 versions with PSO for function ***RS*** in terms of average fitness function value

Pop	Dim	Gne	SMPSO1	SMPSO2	PSO
20	10	1000	0.881465	0.641812	5.5382
	20	1500	5.014802	4.52709	23.1544
	30	2000	13.152097	12.669938	47.4168
40	10	1000	1.241561	0.85634	3.5778
	20	1500	5.91223	5.472557	16.4337
	30	2000	13.005205	14.523385	37.2896
80	10	1000	1.182363	0.813593	2.5646
	20	1500	5.501107	4.97266	13.3826
	30	2000	10.210538	15.028891	28.6293

Table 3.8 Comparison of proposed SMPSO1 and SMPSO2 versions with PSO for function ***GR*** in terms of average fitness function value

Pop	Dim	Gne	SMPSO1	SMPSO2	PSO
20	10	1000	0.006896	0.007877	0.09217
	20	1500	0.009177	0.008486	0.03002
	30	2000	0.025227	0.014541	0.01811
40	10	1000	0.009677	0.009515	0.08496
	20	1500	0.017195	0.012269	0.02719
	30	2000	0.030103	0.011066	0.01267
80	10	1000	0.00886	0.006402	0.07484
	20	1500	0.010828	0.01296	0.02854
	30	2000	0.024265	0.004692	0.01258

Table 3.9 Comparison of proposed SMPSO1 and SMPSO2 versions with PSO or function ***RB*** in terms of average fitness function value

Pop	Dim	Gne	SMPSO1	SMPSO2	PSO
20	10	1000	6.4165	6.4104	94.1276
	20	1500	17.3111	17.2875	204.336
	30	2000	30.5664	28.2597	313.734
40	10	1000	6.4147	6.4011	71.0239
	20	1500	17.2344	17.2504	179.291
	30	2000	28.1147	28.640997	289.593
80	10	1000	6.4161	6.3453	37.3747
	20	1500	17.4405	17.1907	83.6931
	30	2000	28.3247	30.1533	202.672

Table 3.10 Comparison of proposed SMPSO1 and SMPSO2 versions with PSO for the remaining 12 functions in terms of average fitness function value

Function	Dim	Gne	SMPSO1	SMPSO2	PSO
DeJ	10	1000	1.62763e-10	1.69783e-10	1.04431e-07
	20	1500	0.000297	0.000391	0.000801
	30	2000	0.004315	0.003344	0.009211
ACK	10	1000	0.000368	0.000131	0.003435
	20	1500	0.026621	0.018378	0.123742
	30	2000	0.115094	0.140612	1.31424
DeJ-N	10	1000	0.003347	0.006410	0.008474
	20	1500	0.023071	0.031297	0.031376
	30	2000	0.005992	0.084939	0.089811
SWF2.22	10	1000	9.25975e-05	7.01387e-06	0.000102
	20	1500	0.010336	0.008547	0.020129
	30	2000	0.061249	0.089932	0.174977
SWF1.2	10	1000	2.55247e-07	2.90956e-06	0.001198
	20	1500	0.041532	0.036888	4.40991
	30	2000	3.80048	4.09788	271.793
ST	10	1000	0.000000	0.000000	0.000000
	20	1500	0.000000	0.000000	0.000000
	30	2000	0.000000	0.000000	5.8
GP1	10	1000	3.17643e-09	2.61458e-09	6.13307e-07
	20	1500	1.7112e-05	1.32619e-05	0.083184
	30	2000	0.000109	0.000238	0.87767
GP2	10	1000	-1.15042	-1.15044	-1.15007
	20	1500	-1.13147	-1.12577	-0.813208
	30	2000	-1.12011	-1.04616	11.5649
SWF	10	1000	-3439.57	-3456.7	-3308.83
	20	1500	-6355.59	-6593.98	-6258.6
	30	2000	-9221.62	-9830.23	-8872.75
LM	10	1000	-20.5621	-20.604	-20.346
	20	1500	-18.9347	-19.0519	-17.479
	30	2000	-17.2635	-16.1726	-13.4851
SF7	10	1000	0.346679	0.29003	0.612593
	20	1500	1.203	1.07035	2.41555
	30	2000	1.82546	1.7637	4.22028
T2N	10	1000	-78.3323	-78.3323	-78.0496
	20	1500	-75.6455	-76.9185	-74.9025
	30	2000	-75.4883	75.4935	-74.3619

Table 3.11 Comparison results of PSO, GWPSO+UD, GWPSO+GD and GWPSO+ED: in terms of Mean fitness values

Function	PSO	GWPSO+UD	GWPSO+GD	GWPSO+ED
RS	22.33916	21.125507	14.031235	11.077155
DeJ	1.17e-45	9.81e-46	8.41e-46	1.12e-45
GR	0.031646	0.031237	0.002125	0.007694
RB	22.19173	16.408472	12.207816	10.041835
DeJ-N	8.681602	2.865745	2.776164	20.044952
SWF	-6178.56	-6802.169271	-6735.788021	-6868.30208
ACK	3.48E-18	3.08e-18	3.37e-18	3.25e-18
GP2	-1.14934	-1.141943	-1.148241	-1.150072
SF7	1.082386	0.712431	0.640038	0.665112
SB2	-1591.82	-1902.101172	-2400.224219	-2333.74349
GP1	8.29e-13	0.051834	7.36e-13	8.28e-13
T2N	-77.2956	-71.923682	-77.489144	-77.546712
LM	-9.10049	-19.05978	-21.491327	-21.347021
DeJ-1	6.084537	2.863311	5.368709	17.985173
ST	0.00000	0.00000	0.00000	0.00000
SWF2.21	1.69e-08	2.55e-09	7.04e-10	9.75e-10
SWF2.22	7.19e-45	3.74e-45	1.36e-44	5.09e-45
MC	-1.87691	-1.905961	-1.913223	-1.928496
MH	-3.33149	-3.712778	-3.783962	-3.783962
Mic	-1.77459	-1.801301	-1.801301	-1.774591

Table 3.12 Comparison results of PSO, GWPSO+UD, GWPSO+GD and GWPSO+ED: in terms of Standard Deviation

Function	PSO	GWPSO+UD	GWPSO+GD	GWPSO+ED
RS	15.93204	17.228723	7.367918	4.477165
DeJ	5.22e-46	6.42e-46	6.86e-46	5.61e-46
GR	0.025322	0.038953	0.00855	0.014311
RB	1.62e+04	13.576216	11.478216	2.605667
DeJ-N	9.001534	4.847958	5.52229	23.23373
SWF	4.89e+02	425.958668	427.933234	363.798421
ACK	8.36e-19	8.50e-19	1.16e-18	8.67e-19
GP2	0.003296	0.02478	0.004395	0.001972
SF7	1.3815	0.221431	0.22872	0.157151
SB2	162.1692	240.813568	96.004723	83.766385
GP1	4.34e-16	0.122662	3.11e-14	7.09e-16
T2N	1.150406	2.386087	2.119984	2.929179
LM	9.922227	7.731291	0.032962	0.365476
DeJ-1	6.928662	6.823871	9.477964	19.321758
ST	0.00000	0.00000	0.00000	0.00000
SWF2.21	1.79e-08	2.82e-09	9.22e-10	1.78e-09
SWF2.22	1.01e-44	4.05e-45	5.82e-44	2.31e-43
MC	0.081189	0.039106	2.30e-07	0.128774
MH	1.24329	2.488714	3.172452	3.172452
Mic	0.143838	6.36E-07	6.36e-07	0.143838

Table 3.13 t-test values and Improvement (%) of GWPSO+UD, GWPSO+GD and GWPSO+ED in comparison with PSO

Function	GWPSO+UD		GWPSO+GD		GWPSO+ED	
	t-value	Improvement (%)	t-value	Improvement (%)	t-value	Improvement (%)
RS	0.283278	5.43284	1.939151654	37.18995	3.727349	50.4137309
DeJ	1.236393	16.00002	2.163685822	28.00001	0.33395	4.000003425
GR	0.048217	1.292422	3.480259699	93.28509	4.510400	75.68729065
RB	0.001961	26.06040	0.003384872	44.98933	0.0041192	54.74964204
DeJ-N	3.115683	66.99059	3.163673234	68.02244	-2.497924	-
SWF	5.264902	10.09311	4.70446948	9.018737	6.1957669	11.1634782
ACK	1.859792	11.61826	0.531367872	3.319495	1.0516589	6.63899081
GP2	-1.6205	-	-0.240576681	0.095533	1.0452814	0.063775788
SF7	1.448273	34.17958	1.731671894	40.86786	1.643763	38.55131164
SB2	5.853668	19.49223	15.25110038	50.78490	22.26361	46.60851129
GP1	-2.31454	-	4.14892e-12	11.21045	3.250681	0.059518165
T2N	-11.1076	-	0.400130882	0.25035	0.436999	0.32483337
LM	4.336644	109.4369	5.395429072	136.1558	6.7556988	134.5701208
DeJ-1	1.814268	52.94118	0.403170738	11.7647	-3.175529	-
ST	0.000000	0.000000	-	0.000000	-	0.000000
SWF2.21	4.327379	84.88793	4.884777816	95.82213	4.838184	94.21607542
SWF2.22	1.743164	48.05194	-3.227212131	88.96111	0.049907	29.28571905
MC	1.765464	1.547593	2.206845618	1.934505	1.855900	-2.7482346
MH	0.75069	11.44503	0.890837968	13.58173	0.727334	-13.5817389
Mic	1.017094	1.505135	1.017093502	1.50513	0.000000	0.000000

Table 3.14 Result comparison of PSO and MPSO (Mean fitness/Standard deviation/SR (%))

F	Dim	PSO			MPSO		
		Fitness	Std	SR	Fitness	Std	SR
RS	10	8.44052	4.03724	-	3.52897	1.34857	-
DeJ	10	2.49e-08	2.86e-08	-	1.27e-11	2.11e-11	100
GR	10	0.09489	0.03934	-	0.03903	0.01274	30
RB	10	23.7877	34.4229	-	5.09976	6.65036	-
ACK	10	9.9569	9.95228	-	2.19e-08	3.42e-08	50
DeJ-N	10	0.00480	0.00191	-	0.00296	0.00167	-
Mic	2	-1.8013	9.93e-03	100	-1.8013	1.57e-06	100
Mic	5	-4.32364	0.44702	-	-4.66848	0.04535	-
Mic	10	-6.62579	0.67245	-	-9.3255	0.22641	-
ST	10	0.00000	0.00000	60	0.00000	0.00000	70
SWF1.2	10	0.83576	3.76452	-	1.04e-12	2.69e-12	40
SWF2.21	10	0.00015	0.00018	-	9.98e-10	1.26e-09	60
SWF2.22	10	0.09027	0.03896	-	0.01979	0.02145	-
SDP	10	1.53e-11	4.01e-11	100	1.05e-13	2.12e-13	100
ALP	10	0.00667	0.01921	-	0.00021	0.00065	-
GP1	10	-1.1504	4.23e-05	-	-1.15044	2.46e-11	60
GP2	10	1.44e-06	1.90e-06	-	1.72e-12	1.53e-13	90
SWF	10	-3201.6	369.23	20	-3751.61	130.28	70
LM	10	-19.4018	4.2009	-	-21.5023	1.10e-12	30
DeJ1	2	3.87e-11	7.29e-11	30	9.53e-16	2.83e-15	100
HM1	3	-3.86278	3.97e-16	100	-3.86278	3.71e-16	100
HM2	6	-3.18244	0.14725	90	-3.25608	0.06475	100
SF6	2	0.00000	0.00000	100	0.00000	0.00000	100
MT	2	1.81e-15	5.23e-15	100	0.00000	0.00000	100
CB6	2	-1.03163	2.22e-22	-	-1.03163	2.22e-22	80
APH	10	5.58e-13	1.13e-12	80	1.53e-15	4.01e-15	100
CLV	4	0.05054	0.05190	-	0.03758	0.03171	-
GP	2	3.00000	1.60e-15	100	3.00000	2.90e-16	100
MC	2	-1.91322	0.00000	100	-1.91322	0.00000	100
SB1	2	-186.731	4.21e-14	-	-186.731	2.00e-14	-
SB2	10	-98.4351	10.4653	-	-117.776	1.60425	-
SK	2	1.00000	1.85e-16	80	1.00000	9.93e-17	100
BR	2	0.397886	0.00000	50	0.397886	0.00000	100
SF7	10	0.67169	0.20006	-	0.29862	0.08075	-
T2N	10	-78.3323	1.92e-06	-	-78.3323	3.55e-13	90
MH	2	-3.58972	0.388473	30	-3.78396	0.00000	50

Table 3.15 Result comparison of PSO and MPSO (Diversity/NFE/Time (sec))

F	Dim	PSO			MPSO		
		Diversity	NFE	Time	Diversity	NFE	Time
RS	10	3.05529	50050+	2.7	1.02237	50050+	3.2
DeJ	10	0.0805151	50050+	2.7	0.0405957	43570	2.9
GR	10	0.713923	50050+	2.4	2.6207	47540	3.1
RB	10	4.58719	50050+	6.3	0.678667	50050+	8.8
ACK	10	4.1804	50050+	1.7	0.00014	46800	3.7
DeJ-N	10	0.647542	50050+	2.1	0.39727	50050+	4.6
Mic	2	0.00365	38495	0.7	0.0433284	18375	0.5
Mic	5	0.0524459	50050+	2.6	0.0291604	50050+	2.8
Mic	10	2.14912	50050+	5.8	1.35232	50050+	6.4
ST	10	0.793882	49215	0.5	0.14526	46860	0.3
SWF1.2	10	0.59298	50050+	2.1	0.02419	49715	2.6
SWF2.21	10	0.06367	50050+	0.1	2.07e-05	48615	0.24
SWF2.22	10	3.21891	50050+	0.1	0.38093	50050+	0.25
SDP	10	0.16440	34135	0.6	0.32737	22315	0.21
ALP	10	3.73003	50050+	0.2	0.91417	50050+	0.25
GP1	10	1.80214	50050+	5.4	0.38154	49550	7.4
GP2	10	0.443238	50050+	5	0.00436	45745	6.1
SWF	10	0.45694	49630	2.2	0.00704	48510	2.6
LM	10	0.10776	50050+	4.8	0.46810	48680	6.9
DeJ1	2	0.14613	49970	2	0.13494	40755	4.1
HM1	3	0.01927	37945	2.2	0.01299	20465	1.4
HM2	6	0.01558	43920	5.1	0.00763	37295	4.3
SF6	2	0.00693	37020	1.0	0.00053	22525	0.7
MT	2	0.18571	33945	0.2	0.16375	17655	0.1
CB6	2	0.27273	50050+	1.1	0.0069	38575	0.9
APH	10	0.16440	45700	5	0.00153	34135	4.1
CLV	4	0.27199	50050+	2.2	0.16179	50050+	3.0
GP	2	0.00021	40895	1.0	0.00184	19885	0.6
MC	2	0.11743	32970	0.4	0.01701	13855	0.2
SB1	2	1.57827	50050+	0.2	0.91083	50050+	0.4
SB2	10	2.9924	50050+	2.0	3.38776	50050+	2.2
SK	2	0.07937	39390	12.5	0.38103	37840	12.1
BR	2	0.01523	42785	1.0	0.01535	41820	0.6
SF7	10	7.09519	50050+	2.8	1.60477	50050+	3.4
T2N	10	0.05004	50050+	3.9	0.00107	45840	3.6
MH	2	1.08611	47120	1.0	0.15180	35930	0.8

Table 3.16 Comparison of proposed MPSO with PSO in terms of Improvement (%) and t-test values

Function	Improvement (%)	t-value	Function	Improvement (%)	t-value
RS	58.190135	6.320111	LM	10.82632	2.738678
DeJ	99.948996	4.7662	DeJ1	99.99754	2.907591
GR	58.868163	7.398961	HM1	0.00000	0.00000
RB	78.561357	2.919559	HM2	2.313948	2.507455
ACK	100	5.479768	SF6	0.00000	-
DeJ-N	38.333333	3.972251	MT	100	1.89556
Mic	0.00000	0.00000	CB6	0.00000	0.00000
Mic	7.9756872	4.203663	APH	99.72581	2.69725
Mic	40.745481	20.84008	CLV	25.64306	1.16712
ST	0.00000	-	GP	0.00000	0.00000
SWF1.2	100	1.215997	MC	0.00000	-
SWF2.21	99.999335	4.564324	SB1	0.00000	0.00000
SWF2.22	78.07688	8.679908	SB2	19.64838	10.00557
SDP	99.313725	2.075443	SK	0.00000	0.00000
ALP	96.851574	1.840845	BR	0.00000	-
GP1	0.0034771	5.17941	SF7	55.54199	9.47145
GP2	99.999881	4.151155	T2N	0.00000	0.00000
SWF	17.179223	7.694049	MH	5.411007	2.738662

Table 3.17 Comparison of MPSO with PSO and θ-PSO

Fun	Algorithm	Swarm size	Dim	Number of iterations to achieve the goal		
				$w = 0.6$, $c_1 = c_2 = 1.7$		
				Minimum	Average	SR
DeJ	PSO	20	30	722	778	100
	θ-PSO	20	30	523	598	100
	MPSO	20	30	236	324	100
	PSO	40	30	783	847	100
	θ-PSO	40	30	352	406	100
	MPSO	40	30	270	321	100
GR	PSO	20	30	368	455	100
	θ-PSO	20	30	343	512	100
	MPSO	20	30	211	390	100
	PSO	40	30	684	836	100
	θ-PSO	40	30	231	334	100
	MPSO	40	30	219	293	100
RB	PSO	20	30	426	533	100
	θ-PSO	20	30	223	376	100
	MPSO	20	30	208	267	100
	PSO	40	30	544	597	100
	θ-PSO	40	30	194	283	100
	MPSO	40	30	184	214	100

Table 3.18 Comparison of MPSO with PSO and θ-PSO

Fun	Algorithm	Swarm size	Dim	Number of iterations to achieve the goal		
				w = 0.729, $c_1 = c_2 = 1.494$		
				Minimum	Average	SR
DeJ	PSO	20	30	598	682	100
	θ-PSO	20	30	362	734	100
	MPSO	20	30	207	239	100
	PSO	40	30	620	707	100
	θ-PSO	40	30	266	683	100
	MPSO	40	30	203	233	100
GR	PSO	20	30	420	686	100
	θ-PSO	20	30	385	564	95
	MPSO	20	30	178	198	100
	PSO	40	30	883	965	100
	θ-PSO	40	30	263	356	100
	MPSO	40	30	192	294	100
RB	PSO	20	30	363	443	100
	θ-PSO	20	30	328	402	100
	MPSO	20	30	140	185	100
	PSO	40	30	459	537	100
	θ-PSO	40	30	272	325	100
	MPSO	40	30	165	198	100

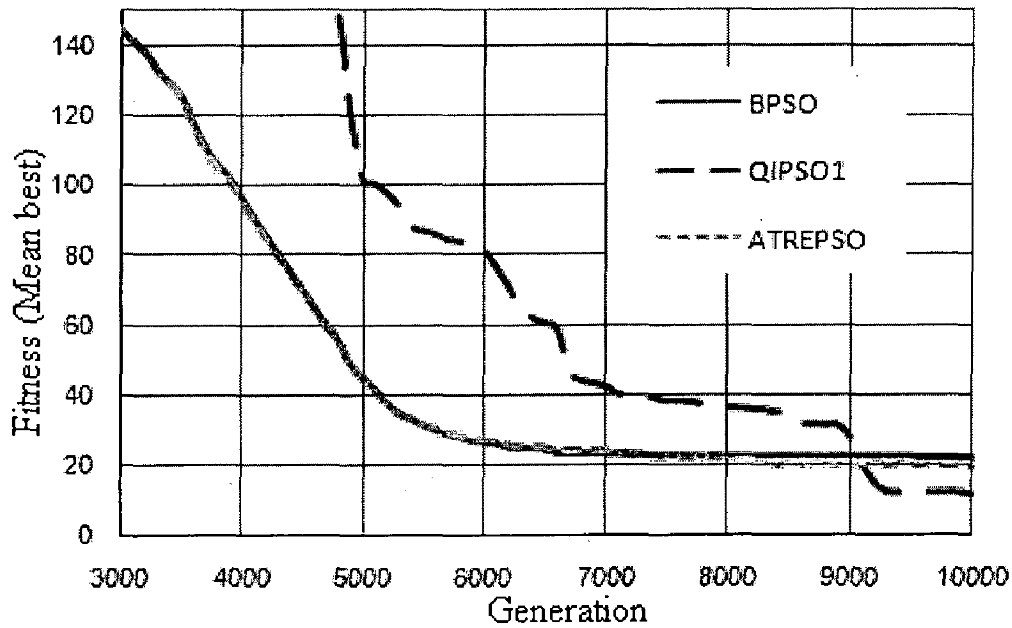

Figure 3.1 (a) Function RS

Table 3.19 Comparison results of all the proposed PSO algorithms in terms of mean fitness, standard deviation (Std) and success rate (SR)

Fun	PSO			ATREPSO			QIPSO1			GMPSO			BMPSO		
	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR
RS	8.44	4.03	-	4.99	1.79	-	5.62	2.15	-	4.98	2.77	-	3.98	2.85	-
DeJ	2.4e-8	2.8e-8	-	1.8e-9	1.4e-9	100	8.6e-9	8.3e-8	100	4.3e-9	6.7e-7	100	6.3e-10	7.2e-9	100
GR	0.094	0.039	-	0.018	0.161	-	0.088	0.132	-	0.026	0.153	-	0.051	0.066	-
RB	23.78	34.42	-	6.035	2.05	-	4.74	1.92	-	5.86	2.32	-	6.11	1.73	-
ACK	9.96	9.95	-	0.0056	0.0052	-	0.0021	0.002	50	0.0014	0.059	40	0.0009	0.013	-
Fun	GAMPSO			BGMPSO			QIPSO2			QIPSO3			QIPSO4		
	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR
RS	5.97	2.56	-	4.98	1.87	-	6.24	3.64	-	3.59	2.78	-	5.67	3.49	-
DeJ	4.8e-9	2.2e-8	100	3.2e-9	2.7e-6	100	2.6e-9	3.2e-6	100	1.7e-10	4.4e-9	100	1.2e-8	2.8e-8	90
GR	0.086	0.028	-	0.023	0.036	-	0.019	0.011	-	0.007	0.080	-	0.015	0.08	-
RB	4.75	2.37	-	5.67	1.56	-	4.64	2.91	-	4.08	2.27	-	5.27	2.40	
ACK	0.008	0.014	-	0.006	0.051	-	0.0044	0.0042	-	0.002	0.002	30	0.004	0.003	-
Fun	SMPSO1			SMPSO2			GWPSO+UD			GWPSO+GD			GWPSO+ED		
	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR	Fitness	Std	SR
RS	6.25	3.77	-	4.97	1.73	-	1.605	1.276	-	2.78	1.58	-	1.95	1.14	-
DeJ	2.9e-9	7.4e-8	100	5.1e-10	1.6e-7	70	1.1e-9	2.6e-9	100	2.7e-10	4.1e-10	100	1.8e-10	3.5e-10	100
GR	0.014	0.009	-	0.014	0.009	-	0.064	0.057	-	1.3e-13	1.9e-13	100	0.008	0.011	-
RB	6.30	0.66	-	6.06	0.708	-	11.51	9.05	-	5.36	0.57	-	4.84	1.24	-
ACK	0.0002	0.0001	40	0.0004	0.0002	60	1.6e-6	2.1e-6	80	5.3e-7	9.1e-7	100	2.5e-7	3.1e-7	60
Fun	MPSO														
	Fitness	Std	SR												
RS	3.52	1.34	-												
DeJ	1.2e-11	2.1e-11	100												
GR	0.039	0.012	30												
RB	5.09	6.65	-												
ACK	2.19e-8	3.42e-8	50												

Table 3.20 Comparison results of all the proposed algorithms in terms of number of function evaluations (NFE) and time (in secs))

Fun	PSO		ATREPSO		QIPSO1		GMPSO		BMPSO		GAMPSO	
	NFE	Time	NFE	Time	NFE	Time	NFE	Time	NFE	Time	NFE	Time
RS	50050	2.1	50050	3.8	50050	3.2	50050	3.8	50050	3.8	50050	3.8
DeJ	50050	2.1	44657	3.7	42356	4.0	45178	3.5	45245	3.4	47600	3.8
GR	50050	2.4	50050	3.7	50050	2.8	50050	3.8	50050	3.8	50050	3.8
RB	50050	6.3	50050	6.7	50050	6.8	50050	6.6	50050	6.8	50050	6.8
ACK	50050	1.7	50050	3.7	48915	2.1	48013	2.9	50050	4.0	50050	4.0
Fun	BGMPSO		QIPSO2		QIPSO3		QIPSO4		SMPSO1		SMPSO2	
	NFE	Time	NFE	Time	NFE	Time	NFE	Time	NFE	Time	NFE	Time
RS	50050	4.0	51050	2.3	51050	2.4	51050	2.4	51050	2.9	51050	2.9
DeJ	45140	3.8	43450	2.2	39760	2.0	40670	2.1	40180	2.2	42840	2.3
GR	50050	3.9	51050	2.6	51050	2.6	51050	2.6	51050	3.1	51050	3.1
RB	50050	7.1	51050	6.5	51050	6.5	51050	6.5	51050	6.8	51050	6.8
ACK	50050	4.2	51050	2.4	46985	2.2	51050	2.3	48730	4.1	45320	3.6
Fun	GWPSO+UD		GWPSO+GD		GWPSO+ED		MPSO					
	NFE	Time	NFE	Time	NFE	Time	NFE	Time				
RS	50050	2.2	50050	2.4	50050	2.2	50050	3.2				
DeJ	34910	0.7	33850	0.7	34910	0.8	43570	2.9				
GR	50050	2.2	35640	1.2	50050	2.3	47540	3.1				
RB	50050	5.6	50050	5.9	50050	5.8	50050	8.8				
ACK	39890	1.9	36935	1.7	39145	1.8	46800	3.7				

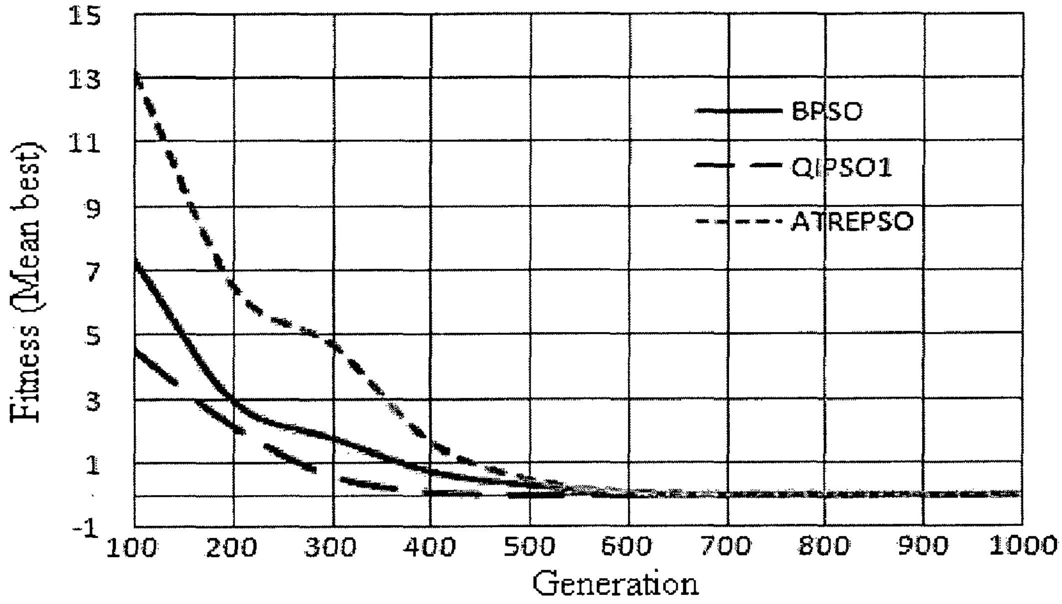

Figure 3.1 (b) Function DeJ

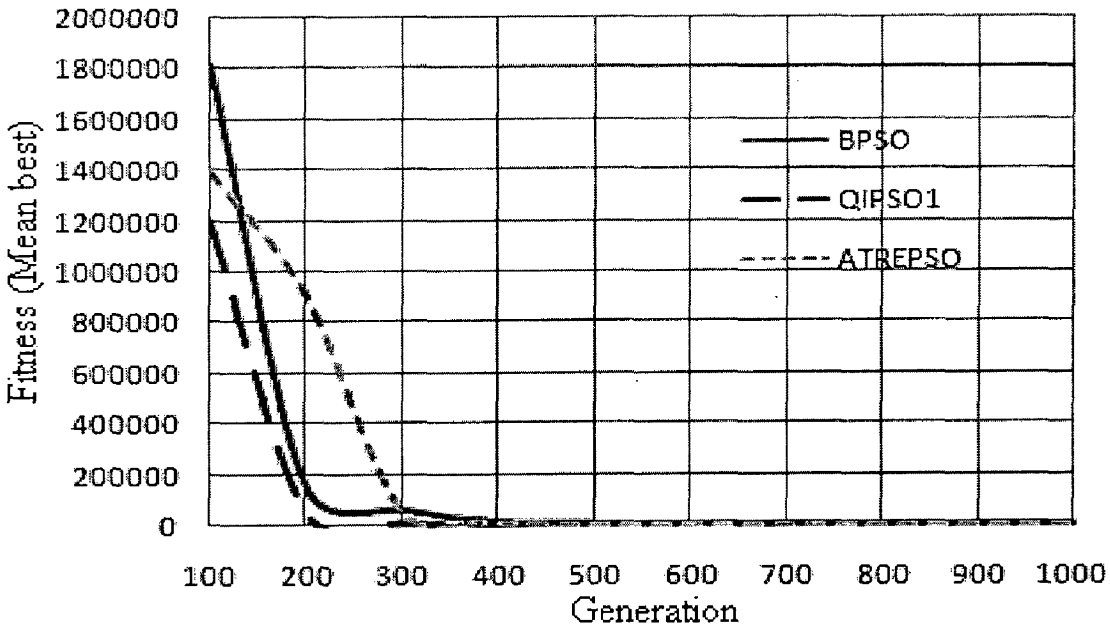

Figure 3.1 (c) Function RB

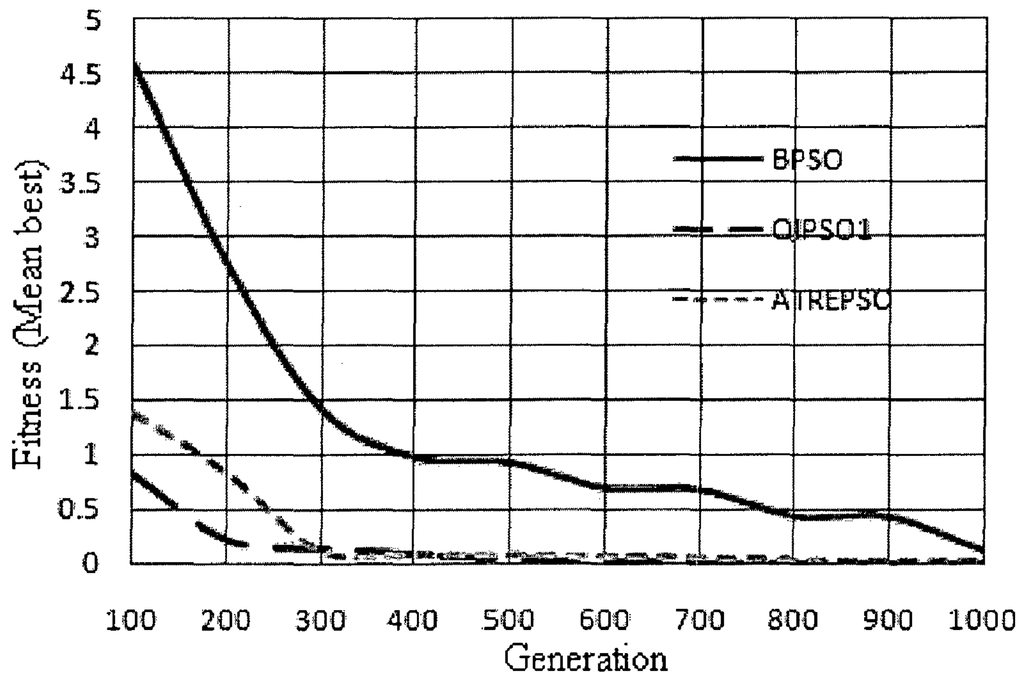

Figure 3.1 (d) Function DeJ – N

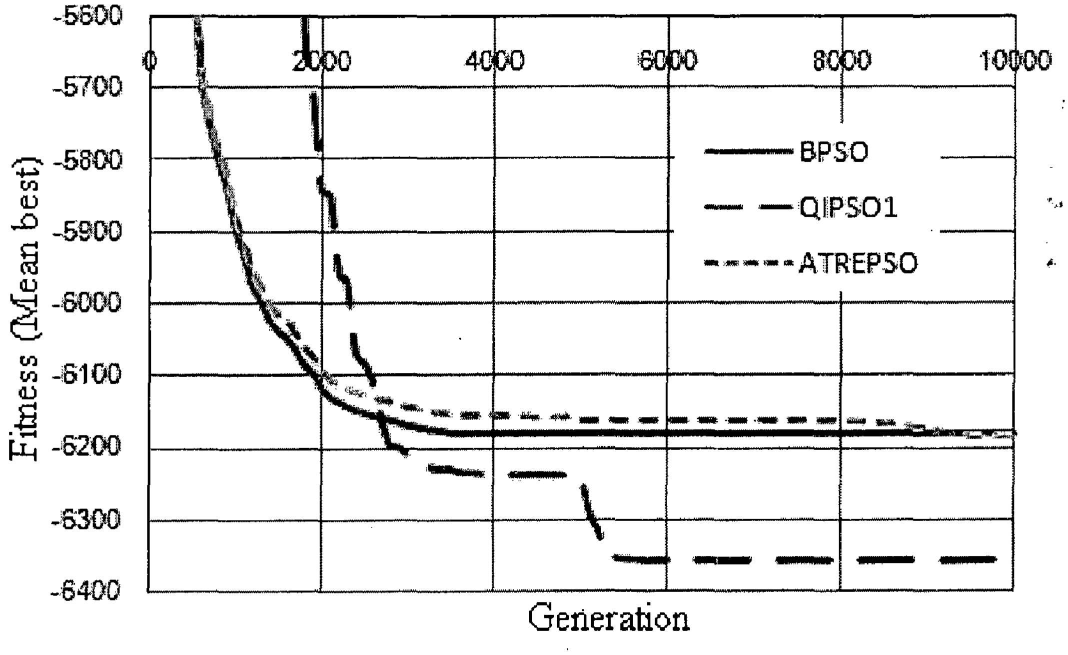

Figure 3.1 (e) Function SWF

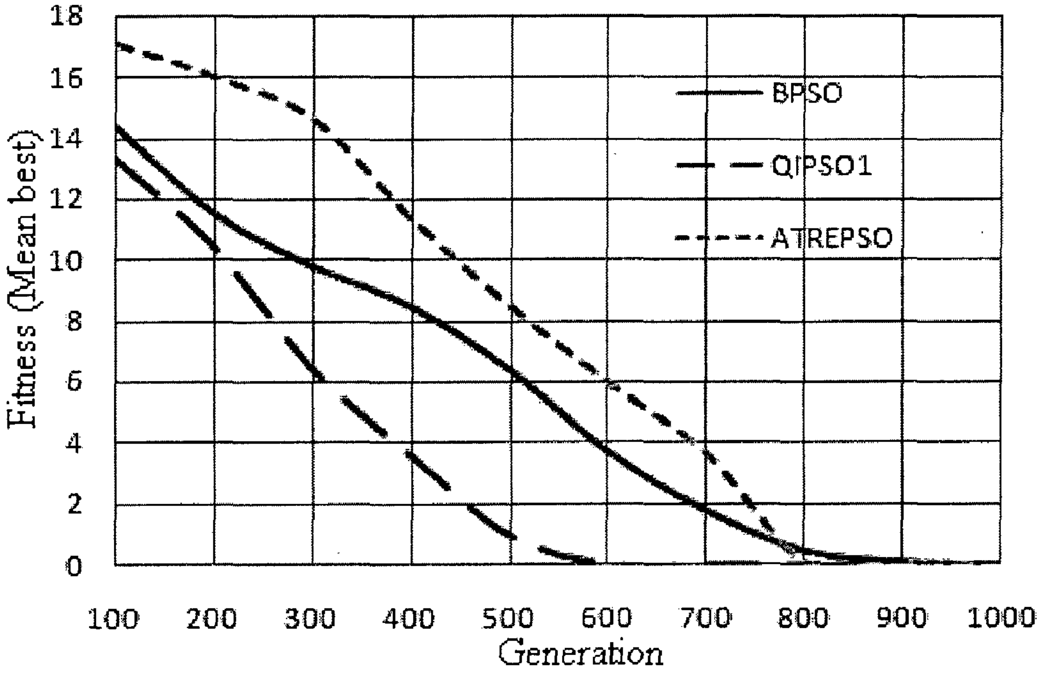

Figure 3.1 (f) Function ACK

Figure 3.1 Performance curves of PSO, QIPSO1 and ATREPSO algorithms

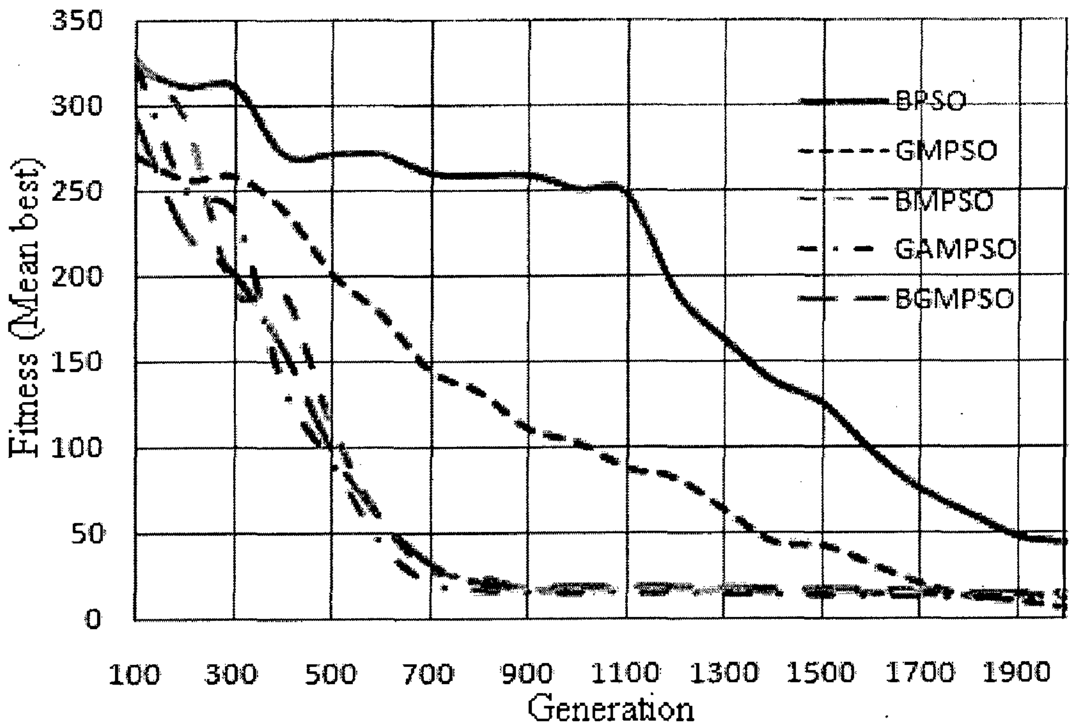

Figure 3.2 (a) Function RS

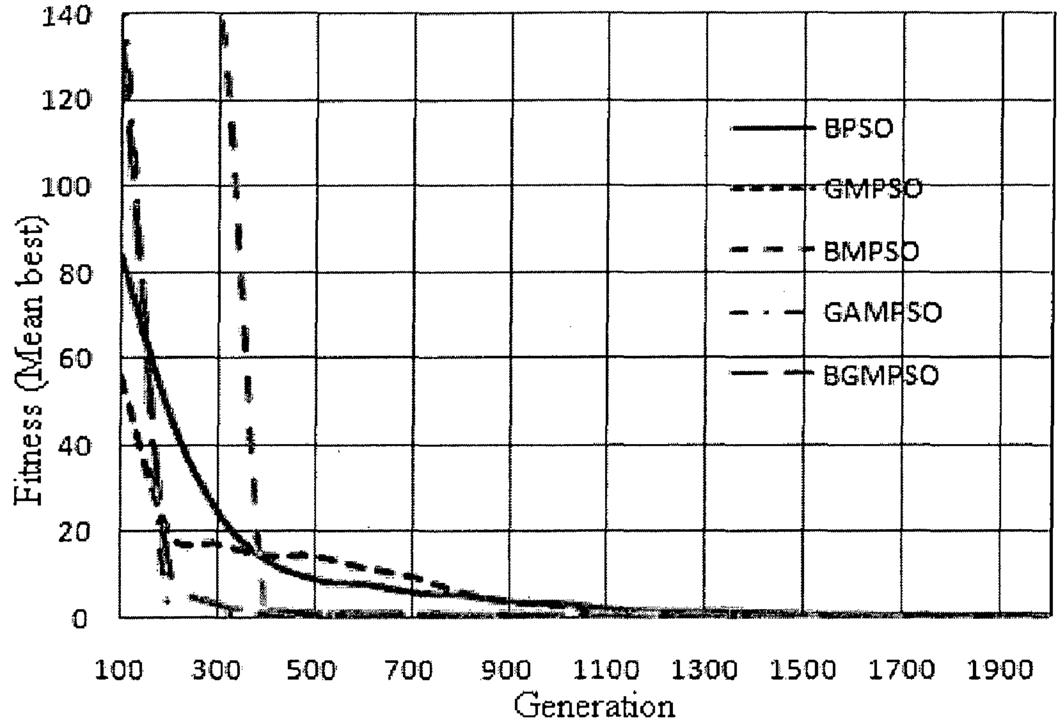

Figure 3.2 (b) Function DeJ-N

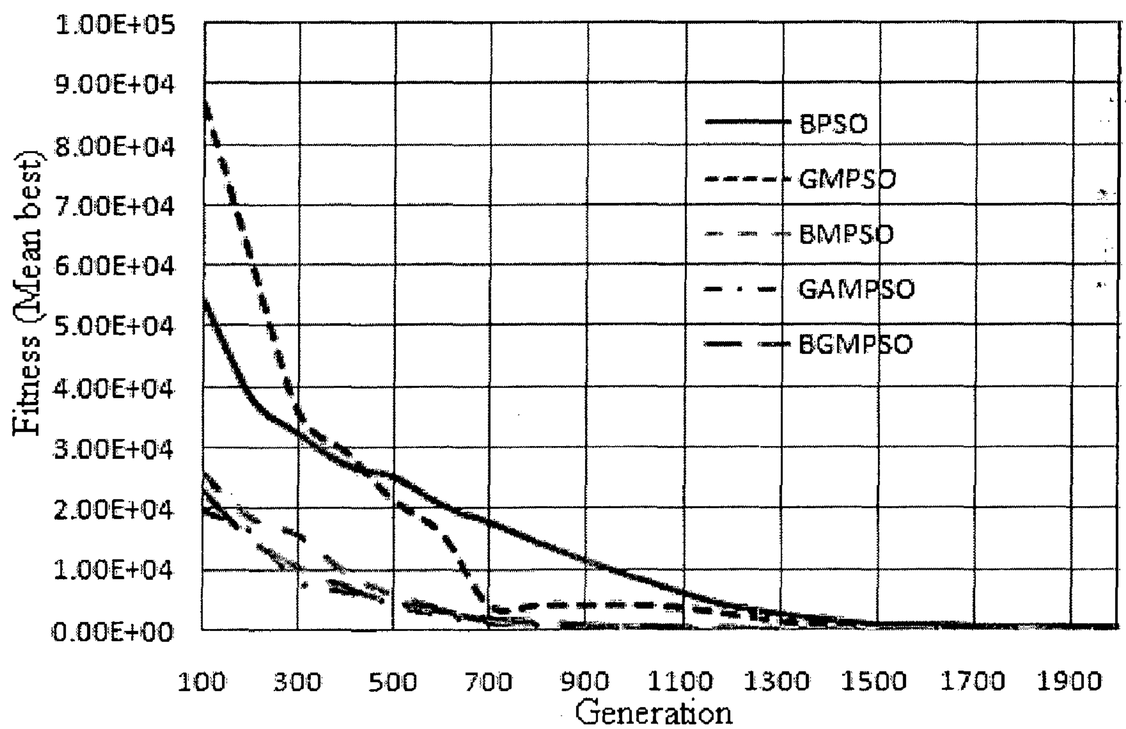

Figure 3.2 (c) Function SWF1.2

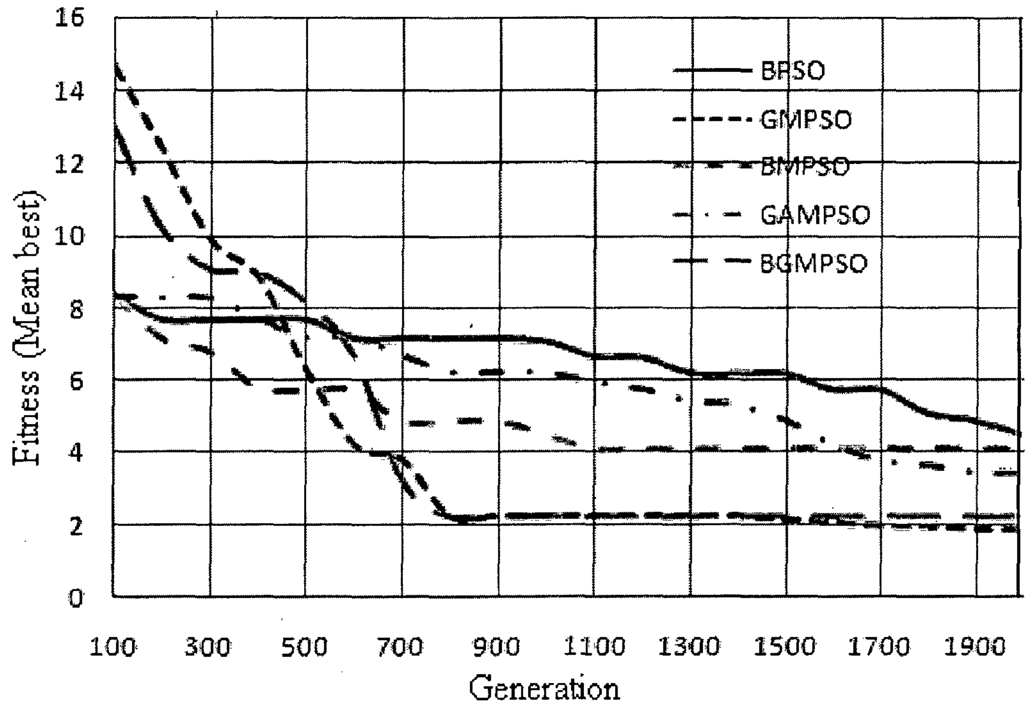

Figure 3.2 (d) Function SF7

Figure 3.2 Performance Curves of PSO, GMPSO, BMPSO, GAMPSO and BGMPSO algorithms

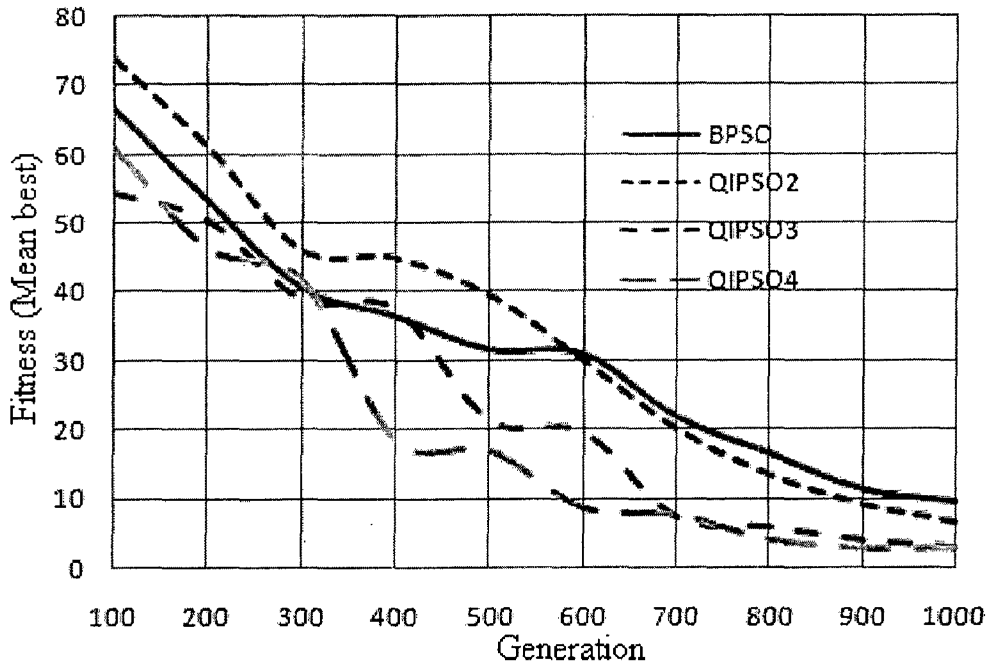

Figure 3.3 (a) Function RS

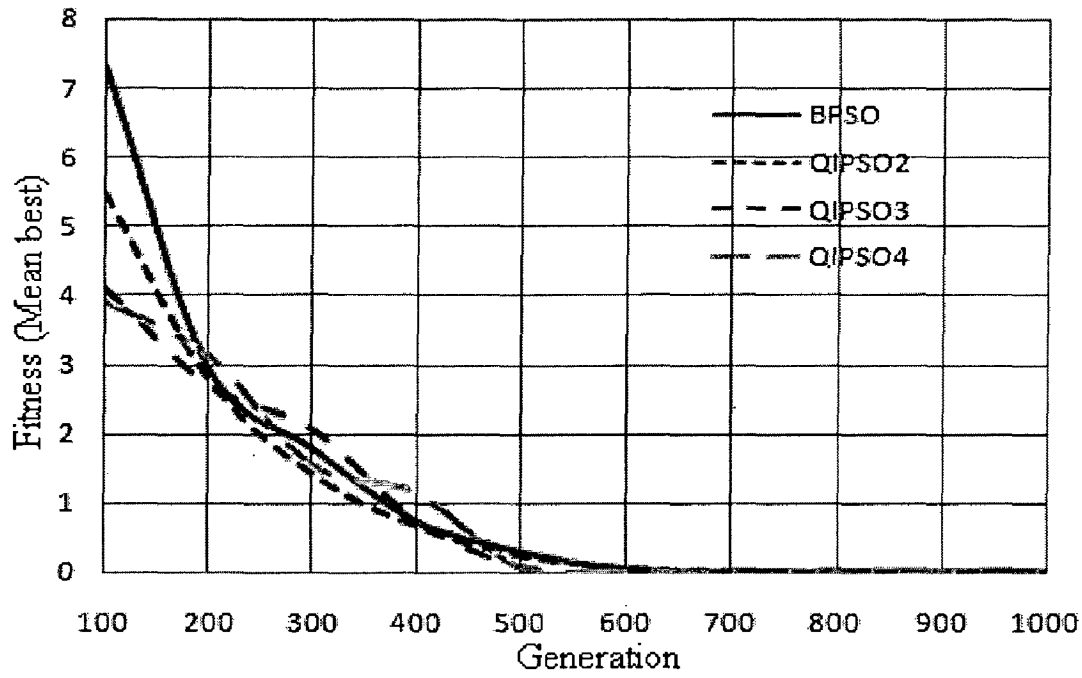

Figure 3.3 (b) Function DeJ

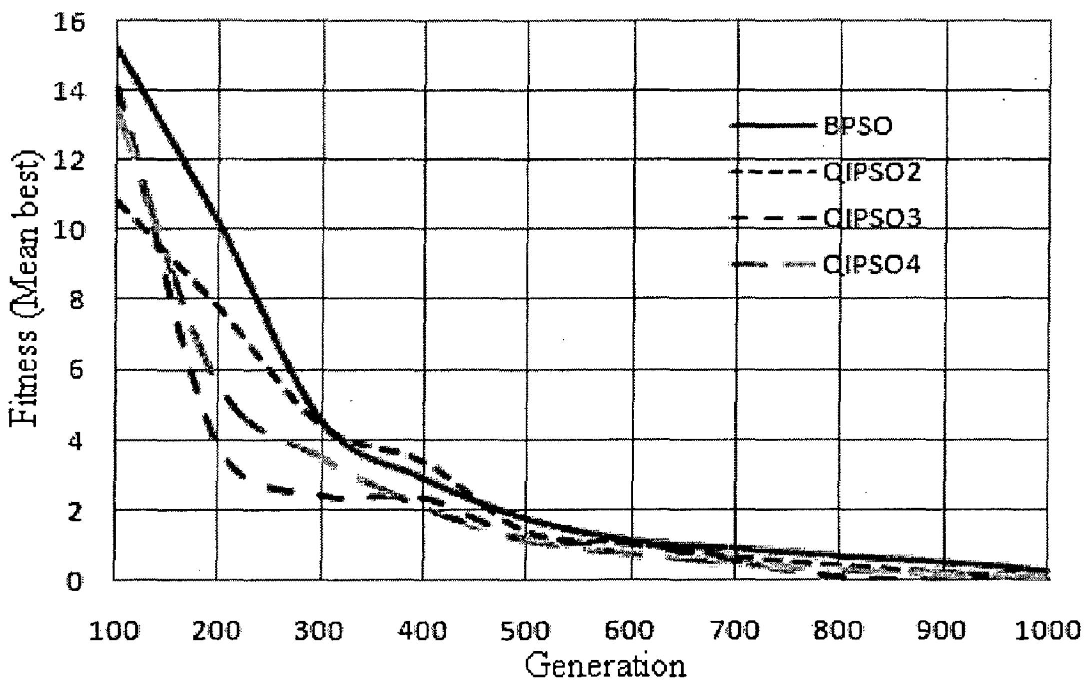

Figure 3.3 (c) Function GR

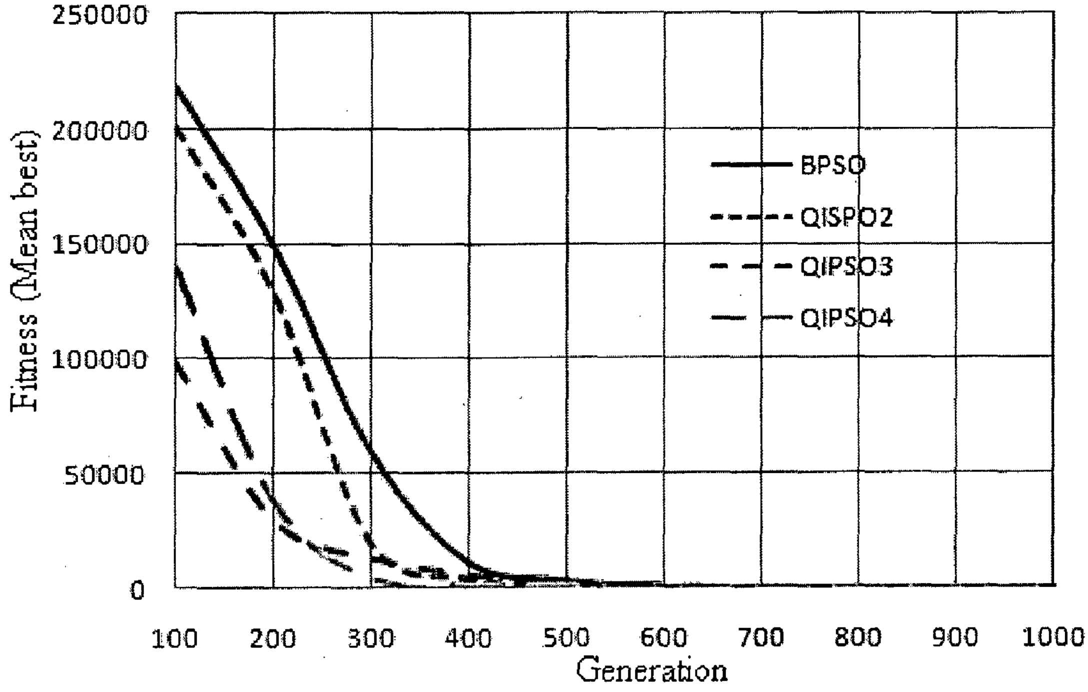

Figure 3.3 (d) Function RB

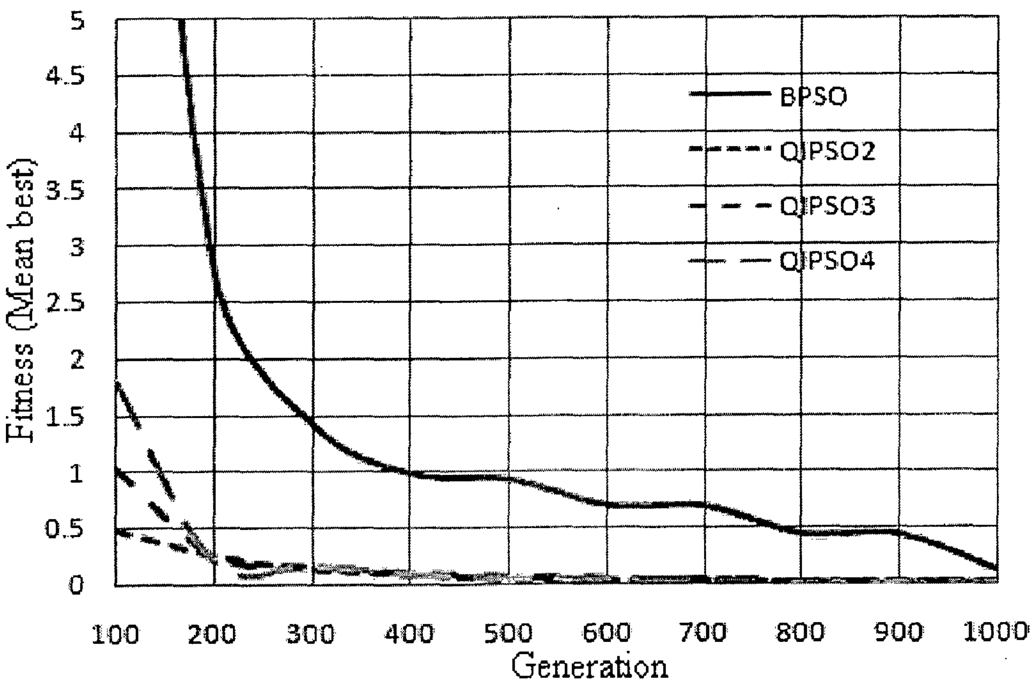

Figure 3.3 (e) Function DeJ-N

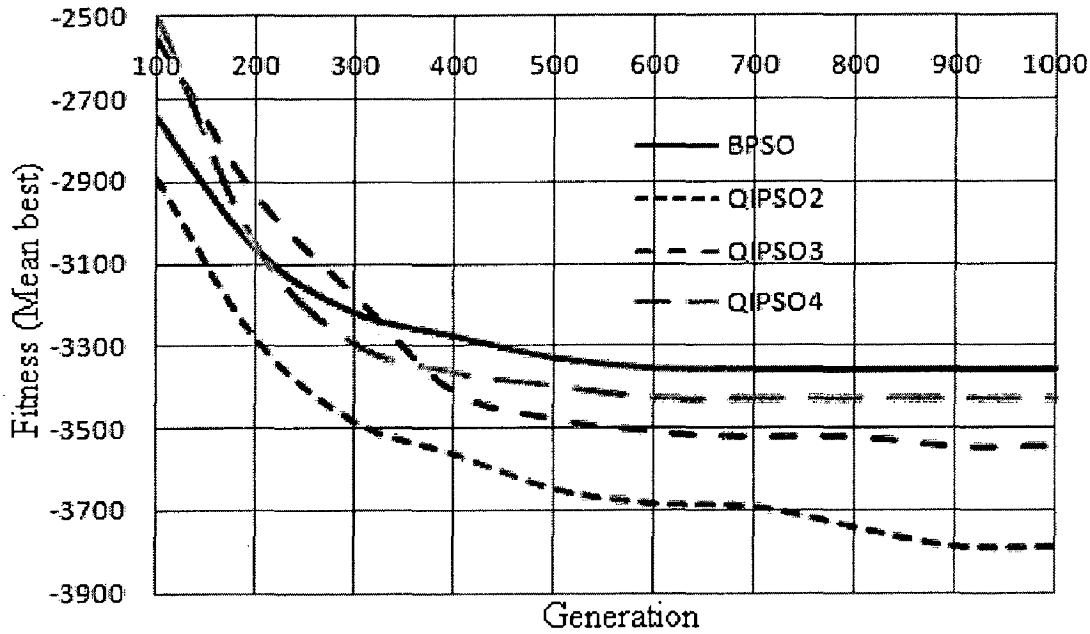

Figure 3.3 (f) Function SWF

Figure 3.3 Performance curves of PSO and the proposed QIPSO2, QIPSO3 and QIPSO4 algorithms

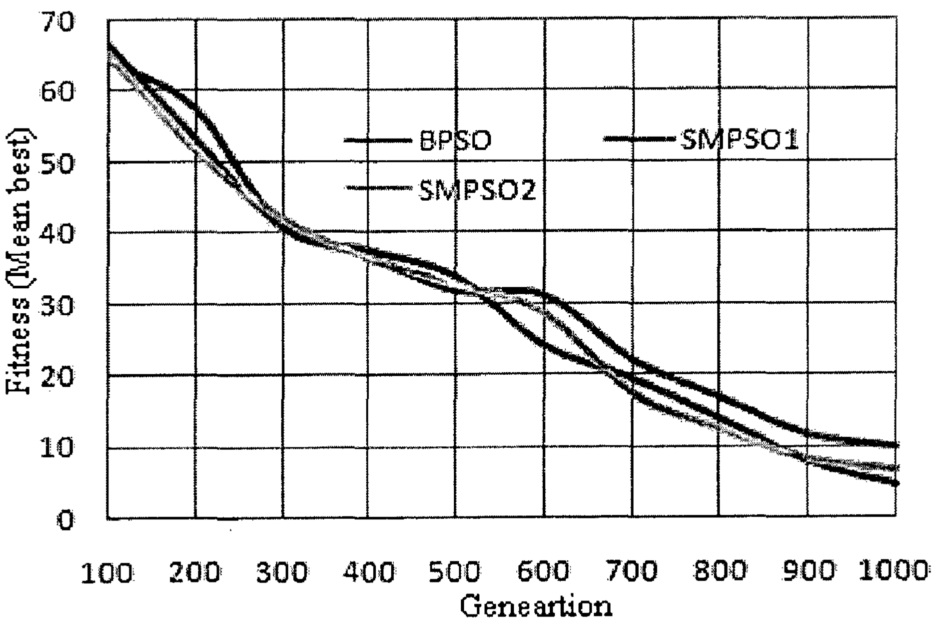

Figure 3.4 (a) Function RS

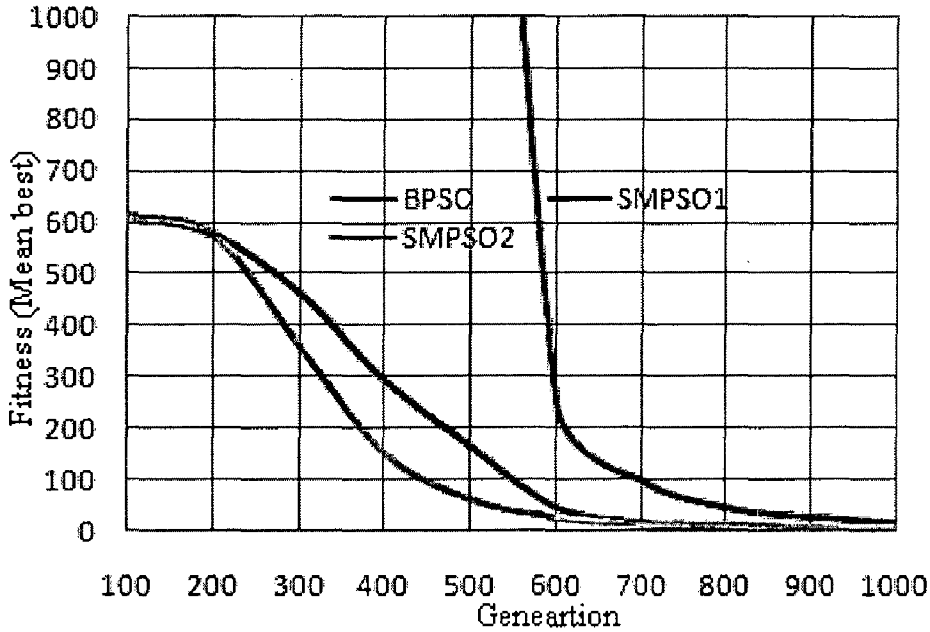

Figure 3.4 (b) Function RB

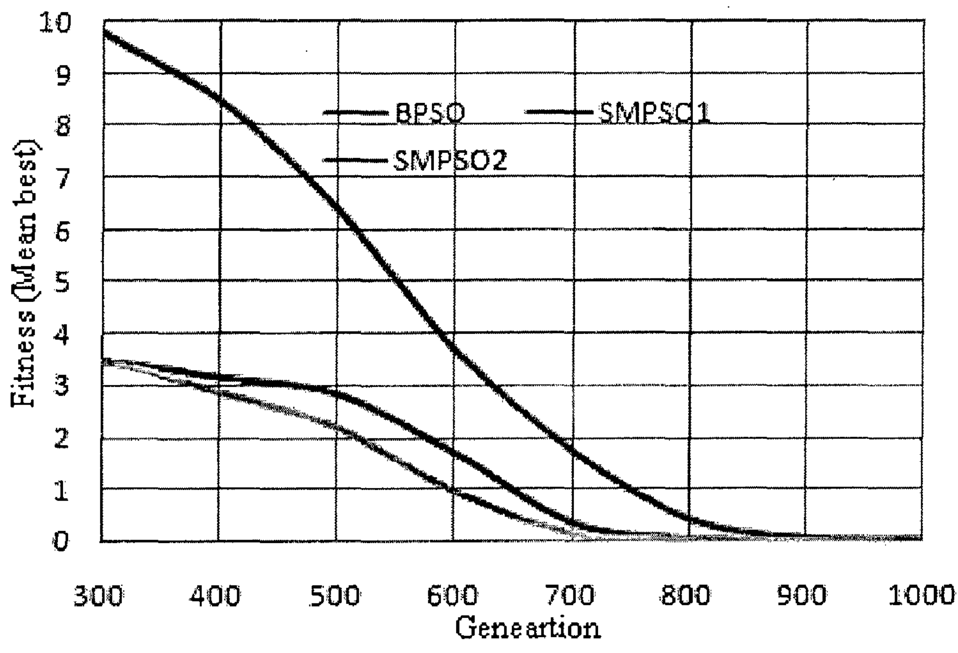

Figure 3.4 (c) Function ACK

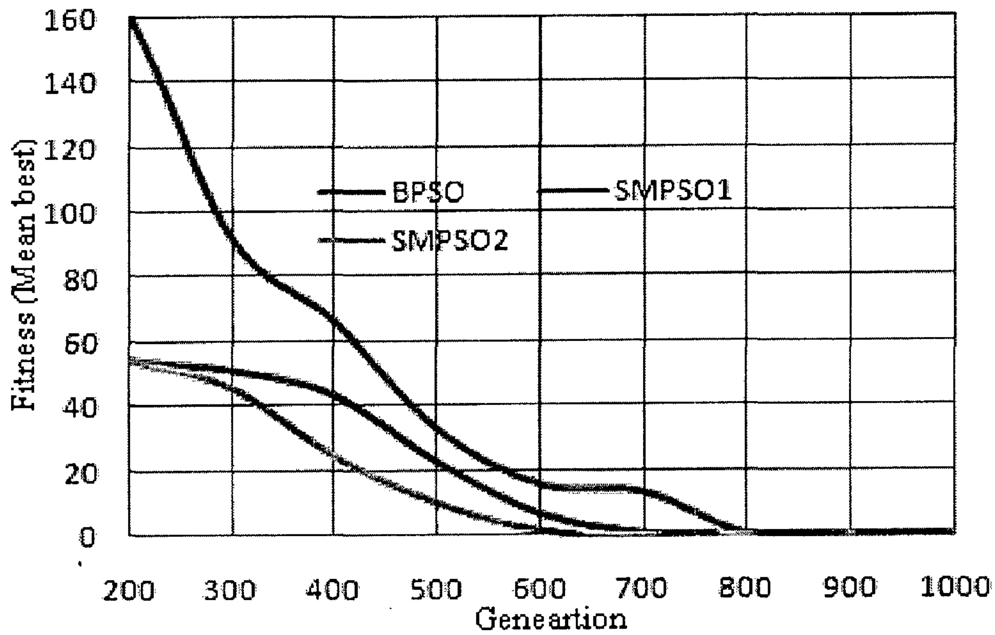

Figure 3.4 (d) Function SWF1.2

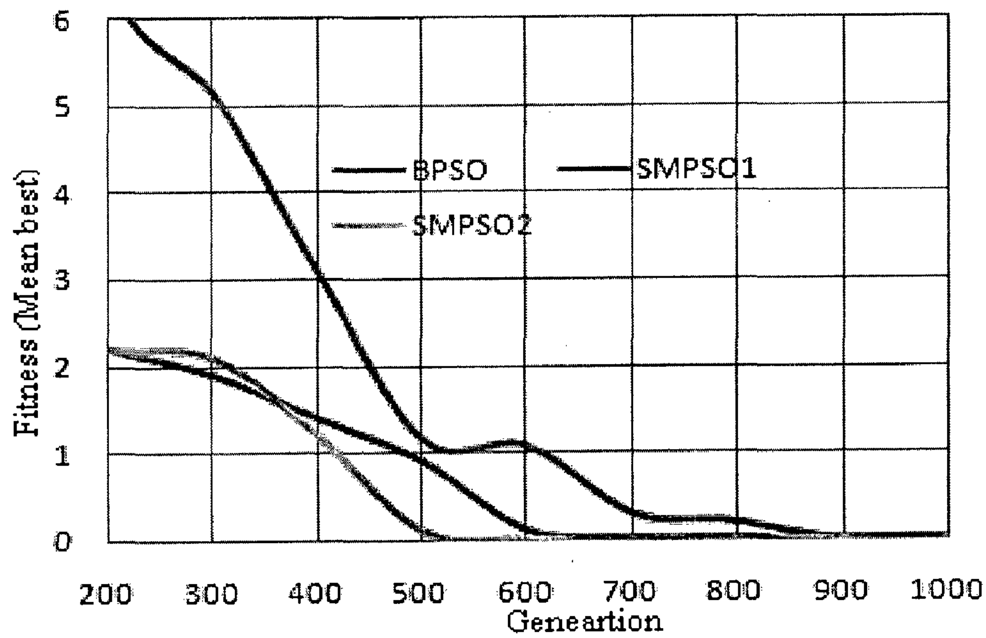

Figure 3.4 (e) Function ST

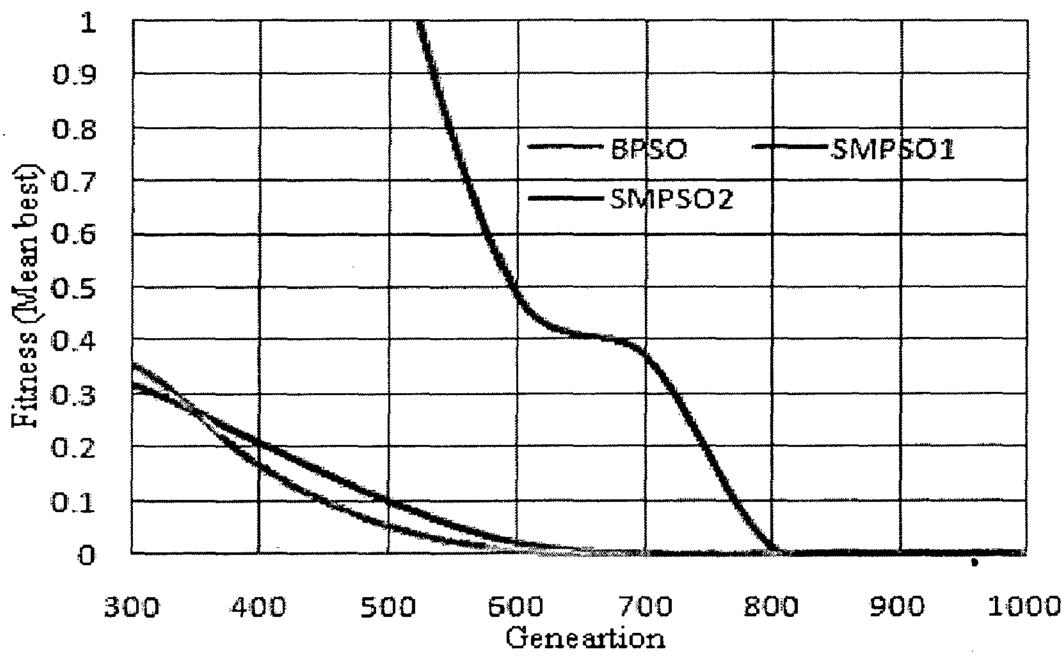

Figure 3.4 (f) Function GP1

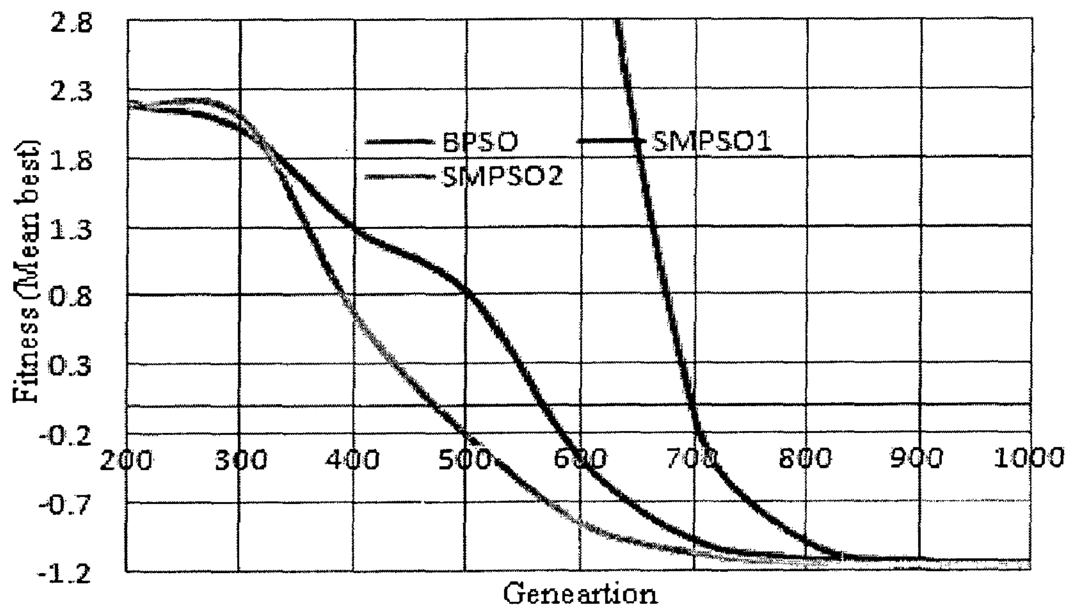

Figure 3.4 (g) Function GP2

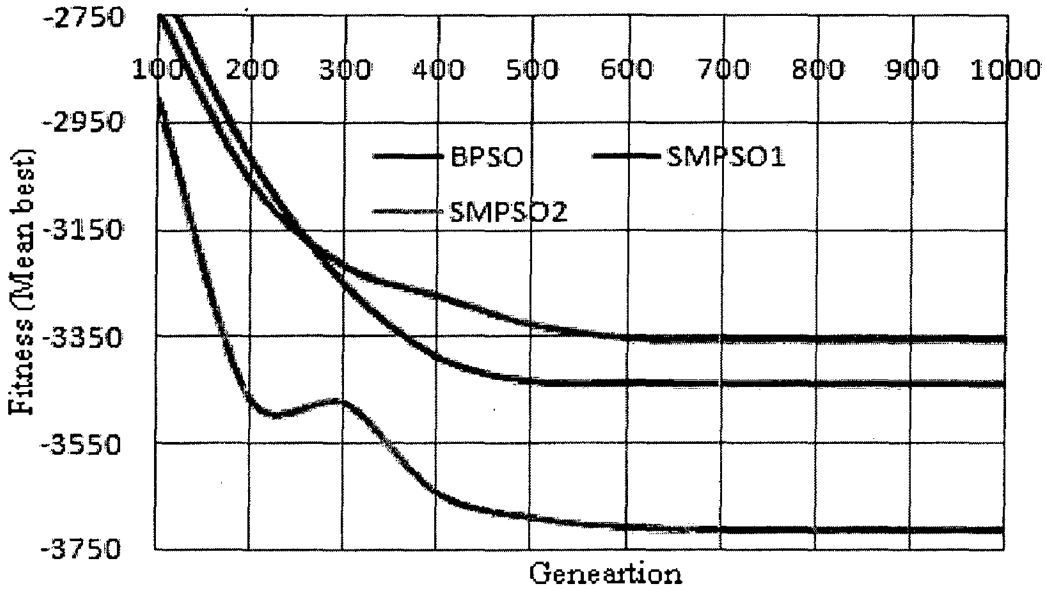

Figure 3.4 (h) Function SWF

Figure 3.4 Performance curves of PSO and proposed SMPSO algorithms

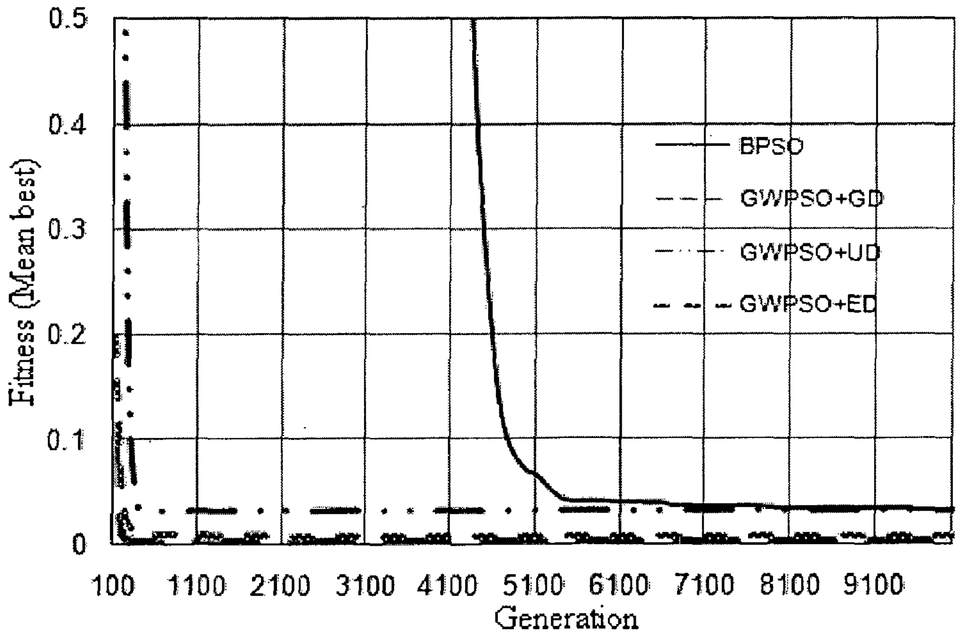

Figure 3.5 (a) Function GR

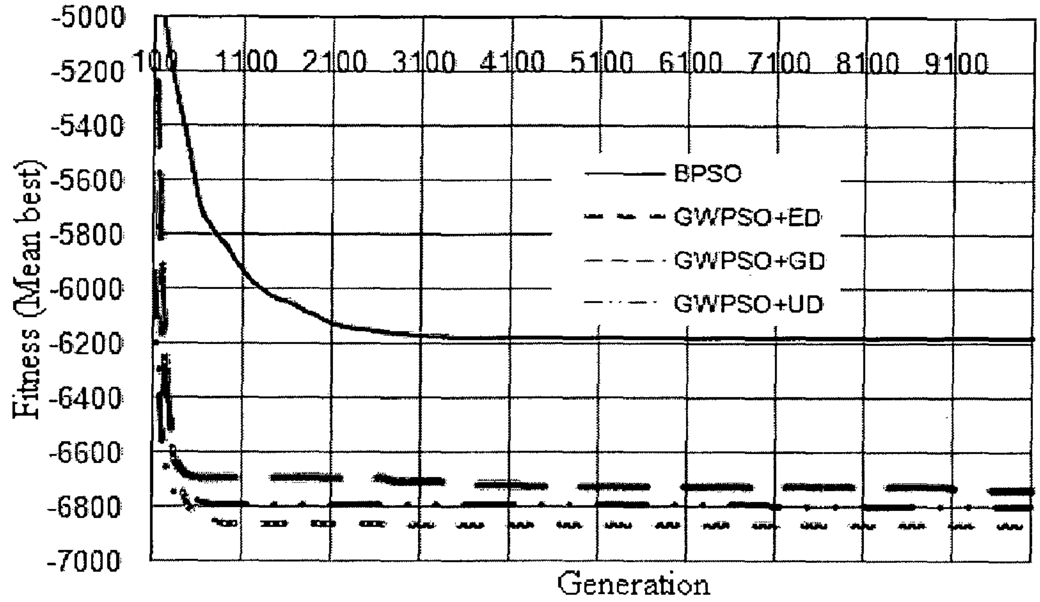

Figure 3.5 (b) Function SWF

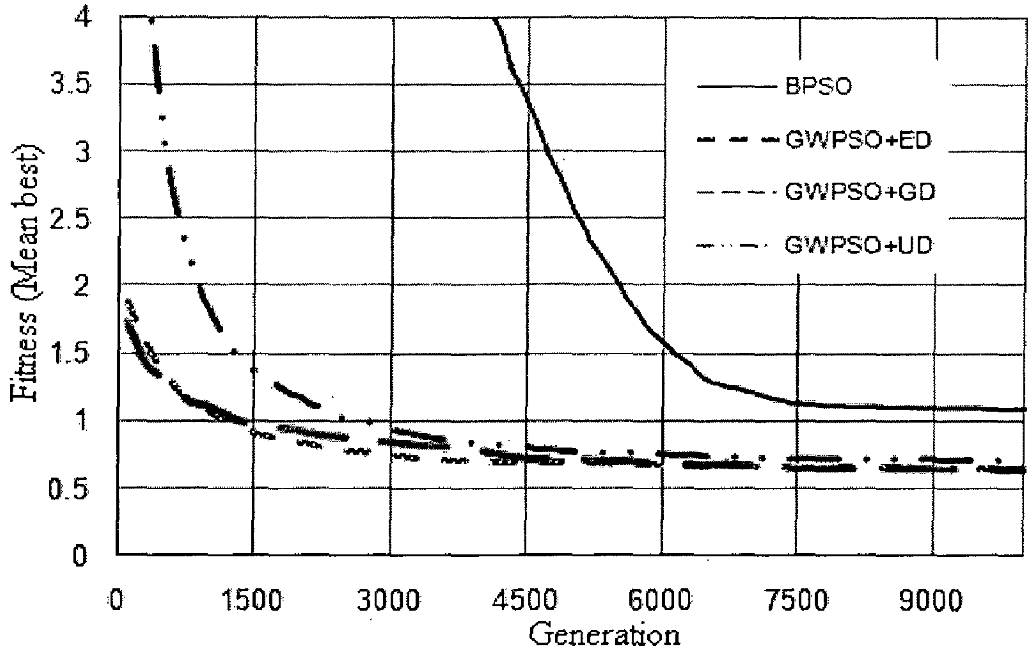

Figure 3.5 (c) Function SF7

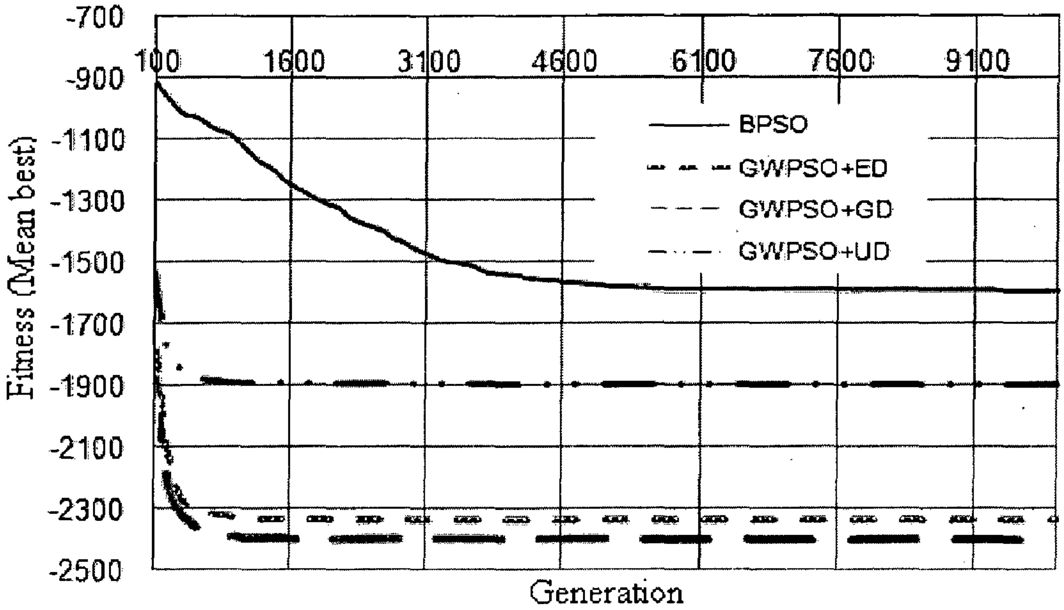

Figure 3.5 (d) Function SB2

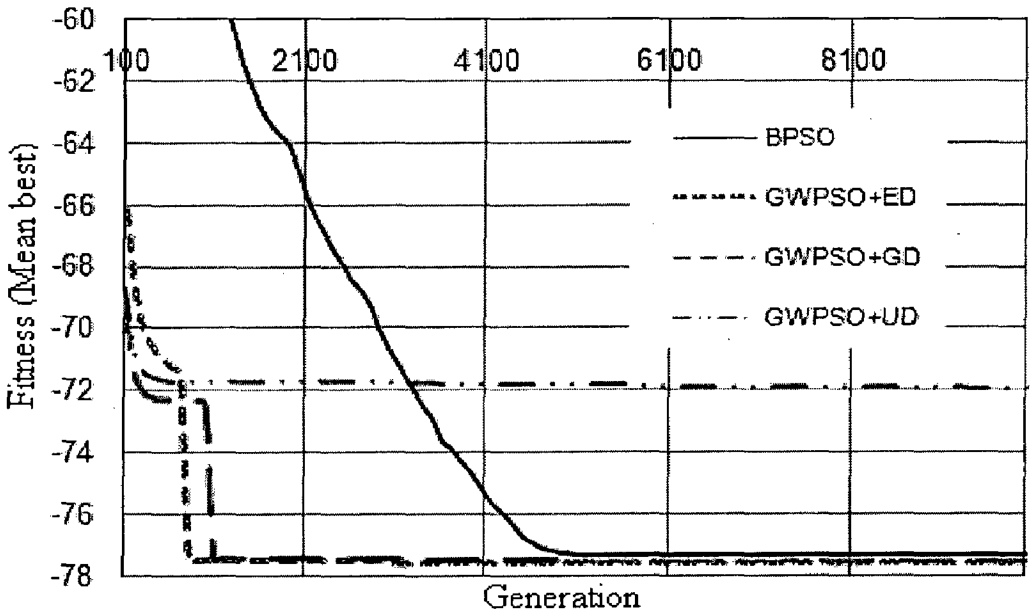

Figure 3.5 (e) Function T2N

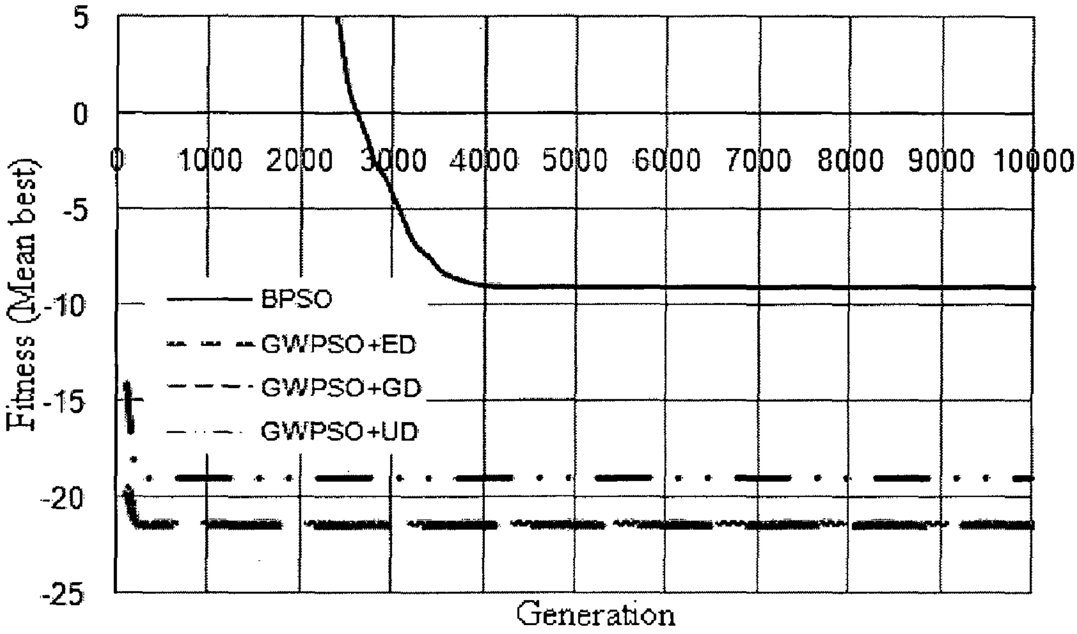

Figure 3.5 (f) Function LM

Figure 3.5 Performance curves of PSO and proposed GWPSO algorithms

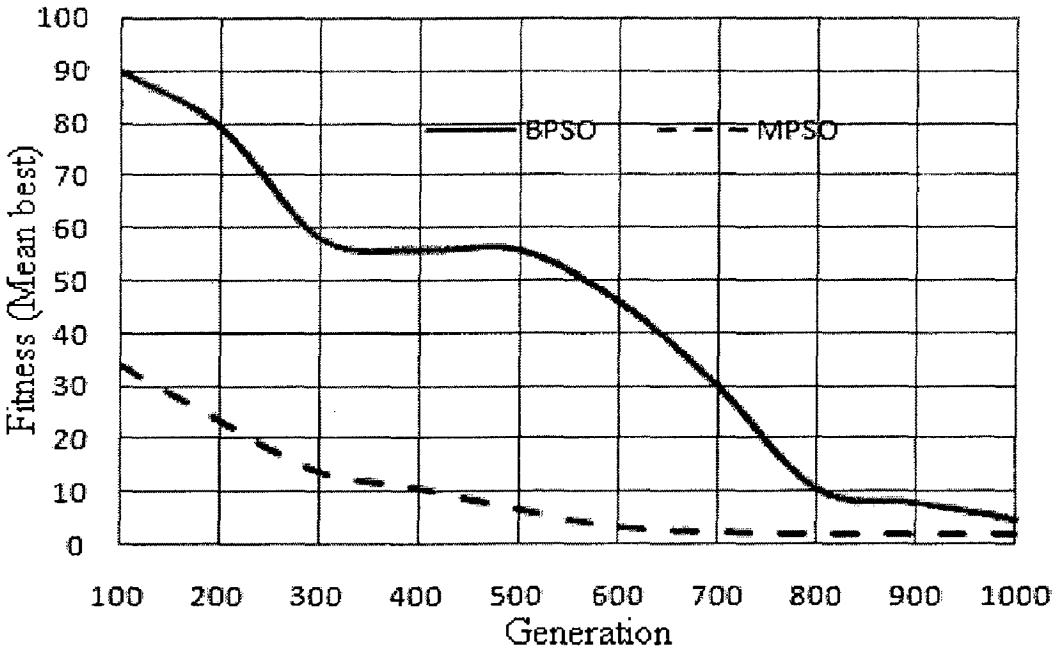

Figure 3.6 (a) Function RS

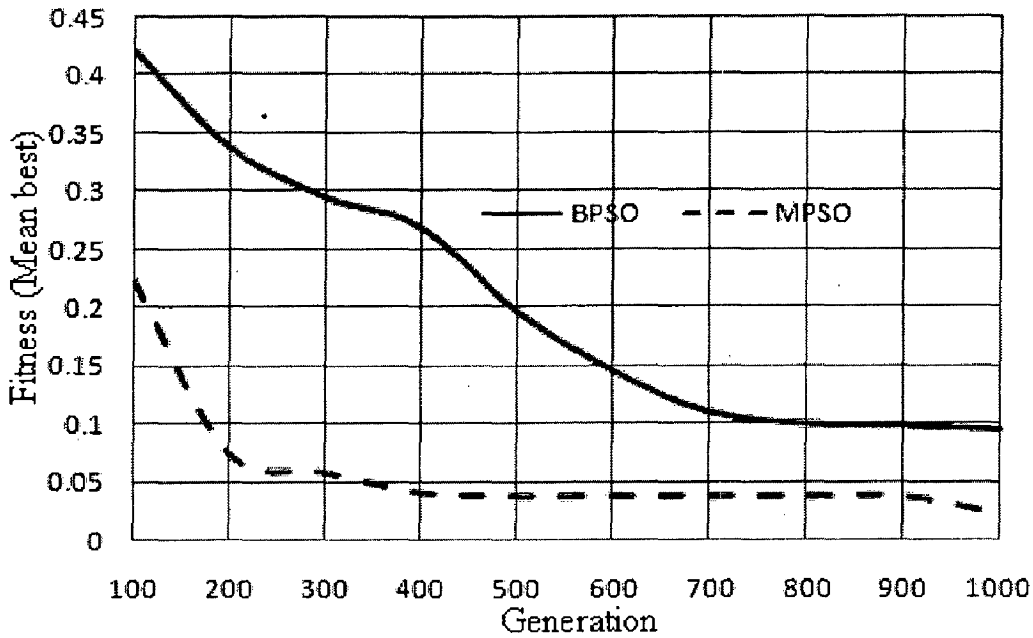

Figure 3.6 (b) Function GR

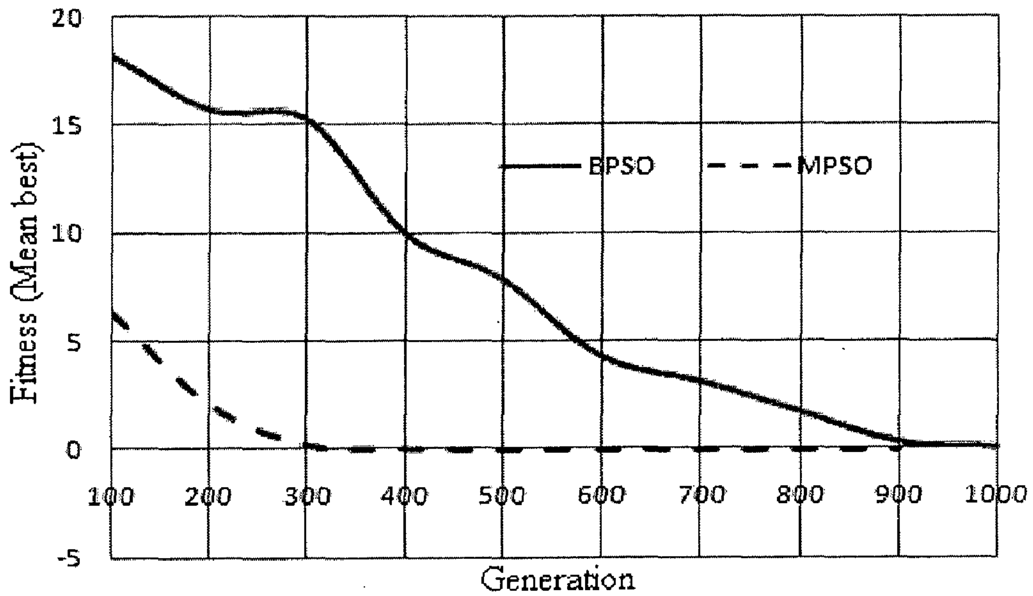

Figure 3.6 (c) Function ACK

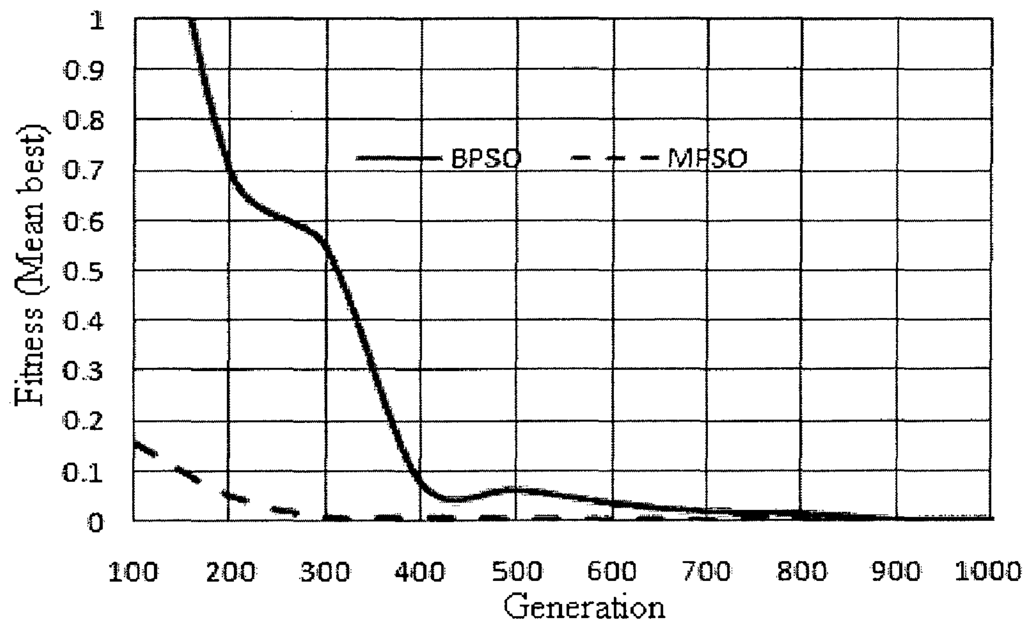

Figure 3.6 (d) Function DeJ-N

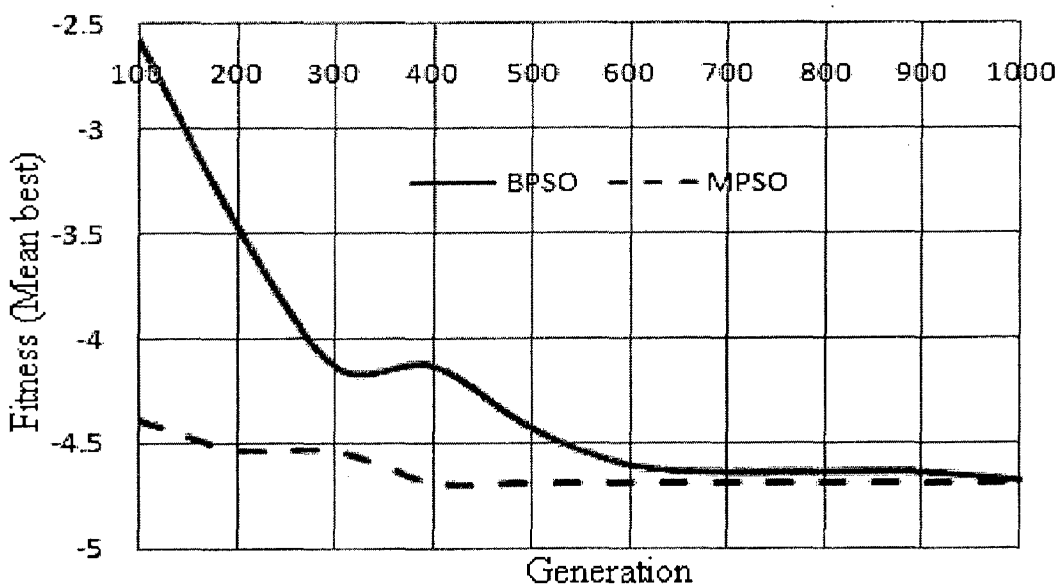

Figure 3.6 (e) Function Mic (5 dimension)

Figure 3.6 Performance curves of PSO and proposed MPSO algorithms

3.10 Conclusion

This chapter presented some modified versions of PSO algorithm. In a total of 15 modified versions of PSO were reported. These algorithms are based on diversity, mutation and crossover. Also a new inertia weight and a new velocity update equation were presented. The modified algorithms are:

- Attraction-Repulsion Particle Swarm Optimization (ATREPSO)
- Quadratic Interpolation Particle Swarm Optimization (QPSO1, QPSO2, QPSO3 and QPSO4)
- Gaussian Mutation Particle Swarm Optimization (GMPSO)
- Beta Mutation Particle Swarm Optimization (BMPSO)
- Gamma Mutation Particle Swarm Optimization (GAMPSO)
- Beta & Gamma Mutation Particle Swarm Optimization (BGMPSO)
- Sobol Mutated Particle Swarm Optimization (SMPSO1 and SMPSO2)
- Gaussian Inertia Weight Particle Swarm Optimization (GWPSO)
- Modified Particle Swarm Optimization with New Velocity (MPSO)

The performance of presented algorithms was tested with some standard benchmark problems. The results obtained by these algorithms on all benchmark problems were either superior or at par with the basic PSO algorithm. In overall comparison, among other algorithms PSO assisted with Quadratic Interpolation operator algorithms gave the best results.

Chapter 3A

Improved Quantum Particle Swarm Optimization Algorithms

[This chapter is an extension of chapter 3. In this chapter a new concept in the field of PSO namely Quantum Particle Swarm Optimization (QPSO) algorithm is discussed and different versions based on QPSO are proposed.]

3A.1 Quantum Particle Swarm Optimization

One of the recent developments in the field of PSO is the application of Quantum laws of mechanics in the structure of PSO. Such PSO's are called Quantum PSO (QPSO). The development in the field of quantum mechanics is mainly due to the findings of Bohr, de Broglie, Schrödinger, Heisenberg and Bohn in 1920's. Their studies gave a different meaning to the concepts of classical mechanics and the traditional understanding of the nature of motions of microscopic objects (Pang, 2005). Recently, the concepts of quantum mechanics and physics have gained considerable attention in the development of optimization techniques (Hogg, Portnov, 2000; Protopescu and Barhen, 2002; Bulger, Baritompa and Wood, 2003).

As per classical PSO, a particle is stated by its position vector x_i and velocity vector v_i, which determine the *trajectory* of the particle. The particle moves along a determined trajectory following Newtonian mechanics. However if we consider quantum mechanics, then the term trajectory is meaningless, because x_i and v_i of a particle cannot be determined simultaneously according to *uncertainty principle*.

Therefore, if individual particles in a PSO system have quantum behavior, the performance of PSO will be far from that of classical PSO (Feng and Xu, 2004). In the quantum model of a PSO, the state of a particle is depicted by wavefunction $\Psi(x,t)$, instead of position and velocity. The dynamic behavior of the particle is widely divergent from that of the particle in traditional PSO systems. In this context, the probability of the particle's appearing in position x_i from probability density function $|\Psi(x,t)|^2$, the form of which depends on the potential field the particle lies (Liu et al, 2006).

The particles move according to the following iterative equations (Sun et al, 2004a; Sun et al, 2004b):

$$x(t+1) = p + \beta * |mbest - x(t)| * \ln(1/u) \ if \ k \geq 0.5$$

$$x(t+1) = p - \beta * |mbest - x(t)| * \ln(1/u) \ if \ k < 0.5 \quad (3A.1)$$

where

$$p = (c_1 P_{id} + c_2 P_{gd})/(c_1 + c_2) \quad (3A.2)$$

$$mbest = \frac{1}{M}\sum_{i-1}^{M} P_i = \left(\frac{1}{M}\sum_{i=1}^{M} P_{i1}, \frac{1}{M}\sum_{i=1}^{M} P_{i2}, \ldots, \frac{1}{M}\sum_{i=1}^{n} P_{id}\right) \quad (3A.3)$$

Mean best (mbest) of the population is defined as the mean of the best positions of all particles, u, k, c_1 and c_2 are uniformly distributed random numbers in the interval [0, 1]. The parameter β is called contraction-expansion coefficient. The computation steps of QPSO algorithm are given below:

Step 1	*Initialize the swarm with uniformly distributed random numbers.*
Step 2	*Calculate mbest using equation (3A.3)*
Step 3	*Update particles position using equation (3A.1)*
Step 4	*Evaluate the fitness value of each particle*
Step 5	*If the current fitness value is better than the best fitness value (Pbest) in history Then update P_i (personal best) by the current fitness value*
Step 6	*Update P_g (global best)*
Step 7	*Go to step 2 until maximum iteration is reached*

QPSO algorithm is depicted only with the position vector without velocity vector, which is a simpler algorithm. And the results show that QPSO performs better than basic PSO on several benchmark test functions and is a promising algorithm due to its global convergence guaranteed characteristic (Liu et al, 2005; Sun et al, 2006; Liu et al, 2006).

3A.2 A Brief Review of QPSO

Xu and Sun (2005) introduced a diversity guided model in QPSO with two phases attraction and repulsion. This algorithm is similar to ARPSO algorithm of Riget et al (2002). Liu et al (2005; 2006) introduced a mutation operator with the help of probability distribution in QPSO, in which particle's global best position and the variable mbest are mutated by using Cauchy distribution. Sun et al (2006) proposed a new QPSO called DCQPSO, which is a method of controlling the diversity of QPSO. Again Sun et al (2006a) explored the applicability of the QPSO to data clustering and proved that QPSO has overall better performance than K-means and PSO clustering algorithms for data clustering, because the QPSO is a global convergent optimization algorithm. A Quantum PSO algorithm with chaotic mutation operator is introduced by Coelho (2006) and he proved that the chaotic mutation based QPSO is a powerful strategy to diversify the QPSO population and improve the QPSO's performance in preventing premature convergence to local minima.

A diversity guided QPSO (DGQPSO) is proposed by Sun et al (2006b), in which a mutation operator is exerted on global best position of the particle to prevent the swarm from clustering, enabling the particle to escape the sub-optimal solution. Again the same authors introduced a diversity maintained algorithm in QPSO (Sun et al, 2006c).

From the update equations of PSO or QPSO, we can see that all particles in PSO or QPSO will converge to a common point, leaving the diversity of the population extremely low and particles stagnated without further search before the iteration is over. To overcome this problem Sun et al (2007) proposed a new QPSO called Revised QPSO by exerting a Gaussian disturbance on the mean best position of the particle. Coelho et al (2008) used the Gaussian probability distribution in QPSO. G-QPSO algorithm is developed by them, in which the constriction factor (β) of QPSO follows the Gaussian distribution and the proposed algorithm is applied to tune the design parameters of Fuzzy Logic Control with Proportional-Integral-derivative conception.

Some other improved versions of QPSO are Improved QPSO by Simulated Annealing (Liu et al, 2006a), QPSO with binary coding (Sun et al, 2007a), QPSO with immune operator (Liu et al, 2006b), QPSO with generalized local search operator (Wang and Zhou, 2007), QPSO with hybrid probability distribution (Sun et al, 2006d), modified QPSO (Sun et al, 2007b), etc.

In order to further improve the performance of QPSO, in this chapter a recombination operator based on quadratic interpolation (QI operator) and a mutation operator based on the low discrepancy sobol sequence (SM operator) are incorporated in the QPSO algorithm. The following sections present four algorithms namely Quadratic Interpolation based QPSO (Q-QPSO1, Q-QPSO2) and Sobol Mutated QPSO (SMQPSO1, SMQPSO2) which uses QI operator and SM operator respectively to improve the performance of the swarm.

3A.3 Quadratic Interpolation based Quantum PSO (Q-QPSO)

The proposed Q-QPSO algorithm is a simple and modified version of QPSO in which we have introduced the concept of recombination. Also the Q-QPSO algorithm is similar to QIPSO algorithms (section 3.4), which uses the QI crossover operator to improve the performance of QPSO. The QI operator is a nonlinear operator which produces a new solution vector lying at the point of minima of the quadratic curve passing through the three selected swarm particles. For more details about the QI operator refer section 3.2.2. Two versions of Q-QPSO algorithms are proposed in this section. In Q-QPSO1, QI operator is applied to the global best (gbest) particle, where as in Q-QPSO2, the crossover operator is applied to the worst particle of the swarm.

The Q-QPSO algorithm starts like the usual QPSO using Eqns. (3A.1), (3A.2) and (3A.3). At the end of each iteration, the QI recombination operator is invoked to generate a new swarm particle. The new particle is accepted in the swarm only if it is better than the global best particle (i.e. the particle having minimum fitness) present in the swarm in Q-QPSO1, whereas in Q-QPSO2, the new particle is accepted in the swarm only if it is better than the worst particle in the swarm. This process is repeated iteratively until a better solution is obtained.

The simple flow of Q-QPSO1 is given below:

Initialize the Swarm

Do

Calculate mbest by Eqn. (3A.3)

Update particle's position vector using Eqn. (3A.1)

Update P_i *and* P_g

Find a new particle using QI operator

If the new particle is better than the global best particle in the swarm then

Replace the global best (P_g) particle by the new particle

While (stopping criterion is reached)

The flow of Q-QPSO2 is same as that of Q-QPSO1, except for the fact that the worst particle in the swarm is mutated instead of the best particle.

3A.4 Sobol Mutated Quantum PSO (SMQPSO)

The proposed SMQPSO algorithm is an extension to the quantum Particle Swarm Optimization, by including the component of mutation in it and it is similar to the SMPSO algorithm (section 3.5). The SMQPSO algorithm uses the SM operator (refer section 3.5) of SMPSO algorithm to mutate the global best and worst particle in the swarm. Two versions of SMQPSO algorithm are proposed; they are: SMQPSO1 and SMQPSO2. The two versions differ from each other in the sense that in SMQPSO1, the global best particle of the swarm is mutated, whereas in SMQPSO2, the worst particle of the swarm is mutated. The idea behind applying the mutation to the worst particle is to push the swarm from the back. The quasi random numbers used in the SM operator allows the worst particle to move forward systemically.

The simple flow of SMQPSO1 is given below:

Initialize the Swarm

Do

Calculate mbest by Eqn. (3A.3)

Update particle's position vector using Eqn. (3A.1)

Update P_i and P_g

Find a new particle using SM operator

If the new particle is better than the global best particle in the swarm then

Replace the global best (P_g) particle by the new particle

While (stopping criterion is reached)

The flow of SMQPSO2 is same as that of SMQPSO1, except for the fact that the worst particle in the swarm is mutated instead of the best particle.

3A.5 Parameter Settings of Proposed QPSO Algorithms

In the present study three benchmark problems (RS, GR and RB), are considered. The mathematical models of the benchmark problems are given in Appendix I. All the test problems are highly multimodal and scalable in nature. The real optimum of all the test problems is zero. Each function is tested with a swarm size of 20, 40 and 80 for dimension 10, 20, 30. The maximum number of generations is set as 1000, 1500 and 2000 corresponding to the dimensions 10, 20 and 30 respectively. A total of 30 runs for each experimental setting are conducted and the average fitness of the best solutions throughout the run is recorded. Also, comparison of the proposed algorithms is done with basic PSO and QPSO.

3A.6 Numerical Results and Discussion

The mean best fitness value for the functions RS, GR and RB are given in Tables 3A.1 – 3A.3, respectively, in which Pop represents the swarm population, Dim represents the dimension and Gne represents the maximum number of permissible generations. Table 3A.4 – 3A.6 shows the improvement (%) of proposed algorithms in comparison with QPSO. Figure 3A.1 shows the performance curves of PSO, QPSO and the proposed QPSO variants.

The numerical results show that in all the test cases except six cases (out of 27 cases) in Griewank function the proposed algorithms perform much better than the other algorithms. If one compares the performance of proposed algorithms with each other then from the numerical results it can be seen Q-QPSO1 gave better results than the other algorithms in 17 test cases out of the total 27 cases tried. Q-QPOS2 algorithm gave better solution than other algorithms in 8 test cases. Remaining two test cases SMQPSO2 performs better than others. From the comparison results of all the proposed algorithms with the algorithms in the literature, it can be seen that all the proposed versions of QPSO gave better results than the variants of QPSO in the literature. Thus from the numerical results, it is concluded that the proposed algorithms improved the performance of QPSO with a noticeable percentage.

Table 3A.1 Comparison results of function RS in terms of average fitness value

Pop	Dim	Gen	PSO	QPSO	Q-QPSO1	Q-QPSO2	SMQPSO1	SMQPSO2	Mutation gbest (Liu et al, 2005)	Mutation gbest Feng and Xu, (2004)
20	10	1000	5.5382	5.2543	4.73e-08	4.92e-19	1.46912	1.216721	5.2216	4.3976
	20	1500	23.1544	16.2673	4.10e-08	7.81e-19	9.242158	4.625661	16.1562	14.1678
	30	2000	47.4168	31.4576	9.07e-07	6.07e-19	11.405516	10.827897	26.2507	25.6415
40	10	1000	3.5778	3.5685	9.11e-19	6.55e-13	0.980288	1.153735	3.3361	3.2046
	20	1500	16.4337	11.1351	2.60e-19	8.67e-19	3.13032	3.813158	10.9072	9.5793
	30	2000	37.2896	22.9594	4.34e-19	5.49e-19	6.381452	9.119192	19.6360	20.5479
80	10	1000	2.5646	2.1245	7.37e-19	0.86794	1.494764	0.095692	2.0185	1.7166
	20	1500	13.3826	10.2759	8.24e-19	0.97712	6.089801	3.409459	7.7928	7.2041
	30	2000	28.6293	16.7768	8.67e-19	5.49e-19	6.006929	5.15692	14.9055	15.0393

Table 3A.2 Comparison results of function GR in terms of average fitness value

Pop	Dim	Gen	PSO	QPSO	Q-QPSO1	Q-QPSO2	SMQPSO1	SMQPSO2	Mutation gbest (Liu et al, 2005)	Mutation gbest Feng and Xu, (2004)
20	10	1000	0.09217	0.08331	0.00466	0.062657	0.07226	0.057871	0.0627	0.0780
	20	1500	0.03002	0.02033	1.115e-06	0.005091	0.022172	0.014902	0.0209	0.0235
	30	2000	0.01811	0.01119	1.756e-08	0.015442	0.012498	0.012113	0.0110	0.0099
40	10	1000	0.08496	0.06912	0.00256	0.057393	0.040943	0.045446	0.0539	0.0641
	20	1500	0.02719	0.01666	2.949e-08	0.005827	0.014368	0.017868	0.0238	0.0191
	30	2000	0.01267	0.01161	1.084e-16	0.007874	0.012306	0.01002	0.0119	0.0098
80	10	1000	0.07484	0.03508	0.00108	0.031527	0.028259	0.039229	0.0419	0.0460
	20	1500	0.02854	0.0146	5.532e-11	0.006633	0.01835	0.013274	0.0136	0.0186
	30	2000	0.01258	0.01136	5.421e-20	0.006648	0.010335	0.012353	0.0120	0.0069

Table 3A.3 Comparison results of function RB in terms of average fitness value

Pop	Dim	Gen	PSO	QPSO	Q-QPSO1	Q-QPSO2	SMQPSO1	SMQPSO2	Mutation gbest (Liu et al, 2005)	Mutation gbest Feng and Xu, (2004)
20	10	1000	94.1276	59.4764	9.80213	5.544203	5.673753	5.507213	27.4620	21.2081
	20	1500	204.336	110.664	28.0156	15.5381	15.890886	15.721613	49.1176	61.9268
	30	2000	313.734	147.609	47.0388	25.68707	26.041815	25.956007	97.5952	86.1195
40	10	1000	71.0239	10.4238	3.66553	4.200496	4.26871	4.210739	7.8741	8.1828
	20	1500	179.291	46.5957	25.7328	14.15802	14.367628	14.156106	28.4435	40.0749
	30	2000	289.593	59.0291	36.4313	24.12632	24.617573	24.397981	62.3854	65.2891
80	10	1000	37.3747	8.63638	2.29337	2.893087	2.745147	2.520018	6.7098	7.3686
	20	1500	83.6931	35.8947	15.3316	12.03305	12.342766	12.42195	31.0929	30.1607
	30	2000	202.672	51.5479	28.4146	22.42601	22.852736	22.742935	43.7622	38.3036

Table 3A.4 Improvement (%) in terms of average fitness function value for RS function in comparison to QPSO

Pop	Dim	Gen	Q-QPSO1	Q-QPSO2	SMQPSO1	SMQPSO2
20	10	1000	100	100	72.03966	76.84333
	20	1500	100	100	43.18567	71.56467
	30	2000	100	100	63.74321	65.57939
40	10	1000	100	100	72.52941	67.66891
	20	1500	100	100	71.88781	65.75551
	30	2000	100	100	72.20549	60.28123
80	10	1000	100	59.14615	29.64161	95.49579
	20	1500	100	90.49115	40.73705	66.82082
	30	2000	100	100	64.19503	69.2616

Table 3A.5 Improvement (%) in terms of average fitness function value for function GR in comparison to QPSO

Pop	Dim	Gen	Q-QPSO1	Q-QPSO2	SMQPSO1	SMQPSO2
20	10	1000	94.40643	24.79054	13.26371	30.53535
	20	1500	99.99452	74.95819	-	26.69946
	30	2000	99.99984	-	-	-
40	10	1000	96.2963	16.96615	40.76534	34.25058
	20	1500	99.99982	65.02401	13.7575	-
	30	2000	100	32.17916	-	13.69509
80	10	1000	96.92132	10.12828	19.44413	-
	20	1500	100	54.56849	-	9.082192
	30	2000	100	41.47887	9.022887	-

Table 3A.6 Improvement (%) in terms of average fitness function value for function RB in comparison to QPSO

Pop	Dim	Gen	Q-QPSO1	Q-QPSO2	SMQPSO1	SMQPSO2
20	10	1000	83.5193	90.67831	90.4605	90.74051
	20	1500	74.68409	85.95921	85.64042	85.79338
	30	2000	68.13284	82.5979	82.35757	82.4157
40	10	1000	64.83499	59.70283	59.04843	59.60457
	20	1500	44.7743	69.61517	69.16533	69.61929
	30	2000	38.28247	59.12808	58.29587	58.66788
80	10	1000	73.44524	66.50116	68.21415	70.8209
	20	1500	57.28729	66.4768	65.61396	65.39336
	30	2000	44.87729	56.49481	55.66699	55.88

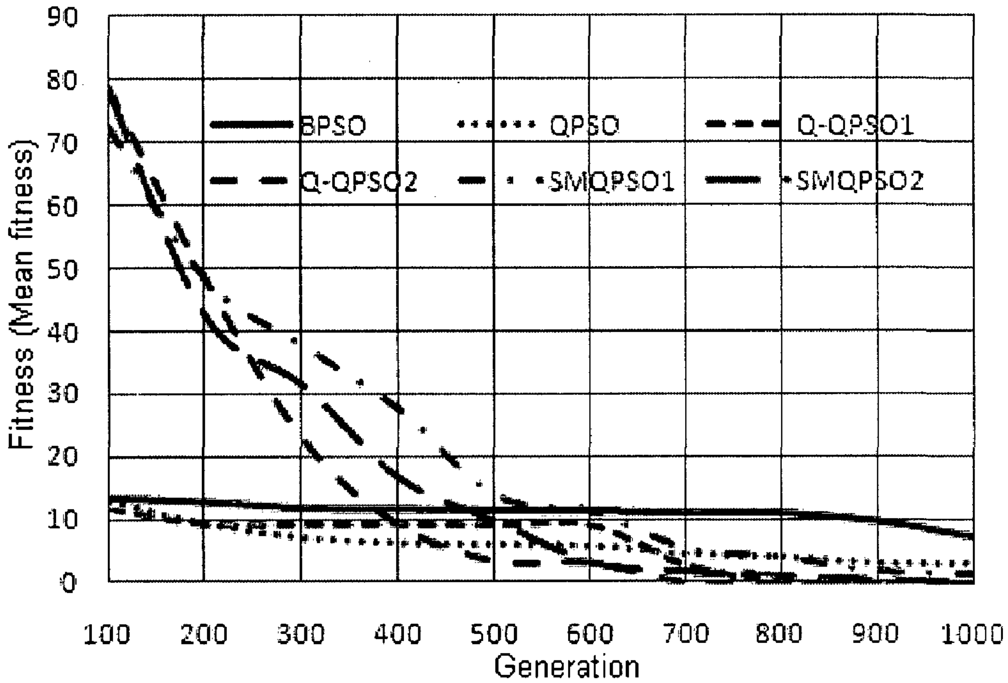

Fig 3A.1 (a) Function RS

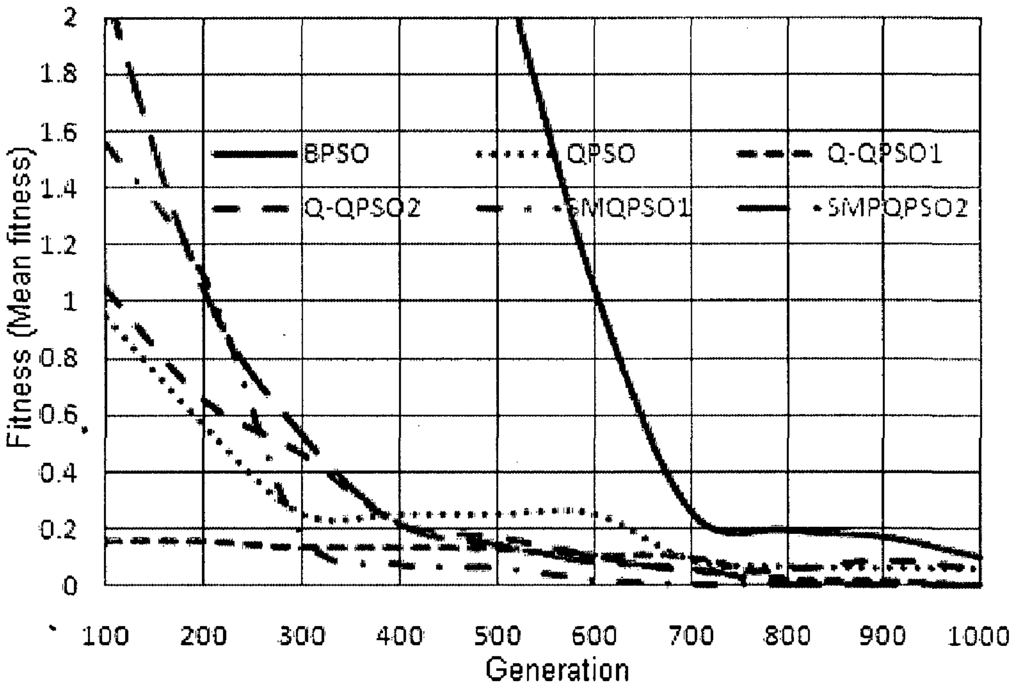

Fig 3A.1 (b) Function GR

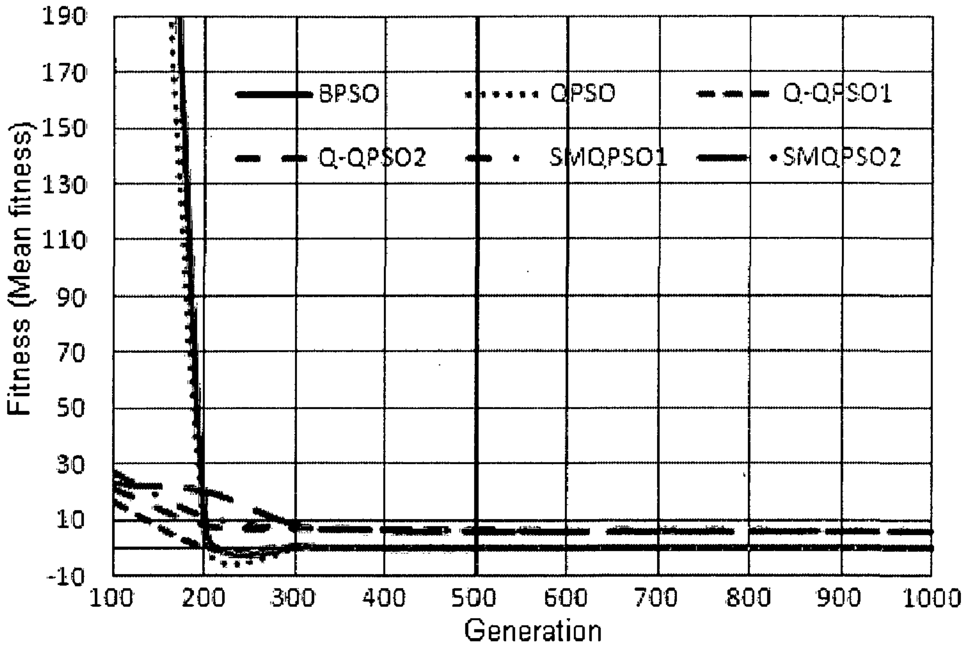

Fig 3A.1 (c) Function RB

Fig 3A.1 Performance curves of PSO, QPSO and the proposed QPSO variants

3A.7 Conclusion

This chapter presented some modified versions of PSO based on the quantum laws application to PSO. It is one of the latest developments in the field of PSO. Four modified versions of Quantum PSO algorithm are discussed in this chapter are:

- Quadratic Interpolation Quantum Particle Swarm Optimization (Q-QPSO1 and Q-QPSO2)
- Sobol Mutated Quantum Particle Swarm Optimization (SMQPSO1 and SMQPSO2)

The performance of presented algorithms was tested with some standard benchmark problems. The numerical results of proposed variants were compared with basic PSO, QPSO and two other variants of QPSO algorithm available in the literature. The results obtained by these algorithms on all benchmark problems were either superior or at par with the basic PSO and Quantum PSO algorithms. In an overall comparison, QPSO assisted with Quadratic Interpolation operator (Q-QPSOs) algorithms gave the best results.

Chapter 4

Improved Differential Evolution Algorithms

[This chapter describes the improved versions of the classical Differential Evolution algorithm. The improved algorithms are based on the mutant vector, the scale factor F and the crossover rate Cr of DE. This chapter proposes two new mutant vectors based on the Laplace probability distribution (LDE) and on the concept of Quadratic Interpolation (DE-QI). Five versions of LDE are proposed namely LDE1, LDE2, LDE3, LDE4 and LDE5. Also this chapter introduces an adaptive scaling factor and Crossover rate. The proposed algorithms are examined on several standard benchmark problems and the results are compared with the classical DE and some other variants of DE in the literature.]

4.1 Introduction

Differential Evolution is a stochastic, population based search strategy developed by Storn and Price (1995). It has been consistently ranked as one of the best search algorithm for solving global optimization problems in several case studies. DE has been designed as a stochastic parallel direct search method, which utilizes concepts borrowed from the broad class of EAs. The method typically requires few, easily chosen control parameters. Experimental results have shown that performance of DE is better than many other well known EAs (Storn and Price, 1997; Storn, 1999). While DE shares similarities with other EAs, it differs significantly in the sense that in DE, distance and direction information is used to guide the search process (Engelbrecht, 2005).

Despite several attractive features, it has been observed that DE sometimes does not perform as good as the expectations. Empirical analysis of DE has shown that it may stop proceeding towards a global optimum even though the population has not converged even to a local optimum (Lampinen and Zelinka, 2000). The situation when the algorithm does not show any improvement though it accepts new individuals in the population is known as stagnation. Besides this, DE also suffers from the problem of premature convergence. This situation arises when there is a loss of diversity in the population. It generally arises when the objective function is multimodal having several local and global optimums. Like other EA, the performance of DE deteriorates with the increase in dimensionality of the objective function. Several modifications have been made in the structure of DE to improve its performance. A brief survey on these modifications of DE is given in Chapter 1.

This chapter has seven sections including the introduction. Section 4.2 describes the proposed LDE algorithms; in section 4.3, the proposed DE-QI algorithm is described. In section 4.4, an adaptive scale factor and crossover rate in classical DE is introduced. Based on these adaptive parameters a new DE algorithm, ACDE algorithm, is proposed. Section 4.5 deals with the parameter settings and the benchmark problems used for this study; section 4.6 gives the result analysis. Finally the chapter concludes with section 4.7.

4.2 Differential Evolution with Laplace Mutation (LDE)

The LDE algorithm is a simple and modified version of basic DE algorithm. The structural difference between the proposed LDE algorithm and the basic DE is in the mutation phase. In this study five new mutation schemes for the basic DE algorithm are proposed. These schemes are based on the absolute difference between the vectors to generate a mutant vector. The amplification factor (or scaling factor), F, is replaced by a random variable following Laplace distribution. The five schemes are named as LDE1, LDE2, LDE3, LDE4 and LDE5. The first scheme, LDE1, uses only two vectors to generate a mutant vector. The second scheme, LDE2, is like target to best scheme of basic DE where the vector having the best fitness function value is used. In LDE3, which is the third scheme two vectors are generated and the one having the better fitness function value is accepted as a mutant vector. In the fourth scheme the original mutation scheme as given by Eqn. (1.6) and the LDE1 scheme are applied stochastically according to the user defined parameter P_{LDE}. Uniformly distributed random numbers between 0 and 1 are generated. If the random number is greater than the parameter P_{LDE}, then LDE1 is applied to generate the mutant vector otherwise the mutant vector is generated using Eqn. (1.6). In the fifth case the mutant vector is generated by adding a random vector to the amplified distance between the best vector and another randomly generated vector.

The mutation schemes for the DE versions are summarized as follows:

A. LDE1 Algorithm

$$v_{i,g+1} = x_{r_1,g} + L^* | x_{r_1,g} - x_{r_2,g} | \qquad (4.1)$$

B. LDE2 Algorithm

$$v_{i,g+1} = x_{best,g} + L^* | x_{r_1,g} - x_{r_2,g} |$$

C. LDE3 Algorithm

$$v'_{i,g+1} = x_{r_1,g} + L^* | x_{r_1,g} - x_{r_2,g} |$$

$$v''_{i,g+1} = x_{r_2,g} + L^* | x_{r_1,g} - x_{r_2,g} |$$

If $(f(v'_{i,g+1}) < f(v''_{i,g+1}))$ *then* $v_{i,g+1} = v'_{i,g+1}$

Else $v_{i,g+1} = v''_{i,g+1}$

D. LDE4 Algorithm

//Generate a uniformly distributed random number between 0 and 1 as U(0,1)

If (U(0,1) > P_{MDE}*) then*

$$v_{i,g+1} = x_{r_1,g} + L^* | x_{r_1,g} - x_{r_2,g} |$$

Else

$$v_{i,g+1} = x_{r_1,g} + F^* (x_{r_2,g} - x_{r_3,g})$$

E. LDE5 Algorithm

$$v_{i,g+1} = x_{r_1,g} + L^* | x_{best,g} - x_{r_2,g} |$$

For all the LDE algorithms, the notations have their usual meaning as described in chapter 1.

As mentioned earlier the amplifying factor in all the cases is a random variable L, following Laplace distribution. The Probability Density Function (pdf) of Laplace distribution is similar to that of normal distribution however, the normal distribution is expressed in terms of squared difference from the mean, Laplace density is expressed in terms of absolute difference from the mean. The density function of Laplace distribution is given as:

$$f(x/\theta) = \frac{1}{2\mu} \exp\left(\frac{-|x-\theta|}{\mu}\right), \quad -\infty \le x \le \infty \qquad (4.2)$$

Its distribution function is given by:

$$= = \frac{1}{2\mu}\begin{cases} \exp(-\frac{x-\theta}{\mu}) & if \quad x \le \theta \\ 1-\exp(-\frac{\theta - x}{\mu}) & if \quad x > 0 \end{cases} \tag{4.3}$$

$\mu > 0$ is the scale parameter.

From the proposed schemes, it can be seen that the newly generated mutant vector will lie in the vicinity of the base vector. However its nearness or distance from base vector will be controlled by **L**. For smaller values of μ, the mutant vector is likely to be produced near the initially chosen vector, whereas for larger values of μ, the mutant vector is more likely to be produced at a distance from the chosen vector. This behavior makes the algorithm self adaptive in nature, which in turn helps in preserving the diversity of the population by exploring the search space more effectively.

4.3 Differential Evolution Algorithm with Quadratic Interpolation Based Mutation (DE-QI)

The proposed DE-QI algorithm is a simple and modified version of basic DE algorithm. In the proposed DE version, a mutation probability P_{qi} is fixed and is having a certain threshold value provided by the user. In every iteration, if the uniformly distributed random number U (0, 1) is less than P_{qi} then the mutant vector is generated by using QI operator otherwise the mutant vector follows the basic DE method. The QI operator, based on quadratic interpolation (Mohan and Shanker, 1994), is a nonlinear operator which produces a new candidate solution lying at the point of minima of the quadratic curve passing through the three selected candidates. Detailed description of QI operator is given in Chapter 3.

A C++ style computational code for the mutation phase of DE-QI algorithm is given as:

//Generate U(0,1), a uniformly distributed random number between 0 and 1

If (U(0,1) <= P_{qi})

$$v_{i,g+1} = 0.5 * \frac{(x_{r_2,g}^2 - x_{r_3,g}^2) * f(x_{r_1,g}) + (x_{r_3,g}^2 - x_{r_1,g}^2) * f(x_{r_2,g}) + (x_{r_1,g}^2 - x_{r_2,g}^2) * f(x_{r_3,g})}{(x_{r_2,g} - x_{r_3,g}) * f(x_{r_1,g}) + (x_{r_3,g} - x_{r_1,g}) * f(x_{r_2,g}) + (x_{r_1,g} - x_{r_2,g}) * f(x_{r_3,g})}$$

Else

$$v_{i,g+1} = x_{r_1,g} + F * (x_{r_2,g} - x_{r_3,g})$$

Where $r_1, r_2, r_3 \in \{1,2,....,NP\}$ *are randomly chosen integers, different from each other and also different from the running index i. NP represents the population size.*

4.4 Differential Evolution Algorithm with Adaptive Control Parameters (ACDE)

Choosing suitable control parameter values is, frequently, a problem-dependent task. The trial-and-error method used for tuning the control parameters requires multiple optimization runs. The appropriate values of these control parameters lead to better (fitter) individuals which in turn are more likely to survive and produce fitter offspring. In this section, an adaptive Differential Evolution (ACDE) algorithm is presented. The main structural difference between the proposed ACDE algorithm and the basic DE is selecting the control parameters. The ACDE algorithm follows adaptive scale factor and cross over rate. The new scaling factor and crossover rate are calculated as:

$$F_{g+1} = \begin{cases} F_l + rand_1 \sqrt{Grand_1^2 + Grand_2^2} & if \quad P_F < rand_2 \\ F_0 & otherwise \end{cases} \tag{4.4}$$

$$Cr_{g+1} = \begin{cases} Cr_l * rand_3 & if \quad P_{Cr} < rand_4 \\ Cr_0 & otherwise \end{cases} \tag{4.5}$$

Eqns. (4.4) and (4.5) are used to produce factors F and Cr in a new generation. Here, $rand_j$, $j \in \{1,2,3,4\}$ are uniform random numbers in the interval *(0, 1]*. $Grand_1$ and $Grand_2$ are Gaussian distributed random numbers with mean 0 and standard deviation 1. P_F and P_{Cr} are the probabilities to adjust the factors F and Cr respectively. In this study, P_F and P_{Cr} are set as $P_F = P_{Cr} = 0.5$. Also the values of the constants F_l, F_0, Cr_l, Cr_0 are set as $F_l = Cr_l = 0.1$, $F_0 = Cr_0 = 0.5$. Also, in this experiment the following bounds for F are used.

$$\begin{aligned} &If \quad F_{g+1} > F_u \quad then \quad F_{g+1} = F_u * rand_5 \\ &If \quad F_{g+1} < F_l \quad then \quad F_{g+1} = F_l * rand_6 \end{aligned} \tag{4.6}$$

Where $rand_j$, $j \in \{5,6\}$ are uniformly distributed random numbers in the interval *(0, 1]*. The value of F_u is set as 0.5. Thus, the new F takes values in the interval *(0, 0.5]* and the new Cr takes values in the interval $(0,0.1] \cup \{0.5\}$. F_{g+1} and Cr_{g+1} are obtained in every iteration. So, they influence the mutation, crossover and selection operations of every new particle. The basic DE algorithm has three control parameters that need to be adjusted by the user. It seems that ACDE has even more parameters, but note that here the values of F_l, F_u, F_0, Cr_l, Cr_0, P_F *and* P_{Cr} *are* fixed for all the test problems in our ACDE algorithm. The user does not need to adjust those additional parameters.

4.5 Parameter Settings and Benchmark Problems

In order to make a fair comparison of DE and all the proposed algorithms, the same seed for random number generation is fixed so that the initial population is same for all the algorithms. The population size is taken as 50 for all the test problems for all algorithms. However, this is a heuristic choice and may be increased, depending on the complexity of the problem. The other parameters, crossover rate and scaling factor F, for classical DE and DE-QI, are fixed at 0.2 and 0.5 respectively. For LDE schemes also the crossover rate is taken as 0.2. The value of additional parameter P_{LDE} in LDE4 scheme is taken as 0.2. As mentioned in section 4.2, the scaling factor for all LDE schemes is a random variable which follows Laplace distribution. For DE-QI algorithm, the mutation probability P_{qi} is taken as 0.1. For each algorithm, the maximum number of iterations allowed is set to 5000 and the error goal is set as 1*e-04. A total of 30 runs for each experimental setting were conducted and the average fitness of the best solutions throughout the run was recorded.

In order to check the compatibility of the proposed LDE, DE-QI and ACDE algorithms a suite of ten benchmark problems are considered; they are: RS, DeJ, GR, RB, DeJ-N, SWF, ACK, Mic, MH and SB1. The mathematical models of the test problems with the true optimum value are given in Appendix I. The performance curves of proposed DE algorithms with classical DE for all benchmark problems are shown in Fig 4.1(a) – Fig 4.1(j).

4.6 Results and Discussion

In order to compare the proposed LDE, DE-QI and ACDE algorithms with basic DE and other modified versions of DE various performance metrics like average fitness function value and standard deviation (STD) are considered to check the efficiency and reliability of the algorithm. To compare the convergence speed of algorithms the average number of function evaluations (NFE) is recorded. Smaller number of function evaluations indicates faster convergence. The speed of the algorithm is also measured by recording the total CPU time and the average CPU time taken by the algorithm to meet the stopping criteria. Besides this success rate (SR) and average success rate (ASR) are also measured. A run is considered a success if the value obtained at the end of the algorithm is within one percent of the desired accuracy. The definitions of performance measures used in this study are given by:

Average NFE = $\dfrac{\sum_{i=1}^{n} NFE(f_i)}{n}$

Improvement (%) in terms of NFE =

$$\frac{Total\ NFE\ (basic\ DE\ algorithm) - Total\ NFE\ (Algorithm\ to\ be\ compared)}{Total\ NFE\ (basic\ DE\ algorithm)} * 100$$

$$Acceleration\ rate\ (AR) = \frac{Total\ NFE\ for\ basic\ DE}{Total\ NFE\ for\ algorithm\ to\ be\ compared}$$

Average CPU time = $\dfrac{\sum_{i=1}^{n} Time(f_i)}{n}$

Improvement (%) in terms of CPU Time =

$$\frac{Total\ time\ (basic\ DE\ algorithm) - Total\ time\ (Algorithm\ to\ be\ compared)}{Total\ NFE\ (basic\ DE\ algorithm)} * 100$$

Average SR = $\dfrac{\sum_{i=1}^{n} SR(f_i)}{n}$

Performance comparisons of all proposed De algorithms with basic DE are given in Tables 4.1 – 4.4. Performance analyses of LDE, DE-QI and ACDE algorithms with ODE (Rahnamayan et al, 2008) and ODE (Rahnamayan et al, 2008) are given in Table 4.5 and Table 4.6 respectively.

4.6.1 Performance Analysis I: Comparison of LDE Schemes with Basic DE

From Table 4.1 which gives the average fitness function value, it can be seen that all the LDE schemes performed better than the basic DE for all the test problems. Particularly in case of function RS (Rastringin function) and function RB (Rosenbrock function), there is a significant improvement in the performance of DE using the proposed LDE3, LDE4 and LDE5 schemes. In case of function RS, there is an improvement of 97% in the function value while using LDE5 scheme. Similarly for function RB, the use of LDE3 scheme improves the function value up to 99%. For other functions also, the proposed schemes outperform the basic DE algorithm. The superior performance of proposed schemes is more evident from Tables 4.2 to 4.4 which give the convergence speed, average CPU time and success rate of the proposed DE schemes and the basic DE. From these tables, it can be seen that there is more than 50% improvement in the convergence speed with the implementation of LDE1, LDE4 and LDE5 schemes. LDE3 scheme improves the performance by 44%. Under the present parameter settings, LDE2 scheme did not show much improvement as the improvement in convergence rate is only 0.33%. When Acceleration Rate (AR) is greater than 1, then it means that the proposed algorithm is better than the basic algorithm. For all the proposed MDE schemes, the AR is greater than 1. When one observe the CPU time given in Table 4.3, it can be seen that the average time taken by all the proposed LDE schemes to solve the given test problems is less than the time taken by DE algorithm. With LDE1 scheme, the improvement is 64% and with LDE3, LDE4 and LDE5 schemes the improvement in time is more than 50%. However with LDE2 scheme, this improvement is only 5%. If we talk about the success rate, which is given in Table 4.4, it can be seen that on an average the proposed LDE1, LDE3 LDE4 and LDE5 gives more than 80% success while LDE2 gives more than 65 % success for all the test problems considered in this study.

4.6.2 Performance Analysis II: Comparison of DE-QI Algorithm with Basic DE

The DE-QI algorithm is compared with the basic DE in terms of Average fitness function value, number of function evaluation, convergence time and success rate. The results are shown for dimension 30 and are given in Table 4.1 to 4.4. From the numerical results it is evident that DE-QI gave a better performance in terms of all considered performance measures for all the test problems. Particularly in case of function RS, there is a significant improvement in the performance of DE using the proposed QI based mutation operator. In this function, there is an improvement of 99.99% in the function value while using DE-QI algorithm. Similarly, for function DeJ-N, which is a noisy function, the improvement of DE-QI algorithm in terms of average fitness function value is 83.87%. Likewise for all other functions also, there is a noticeable improvement in classical DE while using the proposed QI mutation operator. The better performance of DE-QI is more visible from Table 4.2 and 4.3, where NFE and CPU time are reported. From these tables it is clear that the proposed DE-QI algorithm much faster than the classical DE. The total number of function evaluations for solving 10 test problems is 1030110 for DE-QI algorithm whereas for DE the total NFE is 1479540. Therefore, there is an improvement of 30.4% in NFE for DE-QI algorithm in comparison with DE. Similarly, the total time taken by DE-QI is 104.1 seconds whereas the total time taken by DE is 370.6 seconds. Thus from the numerical results, it can be said that the proposed QI based mutation operator improved the performance of classical DE with a noticeable percentage.

4.6.3 Performance Analysis III: Comparison of ACDE Algorithm with Basic DE

Performance comparisons of ACDE algorithm is performed with classical DE in terms of the performance measures average fitness function value, NFE, CPU time and success rate. From the numerical results given in Table 4.1 – 4.4, it can be seen that ACDE algorithm gave better performance than classical DE in all the test cases except for the function RB. From Table 4.1, it can be seen that for the function RS, the difference in the average fitness function values for DE and ACDE is quite significant. The true global minimum for the function RS is located at

0.0. None of the algorithms were able to reach this value. However ACDE gave the value near about the true optimum and it is much better value in comparison to DE. In this case the improvement of ACDE in terms of fitness function value in comparison with DE is 99.55%. Similarly for function DeJ-N, the use of adaptive control parameters improves the function value up to 80.5%. For other functions also, the proposed ACDE algorithm outperform the basic DE algorithm. In terms of NFE the improvement of ACDE algorithm is around 53% in comparison with the DE algorithm. From Table 4.3, it is clear that the proposed ACDE converges much faster than the classical DE; there is an improvement of 83% in CPU time. The total time taken by classical DE is 370.6 seconds whereas the total time taken by ACDE is 61.35 seconds only. If we talk about the success rate, which is given in Table 4.4, it can be seen that on an average the proposed ACDE gives more than 88% success for all the test problems considered in this study.

4.6.4 Performance Analysis IV: Comparison of LDE, DE-QI and ACDE Algorithms with each other

If one compares the performance of proposed LDE, DE-QI and ACDE algorithms with each other then from the numerical results it can be seen that LDE3 and DE-QI algorithms perform better than other algorithms in 2 test cases for each out of 10 test cases in terms of fitness function values. LDE4, LDE5 and DE-QI are performed the same in one test case. If one compares the NFE then LDE1 is the clear winner in comparison with other proposed algorithms; it gave an improvement of 57% in total NFE. But in comparison of CPU time taken by the algorithms, ACDE is the winner. The total time taken by ACDE algorithm is an average of 6.135 seconds. In terms of success rate LDE3 algorithm perform better than other compared algorithms.

4.6.5 Performance Analysis V: Comparison of LDE, DE-QI and ACDE Algorithms with other Variants of DE

Besides using the basic DE for comparison of proposed DE algorithms two recent versions of DE namely Opposition based DE i.e. ODE (Rahnamayan et al, 2008) and Differential Evolution with Preferential crossover or DEPC (Ali, 2007) are also used. While comparing the performance of proposed DE algorithms with DEPC and ODE, the parameter settings of proposed DE algorithms were changed as same as that of the algorithms to which they were compared. This was done to give an equal opportunity to all the algorithms. In these comparisons the average CPU time was not recorded because it was not mentioned in the literature. The remaining performance metrics are kept as same as mentioned in Section 4.6.

Performance analyses of LDE, DE-QI and ACDE algorithms with ODE (Rahnamayan et al, 2008) are given in Table 4.5. From this Table it can be seen that under the changed parameter settings, except for LDE2 algorithm for function RS (Rastringin function) where it failed to give any result, the performance of the remaining proposed DE algorithms is either better or at par with ODE in terms of NFE. In terms of reliability, the SR for ODE is 85% while LDE3, LDE5, DE-QI and ACDE algorithms gave an average of 100% success for all the test problems that were considered. The SR of LDE1 and LDE4 is more than 95%. However, the SR of LDE2 algorithm is only 75%.

In Table 4.6, the performance comparison of proposed DE algorithms is given with DEPC algorithm. Here also the parameter settings of all the proposed algorithms were changed according to DEPC (Ali, 2007). Here, an interesting thing was observed that LDE2 algorithm which was giving the worst performance in previous cases started performing very well under the changed parameter settings. It gave the best results in terms of NFE for function RS for which it failed in previous cases. For function RB, the DE-QI algorithm is failed to give any result. The other algorithms performed more or less in a stable manner giving good results (an average success rate of 90%) which are once again either better or at par with the DEPC algorithm. The success rate for DEPC algorithm was however 98% but this is quite expected because the parameter settings are in favor of DEPC.

Table 4.1 Average fitness function value and standard deviation obtained by basic DE and proposed DE algorithms

Fun.	DE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI	ACDE
RS	29.9076 (1.34989)	5.87024 (2.10827)	27.7223 (7.18165)	4.97478 (1.33962)	2.78592 (0.974859)	0.895465 (0.696468)	4.69e-05 (7.63e-06)	0.132725 (0.338219)
DeJ	6.87e-05 (9.13e-06)	3.99e-06 (1.18e-06)	9.45e-06 (3.70e-06)	5.41e-06 1.54e-06)	4.14e-05 (1.40e-05)	5.09e-06 (1.18e-06)	5.12e-05 (8.93e-06)	5.82e-05 (1.10e-05)
GR	7.70e-05 (8.63e-06)	4.83e-06 (2.22e-06)	2.06491 (0.790521)	4.08e-11 3.53e-09)	4.82e-05 (1.17e-05)	0.017624 (0.052857)	5.35e-05 (4.48e-06)	6.26e-05 (1.76e-05)
RB	26.3194 (1.4247)	8.98702 98.01641)	17.2028 (4.56154)	1.35307 (3.10551)	0.334056 (8.00e-05)	4.79998 (3.29972)	24.3933 (1.0704)	34.3654 (18.1468)
DeJ-N	0.017781 (0.004219)	0.003947 (0.001102)	0.076151 (0.055258)	0.003925 (0.000767)	0.003182 (0.000682)	0.003726 (0.000837)	0.002868 (0.0009)	0.003460 (0.000695)
SWF	-12474.7 (4.73753)	-12534 (3.58375)	-11618.2 (3.5559)	-12545.8 (2.10634)	-12569.5 (0.00000)	-12569.5 (1.31e-06)	-12569.5 (1.17e-05)	-12530 (70.623)
ACK	0.0001830 (2.077e-05)	6.84e-05 (0.0001639)	1.13e-06 (0.739362)	1.25e-05 (0.13524)	0.0001516 (2.38e-05)	1.55e-05 (2.09e-06)	0.000128 (2.25e-05)	0.000172 (3.23e-05)
Mic	-27.095 (0.32179)	-28.6223 (0.215474)	-27.2475 (1.29499)	-28.8925 (0.201602)	-29.1373 (0.181723)	-29.5502 (0.028403)	-29.18 (0.260897)	-29.6199 (0.015583)
MH	-3.28972 (0.388473)	-3.49703 (0.470752)	-3.31278 (0.012865)	-3.78396 (0.00000)	-3.39549 (0.475781)	-3.29837 (0.485592)	-3.49261 (0.445052)	-3.39549 (0.475781)
SB1	-186.731 (1.11e-07)	-186.731 (1.77e-08)	-186.731 (4.01e-09)	-186.731 (8.79e-09)	-186.731 (2.38e-07)	-186.731 (1.94e-08)	-186.731 (7.63e-10)	-186.731 (8.24e-08)

Table 4.2 Comparison Results of DE and Proposed DE algorithms: NFE (Number of Function Evaluations)

Function	DE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI	ACDE
RS	250050	34585	250050	86375	37440	37935	135295	55676
DeJ	57000	18935	19455	45020	16540	18570	16670	17590
GR	175570	27005	26715	78305	24715	29165	25775	26120
RB	250050	178750	197189	192980	216285	242105	250050	125320
DeJ-N	250050	250050	250050	750050	250050	250050	250050	250050
SWF	122525	28290	31735	75425	28025	31460	61335	44786
ACK	100655	32170	19030	82415	28290	32450	29275	30526
Mic	250050	53580	25475	141005	53655	74385	250050	125697
MH	5470	4755	5155	13490	4155	4070	4825	3256
SB1	18120	3950	1610	9545	4675	8315	6785	9553
$\sum$	1479540	632070	826464	1474610	663830	728505	1030110	688574
Average NFE	147954	63207	82646.4	147461	66383	72850.5	103011	68857.4
Improvement (%) of NFE		57.27929	44.14048	0.333212	55.13268	50.76139	30.37633	53.46026
AR		2.340785	1.790205	1.003343	2.228794	2.030926	1.436293	2.148702

Table 4.3 Comparison Results of DE and Proposed DE algorithms: CPU Time (in seconds)

Function	DE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI	ACDE
RS	42.8	5.4	37.3	11.8	5.7	5.9	9.2	3.8
DeJ	8.6	2.8	2.9	5.7	2.5	2.8	1.1	1.2
GR	28.9	4.1	4.1	11.3	3.6	4.3	1.8	1.8
RB	106.9	64.9	66.7	146.4	88.2	87.1	43.3	21.2
DeJ-N	37.9	37.0	37.1	97.4	40.2	37.8	16.4	16.1
SWF	2.3	0.5	0.6	1.2	0.7	0.6	0.6	0.46
ACK	13.9	6.1	3.9	15.7	4.6	5.2	1.9	2.2
Mic	129.1	11.9	9.6	61.4	13.0	18.1	29.5	14.4
MH	0.1	0.1	0.4	0.3	0.1	0.1	0.2	0.13
SB1	0.1	0.01	0.01	0.1	0.01	0.1	0.1	0.06
$\sum$	370.6	132.81	162.61	351.3	158.61	162	104.1	61.35
Average Time	37.06	13.281	16.261	35.13	15.861	16.2	10.41	6.135
Improvement (%) of Time		64.16352	56.1225	5.207771	57.20183	56.2871	71.91042	83.44576

Table 4.4 Comparison Results of DE and Proposed DE algorithms: Success Rate (SR) (%)

Function	DE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI	ACDE
RS	-	100	-	100	100	100	100	100
DeJ	100	100	100	100	100	100	100	100
GR	100	100	100	100	100	100	100	100
RB	-	70	30	90	70	10	-	80
DeJ-N	-	-	-	-	-	-	-	-
SWF	100	100	70	100	100	100	100	100
ACK	100	100	100	100	100	100	100	100
Mic	-	100	100	100	100	100	-	100
MH	100	100	100	100	100	100	100	100
SB1	70	90	70	100	100	100	100	100
Σ	570	860	670	890	870	810	600	880
Average SR	57	86	67	89	87	81	60	88

Table 4.5 Comparison Results of ODE and Proposed Algorithms: NFE and Success Rate

NFE									
Function	Dim	ODE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI	ACDE
RS	10	70389	52866	-	70090	30050	29793	39990	43750
DeJ	30	47716	46893	51500	115630	43440	50133	42940	27175
GR	30	69342	61146	66424	150400	65860	65933	68800	59450
ACK	30	98296	88453	98460	219700	96530	95026	101200	64900
Mic	10	213330	174446	183256	9790	146487	15100	21150	13275
Success Rate (%)									
Function	Dim	ODE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI	ACDE
RS	10	76	94	-	100	100	100	100	100
DeJ	30	100	100	100	100	100	100	100	100
GR	30	96	100	100	100	100	100	100	100
ACK	30	100	100	100	100	100	100	100	100
Mic	10	56	86	76	100	88	100	100	100

Table 4.6 Comparison Results of DEPC and Proposed Algorithms: NFE and Success Rate

NFE									
Function	Dim	DEPC	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI	ACDE
RS	10	26927	25800	11180	60340	24510	24140	28640	25600
GR	10	47963	36866	18400	92050	30490	36770	43460	25800
RB	10	512165	209787	930950	1191700	484033	815130	-	239200
SWF	10	24046	19120	10940	51550	21800	20100	28210	26200
ACK	10	29825	24020	12690	68980	25320	24930	27000	24250
SB1	2	1955	1333	630	3566	1644	1918	5080	1760
Success Rate (%)									
Function	Dim	DEPC	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI	ACDE
RS	10	100	100	100	100	100	100	100	100
GR	10	100	100	100	100	100	100	100	100
RB	10	100	100	40	70	80	40	-	50
SWF	10	100	100	100	100	100	100	100	100
ACK	10	100	100	100	100	100	100	100	100
SB1	2	89	100	100	100	100	100	100	100

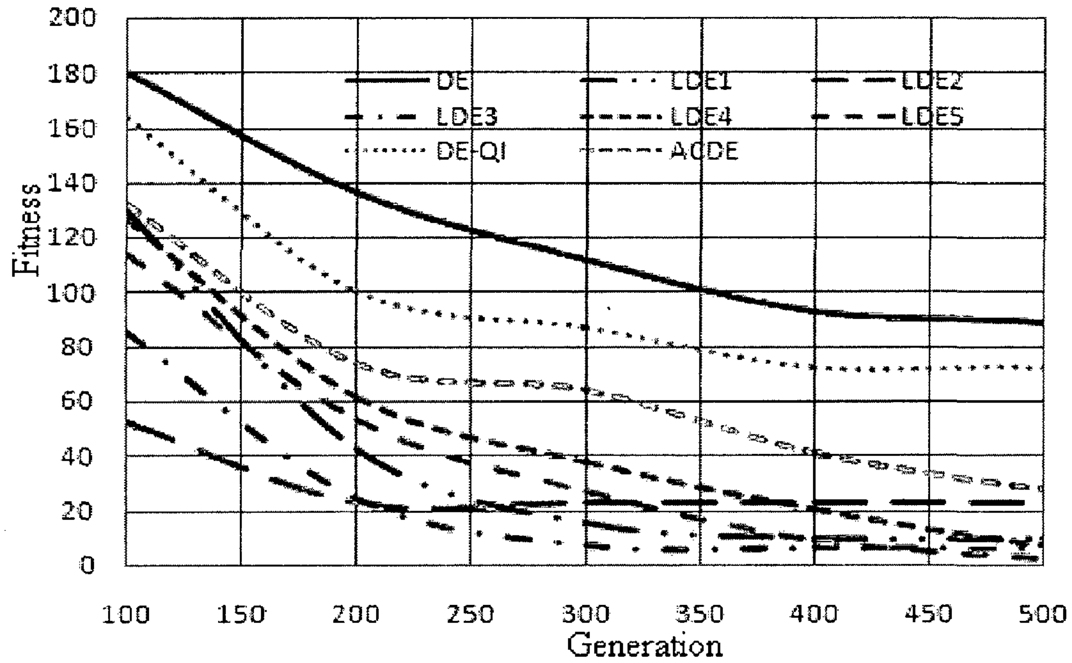

Figure 4.1 (a) Function RS

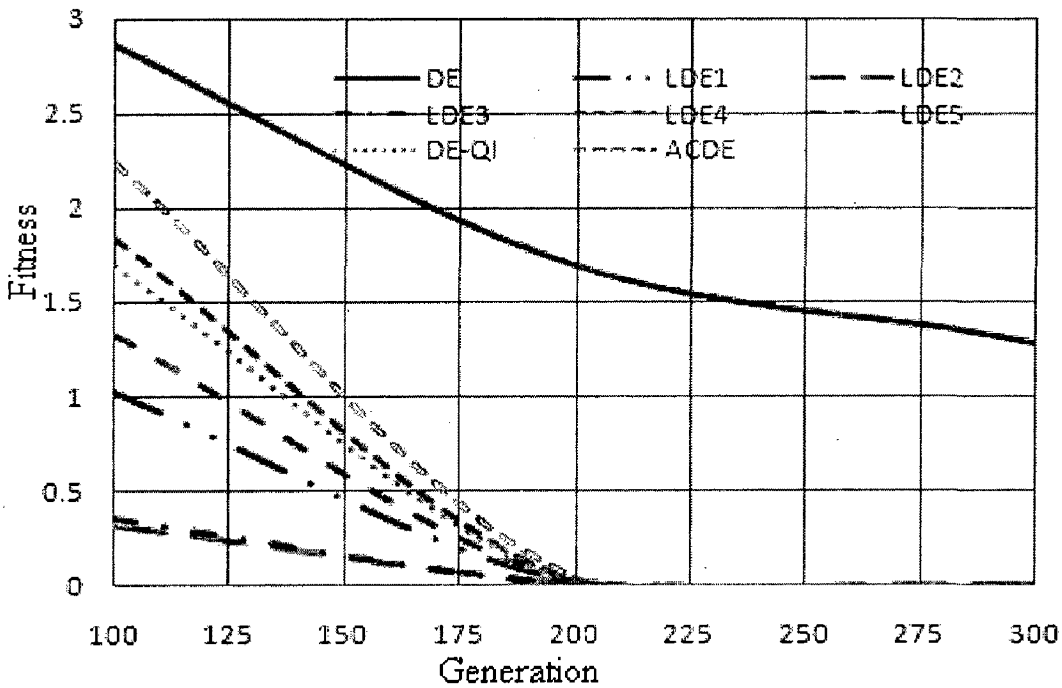

Figure 4.1 (b) Function DeJ

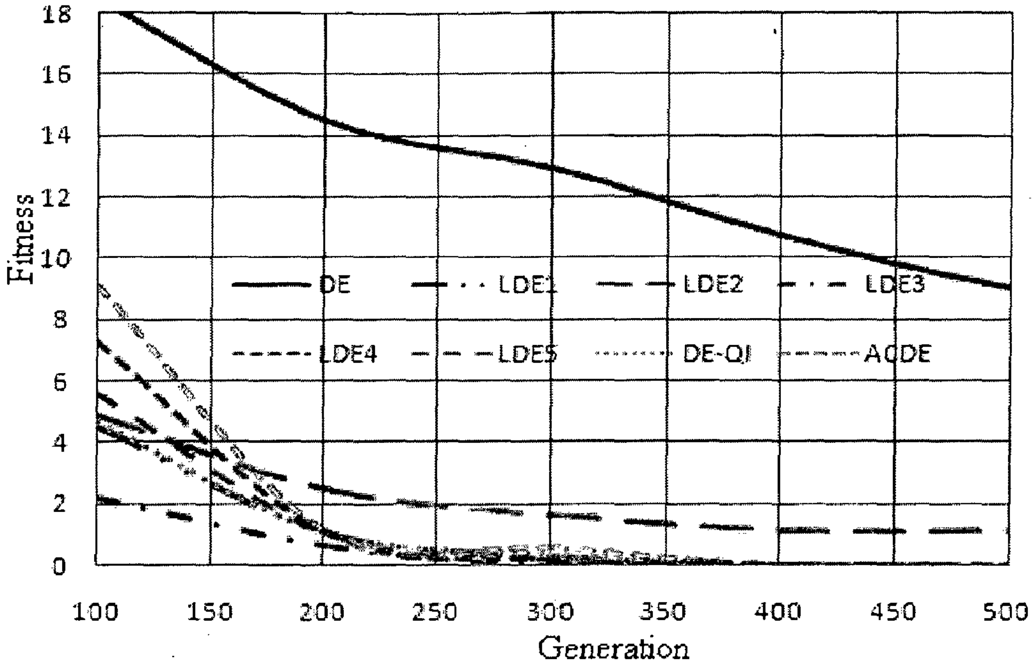

Figure 4.1 (c) Function GR

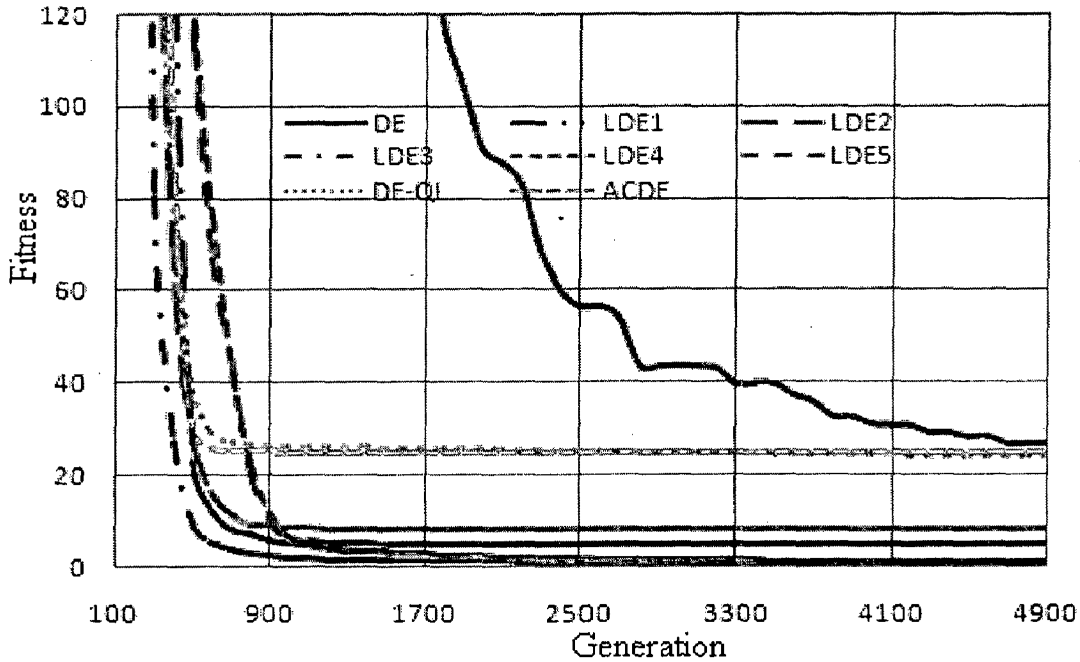

Figure 4.1 (d) Function RB

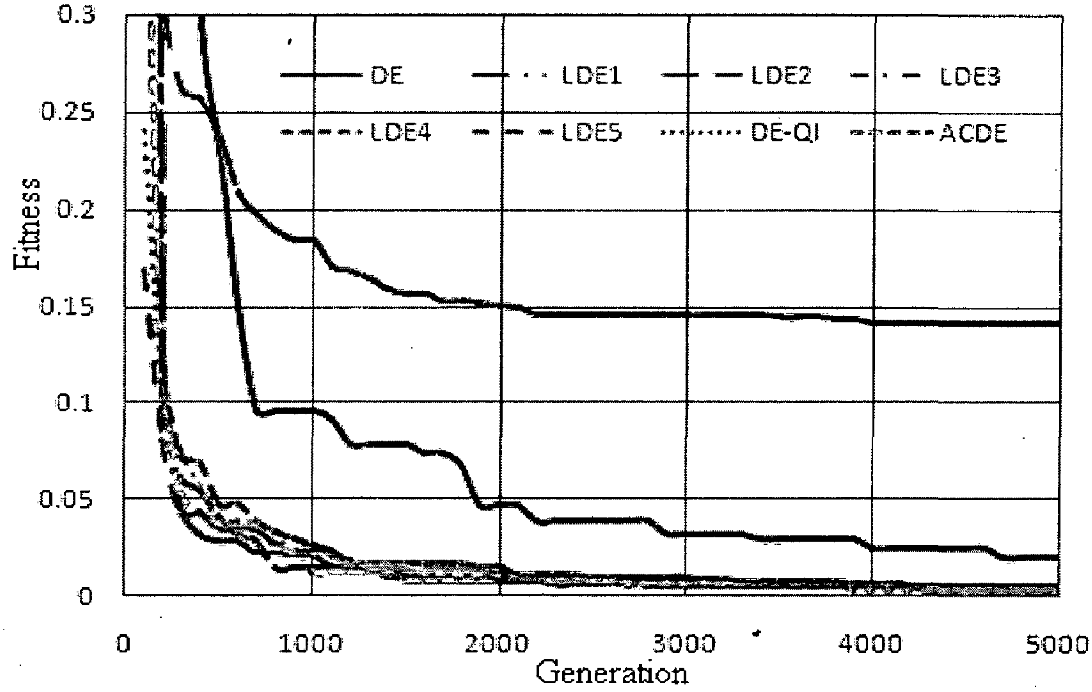

Figure 4.1 (e) Function DeJ-N

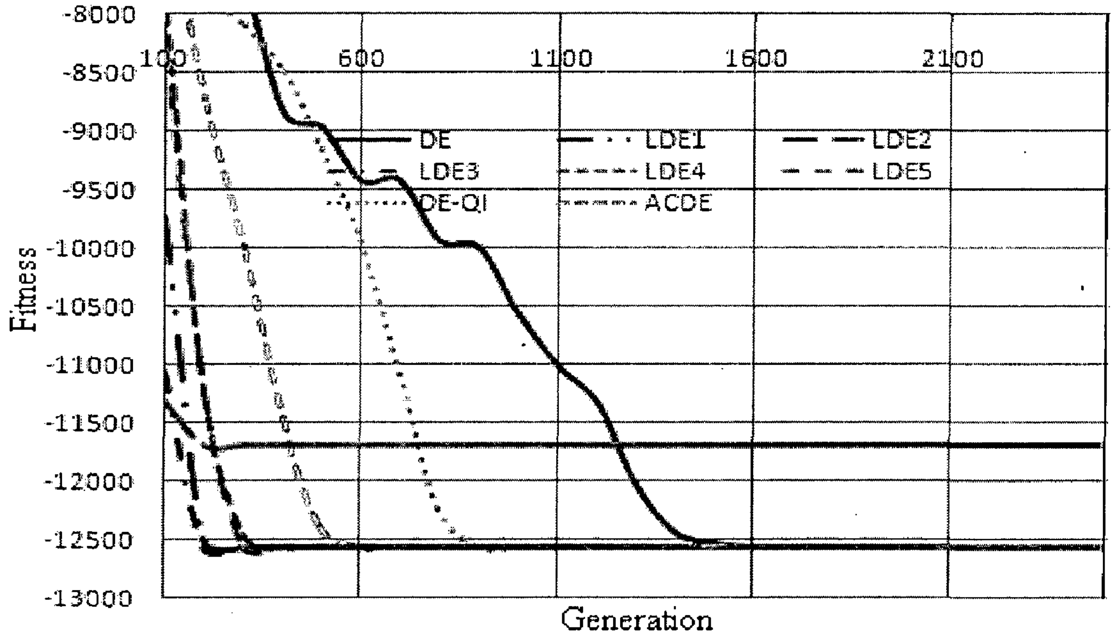

Figure 4.1 (f) Function SWF

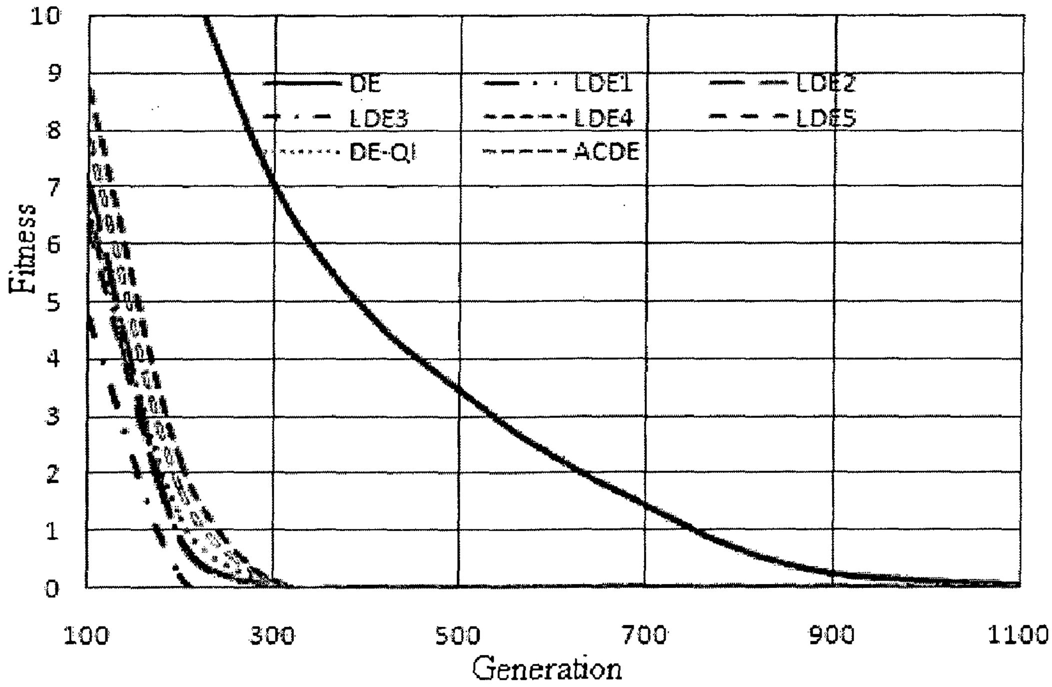

Figure 4.1 (g) Function ACK

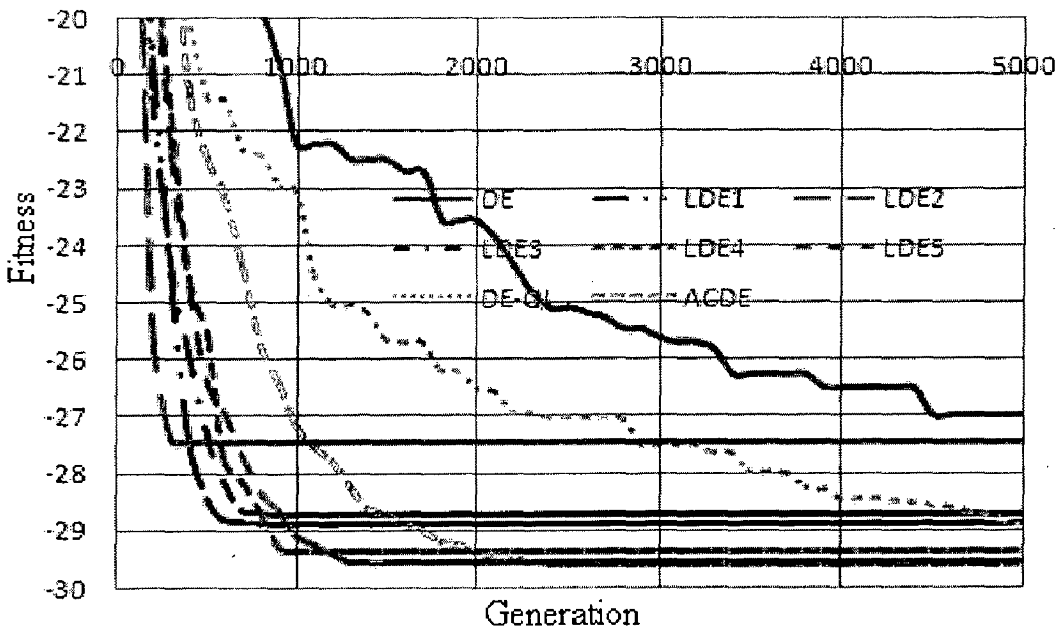

Figure 4.1 (h) Function Mic

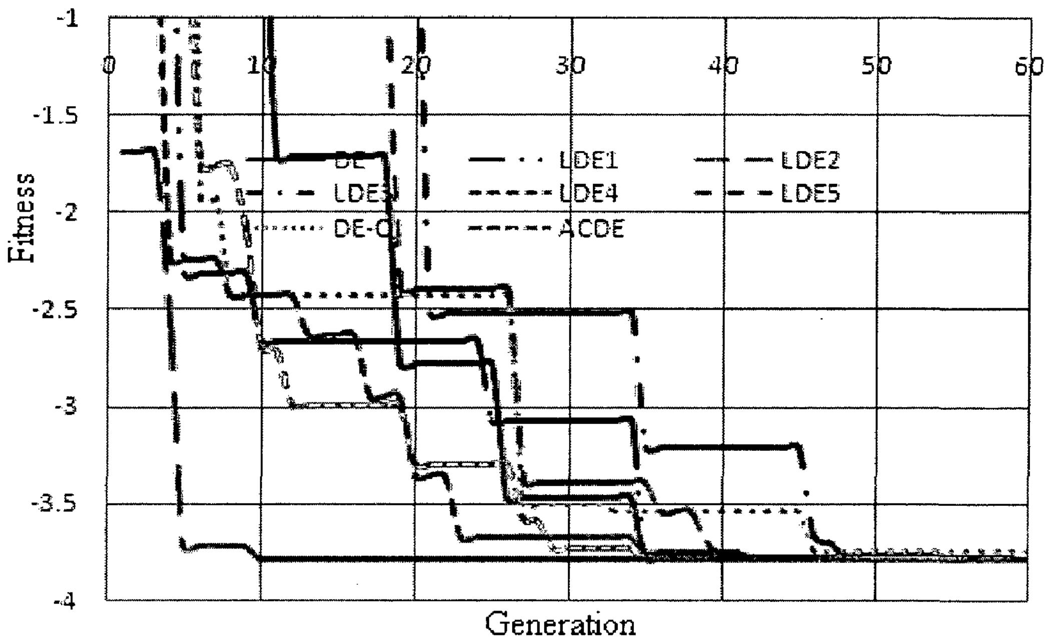

Figure 4.1 (i) Function MH

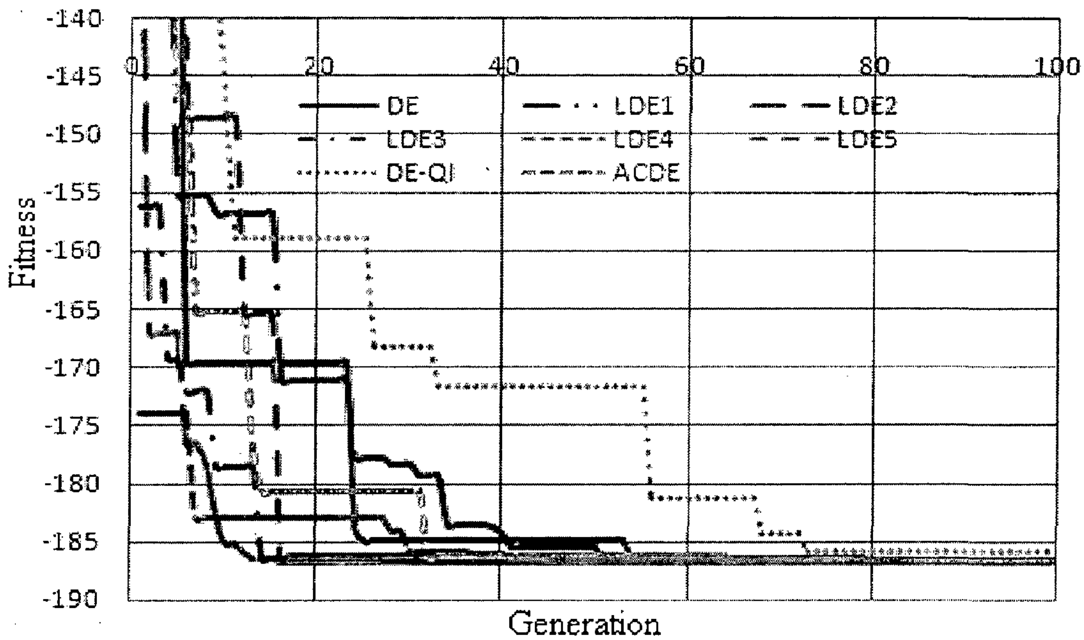

Figure 4.1 (j) Function SB1

Figure 4.1 Performance curves of DE, LDE, DE-QI and ACDE algorithms

4.7 Conclusion

This chapter presented three modified improved versions of DE algorithm. The improved algorithms are based on variation in the mutant vector, the scaling factor F and the crossover rate Cr of DE. The modified algorithms are:

- Differential Evolution with Laplace mutation operator namely LDE1, LDE2, LDE3, LDE4 and LDE5
- Differential Evolution with QI based mutation operator (DE-QI)
- Adaptive Control parameter Differential Evolution (ACDE)

The performance of the proposed algorithms was validated on a set of 10 test problems and the numerical results were compared with basic DE and two other versions of DE. The numerical results show that the proposed algorithms help in improving the convergence rate up to 50% in comparison to the basic DE and at the same time maintain a good SR as well. Also it was observed that out of the seven proposed algorithms LDE2 was most sensitive to the parameter settings as its performance changed drastically when the parameter settings were changed. However the remaining six algorithms performed more or less in a stable manner giving good performance even when the parameter settings were changed according to the algorithms to which they were being compared (i.e. DEPC and ODE).

Chapter 5

Hybrid Algorithms

[This chapter presents hybrid versions of DE and PSO algorithms. The hybridization of these algorithms is done with each other and also with evolutionary programming. The proposed algorithms are compared with each other and also with other variants available in the literature.]

5.1 Introduction

DE and PSO have undergone a plethora of changes since the last few years to improve their performance. One of the class of modified algorithms consists of the hybridization of algorithms, where the two algorithms are combined together to form a new algorithm. Some hybrid versions of DE and PSO include Hendtlass approach (Hendtlass, 2001), where the population evolved by DE is optimized by using PSO, Kannan approach (Kannan et al, 2004); in which DE is applied to each particle for a finite number of iterations to determine the best particle which is then included into the population. Methods of Zhang and Xie (2003) and Talbi and Batauche (2004) apply DE to the best particle obtained by PSO. In the hybrid version of Hao et al (2007), the candidate solution is generated either by DE or by PSO according to some fixed probability distribution. Recently a hybrid version of PSO and DE is proposed by Omran et al (2007) which is named as Barebones DE. In this chapter three hybrid versions of PSO and DE algorithms are proposed. The hybridization methods presented in this chapter consists of merging PSO and DE algorithms with each other and the second type of hybridization consists of combining PSO and DE algorithms with Evolutionary Programming (EP). The algorithms developed in this chapter are: DE-PSO, which as the name suggests is a hybrid version of DE and PSO algorithm; Adaptive Mutation Particle Swarm Optimization or AMPSO which is a hybrid version of PSO and Evolutionary Programming and Modified Differential Evolution or MDE which is a hybridization of DE and Evolutionary Programming.

The remaining of the chapter is organized as follows: in Sections 5.2, 5.3 and 5.4 describe the proposed DE-PSO, AMPSO and MDE algorithms respectively. Section 5.5 deals with the parameter settings and the benchmark problems used for this study; section 5.6 gives the result analysis. Finally the chapter concludes with section 5.7.

5.2 DE-PSO: A Hybrid Algorithm of Differential Evolution and Particle Swarm Optimization

The DE-PSO algorithm is a hybrid version of DE and PSO. It starts like the usual DE algorithm up to the point where the trial vector is generated. If the trial vector is better than the corresponding target vector, then it is included in the population otherwise the algorithm enters the PSO phase and generates a new candidate solution using Particle Swarm's velocity and position update equations with the hope of finding a better solution. The method is repeated iteratively till the optimum value is reached. The inclusion of PSO phase creates a perturbation in the population, which in turn helps in maintaining diversity of the population and producing a good optimal solution.

The pseudo code of the Hybrid DE-PSO algorithm is:

Initialize particle's position and velocity vectors

Do

For i = 1 to N (Population size) do

Select $r_1, r_2, r_3 \in N$ *randomly*

// r_1, r_2, r_3 *are selected such that* $r_1 \neq r_2 \neq r_3$ *//*

For j = 1 to D (dimension) do

Select $j_{rand} \in D$

If (rand () < CR *or* $j = j_{rand}$*)*

// rand () denotes a uniformly distributed random number between 0 and 1 //

$$U_{ij,g+1} = x_{r_1,g} + F * (x_{r_2,g} - x_{r_3,g})$$

End if

End for

If $(f(U_{i,g+1}) < f(X_{i,g}))$ *then*

$$X_{i,g+1} = U_{i,g+1}$$

Else

// PSO phase activated

Find a new particle using Particle Swarm's velocity and position update equations.

Let this particle be $TX = (tx_1, tx_2, ..., tx_D)$ ***//***

For j = 1 to D (dimension) do

$$v_{ij,g+1} = w * v_{ij,g} + c_1 r_1 (P_{ij,g} - x_{ij,g}) + c_2 r_2 (P_{gbestj,g} - x_{ij,g})$$

$$tx_{ij} = x_{ij,g} + v_{ij,g+1}$$

End for

If $(f(TX_i) < f(X_{i,g}))$ ***then***

$$X_{i,g+1} = TX_i$$

Else

$$X_{i,g+1} = X_{i,g}$$

End if

End if

End for.

Until stopping criteria is reached

Here w, c_1, c_2, r_1, r_2 are Particle Swarm's control parameters and P_i and P_{gbest} are particle's personal best and global best positions.

5.3 AMPSO: A Hybrid Algorithm of Particle Swarm Optimization and Evolutionary Programming

In the second type of hybridization, the PSO algorithm is combined with EP. The proposed AMPSO algorithm is a simple, modified version of PSO including EP based adaptive mutation operator using Beta distribution. Two versions of AMPSO namely AMPSO1 and

AMPSO2 are proposed. AMPSO1 and AMPSO2 differ from each other in the sense that in AMPSO1, the personal best (Pbest) position of the swarm particle is mutated and in AMPSO2, the global best or the gbest position of the particle is mutated. The EP phase is activated at the end of each iteration (after the velocity and position vector update is complete), where the particles are mutated according to the following rule:

$$x_{ij} = x_{ij} + \sigma'_{ij} * Betarand_j() \qquad (5.1)$$

where, $\sigma'_{ij} = \sigma_{ij} * \exp(\tau \ N(0,1) + \tau' \ N_j(0,1))$

N(0, 1) denotes a normally distributed random number with mean zero and standard deviation one. $N_j(0, 1)$ indicates that a different random number is generated for each value of j. τ and τ' are set as $1/\sqrt{2n}$ and $1/\sqrt{2\sqrt{n}}$ respectively (Engelbrecht, 2005), where n is the population size. $Betarand_j()$ is a random number generated by beta distribution with parameters less than 1.

The pseudo code of proposed AMPSO1 algorithm is given below:

Initialize particle's position and velocity vectors. Each particle is taken as a pair of real-valued vectors, (X_i, σ_i). *The* X_i *'s give the* i^{th} *particle of the swarm and* σ_i *'s the associated strategy parameters. i varies from 1 to N (population size)*

Do

// Update velocity and position vector

For j = 1 to D (dimension) do

$$v_{ij} = w * v_{ij} + c_1 r_1 (P_{ij} - x_{ij}) + c_2 r_2 (P_{gj} - x_{ij})$$

$$x_{ij} = x_{ij} + v_{ij}$$

End for

Calculate the fitness value, $f(X_i)$*, of each particle*

If($f(X_i) < f(P_i)$ *)* $P_i = X_i$

If($f(P_i) < f(P_g)$ *)* $P_g = P_i$

End if

End if

If (U (0, 1) < 1/D) then

// Apply mutation to P_i ***(personal best position of particle) using Eqn. (5.1) //***

For j = 1 to D (dimension) do

$$P'_{ij} = P_{ij} + \sigma'_{ij} * Betarand_j()$$

$$\sigma'_{ij} = \sigma_{ij} * \exp(\tau\ N(0,1) + \tau'\ N_j(0,1))$$

End for

End if

Evaluate the fitness value of P'_i, $f(P'_i)$

If($f(P'_i) < f(P_i)$ ***)*** $P_i = P'_i$

If($f(P_i) < f(P_g)$ ***)*** $P_g = P_i$

End if

End if

Until stopping criteria is reached

Here U (0, 1) is the uniformly distributed random number in the interval (0, 1) and D is the dimension. The algorithmic steps of AMPSO2 algorithm are the same as the above, just Beta distribution mutating the ***global best particle (Pg)*** instead of personal best position (P_i).

5.4 MDE: A Hybrid Algorithm of Differential Evolution and Evolutionary Programming

The proposed MDE algorithm is a hybrid version of Differential Evolution and Evolutionary Programming and it is similar to DE-PSO algorithm. The only difference between DE-PSO and MDE is DE-PSO uses PSO phase whereas MDE uses EP phase. MDE also starts like the usual DE algorithm up to the point where the trial vector is generated. If the trial vector is better than the target vector, then it is included in the population otherwise the algorithm enters the EP phase and generates a new candidate solution using EP based mutation. The method is repeated iteratively till the optimum value is reached.

The initial numerical results showed that the proposed MDE algorithm gave a better performance than the other hybridized versions presented in this chapter. Therefore in order to further analyze its performance, it was initialized with different distributions. Three versions of MDE based on the initialization scheme are proposed; MDE algorithm initialized with uniform distribution is called as U-MDE, with Gaussian distribution it is termed as G-MDE and with low discrepancy Sobol sequence it is called as S-MDE.

The C++ style pseudo code of the MDE Algorithm is:

Initialize the population using uniform (/Gaussian/ Sobol sequence) distributed random numbers

Do

For i = 1 to N (Population size) do

Select r_1, r_2, r_3 $\in N$ randomly

// r_1, r_2, r_3 are selected such that $r_1 \neq r_2 \neq r_3$//

For j = 1 to D (dimension) do

Select j_{rand} $\in D$

If (rand () < CR or j = j_{rand})

// rand () denotes a uniformly distributed random number between 0 and 1//

$$U_{ij,g+1} = x_{r_1,g} + F * (x_{r_2,g} - x_{r_3,g})$$

End if

End for

If ($f(U_{i,g+1}) < f(X_{i,g})$) then

$$X_{i,g+1} = U_{i,g+1}$$

Else

// EP phase activated

Find a new particle using EP based mutation

Let this particle be** $TX = (tx_1, tx_2, ..., tx_D)$ **//

For j = 1 to D (dimension) do

$$tx_j = x_{ij,g} + \sigma_{ij} * N_j(0,1)$$

$$\sigma'_{ij} = \sigma_{ij} * \exp(\tau\ N(0,1) + \tau'\ N_j(0,1))$$

End for

If ($f(TX_i) < f(X_{i,g})$) then

$X_{i,g+1} = TX_i$

Else

$X_{i,g+1} = X_{i,g}$

End if

End if

End for

Until stopping criteria is reached

Here $N(0,1)$, $N_j(0,1), \tau$ and τ' are same as in section 5.3.

5.5 Parameter Settings and Benchmark Problems

The main parameters of DE are crossover rate Cr and scaling factor F, which are taken as 0.2 and 0.5 respectively for the basic DE as well as for the proposed DE-PSO and MDE algorithm. For basic PSO and for proposed DE-PSO and AMPSO, inertia weight w is taken to be linearly decreasing (0.9 – 0.5) and acceleration constants c_1 and c_2 are taken as 2.0 each. Besides these settings all the problems are tested for dimensions 30 for which the population size is taken as 30 for all the algorithms. Stopping criteria for all the algorithms is decided according to the maximum numbers of generations which is fixed at 3000.

The compatibility of the proposed DE-PSO, AMPSO and MDE algorithms is tested on a suite of twelve benchmark problems; RS, DeJ, GR, RB, SWF, GP1, GP2, ACK, DeJ-N, SWF2.21, SWF2.22 and ST. The mathematical models of the test problems with the true optimum value are given in Appendix I.

Besides using the basic DE and PSO for comparison of proposed DE-PSO and MDE algorithms we have also used two recent versions namely DEPSO (Hao et al, 2007) and BBDE (Omran et al, 2007). When the proposed algorithms are compared with two other algorithms in the literature the same experimental settings as that mentioned in the literature is taken, in order to make a fair comparison. When the proposed algorithms are compared with DEPSO population size of the swarm is taken as 30 and dimension of the problems is also taken as 30, while the maximum number of generation is fixed at 12000 for both the algorithms. In the comparison of proposed algorithms with BBDE, the dimension and population size of the swarm are fixed at 30 each while the stopping criteria is taken as the number of function evaluations instead of number of generations like the previous comparisons. The maximum number of function evaluations is set as 100000. The proposed AMPSO algorithm is compared with three other algorithms in the literature namely: CPSO (Stacey et al, 2003), FEP and CEP (Yao and Liu, 1996) in addition with the comparison of PSO and EP.

5.6 Results and Discussion

The proposed algorithms are compared with basic DE, PSO, EP and 5 other variants of DE, PSO and EP in the literature.

5.6.1 Performance Analysis I: Comparison of DE-PSO with DE and PSO

The proposed DE-PSO algorithm is compared with the basic DE and PSO algorithms and the corresponding numerical results are given in Table 5.1. As expected the proposed DE-PSO algorithm performed much better than the classical PSO and DE. From Table 5.1 it can be seen that when DE-PSO is used to solve the present benchmark problems the improvement in terms of average fitness function is more than 95% in comparison to basic PSO for 8 test cases out of 12 test cases. For all the remaining test cases there is an improvement of more than 20%. If the comparison is made with the performance of DE-PSO with basic DE then it can be seen that improvement in fitness function value is more than 40% in 4 test cases and an improvement of more than 20% in 2 test cases. For functions GR and ST, both DE and DE-PSO gave same results. The performance curves of proposed DE-PSO algorithms with DE for selected benchmark problems are given in Figure 5.1 (a) – 5.1 (d) and the performance curves of DE-PSO and PSO for all benchmark problems are shown in Figure 5.2 (a) – Fig 5.2 (d).

5.6.2 Performance Analysis II: Comparison of AMPSO with PSO and EP

Since AMPSO contains the features of both PSO and EP we compared its performance with basic PSO and EP. In Table 5.2, the comparison of AMPSO1 and AMPSO2 is shown with

PSO and EP. From the numerical results of Table 5.2, it can be seen that the AMPSO algorithms perform better than the PSO and EP algorithms in all the considered test problems except the function SWF, where EP gave a slightly better performance. If the AMPSO algorithms are compared with each other then the AMPSO2 (in which the global best particle is mutated) algorithm gives the best values in terms of average fitness function value for all the test functions except the functions GP1 and ST, where both the algorithms perform the same. The performance curves of the proposed AMPSO algorithms with respect to few selected test problems are given in Figure 5.3 (a) – (d).

5.6.3 Performance Analysis III: Comparison of MDE with DE and EP

In Table 5.3 the comparison of the proposed MDE versions is given with the basic DE and EP in terms of average fitness function value and standard deviation. In terms of average fitness function value all the algorithms gave good performance as it is evident from Table 5.3, though the proposed versions gave a slightly better performance in some of the test cases. If the standard deviation is compared, then also it can be observed that all the algorithms converged to the desired objective function value with small value for standard deviation which is nearer to zero in almost all the test cases. This tendency shows the stability of the algorithms. When the proposed versions are compared with the EP algorithm in terms of fitness function value, it can be clearly seen that the proposed MDE algorithms gave much better results than EP. If the comparisons are made with the proposed MDE algorithms with each other then half of the test cases (6 cases out of 12 test cases) S-MDE, which is the hybrid of DE and EP and follows sobol sequence for initial population, gave better performance than other two (U-MDE and G-MDE) algorithms. In 2 test cases out of 12, G-MDE is better than other compared algorithms, U-MDE is better than others in one test case. The performance curves of the proposed MDE algorithms with respect to few selected problems are given in Figure 5.4 (a) – (d).

5.6.4 Performance Analysis IV: Comparison of DE-PSO, AMPSO and MDE with each other

The numerical results of comparison of all proposed algorithms are given in Table 5.4. From the numerical results of Table 5.4, it can be seen that S-MDE algorithm is superior with other algorithms. It gave better result than other compared algorithms in 5 test cases out of 12 test cases. For the first function, which is Rastringin (RS) function, G-MDE gave better performance than all other proposed algorithms. In this case S-MDE gave slightly higher fitness value than U-MDE. The second function is a simple sphere function, in which all the proposed algorithms perform more or less same in terms of fitness function value but if we do the exact comparison then S-MDE is the winner. From the results of Griewank (GR) function, which is the third test case, DE-PSO, G-MDE and S-MDE algorithms converged to the exact global optimum value. Again S-MDE gave better performance in the 4^{th} test case. In case of Schwefel (SWF) function, U-MDE converged to the exact global minimum and for GP2, DE-PSO algorithm converged to the true optimum. For generalized penalized function 1 (GP1) and for function ST, all the proposed algorithms gave the same fitness value. On Ackley (ACK) function, the performance of G-MDE is better than DE-PSO, AMPSO1, AMPSO2, U-MDE and S-MDE. For the remaining three test problems, S-MDE is a clear winner.

Also, the comparison of all proposed hybrid algorithms is made with PSO and DE using the performance measures average fitness function value, standard deviation, success rate, number of function evaluations, average number of generations and CPU time using a test suit of five benchmark problems (RS, DeJ, RB, GR and ACK). A run is considered as success run if the function value satisfies the accuracy level $|f_{max} - f_{min}| < 10^{-4}$. The comparison results are given in Tables 5.5 and 5.6. From these results also, it can be seen that the superior performance of the proposed hybrid algorithms in comparison with PSO and DE.

5.6.5 Performance Analysis V: Comparison of DE-PSO, AMPSO and MDE with other Algorithms in the Literature

The performances of the proposed DE-PSO and MDE algorithms are further compared with BBDE and DEPSO. In order to make a fair comparison of algorithms the same parameter settings for the DEPSO and BBDE as mentioned in (Omran et al, 2007) and (Hao et al, 2007) are taken. Corresponding numerical results for these algorithms are given in Table 5.7 and 5.8. From Table 5.7, it can be seen that the proposed DE-PSO and MDE algorithms are better or at par with DEPSO for all the test functions. In comparison to BBDE, DE-PSO and MDE algorithms gave a superior performance in three out of five test cases tried. For function ST, all the algorithms gave same performance.

Table 5.9, gives the comparison of AMPSO versions with two versions of EP namely FEP and CEP and one version of PSO namely CPSO. FEP uses self adaptive Cauchy mutation, CEP is the classical EP using self adaptive Gaussian mutation and CPSO is the basic PSO with Gaussian mutation. The difference in the results of EP given in Table 5.2 and CEP given in Table 5.9 is because of different experimental settings (see (Yao and Liu, 1996) for experimental settings of CEP and FEP). Despite different experimental settings, it can be observed from Table 5.9, that AMPSO2 gave a superior performance in 7 out of 12 problems whereas FEP gave better results in 4 out of 12 test problems. In case of function ST, except for CEP, all the algorithms converged to the true global optimum.

Table 5.1 Comparison of proposed DE-PSO with PSO and DE in terms of average fitness function value and standard deviation (std)

Function	PSO	DE	DE-PSO
RS	37.819279 (7.455974)	2.53134 (5.19026)	1.61412 (3.88547)
DeJ	3.542765e-16 (4.267806e-16)	2.55105e-047 (3.03209e-047)	4.07701e-48 (1.59336e-47)
GR	0.018439 (0.023367)	0.00000 (0.00000)	0.00000 (0.00000)
RB	81.27341 (41.218071)	31.1369 (17.1211)	24.2024 (12.3086)
SWF	-10652.332146 (663.174189)	-12534 (54.2753)	-12545.8 (47.3753)
GP2	-1.138451 (0.005241)	-1.149356 (1.85776e-016)	-1.15044 (0.00000)
GP1	0.020733 (0.052857)	5.50443e-013 (0.00000)	5.50585e-013 (0.00000)
ACK	1.026221e-08 (1.906846e-08)	7.25006e-015 (7.74296e-016)	3.69735e-015 (0.00000)
DeJ-N	0.508692 (0.250828)	0.00744261 (0.00170156)	0.00766934 (.00206289)
SWF2.22	5.357442 (3.204075)	4.23925e-006 (1.32717e-006)	2.47426e-006 (1.44023e-006)
SWF2.21	2.063802e-11 (5.853435e-12)	8.91504e-027 (3.34859e-027)	4.79286e-027 (4.52357e-027)
ST	0.05 (0.217945)	0.00000 (0.00000)	0.00000 (0.00000)

Table 5.2 Numerical Results of proposed AMPSO algorithms in comparison with basic PSO and EP in terms of average fitness function value and standard deviation (std)

Function	PSO	AMPSO1	AMPSO2	EP
RS	37.819279 (7.455974)	31.838496 (6.004726)	18.307161 (2.658596)	184.097 (20.4831)
DeJ	3.542765e-16 (4.267806e-16)	2.320141e-36 (4.57339e-36)	4.343247e-40 (7.930461e-40)	25.2578 (5.71493)
GR	0.018439 (0.023367)	1.626303e-19 (9.812327e-09)	8.673617e-20 (3.160965e-20)	1.0735 (0.0250742)
RB	81.27341 (41.218071)	29.137924 (25.014428)	24.171648 (16.361839)	157.385 (154.873)
SWF	-10652.332146 (663.174189)	-11937.8019 (383.462609)	-12214.171614 (212.294602)	-12297.4 (676.107)
GP2	-1.138451 (0.005241)	-1.139848 (0.024136)	-1.144443 (0.010626)	-0.986551 (0.185161)
GP1	0.020733 (0.052857)	5.505851e-13 (0.00000)	5.505851e-13 (0.00000)	14.7636 (5.04623)
ACK	1.026221e-08 (1.906846e-08)	5.616167e-17 (1.046215e-17)	3.187554e-17 (3.913005e-13)	10.3051 (1.05315)
DeJ-N	0.508692 (0.250828)	0.467018 (0.31365)	0.416838 (0.220852)	5.38792 (1.71161)
SWF2.22	5.357442 (3.204075)	0.155533 (0.110062)	0.147111 (0.131858)	9.91998 (1.3015)
SWF2.21	2.063802e-11 (5.853435e-12)	4.555099e-16 (1.821241e-15)	1.392284e-17 (3.438109e-17)	32.8945 (4.47881)
ST	0.05 (0.217945)	0.00000 (0.00000)	0.00000 (0.00000)	1.53333 (2.27645)

Table 5.3 Comparison of proposed MDE with DE and EP in terms of average fitness function value and standard deviation (std)

Function	DE	EP	U-MDE	G-MDE	S-MDE
RS	2.53134 (5.19026)	184.097 (20.4831)	1.43079 (3.31876)	0.11534 (0.296985)	0.198991 (0.397982)
DeJ	2.55e-47 (3.03e-47)	25.2578 (5.71493)	2.62e-47 (1.82e-47)	4.20e-48 (3.51e-48)	9.14e-49 (6.50e-49)
GR	0.00000 (0.00000)	1.0735 (0.025074)	1.08e-20 (2.16e-20)	0.00000 (0.00000)	0.00000 (0.00000)
RB	31.1369 (17.1211)	157.385 (154.873)	30.4642 (12.8225)	24.5408 (0.770056)	23.8297 (0.95337)
SWF	-12534 (54.2753)	-12297.4 (676.107)	-12569.5 (1.67e-005)	-12537.9 (67.923)	-12514.2 (121.299)
GP2	-1.149356 (1.85e-16)	-0.986551 (0.185161)	-1.15037 (1.27e-05)	-1.15036 (1.71e-05)	-1.15038 (1.28e-06)
GP1	5.50e-13 (0.00000)	14.7636 (5.04623)	5.50e-13 (0.00000)	5.50e-13 (0.00000)	5.50e-13 (0.00000)
ACK	7.25e-15 (7.74e-16)	10.3051 (1.05315)	4.05e-15 (1.06e-15)	2.54e-15 (2.36e-16)	3.69e-15 (1.12e-16)
DeJ-N	0.007442 (0.001701)	5.38792 (1.71161)	0.007806 (0.001528)	0.005396 (0.001858)	0.000205 (2.93e-05)
SWF2.22	4.23e-06 (1.32e-06)	9.91998 (1.3015)	4.63e-06 (1.44e-06)	7.72e-08 (3.12e-08)	6.16e-08 (6.05e-08)
SWF2.21	8.91e-27 (3.34e-27)	32.8945 (4.47881)	8.37e-27 (3.68e-27)	1.95e-27 (8.25e-28)	9.94e-28 (3.72e-28)
ST	0.00000 (0.00000)	1.53333 (2.27645)	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)

Table 5.4 Comparison results of proposed DE-PSO, AMPSO1, AMOSO2, U-MDE, G-MDE and S-MDE algorithms

Function	DE-PSO	AMPSO1	AMPSO2	U-MDE	G-MDE	S-MDE
RS	1.61412 (3.88547)	31.8384 (6.0047)	18.307161 (2.658596)	1.43079 (3.31876)	0.11534 (0.296985)	0.198991 (0.397982)
DeJ	4.07e-48 (1.59e-47)	2.32e-36 (4.57e-36)	4.34e-40 (7.93e-40)	2.62e-47 (1.82e-47)	4.20e-48 (3.51e-48)	9.14e-49 (6.50e-49)
GR	0.00000 (0.00000)	1.62e-19 (9.81e-09)	8.67e-20 (3.16e-20)	1.08e-20 (2.16e-20)	0.00000 (0.00000)	0.00000 (0.00000)
RB	24.2024 (12.3086)	29.137924 (25.014428)	24.171648 (16.361839)	30.4642 (12.8225)	24.5408 (0.770056)	23.8297 (0.95337)
SWF	-12545.8 (47.3753)	-11937.80 (383.4626)	-12214.17 (212.29)	-12569.5 (1.67e-05)	-12537.9 (67.923)	-12514.2 (121.299)
GP2	-1.15044 (0.00000)	-1.139848 (0.024136)	-1.144443 (0.010626)	-1.15037 (1.27e-05)	-1.15036 (1.71e-05)	-1.15038 (1.28e-06)
GP1	5.50e-13 (0.00000)	5.50e-13 (0.00000)	5.50e-13 (0.00000)	5.50e-13 (0.00000)	5.50e-13 (0.00000)	5.50e-13 (0.00000)
ACK	3.69e-15 (0.00000)	5.61e-17 (1.04e-17)	3.18e-17 (3.91e-13)	4.05e-15 (1.06e-15)	2.54e-15 (2.36e-16)	3.69e-15 (1.12e-16)
DeJ-N	0.007669 (0.002062)	0.467018 (0.31365)	0.416838 (0.220852)	0.007806 (0.001528)	0.005396 (0.001858)	0.000205 (2.93e-05)
SWF2.22	2.47e-06 (1.44e-06)	0.155533 (0.110062)	0.147111 (0.131858)	4.63e-06 (1.44e-06)	7.72e-08 (3.12e-08)	6.16e-08 (6.05e-08)
SWF2.21	4.79e-27 (4.52e-27)	4.55e-16 (1.82e-15)	1.39e-17 (3.43e-17)	8.37e-27 (3.68e-27)	1.95e-27 (8.25e-28)	9.94e-28 (3.72e-28)
ST	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)

Table 5.5 Comparison results of all the proposed hybrid algorithms in terms of mean fitness, standard deviation (Std), success rate (SR), number of function evaluations (NFE) and convergence time (in seconds)

Fun	PSO					DE					DE-PSO				
	Fitness	Std	SR	NFE	Time	Fitness	Std	SR	NFE	Time	Fitness	Std	SR	NFE	Time
RS	8.44	4.03	-	50050	2.1	1.5e-5	3.6e-6	100	18625	0.8	1.3e-5	6.4e-6	100	33020	1.2
DeJ	2.4e-8	2.8e-8	-	50050	2.1	1.4e-5	5.7e-6	100	7555	0.2	1.3e-6	3.4e-6	100	11753	0.2
GR	0.094	0.039	-	50050	2.4	8.3e-5	1.2e-5	100	27900	0.7	4.0e-7	2.9e-6	100	51999	1.2
RB	23.78	34.42	-	50050	6.3	5.54	1.97	-	50050	2.8	1.87	1.75	-	93668	5.7
ACK	9.96	9.95	-	50050	1.7	4.7e-5	1.2e-5	100	14395	0.3	2.5e-5	1.2e-5	100	22988	0.6
Fun	AMPSO1					AMPSO2					U-MDE				
	Fitness	Std	SR	NFE	Time	Fitness	Std	SR	NFE	Time	Fitness	Std	SR	NFE	Time
RS	5.07	3.68	90	37115	0.9	4.97	2.37	90	35375	0.9	8.5e-6	7.0e-6	100	32695	1.8
DeJ	6.1e-9	7.3e-9	100	14640	0.4	3.0e-9	7.5e-9	100	15390	0.4	1.1e-6	2.9e-5	100	11812	0.3
GR	0.041	0.026	50	41755	1.0	0.062	0.021	60	41560	1.0	3.2e-7	1.3e-6	100	57057	1.5
RB	0.996	1.52	-	50050	6.8	0.597	1.15	-	50050	6.9	3.58	1.72	-	94619	5.5
ACK	1.0e-6	9.3e-7	100	21490	0.5	7.9e-7	9.3e-7	100	21700	0.5	5.2e-7	1.4e-6	100	22620	0.6
Fun	G-MDE					S-MDE									
	Fitness	Std	SR	NFE	Time	Fitness	Std	SR	NFE	Time					
RS	9.5e-6	4.1e-6	100	29358	1.6	1.3e-6	6.7e-6	100	26412	1.4					
DeJ	9.8e-6	3.7e-6	100	10380	0.3	5.6e-8	5.1e-6	100	9323	0.2					
GR	5.9e-6	4.7e-6	100	8707	0.2	1.2e-7	4.4e-6	100	7568	0.3					
RB	4.95	0.173	-	91185	5.8	1.748	0.889	-	95219	5.6					
ACK	5.1e-6	9.6e-6	100	17795	0.4	5.6e-8	1.4e-5	100	16723	0.4					

Table 5.6 Comparison results of all the proposed hybrid algorithms in terms of average number of generations to achieve the accuracy level $|f_{max} - f_{min}| < 10^{-4}$

Function	PSO	DE	DE-PSO	AMPSO1	AMPSO2	U-MDE	G-MDE	S-MDE
RS	1000	371	312	741	706	327	322	301
DeJ	1000	150	137	291	306	144	122	110
GR	1000	557	477	834	830	482	102	94
RB	1000	1000	1000	1000	1000	1000	1000	1000
ACK	1000	286	236	428	433	261	216	196

Table 5.7 Comparison of DE-PSO, U-MDE, G-MDE and S-MDE with DEPSO (Mean best (std))

Function	DEPSO	DE-PSO	U-MDE	G-MDE	S-MDE
RS	24.216 (6.417)	0.00000 (0.00000)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)
GR	6.2e-16 (4.1e-16)	0.00000 (0.00000)	0.0000 (0.0000)	0.0000 (0.0000)	0.0000 (0.0000)
SWF	-12547.7 (66.25)	-12554.3 (95.3042)	-12569.5 (0.00000)	-12569.5 (0.00000)	-12569 (1.04275)
GP1	3.9e-20 (4.1e-21)	5.505e-13 (0.00000)	5.51e-13 (0.0000)	5.32e-18 (0.0000)	4.71e-22 (0.0000)
ACK	-0.0002 (0.0002)	3.697e-15 (0.00000)	3.69e-15 (0.0000)	3.69e-15 (0.0000)	3.69e-15 (0.0000)

Table 5.8 Comparison of DE-PSO, U-MDE, G-MDE and S-MDE with BBDE (Mean best (std))

Function	BBDE	DE-PSO	U-MDE	G-MDE	S-MDE
RS	72.185823 (3.018019)	31.2688 (4.87695)	1.73e-13 (1.36e-13)	1.99e-18 (3.66e-18)	0.0000 (0.0000)
GR	0.269504e-01 (0.767095e-02)	8.13e-21 (1.93e-020)	2.16e-20 (2.65e-20)	5.42e-21 (1.62e-20)	5.42e-21 (1.62e-20)
RB	14.295707 (0.948028)	25.254 (0.873509)	48.129 (25.23)	25.51 (0.9108)	25.69 (3.91)
ACK	2.136173 (0.159471)	9.20e-15 (1.76e-15)	2.18e-14 (4.03e-15)	1.08e-14 (3.55e-15)	7.25e-15 (2.31e-15)
ST	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)	0.00000 (0.00000)

Table 5.9 Numerical Results of proposed AMPSO algorithms in comparison with FEP, CEP and CPSO

Function	AMPSO1	AMPSO2	FEP	CEP	CPSO
RS	31.838496 (6.004726)	18.307161 (2.658596)	4.6e-2 (1.2e-2)	89.0 (23.1)	69.050 (16.419)
DeJ	2.320141e-36 (4.57339e-36)	4.343247e-40 (7.930461e-40)	5.7e-4 (1.3e-4)	2.2e-4 (5.9e-4)	2.362e-26 (1.500e-25)
GR	1.626303e-19 (9.812327e-09)	8.673617e-20 (3.160965e-20)	1.6e-2 2.2e-10	8.6e-2 0.12	NA
RB	29.137924 (25.014428)	24.171648 (16.361839)	5.06 (5.87)	6.17 (13.61)	29.084 (31.460)
SWF	-11937.8019 (383.462609)	-12214.171614 (212.294602)	-12554.5 (52.6)	-7917.1 (634.5)	NA
GP2	-1.139848 (0.024136)	-1.144443 (0.010626)	1.6e-4 (7.3e-5)	1.4 (3.7)	NA
GP1	5.505851e-13 (1.75799e-12)	5.505851e-13 (1.75799e-12)	9.2e-6 (3.6e10)	1.76 (2.4)	NA
ACK	5.616167e-17 (1.046215e-17)	3.187554e-17 (3.913005e-13)	1.8e-2 (2.1e-3)	9.2 (2.8)	5.651 (1.333)
DeJ-N	0.467018 (0.31365)	0.416838 (0.220852)	7.6e-3 (2.6e-3)	1.8e-2 (6.4e-3)	NA
SWF2.22	0.155533 (0.110062)	0.147111 (0.131858)	0.3 (0.5)	2.0 (1.2)	NA
SWF2.21	4.555099e-16 (1.821241e-15)	1.392284e-17 (3.438109e-17)	8.1e-3 (7.7e-4)	2.6e-3 (1.7e-4)	NA
ST	0.00000 (0.00000)	0.00000 (0.00000)	0.0000 (0.0000)	577.76 (1125.76)	NA

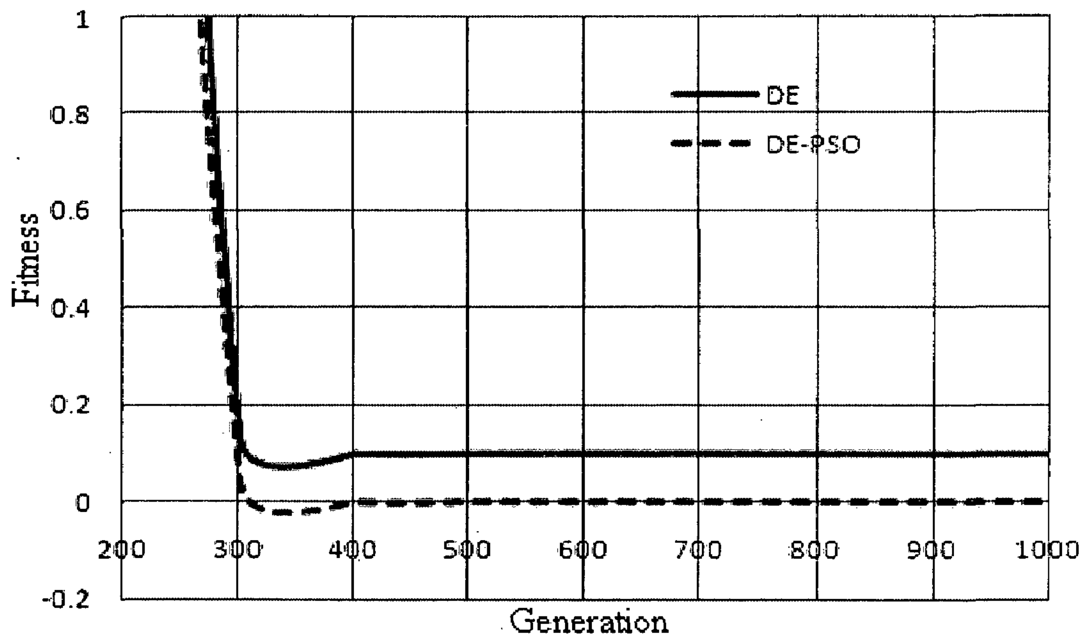

Figure 5.1 (a) Function RS

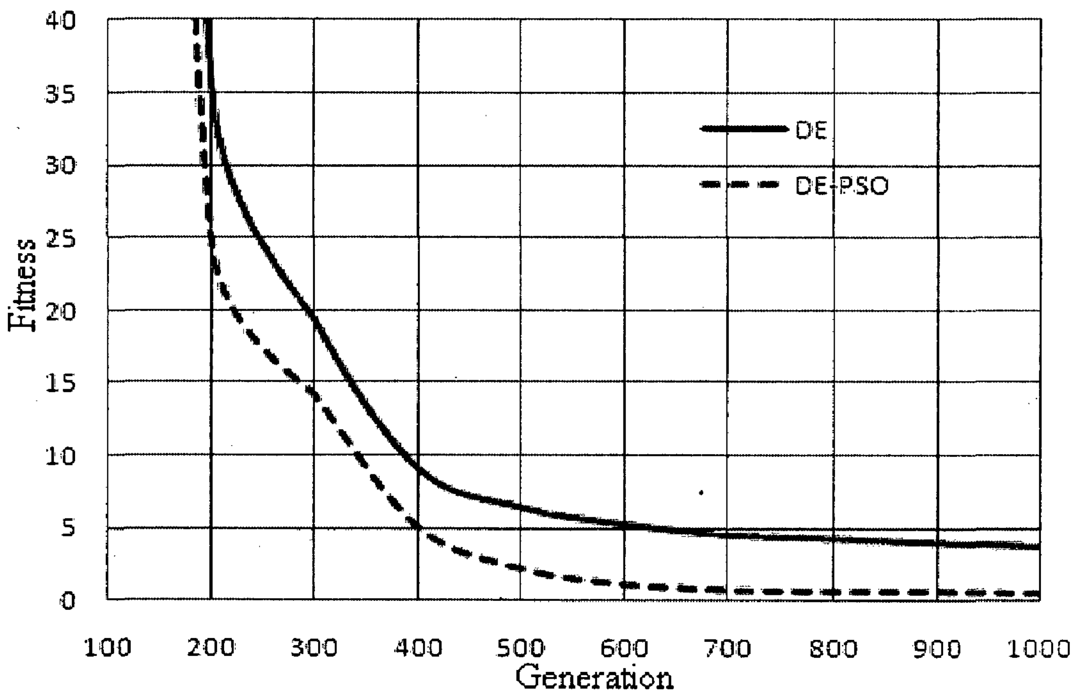

Figure 5.1 (b) Function RB

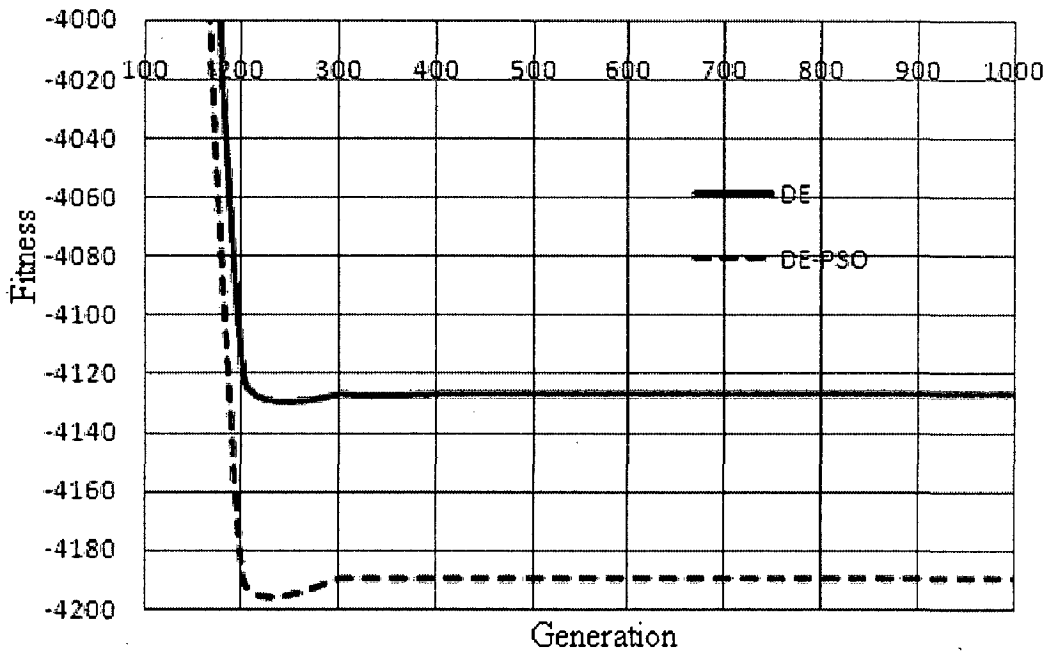

Figure 5.1 (c) Function SWF

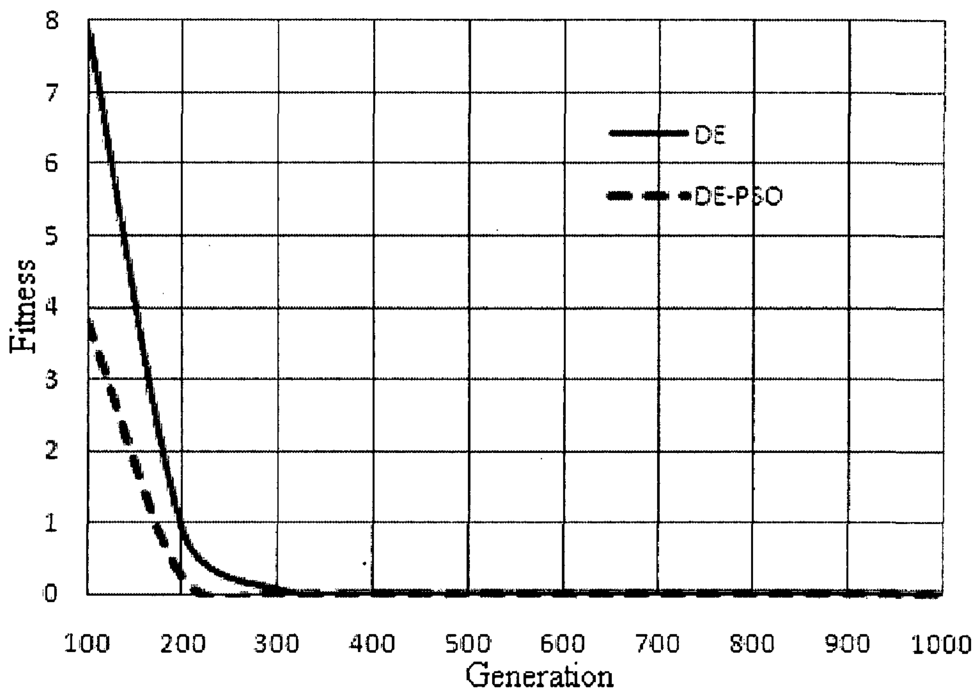

Figure 5.1 (d) Function SWF 2.22

Figure 5.1 Performance curves of DE and DE-PSO algorithms

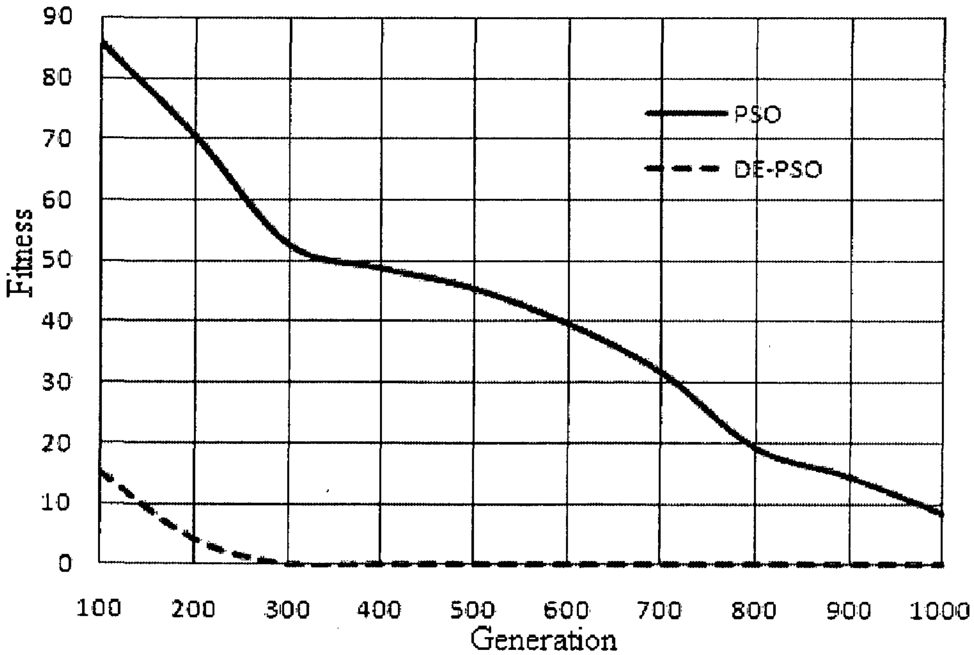

Figure 5.2 (a) Function RS

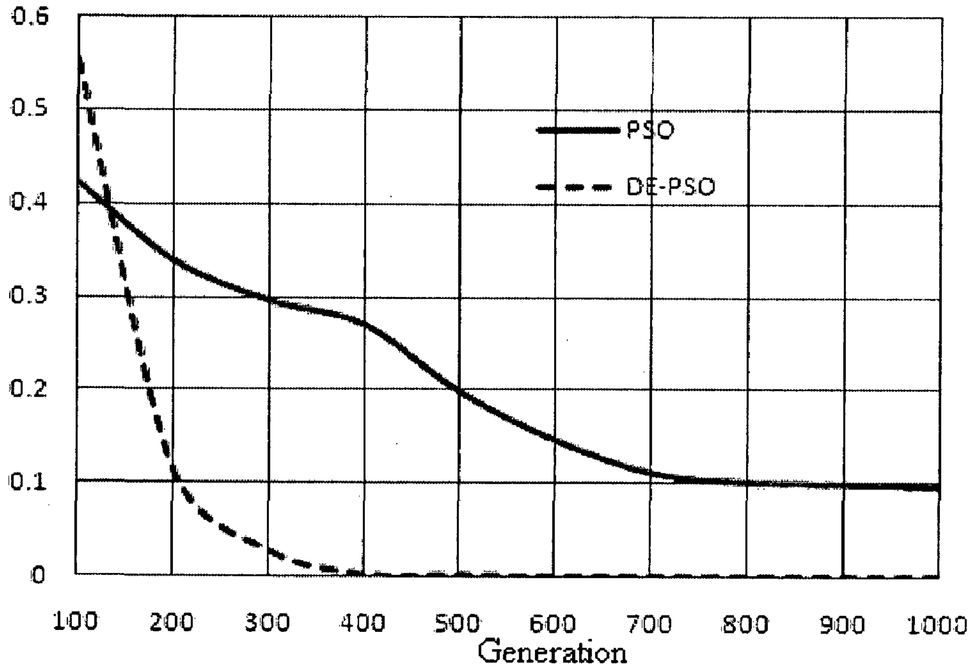

Figure 5.2 (b) Function GR

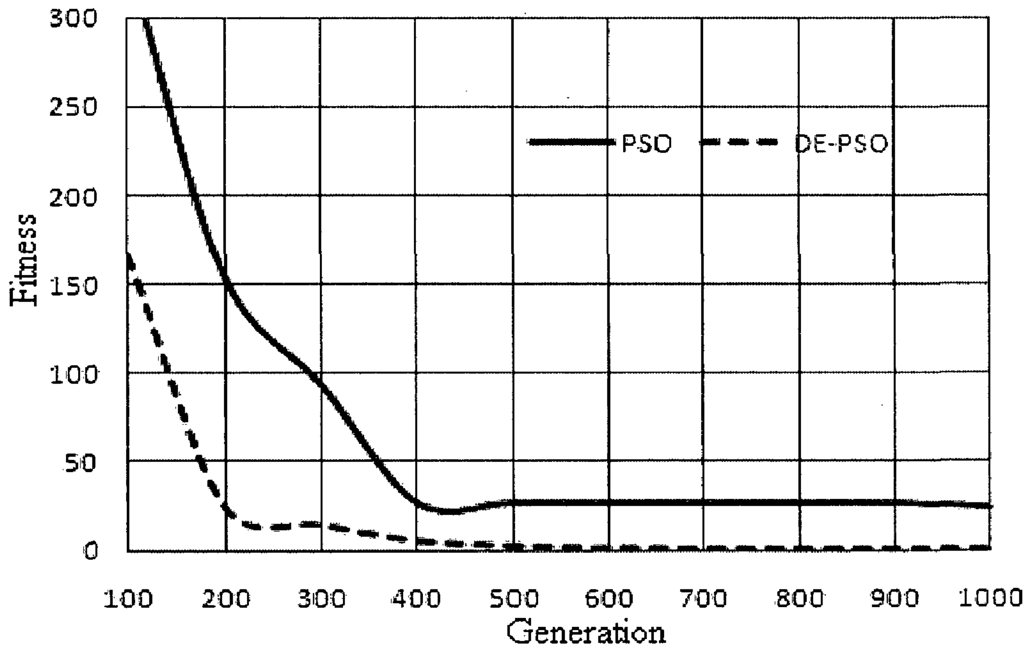

Figure 5.2 (c) Function RB

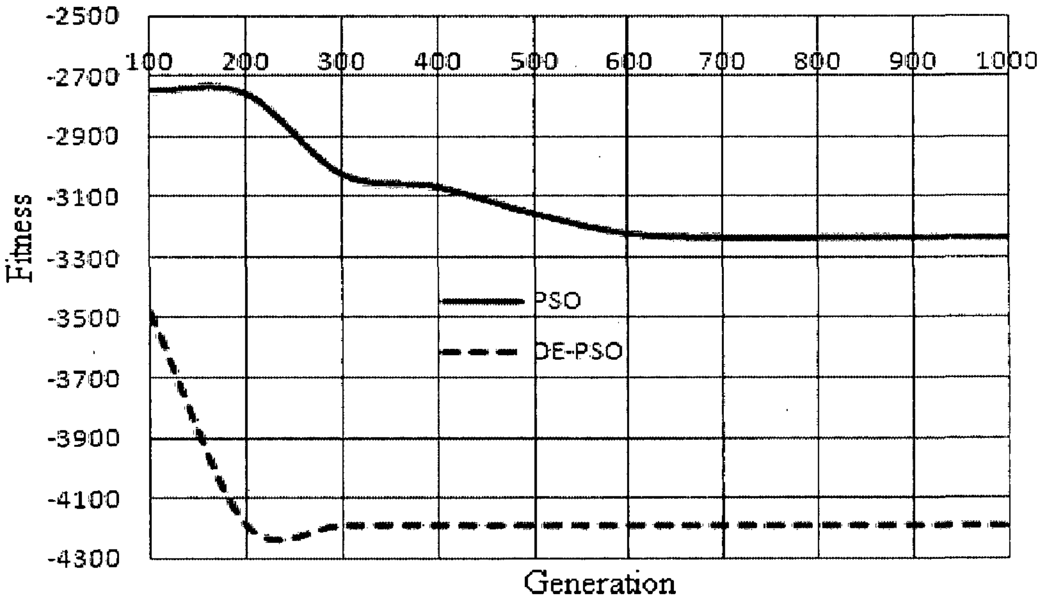

Figure 5.2 (d) Function SWF

Figure 5.2 Performance curves of PSO and DE-PSO algorithms

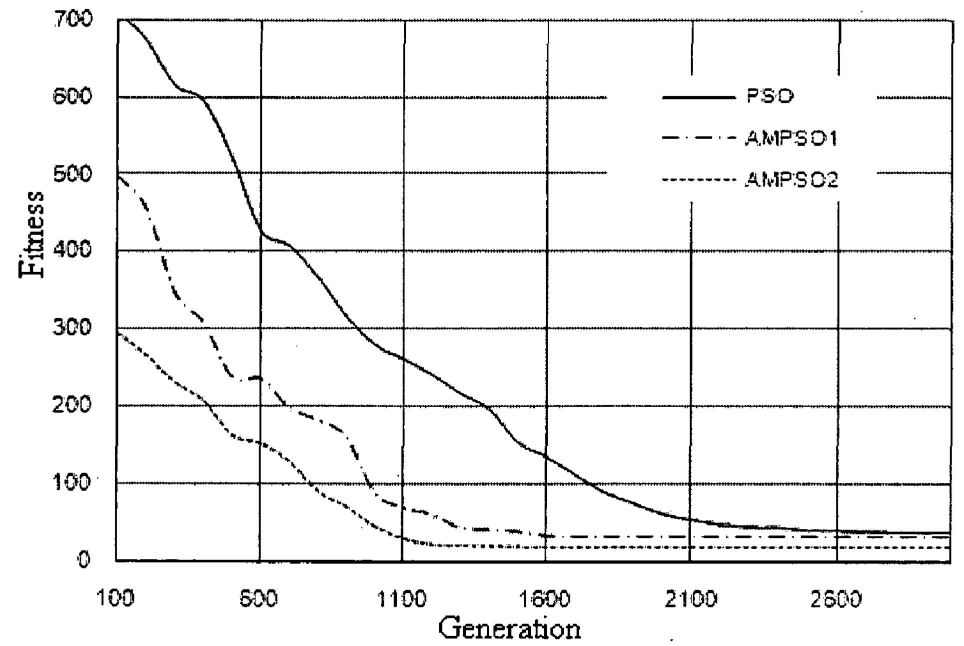

Figure 5.3 (a) Function RS

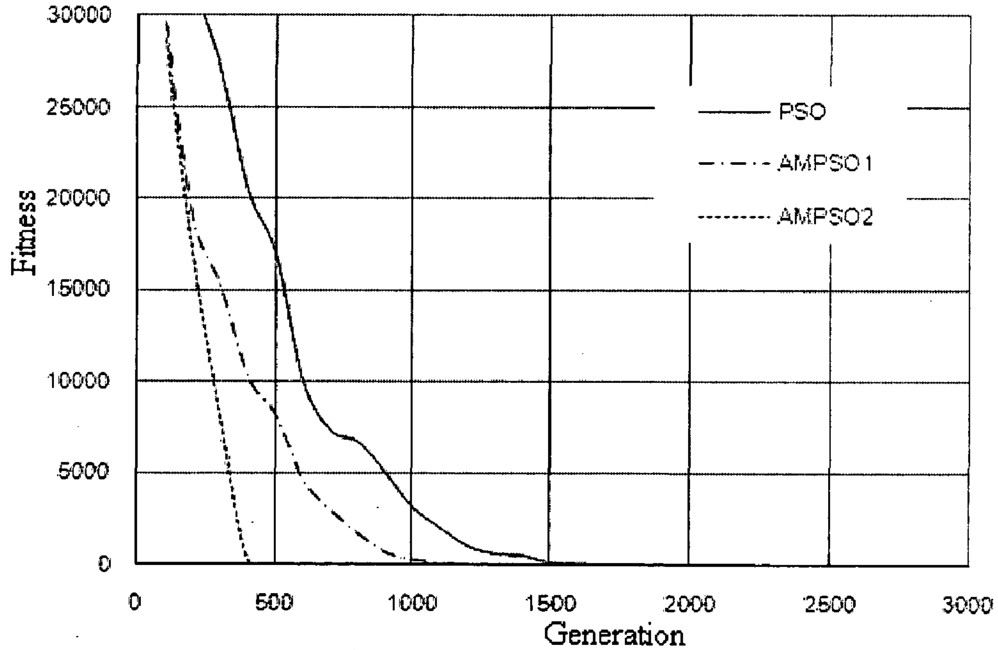

Figure 5.3 (b) Function DeJ

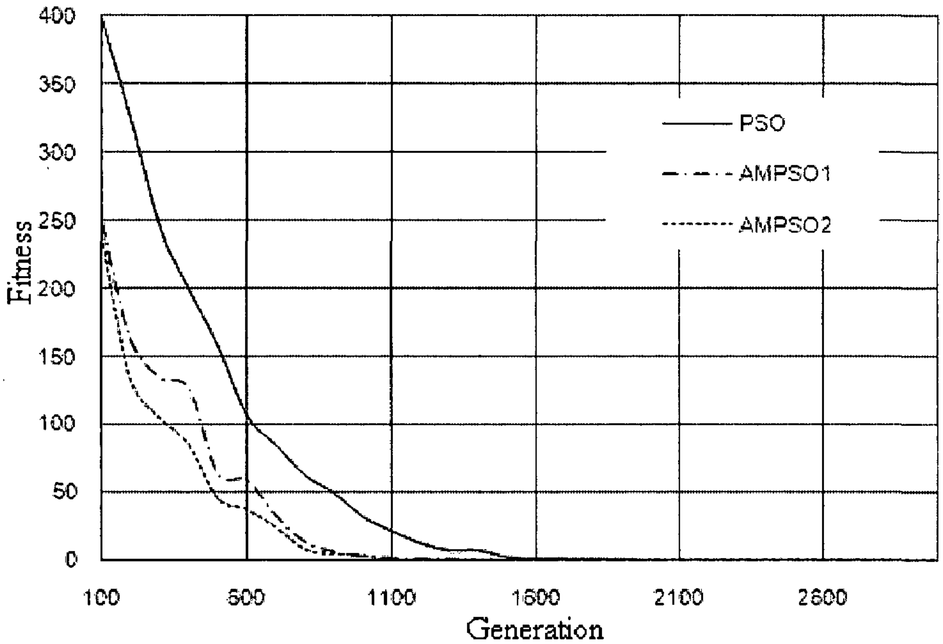

Figure 5.3 (c) Function GR

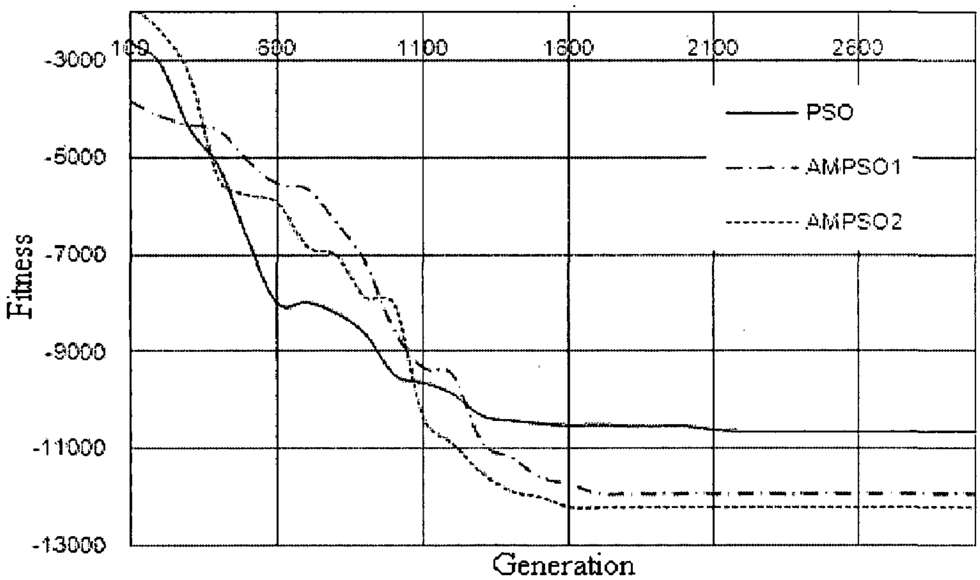

Figure 5.3 (d) Function SWF

Figure 5.3 Performance curves of PSO, AMPSO1 and AMPSO2 algorithms

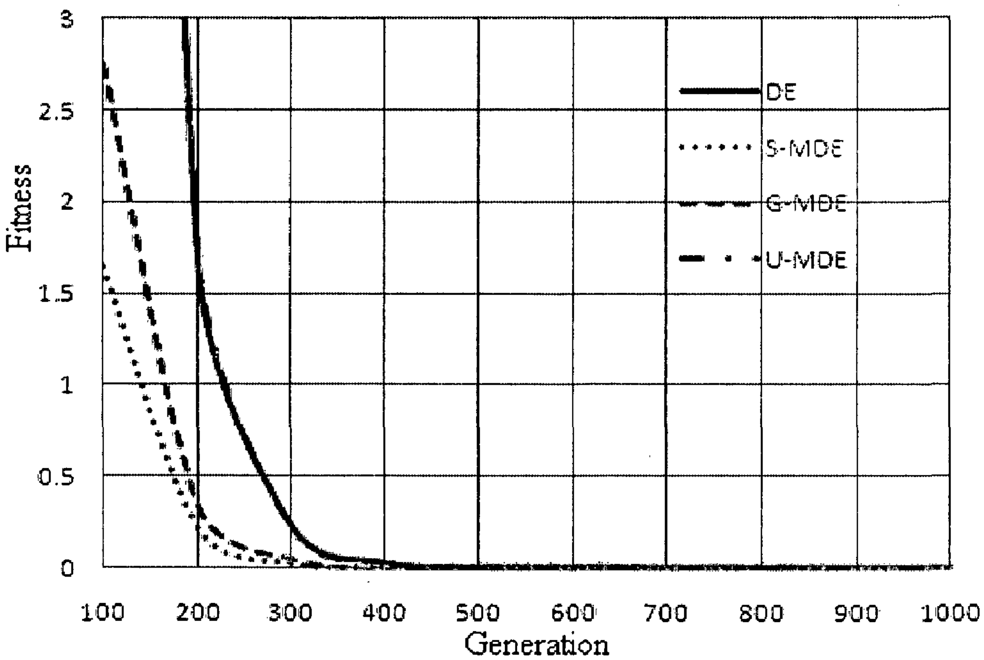

Figure 5.4 (a) Function SWF 2.21

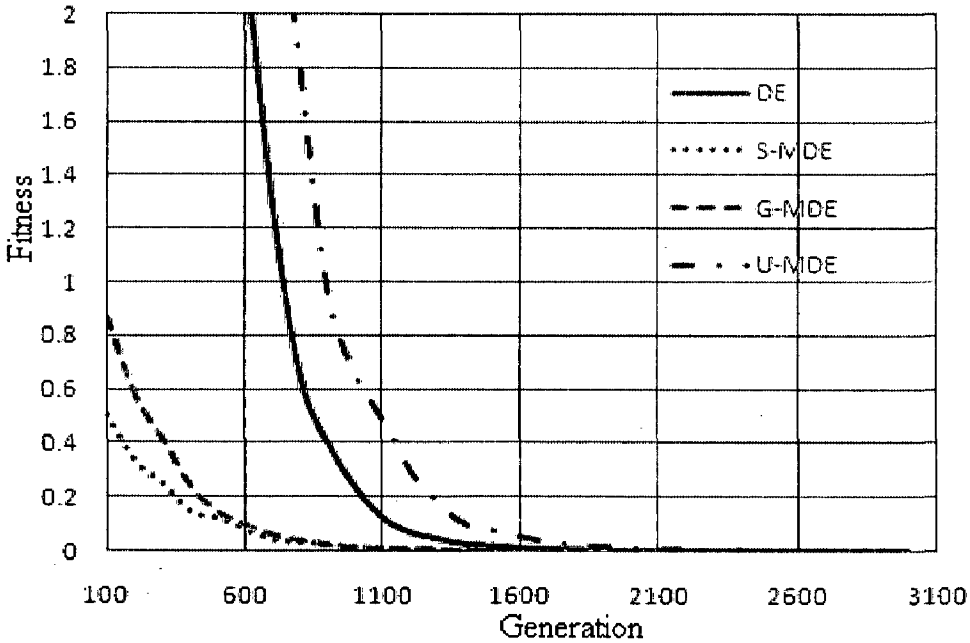

Figure 5.4 (b) Function SWF2.22

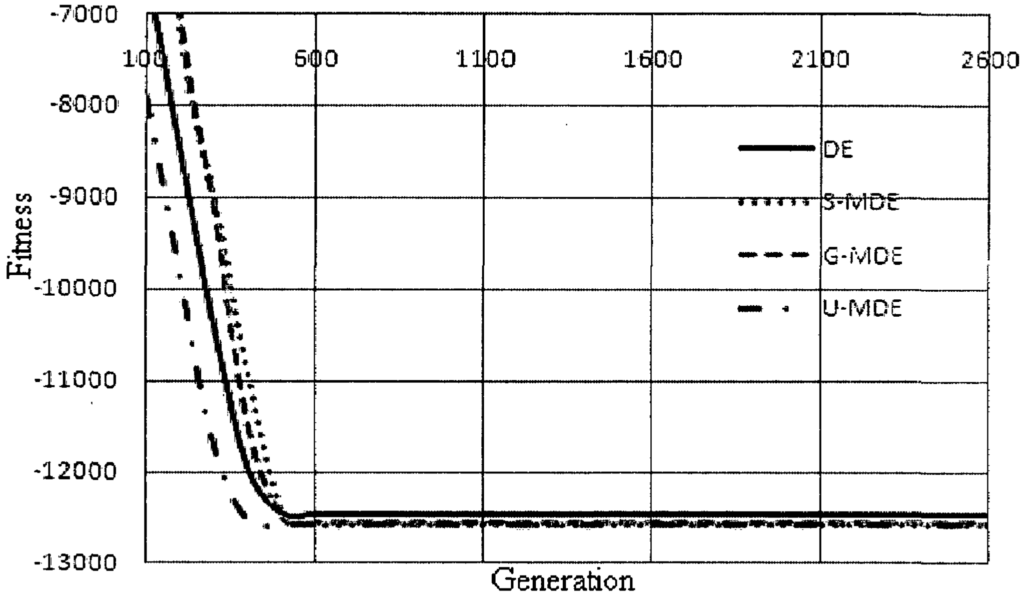

Figure 5.4 (c) Function SWF

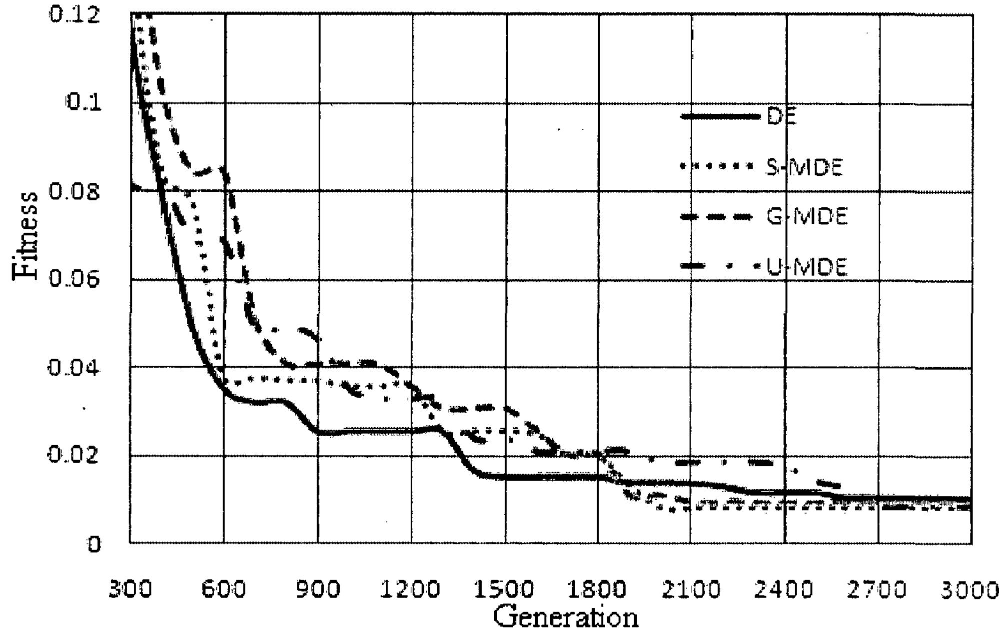

Figure 5.4 (d) Function DeJ-N

Figure 5.4 Performance curves of DE, S-MDE, G-MDE and U-MDE algorithms

5.7 Conclusion

Hybridization is a popular concept being applied to evolutionary algorithms to increase their efficiency and robustness. In this chapter three simple and efficient hybridised versions of DE and PSO algorithms are presented. They are:

- Hybrid of Differential Evolution and Particle Swarm Optimization (DE-PSO)
- Hybrid of Particle Swarm Optimization and Evolutionary Programming (AMPSO)
- Hybrid Differential Evolution and Evolutionary Programming (MDE)

The performance of proposed algorithms were validated on a set of 12 benchmark problems and the numerical results were compared with classical DE, PSO, EP and 5 other variants of DE, PSO and EP in the literature. Although all the modified versions presented in this chapter performed either superior or at par with the basic DE and PSO algorithms, it was observed that MDE, the hybridized version of DE with EP was better than other suggested algorithms. In order to further analyze the performance of MDE it was initialized with different probability distributions which showed that S-MDE (MDE initialized with sobol sequence) outperformed the other versions by a significant difference in terms of average fitness function value and number of function evaluations.

Chapter 6

Constraint Handling Mechanism for PSO and DE Algorithms

[This chapter proposes constraint handling mechanism for PSO and DE algorithms. It is a simple approach and does not need any additional parameters. Based on this approach, two algorithms namely ICPSO and ICDE are proposed. The Improved Constrained Particle Swarm Optimization (ICPSO) and the Improved Constrained DE (ICDE) algorithm differs from basic PSO and DE algorithms for unconstrained optimization problems only in the phase of selection of particles for the next generation and during the sorting of the final results. Besides the selection and sorting rules, the ICPSO algorithm starts with quasi random sequence and ICDE uses a dynamic scaling factor. The performance of ICPSO and ICDE algorithms are analyzed on a test suite of twenty constrained benchmark problems. The numerical results show that the competence of proposed algorithms for solving constrained optimization problems.]

6.1 Introduction

The search space in Constrained Optimization Problems (COPs) consists of two kinds of solutions: feasible and infeasible. Feasible points satisfy all the constraints, while infeasible points violate at least one of them. Therefore, the final solution of an optimization problem must satisfy all constraints.

The general NLP is given by nonlinear objective function f, which is to be minimized /maximized with respect to the design variables $\bar{x} = (x_1, x_2, \ldots\ldots, x_n)$ and the nonlinear inequality and equality constraints. This can be formulated by,

Minimize / Maximize $f(\bar{x})$

Subject to: $g_j(\bar{x}) \leq 0, \; j = 1, \ldots\ldots, p$ (6.1)

$h_k(\bar{x}) = 0, \; k = 1, \ldots\ldots, q$ (6.2)

$x_{i\min} \leq x_i \leq x_{i\max} \; (i = 1, \ldots\ldots, n)$.

Where p and q are the number of inequality and equality constraints respectively and n is the number of variables. A measure of the constraint violation is often useful when handling constraints. A solution $\bar{x}$ is called as feasible solution if

$g_j(\bar{x}) \leq 0,$ for all $j = 1,....., p$

$|h_k(\bar{x})| - \varepsilon \leq 0,$ for all $k = 1,...,q$

Here equality constraints are transformed into inequality constraints and usually ε is set as 0.0001. The constraint violation is defined as:

$$\bar{v} = \sum_{j=1}^{p} G_j(\bar{x}) + \sum_{k=1}^{q} H_k(\bar{x}) \tag{6.3}$$

Where
$$G_j(\bar{x}) = \begin{cases} g_j(\bar{x}) & if \quad g_j(\bar{x}) > 0 \\ 0 & if \quad g_j(\bar{x}) \leq 0 \end{cases} \tag{6.4}$$

$$H_k(\bar{x}) = \begin{cases} |h_k(\bar{x})| & if \quad |h_k(\bar{x})| - \varepsilon > 0 \\ 0 & if \quad |h_k(\bar{x})| - \varepsilon \leq 0 \end{cases} \tag{6.5}$$

There are many traditional methods in the literature for solving NLP. However, most of the traditional methods require certain auxiliary properties (like convexity, continuity etc.) of the problem and also most of the traditional techniques are suitable for only a particular type of problem (for example Quadratic Programming Problems, Geometric Programming Problems etc). Keeping in view the limitations of traditional techniques researchers have proposed the use of stochastic optimization methods and intelligent algorithms for solving constrained NLP. Based on the research efforts in literature, constraint handling methods have been categorized in a number of classes (Engelbrecht, 2005). They are:

Reject infeasible solutions:

It is one of the simplest ways to deal with constraints. In this method particles that violate any of the constraints are rejected. Infeasible particles can be treated in a number of ways: Do not allow infeasible particles to be selected as personal best or neighborhood global best positions.

Penalty function methods:

Many evolutionary algorithms incorporate a constraint-handling method based on the concept of exterior penalty functions which penalize infeasible solutions. These methods differ in important details, how the penalty function is designed and applied to infeasible solutions.

Convert the constrained problem to an unconstrained problem:

Sometimes the constrained problem may be converted into an unconstrained one using a suitable penalty approach.

Preserving feasibility methods:

Preserving feasibility methods ensure that adjustments to particles do not violate any constraints. The particles are initialized to contain only feasible particles, and particles are not allowed to move into infeasible space.

Repair methods:

Repair methods allow particles to move into infeasible space, but special operators/methods are then applied to either change the particle into a feasible one or to direct the particle to move towards feasible space.

Other hybrid methods:

These methods combine evolutionary computation techniques with deterministic procedures for numerical optimization problems.

Out of the aforesaid techniques, penalty methods have been most frequently used for solving constrained optimization problem. However the drawback of this approach is the selection of a suitable penalty parameter. Repair method is another approach which is commonly used for constraint handling. The advantage of this approach over the penalty method is that it does not require any additional parameter and gives good results. In this chapter a new constraint handling mechanism based on repair methods is proposed which uses simple selection and sorting rules.

The rest of the chapter is organized as follows: section 6.2 explains the constraint handling method, section 6.3 and 6.4 describes the proposed ICPSO and ICDE algorithms. In section 6.5, the experimental results and discussion are given and finally the chapter concludes with section 6.6.

6.2 Constraint Handling Mechanism Used in this Thesis

The constraint handling method used in this thesis is easy to implement and does not require the user to set any additional parameters except providing the constraint functions when programming the evaluation function routine. *It differs from unconstrained optimization algorithm only in the phase of selecting candidate solution for the new generation and sorting the final results.* It follows the following selection and sorting rules.

Selection Rules:

The following three selection rules are used for selecting the next generation candidate solution:

1) If both the compared solutions are feasible then select the one having the minimum (in case of minimize problem) or maximum (in case of maximize problem) function value.
2) If both the compared solutions are infeasible then select the one with less constraint violation
3) If one is feasible and another one is infeasible then select the feasible solution

Sorting Rules:

Also at the end of every iteration, the particles are sorted by using the three criteria:

1) Sort feasible solutions in front of infeasible solutions
2) Sort feasible solutions according to their fitness function values
3) Sort infeasible solutions according to their constraint violations.

Based on the above rules, two algorithms are proposed namely Improved Constrained Particle Swarm Optimization (ICPSO) and Improved Constrained Differential Evolution (ICDE).

6.3 Improved Constraint Particle Swarm Optimization Algorithm (ICPSO)

The proposed algorithm ICPSO algorithm is a simple algorithm for solving constraint optimization problems, it is easy to implement. The additional feature of ICPSO algorithm is that it uses quasi random *Vander Corput sequence* to initialize the swarm besides using the selection and sorting rules given in Section 6.2.

The computational steps of ICPSO algorithm are given below:

Step 1 ***Initialize the population (X_i) using low discrepancy Van der Corput Sequence***

Step 2 *For all particles*

Evaluate the objective function

Calculate the constraint violation

End for

Step 3 *w linearly decreases from 0.9 to 0.4*

Step 4 ***For all particles***

//Update velocity vector V_i $(=(v_{i1}, v_{i2},....v_{iD}))$ and find a new particle NX_i $(=(nx_{i1}, nx_{i2},....nx_{iD}))$ using the previous particle X_i^t $(=(x_{i1}^t, x_{i2}^t,....x_{iD}^t))$ where D is the dimension//

$$v_{ij}^{t+1} = w * v_{ij}^t + c_1 r_1 (P_{ij}^t - x_{ij}^t) + c_2 r_2 (P_{gj}^t - x_{ij}^t)$$

$$nx_{ij} = x_{ij}^t + v_{ij}^{t+1}$$

If (NX and X_i^t are feasible) Then

If ($f(NX) < f(X_i^t)$) Then

$$X_i^{t+1} = NX$$

Else

$$X_i^{t+1} = X_i^t$$

End if

End if

If (NX and X_i^t are infeasible) Then

If (constraint violate (NX) < constraint violate (X_i^t)) Then

$$X_i^{t+1} = NX$$

Else

$$X_i^{t+1} = X_i^t$$

End if

End if

If (NX is feasible and X_i^t is infeasible) Then

$$X_i^{t+1} = NX$$

Else

$$X_i^{t+1} = X_i^t$$

End if

Update Personal best position (P_i) and global best position (P_g)

End for

Step 5 ***Sort the particles using the three sorting rules***

Step 6 *Go to Step 3 and repeat the loop until stopping criteria is reached.*

6.4 Improved Constraint Differential Evolution Algorithm (ICDE)

Like ICPSO is a constrained version of PSO, ICDE algorithm is a simple and modified version of DE algorithm for solving constraint optimization problems. It uses the selection and sorting rules mentioned in Section 6.2 and its additional feature is the use of *dynamic scaling factor F* which follows Rayleigh distribution to generate a random number.

The computational steps of ICDE algorithm is given below:

Step 1 *Initialize the population and set control parameters of DE except F (i.e. population size and Crossover rate Cr)*

Step 2 *For all particles*

Evaluate the objective function

Calculate the constraint violation

End for

Step 3 ***// dynamic scaling factor: F ranges in the interval (0, 0.9] //***

$$F = \begin{cases} 0.1 + U(0,1)\sqrt{Grand_1^{\,2} + Grand_2^{\,2}} & if \quad 0.5 < U(0,1) \\ 0.5 & otherwise \end{cases}$$

If $F > 0.9$ *then* $F = 0.9$

Where $U(0,1)$ *is a uniformly distributed random number in the interval (0, 1],* $Grand_1$ *and* $Grand_2$ *are Gaussian distributed random number with mean zero and standard deviation one.*

Step 4 *// Mutation*

For all particles

$$V_{i,g+1} = X_{r_1,g} + F*(X_{r_2,g} - X_{r_3,g})$$

// where x_{r_1}, x_{r_2} *and* x_{r_3} *are randomly selected particles and are different from each other //*

End for

Step 5 *// Crossover (Generate trial vector* $U_{i,g+1}$*)*

For all particles

Select $j_{rand} \in \{1,...,D\}$

For j = 1 to D

If (U(0,1] $\leq$ *Cr or j=* j_{rand}*) Then*

$$u_{ij,g+1} = v_{ij,g+1}$$

Else

$$u_{ij,g+1} = x_{ij,g}$$

End if

End for

End for

Step 6 ***// Selection***

For all particles

If ($U_{i,g+1}$ ***and*** $X_{i,g}$ ***are feasible) Then***

If ($f(U_{i,g+1}) < f(X_{i,g})$***) Then***

$$X_{i,g+1} = U_{i,g+1}$$

Else

$$X_{i,g+1} = X_{i,g}$$

End if

End if

If $(U_{i,g+1}$ *and* $X_{i,g}$ *are infeasible) Then*

If (constraint violate $(U_{i,g+1})$ < *constraint violate* $(X_{i,g})$*) Then*

$X_{i,g+1}=U_{i,g+1}$

Else

$X_{i,g+1}=X_{i,g}$

End if

End if

If $(U_{i,g+1}$ *is feasible and* $X_{i,g}$ *is infeasible) Then*

$X_{i,g+1}=U_{i,g+1}$

Else

$X_{i,g+1}=X_{i,g}$

End if

End for

Step 7 ***Sort the particles using the three sorting rules***

Step 8 *Go to step 3 and repeat the loop until stopping criteria is reached.*

6.5 Results and Discussion

A set of 20 constrained benchmark problems is considered to evaluate the performance of the proposed ICPSO and ICDE. All the problems are nonlinear in nature i.e. either the objective function or the constraints or both have a nonlinear term in it. The mathematical models of the problems along with the optimal solution are given in Appendix II. A total of 25 runs for each experimental setting are conducted and the average fitness of the best solutions throughout the run is recorded. The population size is taken as 50 for both the algorithms.

For ICPSO, a linearly decreasing inertia weight is used which starts at 0.9 and ends at 0.4, with the user defined parameters $c_1= c_2=2.0$ and r_1, r_2 as uniformly distributed random numbers between 0 and 1. The proposed ICPSO algorithm is compared with two more variants of PSO namely ZRPSO (Zielinski and Rainer, 2006) and PESO (Angel et al, 2006).

For ICDE, the crossover rate is set as 0.9 the scaling factor F as already mentioned follows Rayleigh distribution. The proposed ICDE algorithm is also compared with two more variants of DE namely: ZRDE (Zielinski and Rainer, 2006a) and jDE-2 (Brest et al, 2006).

Several criteria are used to measure the performance of the proposed ICPSO and ICDE algorithms and to compare them with other versions available in the literature. In Tables 6.1 – 6.4 the performance of the proposed ICPSO is recorded in terms of best, worst and average fitness function value along with the standard deviation (Std) while increasing the NFE (number of function evaluations) to three different values 5×10^3, 5×10^4, 5×10^5. In Table 6.5 the performance of ICPSO is compared with ZRPSO and PESO for solving constrained optimization problems. Tables 6.6 – 6.10 shows the numerical results of proposed ICDE with respect to the above said performance measures.

Besides using the performance indices mentioned above, the following comparison criteria (Liang et al, 2006) are also used to analyze the performance of the algorithms considered in the present study. These criteria are:

Feasible Run: A run during which at least one feasible solution is found in maximum NFE.

Successful Run: A run during which the algorithm finds a feasible solution x satisfying (f(x) – f(x*) <= 0.0001·

Feasible Rate (FR) = (Number of feasible runs) / total runs

Success Rate (SR) = (Number of successful runs) / total runs

Success Performance (SP) = mean (FEs for successful runs) x (Number of total runs) / (Number of successful runs)

Average Feasibility Rate (AFR) = $\dfrac{\sum_{i=1}^{N}(FR)_i}{N}$

Average Success Rate (ASR) = $\dfrac{\sum_{i=1}^{N}(SR)_i}{N}$

Average Success Performance (ASP) = $\dfrac{\sum_{i=1}^{N}(SP)_i}{N}$

Where N stands for the number of problems (=20 in the present study).

6.5.1 Analysis of Numerical Results for ICPSO and ICDE

From Tables 6.1 – 6.4, it can be seen the performance of the proposed ICPSO improves with the increase in the number of function evaluations. This is quite an expected out come. However it can be said that 5 x 10^4 NFE is sufficient for reaching a good optimum solution which lies in the vicinity of the true optimum value, under the present parameter settings. Also, it can be seen that except for problem number 5 (g05), where the standard deviation (std) is 56.177, the std for all the remaining problems is quite low. This shows the consistency and stability of the proposed ICPSO algorithm. The superior performance of ICPSO is more visible from Table 6.5 where the results are recorded after fixing the accuracy at 0.0001. In this table it can be seen that the proposed ICPSO gave a better or at par performance with the other two algorithms. We will now take the comparison criteria one-by-one and discuss them briefly. The first criterion is that of a feasible run. A run is said to be feasible if at least one feasible solution is obtained in maximum number of function evaluations. According to this criterion all the algorithm gave 100% feasible rate for all the test problems except ICPSO which gave 96% feasible rate for test problem g17. However, from the observation of second criterion which is of successful run and is recorded when the algorithm finds a feasible solution satisfying the given accuracy (=0.0001 in the present study) it can be seen that the proposed ICPSO outperforms the other algorithms in all the test cases including g17. In g17, the percentage of success rate for ICPSO is 72, whereas the other algorithms were not able to reach the prescribed accuracy in any of the run. The third criterion is that of the success performance which depends on the feasibility rate and success rate, as described in the previous subsection. Here also ICPSO gave a better performance in comparison to the other two algorithms taken for comparison.

The performance of proposed ICDE algorithm is shown is Table 6.6 – 6.9 in terms of best, worst and mean fitness function values with standard deviation values for three different NFEs. Table 6.10 shows the number of function evaluations to achieve the fixed accuracy level (f(x) – f(x*) <=0.0001), where f(x*) is the true of the problem, feasible rate, success rate and success performance performed by ICDE algorithm for the given set of 20 constrained test problems. The proposed ICDE algorithm found at least one feasible solution for all the test problems except for the test problem g03. The success rate of ICDE is 100% for 16 test problems. For the remaining four test problems (g03, g11, g13, g17), ICDE was not able to give 100% the success rate, but in these cases also it gave more than 25% success rate except g03. The overall success rate for all 20 test problems is 87.7%. The proposed ICDE algorithm is also compared with two other variants of DE namely: ZRDE (Zielinski and Rainer, 2006a) and jDE-2 (Brest et al, 2006). From the numerical results of Table 6.10, it can be seen that the proposed ICDE gave a better or at par performance with the other two compared algorithms.

In comparison of ICDE and ICPSO algorithms with each other, it can be seen that both the algorithms gave more or less similar results in terms of the performance measures considered in the present study. Further, g03 was the only problem for which neither ICDE nor ICPSO were able to achieve any success rate.

Table 6.1 Results of ICPSO: Fitness function values achieved when NFE=5 x 10^3, NFE=5 x 10^4 and NFE=5 x 10^5 for problems g01 – g05

NFE		g01	g02	g03	g04	g05
5 x 10^3	Best	-12.7810	0.412234	-0.5123	-30665.5314	5126.2298
	Worst	-10.3994	0.354648	-0.2144	-30665.3480	5189.3433
	Mean	-11.3257	0.363072	-0.4231	-30665.3712	5165.7069
	Std	0.77603	0.021727	0.0393	0.228162	56.177
5 x 10^4	Best	-15	0.803138	-0.7181	-30665.5386	5126.4967
	Worst	-15	0.784856	-0.3990	-30665.5386	5126.4967
	Mean	-15	0.793258	-0.6495	-30665.5386	5126.4967
	Std	9.3e-09	0.00976	0.1294	1.02e-12	5.53e-05
5 x 10^5	Best	-15	0.803618	-0.8324	-30665.5386	5126.4967
	Worst	-15	0.794661	-0.4751	-30665.5386	5126.4967
	Mean	-15	0.803113	-0.7563	-30665.5386	5126.4967
	Std	1.5e-11	0.009781	0.0245	0.0000	2.40e-12

Table 6.2 Results of ICPSO: Fitness function values achieved when NFE=5 x 10^3, NFE=5 x 10^4 and NFE=5 x 10^5 for problems g06 – g10

NFE		g06	g07	g08	g09	g10
5 x 10^3	Best	-6961.8127	25.5805	-0.095826	680.6481	8207.3551
	Worst	-6939.9306	28.9778	-0.095826	681.1337	8399.2033
	Mean	-6958.7191	27.6499	-0.095826	680.7835	8344.4623
	Std	10.1347	0.9488	2.77e-18	0.1498	2.734
5 x 10^4	Best	-6961.8138	24.3062	-0.095826	680.6303	7049.2533
	Worst	-6961.8138	24.3118	-0.095826	681.0767	7049.2738
	Mean	-6961.8138	24.4006	-0.095826	680.6683	7049.2697
	Std	9.09e-13	0.01911	4.80e-19	0.089227	0.01456
5 x 10^5	Best	-6961.8138	24.3062	-0.095826	680.6301	7049.2480
	Worst	-6961.8138	24.3246	-0.095826	680.6435	7049.2480
	Mean	-6961.8138	24.3073	-0.095826	680.6329	7049.2480
	Std	1.98e-15	0.00402	0.0000	0.0525	1.81e-13

Table 6.3 Results of ICPSO: Fitness function values achieved when NFE=5 x 10^3, NFE=5 x 10^4 and NFE=5 x 10^5 for problems g11 – g15

NFE		g11	g12	g13	g14	g15
5 x 10^3	Best	0.7499	-1	0.4923	-44.4379	961.7302
	Worst	0.8539	-1	0.9997	-39.8987	962.0497
	Mean	0.8102	-1	0.8807	-42.1293	961.7565
	Std	0.0733	0.0000	0.1940	1.3022	1.9369
5 x 10^4	Best	0.7499	-1	0.3212	-47.6380	961.7150
	Worst	0.7499	-1	0.6389	-45.7222	962.6006
	Mean	0.7499	-1	0.4783	-46.2218	962.2491
	Std	2.22e-16	0.0000	0.1067	1.0495	0.8847
5 x 10^5	Best	0.7499	-1	0.0531	-47.7648	961.7150
	Worst	0.7499	-1	0.434	-47.7648	961.7150
	Mean	0.7499	-1	0.32736	-47.7648	961.7150
	Std	2.22e-16	0.0000	0.171017	4.71e-15	4.42e-13

Table 6.4 Results of ICPSO: Fitness function values achieved when NFE=5 x 10^3, NFE=5 x 10^4 and NFE=5 x 10^5 for problems g16 – g20

NFE		g16	g17	g18	g19	g20
5 x 10^3	Best	-1.9015	8967.5800	-0.6485	-5.50801	1.39331
	Worst	-1.8991	11028.714	-0.4833	-5.50801	1.39331
	Mean	-1.9001	9311.915	-0.5261	-5.50801	1.39331
	Std	0.00251	7.618	0.05928	2.348e-08	2.693e-08
5 x 10^4	Best	-1.9051	8868.7455	-0.8657	-5.50801	1.39331
	Worst	-1.9051	10903.986	-0.8644	-5.50801	1.39331
	Mean	-1.9051	9070.5204	-0.8650	-5.50801	1.39331
	Std	7.90e-16	4.8995	0.00066	8.580e-16	2.220e-16
5 x 10^5	Best	-1.9051	8853.5338	-0.8660	-5.50801	1.39331
	Worst	-1.9051	8853.5338	-0.8660	-5.50801	1.39331
	Mean	-1.9051	8853.5338	-0.8660	-5.50801	1.39331
	Std	8.05e-17	1.00e-12	1.06e-16	8.580e-16	2.220e-16

Table 6.5 Comparative Results of ICPSO with ZRPSO and PESO: NFE to achieve the fixed accuracy level ((f(x) – f(x*)) <= 0.0001), success rate, Feasible Rate and Success Performance for problems g01 – g20

Problem	Algorithm	Best	Worst	Mean	Feasible Rate (%)	Success Rate (%)	Success Performance
g01	ICPSO	25250	55250	29796	100	100	29796
	ZRPSO	25273	346801	76195	100	52	146530
	PESO	95100	106900	101532	100	100	101532
g02	ICPSO	81800	135750	115850	100	100	115850
	ZRPSO	-	-	-	100	0	-
	PESO	180000	327900	231193	100	56	412844.3878
g03	ICPSO	-	-	-	100	0	-
	ZRPSO	-	-	-	100	0	-
	PESO	450100	454000	450644	100	100	450644
g04	ICPSO	7750	12650	9568	100	100	9568
	ZRPSO	15363	25776	20546	100	100	20546
	PESO	74300	85000	79876	100	100	79876
g05	ICPSO	13350	65400	19286	100	100	19286
	ZRPSO	94156	482411	364218	100	16	2276363
	PESO	450100	457200	452256	100	100	452256
g06	ICPSO	7300	9600	8252	100	100	8252
	ZRPSO	16794	22274	20043	100	100	20043
	PESO	47800	61100	56508	100	100	56508
g07	ICPSO	29050	57800	40046	100	100	40046
	ZRPSO	315906	338659	327283	100	8	4091031
	PESO	198600	444100	352592	100	96	367282.9861
g08	ICPSO	1050	1350	1158	100	100	1158
	ZRPSO	1395	3921	2360	100	100	2360
	PESO	2800	8400	6124	100	100	6124
g09	ICPSO	10450	29550	16248	100	100	16248
	ZRPSO	45342	84152	58129	100	100	58129
	PESO	77000	129000	97544	100	100	97544
g10	ICPSO	66050	84900	75920	100	100	75920
	ZRPSO	290367	486655	426560	100	32	1332999
	PESO	398000	475600	452575	100	16	2828593.75

Table 6.5 Contd...

Problem	Algorithm	Best	Worst	Mean	Feasible Rate (%)	Success Rate (%)	Success Performance
g11	ICPSO	1650	24250	13630	100	100	13630
	ZRPSO	5475	21795	16386	100	100	16386
	PESO	450100	450100	450100	100	100	450100
g12	ICPSO	850	1100	976	100	100	976
	ZRPSO	1409	9289	4893	100	100	4893
	PESO	3300	10900	8088	100	100	8088
g13	ICPSO	88700	111100	102512	100	16	640700
	ZRPSO	-	-	-	100	0	-
	PESO	450100	453200	450420	100	100	450420
g14	ICPSO	21250	339550	50614	100	100	50614
	ZRPSO	-	-	-	100	0	-
	PESO	-	-	-	100	0	-
g15	ICPSO	7400	128100	54306	100	100	54306
	ZRPSO	17857	348138	176827	100	80	221033
	PESO	450100	450100	450100	100	100	450100
g16	ICPSO	7100	10650	8732	100	100	8732
	ZRPSO	24907	51924	33335	100	100	33335
	PESO	43400	53900	49040	100	100	49040
g17	ICPSO	256800	463350	408506	96	72	567369
	ZRPSO	-	-	-	100	0	-
	PESO	-	-	-	100	0	-
g18	ICPSO	53600	89900	71694	100	100	71694
	ZRPSO	85571	455907	191220	100	80	239026
	PESO	120800	394900	214322	100	92	232958.4121
g19	ICPSO	2200	2850	2600	100	100	2600
	ZRPSO	9833	18382	13791	100	100	13791
	PESO	14600	22700	19980	100	100	19980
g20	ICPSO	3000	3450	3263.33	100	100	3263.33
	ZRPSO	NA	NA	NA	NA	NA	NA
	PESO	NA	NA	NA	NA	NA	NA

Table 6.6 Results of ICDE: Fitness function values achieved when NFE=5 x 10^3, NFE=5 x 10^4 and NFE=5 x 10^5 for problems g01 – g05

FES		g01	g02	g03	g04	g05
5 x 10^3	Best	-12.912	0.533948	0.097667	-30665.5	5126.62
	Worst	-10.6738	0.393898	-0.001257	-30653.8	5325.05
	Mean	-11.6328	0.45796	-0.053420	-30658.1	5269.37
	Std	0.753357	0.036574	0.11039	12.9959	1.7625
5 x 10^4	Best	-15	0.803366	-0.395644	-30665.5	5126.498
	Worst	-14.9827	0.785104	-0.025106	-30665.5	5126.498
	Mean	-14.9988	0.795961	-0.11585	-30665.5	5126.498
	Std	0.004318	0.009150	0.109245	0.232389	1.5367e-02
5 x 10^5	Best	-15	0.803618	-0.525045	-30665.5	5126.498
	Worst	-15	0.803602	-0.239068	-30665.5	5126.498
	Mean	-15	0.803611	-0.34022	-30665.5	5126.498
	Std	0.00000	0.0020312	0.118131	1.40132e-06	1.194e-05

Table 6.7 Results of ICDE: Fitness function values achieved when NFE=5 x 10^3, NFE=5 x 10^4 and NFE=5 x 10^5 for problems g06 – g10

NFE		g06	g07	g08	g09	g10
5 x 10^3	Best	-6961.81	28.7617	-0.0958259	680.67	7421.73
	Worst	-5783.19	35.0193	-0.0958259	682.35	8231.56
	Mean	-6643.96	31.2817	-0.0958259	680.881	7890.08
	Std	71.2902	2.42755	2.888e-17	0.398674	288.741
5 x 10^4	Best	-6961.81	24.3062	-0.0958259	680.63	7049.3
	Worst	-6961.81	24.3066	-0.0958259	680.63	7066.38
	Mean	-6961.81	24.3064	-0.0958259	680.63	7052.41
	Std	1.2125e-04	0.000831	1.8619e-17	2.807e-07	82.1743
5 x 10^5	Best	-6961.81	24.3062	-0.0958259	680.63	7049.25
	Worst	-6961.81	24.3062	-0.0958259	680.63	7049.25
	Mean	-6961.81	24.3062	-0.0958259	680.63	7049.25
	Std	1.27e-07	2.726e-12	2.775e-17	8.56e-016	0.000493

Table 6.8 Results of ICDE: Fitness function values achieved when NFE=5 x 10^3, NFE=5 x 10^4 and NFE=5 x 10^5 for problems g11 – g15

NFE		g11	g12	g13	g14	g15
5 x 10^3	Best	0.750673	-1	0.492094	-45.5992	961.723
	Worst	0.928549	-1	0.997157	-39.1403	963.337
	Mean	0.86372	-1	0.78893	-41.2485	962.457
	Std	0.06879	0.0000	0.170059	2.30428	0.684965
5 x 10^4	Best	0.7499	-1	0.053123	-47.6919	961.715
	Worst	0.750062	-1	0.434013	-46.8961	962.288
	Mean	0.749906	-1	0.329706	-47.2343	961.753
	Std	0.000103	0.0000	0.146536	0.366103	1.45398
5 x 10^5	Best	0.7499	-1	0.053122	-47.7649	961.715
	Worst	0.7499	-1	0.434008	-47.7649	961.715
	Mean	0.7499	-1	0.332439	-47.7649	961.715
	Std	1.110e-16	0.0000	0.168434	9.53e-15	3.497e-13

Table 6.9 Results of ICDE: Fitness function values achieved when NFE=5 x 10^3, NFE=5 x 10^4 and NFE=5 x 10^5 for problems g16 – g20

NFE		g16	g17	g18	g19	g20
5 x 10^3	Best	-1.90357	8890.67	-0.77478	-5.50801	1.39331
	Worst	-1.89976	9818.1	-0.619384	-5.38881	1.39331
	Mean	-1.90023	8942.23	-0.735716	-5.50006	1.39331
	Std	0.0023671	7.7282	0.0462197	0.029734	3.05e-05
5 x 10^4	Best	-1.90516	8877.24	-0.866025	-5.50801	1.39331
	Worst	-1.90516	8968.57	-0.866025	-5.50801	1.39331
	Mean	-1.90516	8940.43	-0.866025	-5.50801	1.39331
	Std	8.347e-16	7.63093	5.416e-07	1.36e-05	1.09e-06
5 x 10^5	Best	-1.90516	8853.53	-0.866025	-5.50801	1.39331
	Worst	-1.90516	8927.59	-0.866025	-5.50801	1.39331
	Mean	-1.90516	8894.44	-0.866025	-5.50801	1.39331
	Std	9.064e-16	5.60309	1.251e-09	1.29e-12	5.23e-07

Table 6.10 Comparative Results of ICDE with ZRDE and jDE-2: NFE to achieve the fixed accuracy level ((f(x) – f(x*)) <= 0.0001), success rate, Feasible Rate and Success Performance for problems g01 – g20

Problem	Algorithm	Best	Worst	Mean	Feasible Rate (%)	Success Rate (%)	Success Performance
g01	ICDE	25350	62600	32466	100	100	32466
	ZRDE	30511	38028	33414	100	100	33414
	jDE-2	46559	56968	50386	100	100	50386
g02	ICDE	66850	124900	111503	100	100	111503
	ZRDE	95501	129363	113298	100	84	134879
	jDE-2	101201	173964	123490	100	92	145899
g03	ICDE	-	-	-	100	0	-
	ZRDE	-	-	-	100	0	-
	jDE-2	-	-	-	100	0	-
g04	ICDE	6750	11300	9836.67	100	100	9836.67
	ZRDE	14048	18362	15986	100	100	15986
	jDE-2	38288	42880	40728	100	100	40728
g05	ICDE	13450	29550	18460	100	100	18460
	ZRDE	16994	204151	107076	100	100	107076
	jDE-2	133340	482304	206620	100	68	446839
g06	ICDE	5900	15200	13933.3	100	100	13933.3
	ZRDE	6147	7995	7143	100	100	7143
	jDE-2	26830	31299	29488	100	100	29488
g07	ICDE	32700	43450	38550	100	100	38550
	ZRDE	84889	104026	93793	100	100	93793
	jDE-2	114899	141847	127740	100	100	127740
g08	ICDE	758	1400	1346.67	100	100	1346.67
	ZRDE	831	1337	1086	100	100	1086
	jDE-2	1567	4485	3236.4	100	100	3236
g09	ICDE	13150	296600	35373.3	100	100	35373.3
	ZRDE	23828	27424	25805	100	100	25805
	jDE-2	49118	58230	54919	100	100	54919
g10	ICDE	59450	216900	168727	100	100	168727
	ZRDE	105673	132270	119217	100	100	119217
	jDE-2	139095	165498	146150	100	100	146150

Table 6.10 Contd...

Problem	Algorithm	Best	Worst	Mean	Feasible Rate (%)	Success Rate (%)	Success Performance
g11	ICDE	1950	37400	29030	100	67	43545
	ZRDE	1384	24356	13380	100	100	13380
	jDE-2	17834	432169	49700	100	96	53928
g12	ICDE	850	1200	1083.33	100	100	1083.33
	ZRDE	342	7307	5104	100	100	5104
	jDE-2	1820	9693	6355.6	100	100	6356
g13	ICDE	46150	71000	60150	100	27	225562
	ZRDE	242289	288226	265703	100	32	830322
	jDE-2	-	-	-	100	0	-
g14	ICDE	89400	111900	104337	100	100	104337
	ZRDE	57727	81392	68226	100	100	68226
	jDE-2	88954	107951	97845	100	100	97845
g15	ICDE	18450	90200	82653.3	100	100	82653.3
	ZRDE	7151	137487	57968	100	100	57968
	jDE-2	51321	432766	222460	100	96	241383
g16	ICDE	7650	9500	8546.67	100	100	8546.67
	ZRDE	9837	12619	11592	100	100	11592
	jDE-2	28230	34182	31695	100	100	31695
g17	ICDE	222400	448950	347327	100	60	578878
	ZRDE	201798	328448	265692	100	20	1328459
	jDE-2	449306	449306	179710	100	4	11232650
g18	ICDE	16300	21600	19443.3	100	100	19443.3
	ZRDE	70290	96334	79557	100	100	79557
	jDE-2	91049	142674	104460	100	100	104462
g19	ICDE	2450	4250	3146.67	100	100	3146.67
	ZRDE	2514	3356	3024	100	100	3024
	jDE-2	7587	11550	10196	100	100	10196
g20	ICDE	2050	2550	2410	100	100	2410
	ZRDE	NA	NA	NA	NA	NA	NA
	jDE-2	NA	NA	NA	NA	NA	NA

6.6 Conclusion

This chapter presented a new constraint handling mechanism for solving constrained optimization problems. It is a simple approach for handing constraints and no need of additional parameters. Based on the new constraint handling mechanism, two algorithms were proposed namely ICPSO and ICDE. The Improved Constraint Particle Swarm Optimization (ICPSO) algorithm was initialized using quasi random Vander Corput sequence and differs from unconstrained PSO algorithm in the phase of updating the position vectors and sorting every generation solutions. Similarly, the Improved Constraint DE (ICDE) algorithm differs from unconstrained DE algorithm only in the phase of selection of particles to the next generation and sorting the final results. Also the proposed ICDE uses dynamic scaling factor following Rayleigh distribution. The performance of ICPSO and ICDE algorithms are validated on twenty constrained benchmark problems and compared with two other variants of PSO and DE in the literature. From the empirical analysis it can be seen that both the proposed algorithms gave an average feasibility rate of 100%. In terms of success rate, ICPSO gave an ASR of 89% while ICDE gave an ASR of 87%. Finally in terms of success performance, ICPSO gave an ASP of 91053, while ICDE gave an ASP of 78936.

Chapter 7

In-Situ Efficiency Determination of Induction Motor

[This chapter investigates the performance of PSO, DE and their proposed variants with the real life problem namely In-situ efficiency determination of Induction Motor (5hp). By the application of PSO and DE algorithms in this problem, the motor efficiency can be obtained without performing no-load test, which is not easily possible for the motors working in process industries where continuous operation is required. Results are compared with Genetic Algorithm and a physical efficiency measurement method, called torque-gauge method. The performances in terms of objective function (error in the efficiency) and convergence time prove the effectiveness of the optimization algorithms used for comparison in this chapter.]

7.1 Introduction

Induction motors (IMs) have become the most widely used machines in any industry. These motors consume around 70% of the electricity used. Indian electricity tariff, on which electricity and other public utility rates are highly dependent, are increasing and hence many industrial consumers have migrated away from the grid. Also, process industries are found to be energy intensive (4% of energy cost in the total input cost in case of textile industry) compared to other industries like chemical, food, computer manufacturing etc., (Palanichamy et al, 2001). Therefore, a small increment in the efficiency of these motors can result in substantial saving in the long period. Since the Induction Motors operate at part load in industries is unavoidable, there exist a large opportunity for energy savings by implementing efficient controller with Adjustable Speed Drives (ASD). Adoption of energy conservation in these motors by providing ASD or replacing it by energy efficient motors is highly depends on the savings and payback periods. In this situation, accurate in-situ efficiency determination in these motors is essential but it requires the motor's electrical parameters.

Many non-linear programming techniques like the Newton-Raphson technique, cyclic method, Hook and Jeeves and Rosenbrock methods have been applied to parameter estimation and hence efficiency determination of IM. The optimum determined by the Newton-Raphson technique depends heavily on the initial guess of the parameter, with the possibility of a slightly different initial value causing the algorithm to converge to an entirely different solution (Nangsue et al, 1999). Also this algorithm needs derivative during the optimization process, which may be difficult to calculate. Bounekhla et al (2005) have proved Rosenbrock method is better than scatter search and Hook and Jeeves methods in terms of fast and efficient search. Apart from conventional methods, some of the evolutionary techniques like GA (Pillay et al, 1998), Genetic Programming (Nangsue et al, 1999), PSO (Benaidja and Khenfer, 2006), DE (Ursem and Vadstrup, 2003), Evolution Strategy (Ursem and Vadstrup, 2004), and a variant of PSO based on diversity (Ursem and Vadstrup, 2004) have also been successfully applied to Induction Motor parameter (electrical and mechanical) estimation. In this chapter, PSO, QPSO and DE and their variants *(ATREPSO, GMPSO, SMPSO1, LDE1 and DE-QI)* are applied to the in-situ efficiency determination of Induction Motor. A wide comparison is performed on the results obtained from these algorithms along with GA (Pillay et al, 1998), and torque-gauge method (Ontario Hydro Report, 1990).

The rest of the chapter is organized as follows: Section 7.2 explains the standards for Induction Motor efficiency determination. In section 7.3, mathematical model of the in-situ efficiency determination is given; section 7.4 gives the method of solution and result discussion. Finally the chapter concludes with section 7.5.

7.2 Standards for Induction Motor Efficiency Determination

The methods for efficiency measurements can roughly be divided into two categories: direct and indirect methods. In direct method, shaft torque is directly measured and calculate the efficiency by using the ratio of motor output power to the input power. But the preferred method of determining efficiency in three-phase Induction Motor is the summation-of-losses method. The IEEE 112 (USA) (IEEE standard, 2004), IEC 34-2 (European) (IEC standard, 1972) are the international standards and the Indian standard IS 325: sub rule IS 4029 (Indian standard, 2002), represent the most important references for the three-phase Induction Motor efficiency measurements. These standards recommend different measurement procedures, in particular for the stray losses determination and the temperature corrections of the copper losses.

The IEEE 112-2004 (revised of IEEE 112-1996) (IEEE standard, 2004) consists of five basic methods to determine the efficiency: A, B, C, E and F. In method A, the input and output power is measured and the efficiency is directly obtained. This method is only used for very small machines. Method B employs a direct method to obtain the stray load losses. It is not a direct method for determining the motor efficiency.

Method B is the recommended method for testing of induction machines up to 180 kW. Method C is a back to back machine test. The total stray load losses are also obtained via a separation of losses for both motor and generator. The stray load losses are then divided between the motor and generator proportional to the rotor currents. Method E and E_1 are indirect methods; the output power is not measured. In method E the stray load losses are directly measured using the reverse rotation test. In method E_1, the stray load losses are set to an assumed value. From an experimentation (Thangaraj et al, 2007), IS 325 is similar to IEEE 112-E.

In method F and F_1, the equivalent circuit of the machine is used and offer more advantageous when test under load are not feasible or not desirable (Pillay et al, 1998). The stray load losses are again directly measured or in the case of F_1 an assumed value is used. There also exist some additional methods, e.g. the use of the equivalent circuit but calibrated at a load point. These methods are less suitable field measurements (in-situ motor) because they require no-load land locked rotor results. The method developed by Otaduy (1996) requires only speed measurement and nameplate data to construct an equivalent circuit with electrical parameters.

7.3 In-Situ Efficiency Determination

The method to be described in this chapter considers IEEE E_1 and F_1 methods of efficiency determination. The procedure followed in this chapter is same as of Pillay et al (1998) but PSO, DE and their variants are used to solve the algebraic equations instead of GA. Because, PSO and DE techniques have become very popular since last decade to solve multi dimension non linear programming problems due to its less complexity, fast convergence, etc., than Genetic Algorithm and Evolutionary Programming. The general block diagram of in-situ efficiency determination of Induction Motor using optimization algorithms is shown in Figure 7.1.

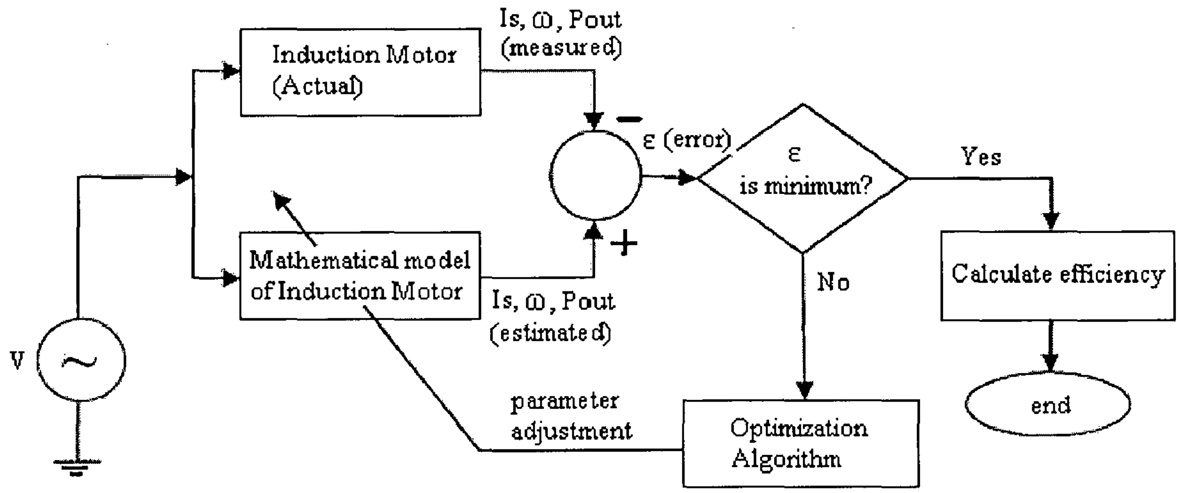

Figure 7.1 Block diagram of Induction Motor in-situ efficiency determination

First, the stator line resistance is measured after shutting down the motor. 5HP, 4 pole Induction Motor considered as test motor. Summation of losses method of efficiency determination is used with the assumption of stray load losses. The winding arrangement of a star connected motor is shown in Figure 7.2 and the resistance per phase is calculated as in Eqn. (7.1).

$$r_1 = \frac{r_{1line}}{2} \tag{7.1}$$

where r_{1line} is stator line resistance

r_1 is stator phase resistance

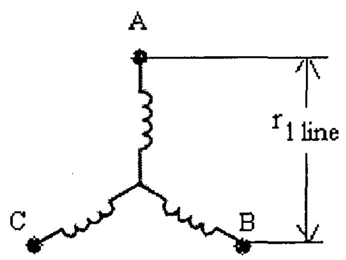

Figure 7.2 Winding arrangement of star connected motor

Before entering into the development of algorithm some measurements are required to find the equivalent circuit parameters which are: stator line to line voltage V_1, stator current I_1, input power P_{inp} and rpm at difference load points. Then, power factor can be calculated as in Eqn. (7.2). The measured and calculated values of the test motor for a wide range of loads and the equivalent circuits considered in this chapter are taken from (Pillay et al, 1998) and are shown in Table 7.1 and Figures 7.3 and 7.4 respectively.

$$pf = \frac{P_{inp}}{\sqrt{3}V_1 I_1} \tag{7.2}$$

Table 7.1 Measured data of the test motor

% load	voltage V, Volts	current I, Ampere	input power Pinp, Watts	power factor, pf	Speed, rpm
25	460	2.7	381	0.177	1794
50	460	4	2047	0.642	1764
75	460	5.3	3272	0.775	1741
100	460	6.7	4326	0.81	1719

Two variations were performed in these circuits from the conventional exact equivalent circuit for comparison that are: (i) inclusion of stray load loss resistance $r_{st,}$ shown in Figure (7.3) and (ii) parallel connection of X_m, r_m is transformed into series connection as X'_m, r'_m, shown in Figure (7.4), and its calculation can be performed as in Eqn. (7.3).

$$r'_m = \frac{r_m x_m^2}{r_m^2 + x_m^2};\; x'_m = \frac{r_m^2 x_m}{r_m^2 + x_m^2} \tag{7.3}$$

where X_m is mutual inductance and r_m is core loss component.

Next step is to find the stray load losses at any load point from its assumed value at full load as per the recommendation in IEEE standards section 5.7.4 (IEEE standard, 2004). The recommended value of stray load losses for different capacity motors is shown in Table 7.2. In the present study, we have considered this loss at full load is 1.8% and its calculation at different load point is shown in Eqn. (7.4). The remaining Eqns. (7.5) - (7.20) of Induction Motor which is involved in the present study are as follows (Pillay et al, 1998).

$$P_{st} = P_{st\,fl} \frac{I_2^{\,2}}{I_{2fl}^2} \tag{7.4}$$

where P_{st}, P_{stfl} are stray load losses at any point and full load respectively; I_2, I_{2fl} are rotor currents at these load points.

The stray load loss resistance r_{st} is

$$r_{st} = 0.018 r_2 \frac{(1 - s_{fl})}{s_{fl}} \tag{7.5}$$

where s_{fl} slip of the motor under full load.

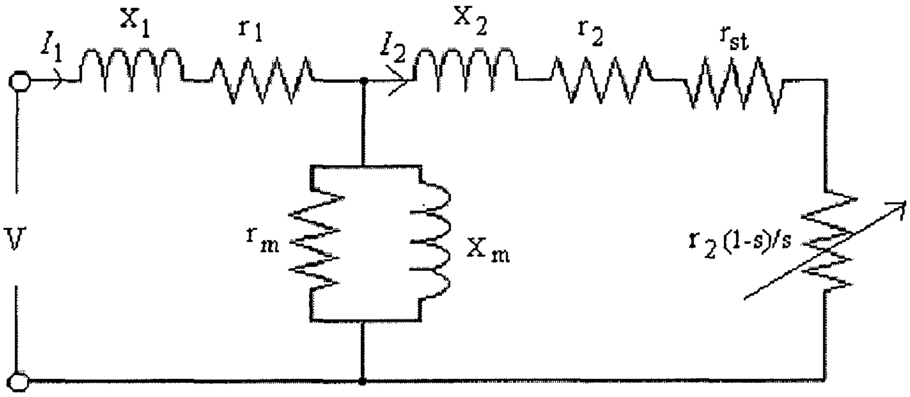

Figure 7.3 Equivalent circuit of Induction Motor with stray loss resistance

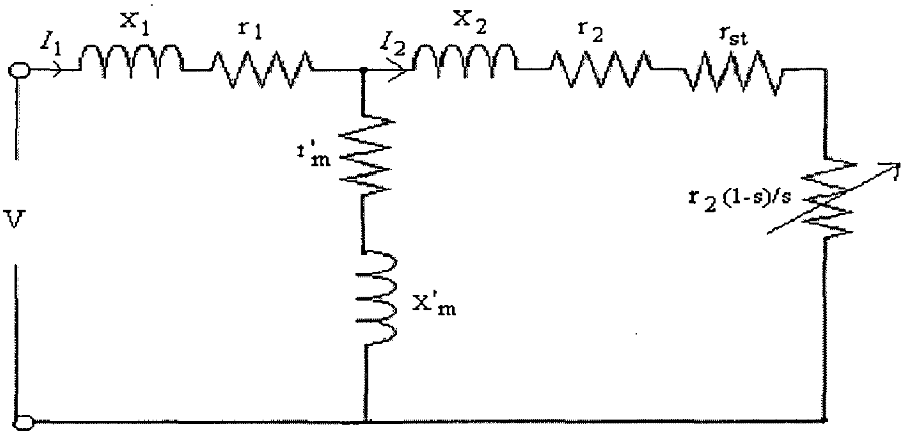

Figure 7.4 Equivalent circuit of Induction Motor (Modified)

Table 7.2 Stray load losses for the different capacity motors

Motor rated power	Stray load losses relative to the output power
0.75 – 90 kW	1.8 %
91 – 375 kW	1.5 %
376 – 1800 kW	1.2 %
1800 kW and higher	0.9 %

Temperatures of stator and rotor windings are assumed to be the same and calculated as in Eqn. (7.6) with the IEEE recommended reference temperature.

$$T_t = \frac{I_1 - I_0}{I_{fl} - I_0}(T_r - T_s) + T_s \quad (7.6)$$

where I_1, I_{fl} are the measured and nameplate stator currents

I_0 is the stator current under no-load DC test

T_r = 75°C is the reference temperature for the insulation system of class A (IEEE standard, 2004)

T_s = 25°C is the ambient temperature.

Eqns. (7.7), (7.8) are used to correct the stator and rotor resistances to the test temperature

$$r_{1c} = \frac{r_1(T_t + k_c)}{T_s + k_c} \quad (7.7)$$

$$r_{2c} = \frac{r_2(T_t + k_a)}{T_s + k_a} \quad (7.8)$$

where r_1 is the stator resistance measured during DC test.

r_2 is the assumed rotor resistance.

The complex admittances of the branches of the equivalent circuits of Figures 7.3 and 7.4 are given below (Pillay et al, 1998).

$$\overline{Y}_2 = \frac{1}{r_{2c}/s + r_{st} + jx_2} \quad (7.9)$$

For the equivalent circuit Figure 7.3: $\overline{Y}_m = \frac{-j}{x_m} + \frac{1}{r_m}$ (7.10)

For the equivalent circuit Figure 7.4: $\overline{Y}_m = \dfrac{1}{r'_m + jx'_m}$ (7.11)

$$\overline{Y}_1 = \frac{1}{r_{1c} + jx_1} \quad (7.12)$$

The stator current can be estimated as

$$I_{1est} = |\bar{I}_1| = \left|\frac{\overline{V}_1\overline{Y}_1(\overline{Y}_2 + \overline{Y}_m)}{\overline{Y}_1 + \overline{Y}_2 + \overline{Y}_m}\right| \quad (7.13)$$

where $\overline{V}_1 = V_1 / \sqrt{3}$

The power factor and rotor current can be estimated as

$$pf_{est} = \frac{\Re(\bar{I}_1)}{I_{1est}};\ I_2 = \left|\frac{\overline{V}_1\overline{Y}_1\overline{Y}_2}{\overline{Y}_1 + \overline{Y}_2 + \overline{Y}_m}\right| \quad (7.14)$$

The current through r_m for the circuit of Figure 7.3: $I_m = \left|\dfrac{\overline{V}_1\overline{Y}_1}{r_m(\overline{Y}_1 + \overline{Y}_2 + \overline{Y}_m)}\right|$ (7.15)

The current through r_m for the circuit of Figure 7.4: $I_m = \left|\dfrac{\overline{V}_1\overline{Y}_1\overline{Y}_m}{\overline{Y}_1 + \overline{Y}_2 + \overline{Y}_m}\right|$ (7.16)

The input power of the circuit of Figure 7.3 can be estimated as

$$P_{inpest} = 3(I_1{}^2 r_{1c} + I_2{}^2(r_{2c} / s + r_{st}) + I_m{}^2 r_m) \quad (7.17)$$

The input power of the circuit of Figure 7.4 can be estimated as

$$P_{inpest} = 3(I_1{}^2 r_{1c} + I_2{}^2(r_{2c} / s + r_{st}) + I_m{}^2 r'_m) \quad (7.18)$$

The output power can be estimated as

$$P_{outest} = 3I_2{}^2 r_{2c}\frac{1-s}{s} \quad (7.19)$$

The efficiency can be estimated as

$$\eta = \frac{P_{outest}}{P_{\inf est}}100\% \quad (7.20)$$

The goal of optimization algorithms is to minimize the errors between the measured and calculated parameters. Four methods are considered in this study. They are:

Method 1: The objective function is,

$$Maximize\ ff_1 = \frac{1}{f_1^2 + f_2^2},$$

where $f_1 = (I_{1est} - I_1)100 / I_1$ and $f_2 = (P_{\inf est} - P_{\inf})100 / P_{\inf}$

Method 2: The objective function included stator current, input power and power factor, which is: $Maximize\ ff_2 = \frac{1}{f_1^2 + f_2^2 + f_3^2}$

Where f_1, f_2 are as same as in objection function ff_1 and $f_3 = (pf_{est} - pf)100 / pf$

Method 3: The objective function included stator current, input power and output power, which is: $Maximize\ ff_3 = \frac{1}{f_1^2 + f_2^2 + f_4^2}$

Where f_1, f_2 are as same as in objection function ff_1 and $f_4 = (P_{outest} - P_{out})100 / P_{out}$

Method 4: The objective function included four input parameters. They are stator current, input power, power factor and output power. The objective function is:

$$Maximize\ ff_4 = \frac{1}{f_1^2 + f_2^2 + f_3^2 + f_4^2}$$

The unknown variables of the above objective functions are x_1, r_2, x_m (or x_m') and r_m (or r_m'). The optimization algorithms are used to determine the above said unknown variables. The assigned parameters of the given motors are: $K_a = 225$, $K_c = 234.5$, $r_1 = 1.635$ ohm.

7.4 Method of Solution and Discussion of Results

For solving the above optimization model, basic PSO, DE and some of their variants discussed in the previous chapters viz. QPSO, ATREPSO, GMPSO, SMPSO1, LDE1, DE-QI are used. These algorithms gave slightly better results than the other algorithms developed in this thesis for this particular problem. For comparison, the results of Genetic Algorithm (Pillay et al, 1998) and tarque-gauge method (Ontario Hydro Report, 1990) are used.

Parameter settings of PSO and DE algorithms:

The main parameters of PSO algorithm are inertia weight w and acceleration constants c_1 and c_2. For all the PSO algorithms, the inertia weight w is taken to be linearly decreasing from 0.9 to 0.5 and the acceleration constants c_1 and c_2 are taken as 2.0. For QPSO algorithm, the parameter β is linearly decreased from 1.0 to 0.5. The main parameters of DE are crossover rate Cr and scaling factor F, which are taken as 0.2 and 0.5 respectively for the entire DE algorithms. Besides these settings a total of 30 runs for each experimental setting were conducted and the average efficiency was recorded.

Result Analysis:

The comparison result of all algorithms corresponding to Figure 7.3 is given in Table 7.3 – 7.10. Table 7.11 – 7.14 shows the comparison results of given algorithms corresponding to Figure 7.4. Performance curves of algorithms are given in Figures 7.5 – 7.11. Figure 7.12 – 7.15 shows the comparison among the optimization algorithms at 25% load for Figure 7.4.

In the comparison among the four methods, the results indicate that the methods corresponding to Figure 7.4 (which is considered the equivalent circuit with a series connection of x'_m, r'_m) gave better results than the one with parallel connection in term of error in the efficiency estimation. The addition of fourth input (method 3), nameplate output power, to the input parameters of the algorithm helps to achieve minimum convergence time and number of function evaluations (NFE) at all the loads. But, no positive effect of this variable in the error minimization. The standard deviation (SD) of method 3 is $6.08e^{-7}$, which proves its superiority whereas SD value in method 1 is $4.65e^{-5}$. The performance of method 4, which is also considered f_4 (output nameplate power) in addition with current, input power and power factor as input parameter of the algorithm, is poor than method 3 in terms of convergence time and NFE.

The results from method 3 and 4 indicate that the effect of third (power factor) input parameter in the performance of the algorithms is almost negligible. But its contribution in method 2 is significant to achieve good performances in comparison with method 1. Hence, the parameters namely, current, input power, speed and output power are sufficient to determine the motor parameters quickly and if the knowledge of output power is not available, power factor should be considered in addition with the first two inputs to get better performance.

In the comparison among the algorithms in terms of error, the results indicate that LDE1 outperformed the other algorithms at 100% load. ATREPSO and DE-QI algorithms produced better results at 75% and 50% loads respectively. At 25% load, SMPSO1 and ATREPSO are outperformed the other algorithms. If we compare the algorithms in terms of convergence speed, then the results indicate that DE-QI offered better performance than all the other algorithms at 100% and 75% loads. SMPSO1 at 50% load and at 25% load, almost all the algorithms performed well in term of speed of convergence.

Over all, it can be said that method 2 which is solved by ATREPSO, SMPSO1, LDE1 and DE-QI algorithms gave better results in many cases in terms of accuracy in efficiency estimation. The method 3 solved by the algorithms SMPSO1 and DE-QI offered very good results in terms of convergence time at all the load points.

Table 7.3 Comparison results of algorithms in terms of efficiency and error: Objective function ff_1 of Figure 7.3

Algorithm	25%		50%		75%		100%	
	Efficiency	Error	Efficiency	Error	Efficiency	Error	Efficiency	Error
PSO	61.5622	13.7622	89.374	5.074	87.7029	2.9029	85.5525	2.0525
SMPSO1	48.2914	0.4914	89.2255	4.9255	87.7029	2.9029	85.522	2.022
GMPSO	47.818	0.018	89.1885	4.8885	87.7027	2.9027	85.5286	2.0286
ATREPSO	57.4311	9.6311	89.1885	4.8885	87.7023	2.9023	85.5266	2.0266
QPSO	55.8678	8.0678	89.0964	4.7964	87.681	2.881	85.5479	2.0479
DE	56.9297	9.1297	89.0775	4.7775	87.7051	2.9051	85.5337	2.0337
LDE1	54.9057	7.1057	89.0764	4.7764	87.6568	2.8568	85.4922	1.9922
DE-QI	55.4705	7.6705	89.0741	4.7741	87.7023	2.9023	85.5241	2.0241
GA	68.99	21.19	89.44	5.14	86.68	1.88	85.86	2.36
TG	47.8		84.3		84.8		83.5	

Table 7.4 Comparison results of algorithms in terms of efficiency and error: Objective function ff_2 of Figure 7.3

Algorithm	25%		50%		75%		100%	
	Efficiency	Error	Efficiency	Error	Efficiency	Error	Efficiency	Error
PSO	60.0291	12.2291	89.393	5.093	87.7468	2.9468	85.5198	2.0198
SMPSO1	57.6565	9.8565	89.3888	5.0888	87.7125	2.9125	85.5181	2.0181
GMPSO	58.3305	10.5305	89.3791	5.0791	87.7062	2.9062	85.5174	2.0174
ATREPSO	59.9916	12.1916	89.3659	5.0659	87.7108	2.9108	85.5166	2.0166
QPSO	60.5208	12.7208	89.263	4.963	87.7503	2.9503	85.5122	2.0122
DE	58.4146	10.6146	89.2604	4.9604	87.7106	2.9106	85.5169	2.0169
LDE1	52.2689	4.4689	89.1821	4.8821	87.687	2.887	85.4965	1.9965
DE-QI	55.0697	7.2697	89.2566	4.9566	87.7103	2.9103	85.5117	2.0117
GA	65.47	17.67	87.04	2.74	85.87	1.07	84.77	1.27
TG	47.8		84.3		84.8		83.5	

Table 7.5 Comparison results of algorithms in terms of efficiency and error: Objective function ff_3 of Figure 7.3

Algorithm	25%		50%		75%		100%	
	Efficiency	Error	Efficiency	Error	Efficiency	Error	Efficiency	Error
PSO	64.6357	16.8357	88.9229	4.6229	87.7134	2.9134	85.9546	2.4546
SMPSO1	54.3489	6.5489	88.8516	4.5516	87.4601	2.6601	85.9497	2.4497
GMPSO	61.2841	13.4841	88.7275	4.4275	87.4898	2.6898	85.947	2.447
ATREPSO	61.772	13.972	88.6545	4.3545	87.626	2.826	85.9731	2.4731
QPSO	63.5507	15.7507	89.0644	4.7644	87.4738	2.6738	85.9489	2.4489
DE	66.1704	18.3704	89.121	4.821	87.4895	2.6895	85.9515	2.4515
LDE1	64.2549	16.4549	88.7976	4.4976	87.4413	2.6413	85.9426	2.4426
DE-QI	60.2366	12.4366	88.9832	4.6832	87.3902	2.5902	85.942	2.442
GA	71.24	23.44	88.88	4.58	88.04	3.24	86.36	2.86
TG	47.8		84.3		84.8		83.5	

Table 7.6 Comparison results of algorithms in terms of efficiency and error: Objective function ff_4 of Figure 7.3

Algorithm	25%		50%		75%		100%	
	Efficiency	Error	Efficiency	Error	Efficiency	Error	Efficiency	Error
PSO	55.4919	7.6919	88.4656	4.1656	87.6683	2.8683	85.997	2.497
SMPSO1	55.4919	7.6919	88.1818	3.8818	87.4651	2.6651	85.9256	2.4256
GMPSO	57.2915	9.4915	88.2892	3.9892	87.6405	2.8405	85.9753	2.4753
ATREPSO	57.2962	9.4962	88.2231	3.9231	87.5096	2.7096	85.9435	2.4435
QPSO	55.7984	7.9984	88.9104	4.6104	87.4671	2.6671	85.9165	2.4165
DE	55.3947	7.5947	88.8762	4.5762	87.5096	2.7096	85.9461	2.4461
LDE1	53.7438	5.9438	88.725	4.425	87.4778	2.6778	85.9091	2.4091
DE-QI	57.0967	9.2967	88.0563	3.7563	87.4972	2.6972	85.9447	2.4447
GA	71.66	23.86	88.85	4.55	87.91	3.11	86.15	2.65
TG	47.8		84.3		84.8		83.5	

Table 7.7 Comparison results of algorithms in terms of NFE and time (sec): Objective function ff_1 of Figure 7.3

Algorithm	25%		50%		75%		100%	
	NFE	Time	NFE	Time	NFE	Time	NFE	Time
PSO	15030	1.7	8007	0.9	5985	0.7	4521	0.6
SMPSO1	15030	1.8	7161	0.8	5823	0.7	4491	0.5
GMPSO	15030	1.8	6357	0.8	4752	0.6	4062	0.5
ATREPSO	15030	2.0	6357	0.9	4644	0.6	4140	0.5
QPSO	15030	1.8	5704	0.8	4329	0.5	4161	0.4
DE	15030	1.7	4104	0.4	2679	0.2	2751	0.2
LDE1	15030	1.8	2949	0.3	2370	0.2	2346	0.2
DE-QI	15030	1.9	2808	0.4	2085	0.2	2172	0.2
GA	NA	NA	NA	NA	NA	NA	NA	NA

Table 7.8 Comparison results of algorithms in terms of NFE and time (sec): Objective function ff_2 of Figure 7.3

Algorithm	25%		50%		75%		100%	
	NFE	Time	NFE	Time	NFE	Time	NFE	Time
PSO	93	0.01	6921	0.8	5958	0.7	5010	0.6
SMPSO1	93	0.01	6921	0.7	6246	0.6	4890	0.6
GMPSO	74	0.01	5463	0.7	4842	0.7	4209	0.6
ATREPSO	80	0.01	5484	0.7	4902	0.7	4368	0.6
QPSO	288	0.1	5431	0.6	4392	0.5	4167	0.5
DE	324	0.1	4248	0.5	2973	0.3	3024	0.3
LDE1	117	0.01	2835	0.3	2337	0.3	2661	0.4
DE-QI	303	0.1	2817	0.3	2079	0.3	2319	0.3
GA	NA	NA	NA	NA	NA	NA	NA	NA

Table 7.9 Comparison results of algorithms in terms of NFE and time (sec): Objective function ff_3 of Figure 7.3

Algorithm	25%		50%		75%		100%	
	NFE	Time	NFE	Time	NFE	Time	NFE	Time
PSO	87	0.01	858	0.2	1938	0.3	3660	0.4
SMPSO1	87	0.01	750	0.1	1886	0.3	3510	0.4
GMPSO	87	0.01	789	0.1	1746	0.2	3246	0.4
ATREPSO	87	0.01	825	0.2	1731	0.2	3414	0.5
QPSO	93	0.01	792	0.1	1448	0.2	2059	0.4
DE	165	0.01	1194	0.2	1710	0.3	2403	0.3
LDE1	126	0.01	927	0.2	1557	0.2	2130	0.3
DE-QI	135	0.01	915	0.1	1353	0.1	1947	0.2
GA	NA	NA	NA	NA	NA	NA	NA	NA

Table 7.10 Comparison results of algorithms in terms of NFE and time (sec): Objective function ff_4 of Figure 7.3

Algorithm	25%		50%		75%		100%	
	NFE	Time	NFE	Time	NFE	Time	NFE	Time
PSO	60	0.01	798	0.1	2082	0.3	3702	0.4
SMPSO1	60	0.01	600	0.1	1800	0.3	3420	0.4
GMPSO	60	0.01	741	0.1	1719	0.2	3507	0.4
ATREPSO	60	0.01	720	0.1	1800	0.3	3570	0.5
QPSO	60	0.01	898	0.1	1765	0.3	2954	0.4
DE	60	0.01	1077	0.1	1749	0.3	2739	0.3
LDE1	60	0.01	849	0.2	1692	0.3	2115	0.3
DE-QI	60	0.01	939	0.1	1461	0.2	2010	0.3
GA	NA	NA	NA	NA	NA	NA	NA	NA

Table 7.11 Comparison results of algorithms in terms of efficiency and error: Objective function ff_1 of Figure 7.4

Algorithm	25%		50%		75%		100%	
	Efficiency	error	Efficiency	error	Efficiency	error	Efficiency	error
PSO	52.8327	5.0327	87.3836	3.0836	87.2989	2.4989	85.6585	2.1585
SMPSO1	50.3145	2.5145	87.3403	3.0403	86.6205	1.8205	85.3841	1.8841
GMPSO	47.866	0.066	86.4765	2.1765	86.9479	2.1479	85.5267	2.0267
ATREPSO	57.8555	10.0555	85.9917	1.6917	86.468	1.668	85.5712	2.0712
QPSO	64.3479	16.5479	86.5772	2.2772	86.2004	1.4004	84.8355	1.3355
DE	64.7172	16.9172	83.4023	-0.8977	86.3019	1.5019	84.7322	1.2322
LDE1	59.011	11.211	83.8205	-0.4795	84.9311	0.1311	83.5694	0.0694
DE-QI	56.8356	9.0356	84.2825	-0.0175	86.2282	1.4282	84.2291	0.7291
GA	56.38	8.58	83.51	-0.79	85.59	0.79	83.18	-0.32
TG	47.8	0	84.3	0	84.8	0	83.5	0

Table 7.12 Comparison results of algorithms in terms of efficiency and error: Objective function ff_2 of Figure 7.4

Algorithm	25%		50%		75%		100%	
	Efficiency	error	Efficiency	error	Efficiency	error	Efficiency	error
SPSO	58.0905	10.2905	85.3179	1.0179	84.8403	0.0403	83.8705	0.3705
SMPSO	48.6996	0.8996	85.7073	1.4073	84.2272	-0.5728	82.9782	-0.5218
GMPSO	53.8224	6.0224	84.8432	0.5432	85.9331	1.1331	83.2451	-0.2549
ATREPSO	50.5116	2.7116	84.8611	0.5611	85.9443	1.1443	83.8515	0.3515
QPSO	52.2799	4.4799	84.8585	0.5585	85.1353	0.3353	82.7034	-0.7966
DE	56.5057	8.7057	85.5052	1.2052	86.6821	1.8821	84.3514	0.8514
LXDE	54.7454	6.9454	84.1863	-0.1137	86.6216	1.8216	84.7473	1.2473
DE-QI	54.4074	6.6074	85.6781	1.3781	85.95	1.15	84.7072	1.2072
GA	58.77	10.97	86.93	2.63	85.74	0.94	82.91	-0.59
TG	47.8	0	84.3	0	84.8	0	83.5	0

Table 7.13 Comparison results of algorithms in terms of efficiency and error: Objective function ff_3 of Figure 7.4

Algorithm	25%		50%		75%		100%	
	Efficiency	error	Efficiency	error	Efficiency	error	Efficiency	error
SPSO	54.3575	6.5575	90.2422	5.9422	89.3893	4.5893	86.1882	2.6882
SMPSO	50.8532	3.0532	88.7778	4.4778	87.2944	2.4944	86.1882	2.6882
GMPSO	51.6536	3.8536	88.7165	4.4165	87.2927	2.4927	86.1852	2.6852
ATREPSO	57.9641	10.1641	88.2677	3.9677	87.3953	2.5953	86.1882	2.6882
QPSO	57.9816	10.1816	90.5271	6.2271	89.4017	4.6017	86.1898	2.6898
DE	62.6156	14.8156	90.6982	6.3982	89.3959	4.5959	86.1923	2.6923
LXDE	53.3793	5.5793	89.5805	5.2805	87.3838	2.5838	85.9544	2.4544
DE-QI	58.0465	10.2465	88.5345	4.2345	87.3911	2.5911	86.1886	2.6886
GA	72.37	24.57	88.87	4.57	87.95	3.15	86.13	2.63
TG	47.8		84.3		84.8		83.5	

Table 7.14 Comparison results of algorithms in terms of efficiency and error: Objective function ff_4 of Figure 7.4

Algorithm	25%		50%		75%		100%	
	Efficiency	error	Efficiency	error	Efficiency	error	Efficiency	error
SPSO	50.7735	2.9735	88.3947	4.0947	87.3534	2.5534	85.2512	1.7512
SMPSO	50.1228	2.3228	88.1714	3.8714	87.1219	2.3219	84.4474	0.9474
GMPSO	51.4569	3.6569	88.2749	3.9749	84.7294	-0.0706	83.9022	0.4022
ATREPSO	51.4188	3.6188	88.1275	3.8275	84.8454	0.0454	82.8624	-0.6376
QPSO	59.0119	11.2119	88.7229	4.4229	85.7053	0.9053	83.6247	0.1247
DE	59.3771	11.5771	86.4578	2.1578	86.6514	1.8514	84.2524	0.7524
LXDE	59.2987	11.4987	86.2706	1.9706	85.6621	0.8621	84.1691	0.6691
DE-QI	55.763	7.963	86.4489	2.1489	86.3441	1.5441	84.061	0.561
GA	72.92	25.12	89.02	4.72	88.04	3.24	86.19	2.69
TG	47.8	0	84.3	0	84.8	0	83.5	0

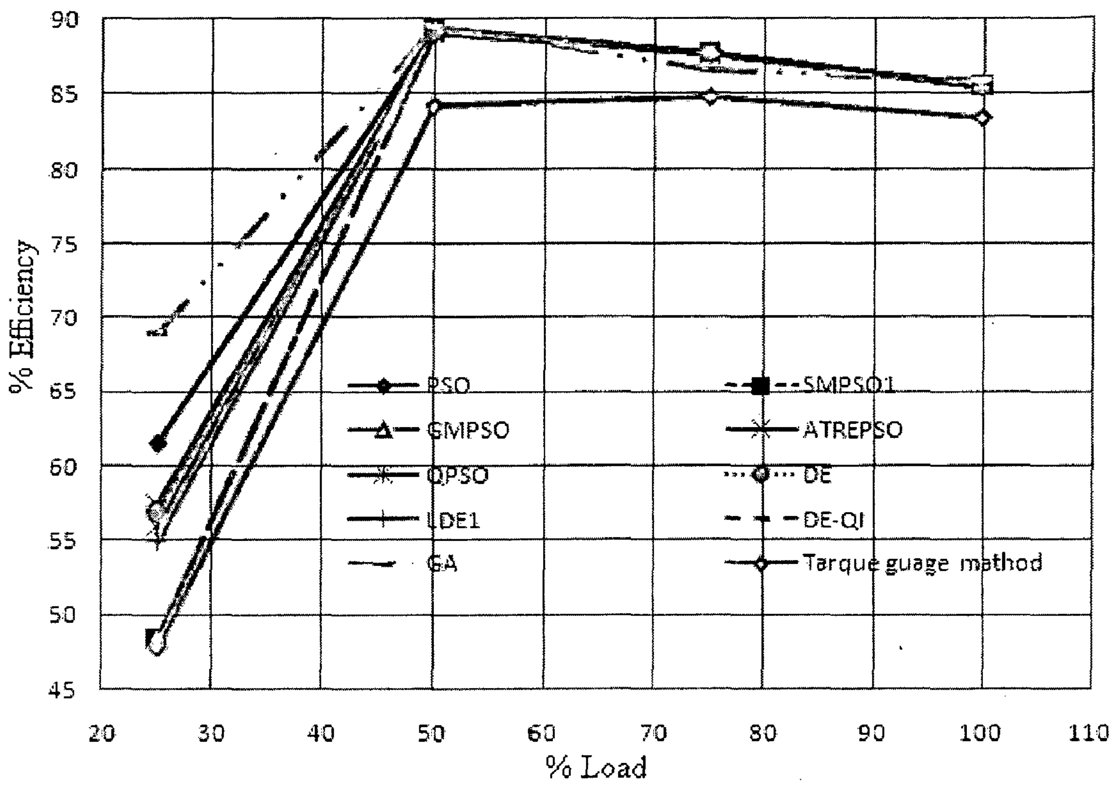

Figure 7.5 Performance curves of algorithms using objective function ff_1 of Figure 7.3

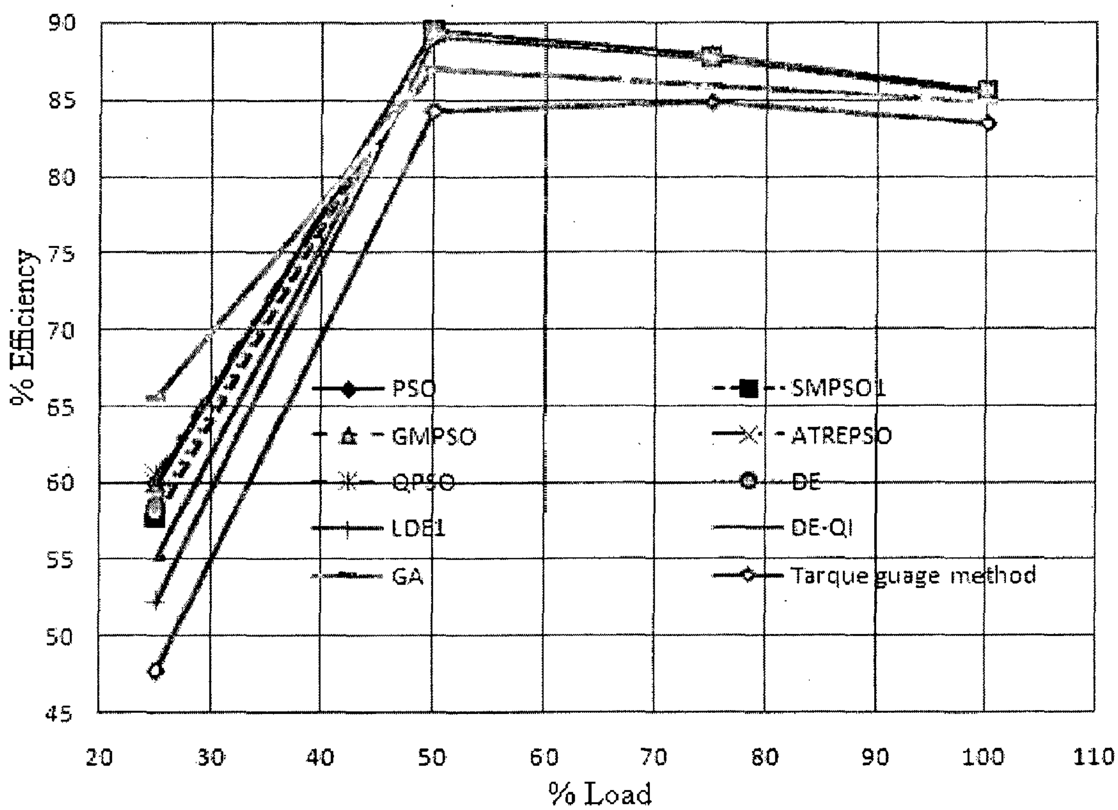

Figure 7.6 Performance curves of algorithms using objective function ff_2 of Figure 7.3

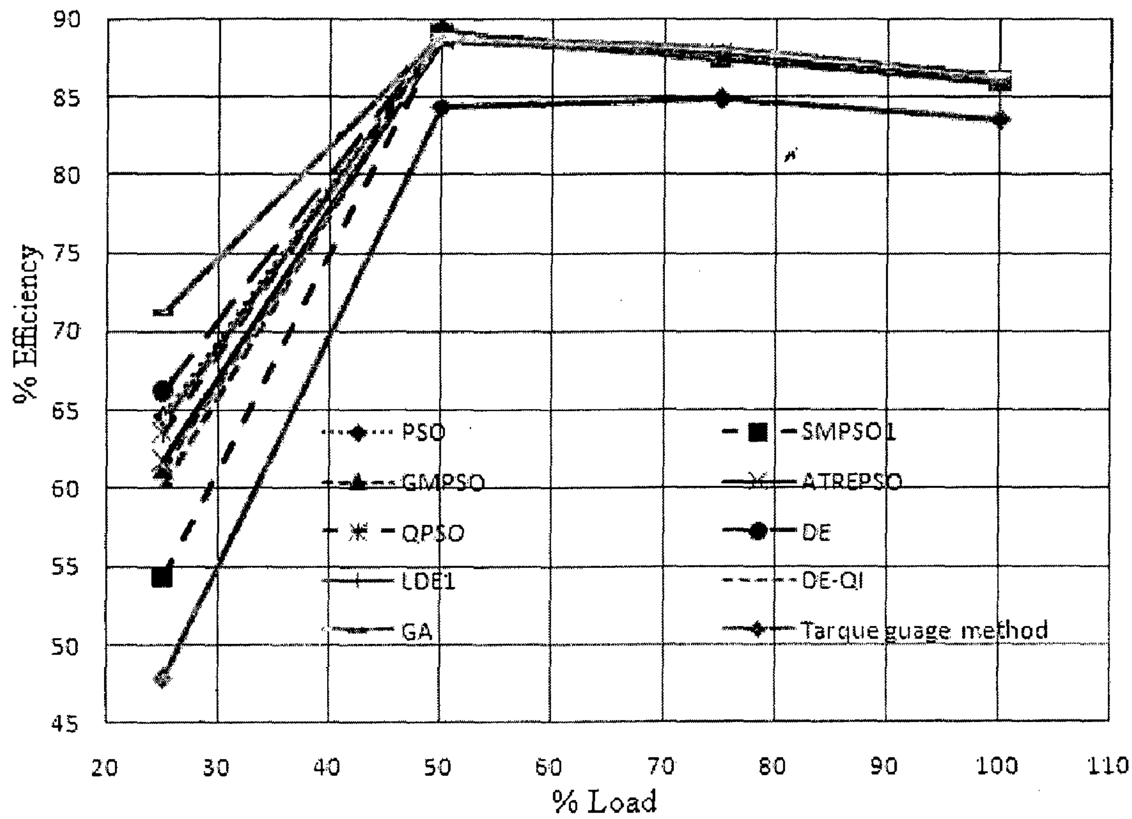

Figure 7.7 Performance curves of algorithms using objective function ff_3 of Figure 7.3

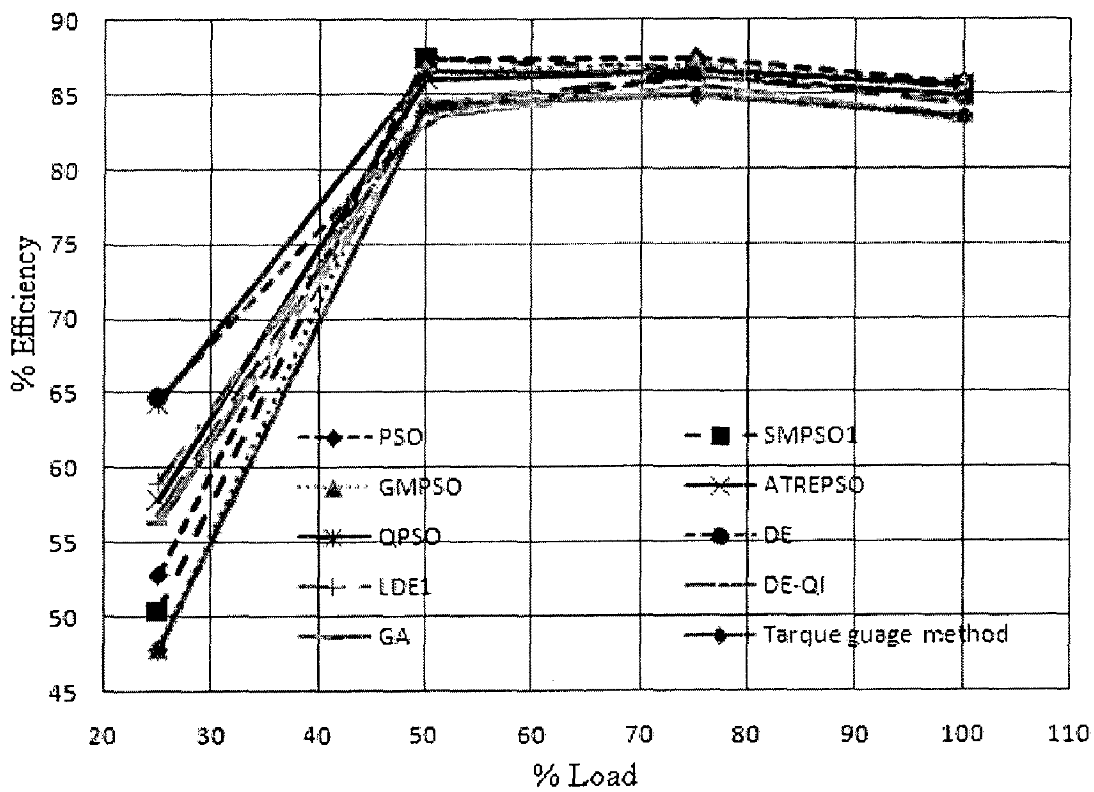

Figure 7.8 Performance curves of algorithms using objective function ff_1 of Figure 7.4

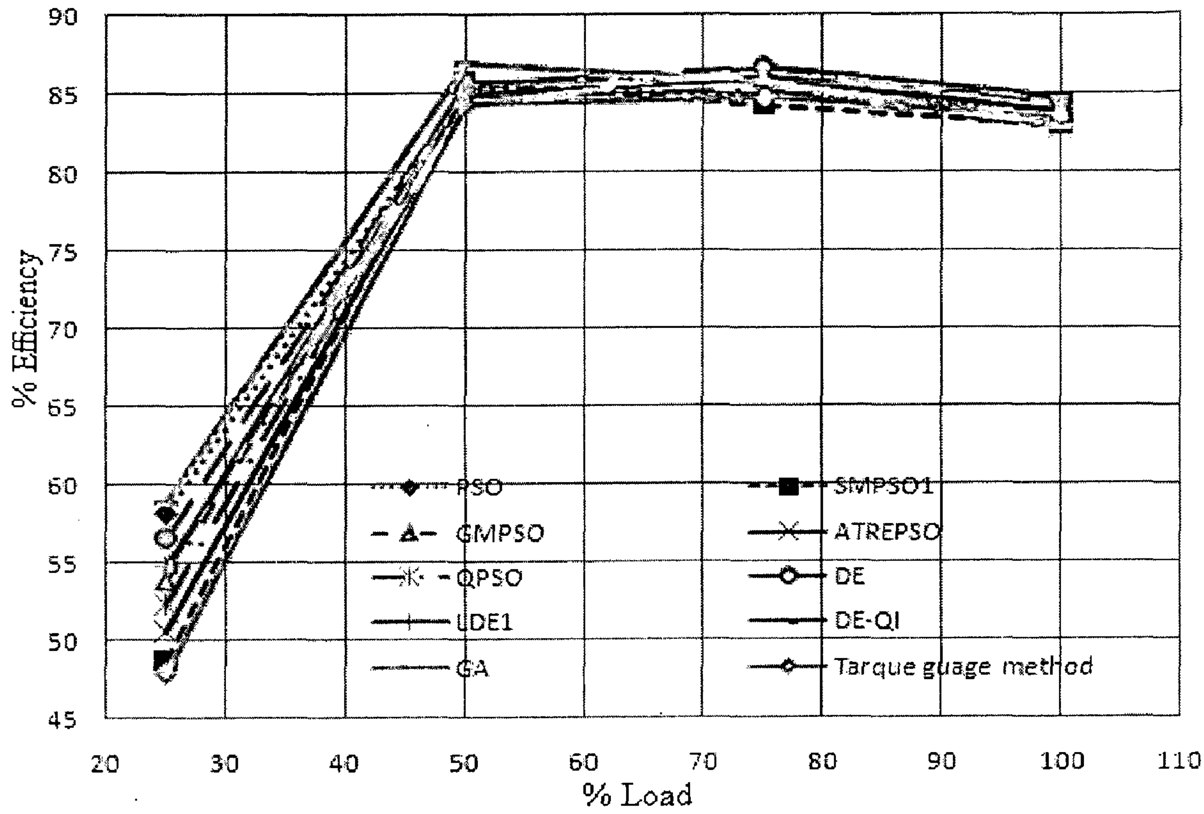

Figure 7.9 Performance curves of algorithms using objective function ff_2 of Figure 7.4

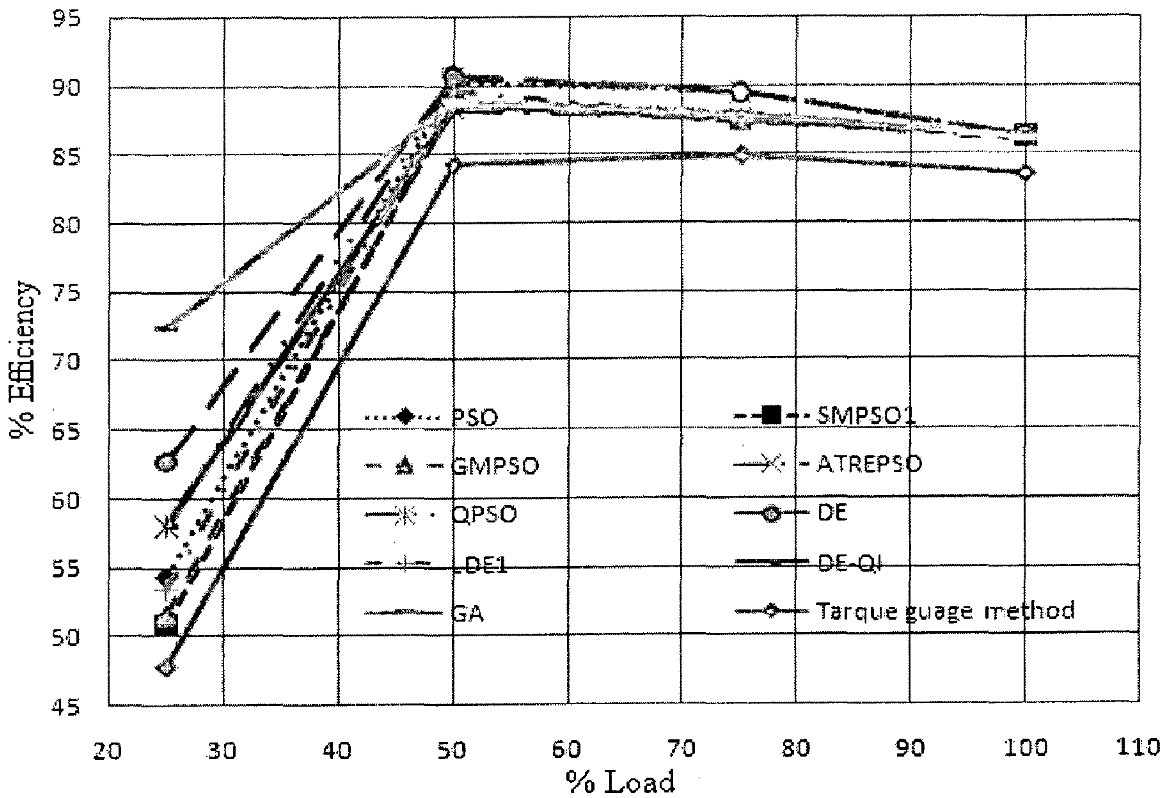

Figure 7.10 Performance curves of algorithms using objective function ff_3 of Figure 7.4

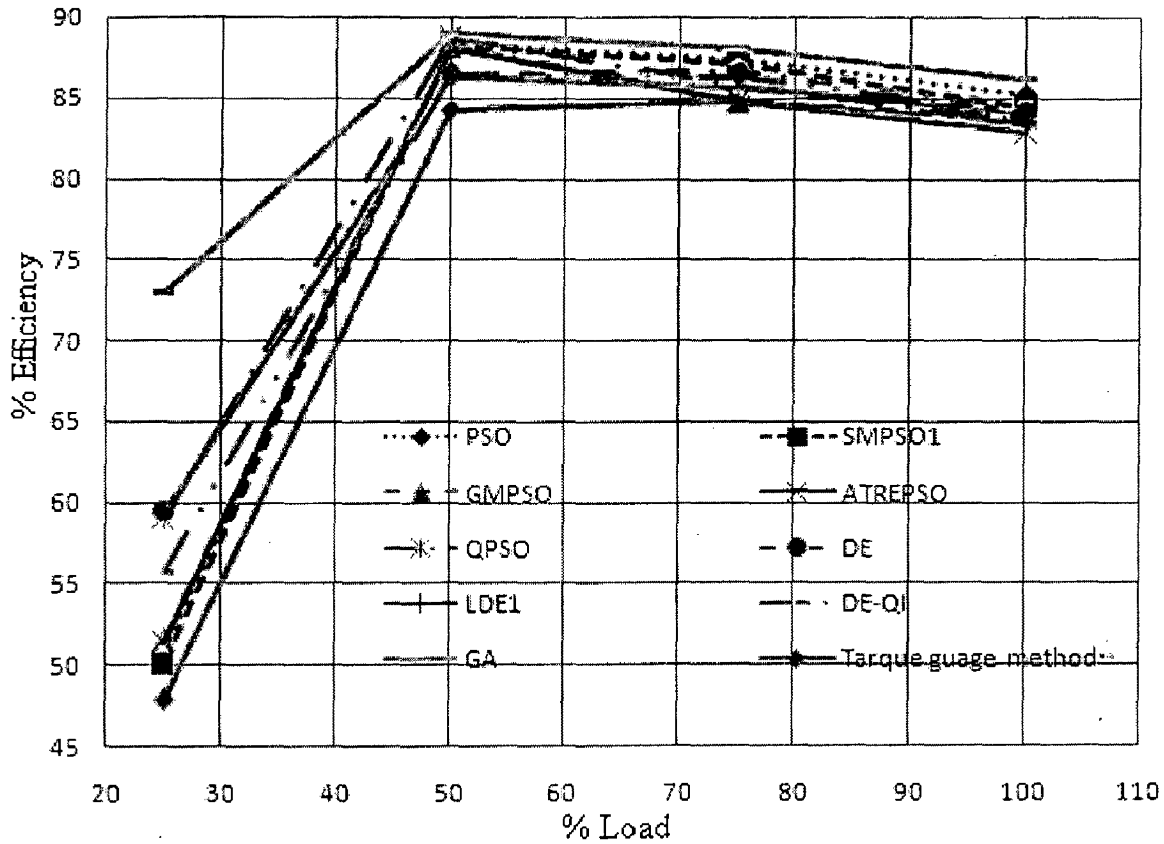

Figure 7.11 Performance curves of algorithms using objective function ff_4 of Figure 7.4

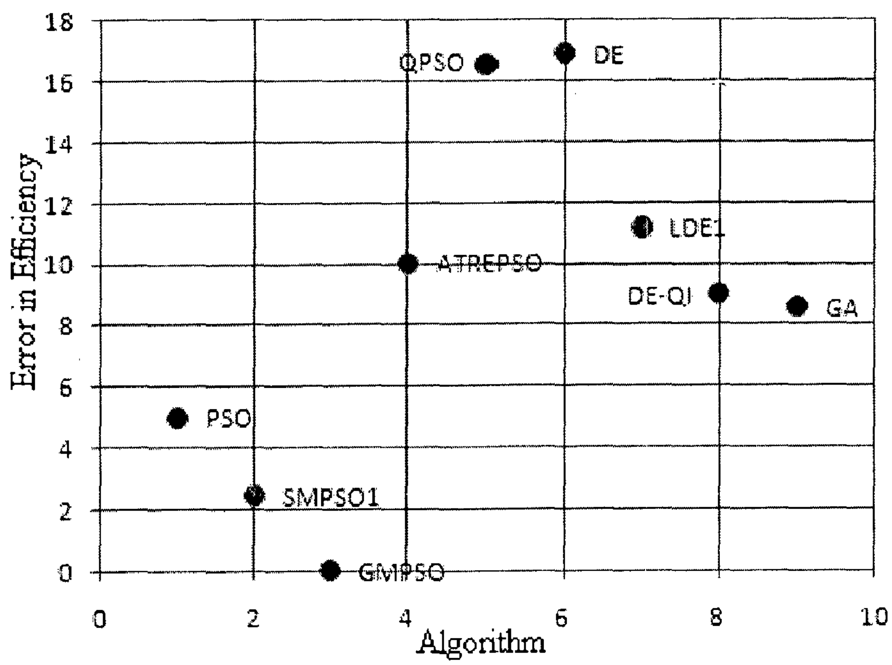

Figure 7.12 Comparison of algorithms for objective function ff_1 at 25% load corresponding to Figure 7.4

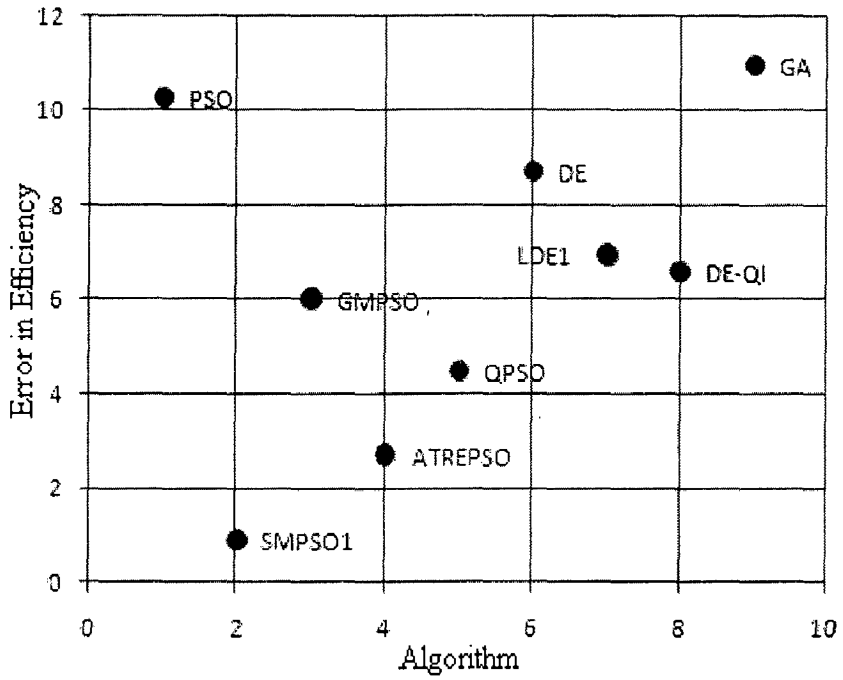

Figure 7.13 Comparison of algorithms for objective function ff_2 at 25% load corresponding to Figure 7.4

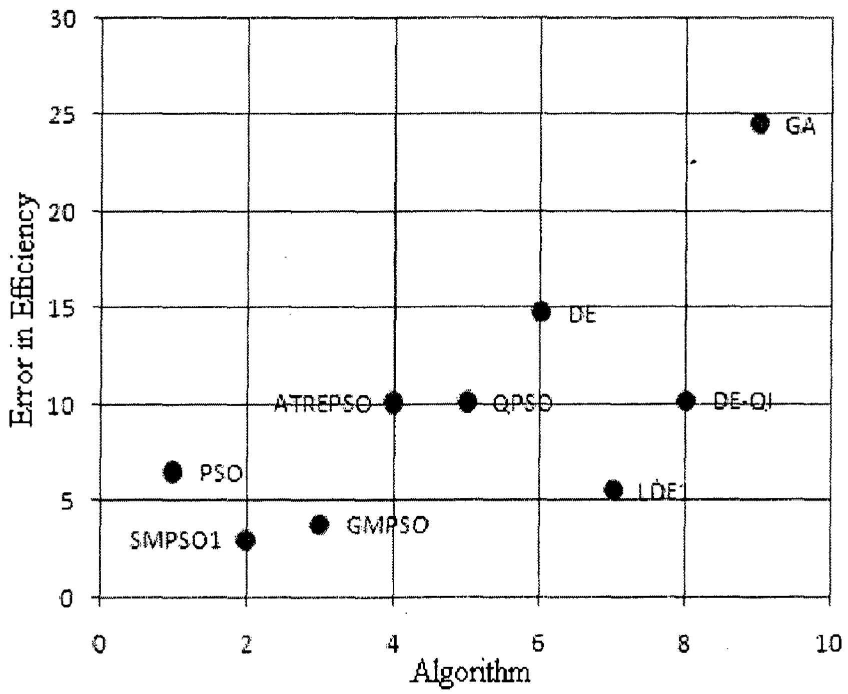

Figure 7.14 Comparison of algorithms for objective function ff_3 at 25% load corresponding to Figure 7.4

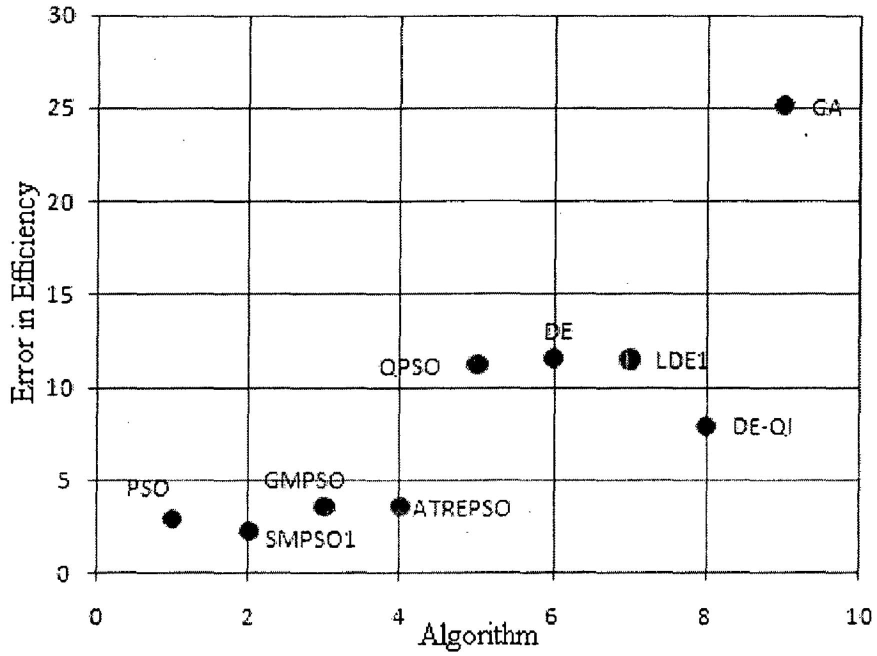

Figure 7.15 Comparison of algorithms for objective function ff_4 at 25% load corresponding to Figure 7.4

7.5 Conclusion

In this chapter, a comparison was made among PSO, QPSO, DE and their variants (five improved versions) with Genetic Algorithm and torque-gauge methods for in-situ efficiency determination of an induction motor through its parameter identification. This problem was framed by four different methods. The differences in the method were based on the number of input parameters used to the optimization algorithms and modifications in the equivalent circuit of the motor. All the algorithms have proven their numerical stability and their robustness towards error minimization in a short time.

In summary, ATREPSO, SMPSO1, LDE1 and DE-QI outperformed the other algorithms in many cases in terms of efficiency evaluation with minimum error. In case of speed of convergence, mutation based variants of PSO and DE were the winners in all the cases. The influence of output power as an input parameter of the algorithm is significant. Finally, induction motor in-situ efficiency can be accurately calculated by using input power, current, speed and output power as the input parameters of optimization algorithms. Modification in the equivalent circuit of the induction motor helped to estimate the efficiency with high accuracy.

Chapter 8

Optimization of Over-current Relay Settings in Electric Power Systems

---

[This chapter presents the model of Directional Over-current Relay settings in Electrical Power Systems, modeled as a constrained nonlinear optimization problem. The optimization problem corresponding to IEEE 3-bus, IEEE 4-bus and IEEE 6-bus system is considered. The six DE algorithms namely: LDE1, LDE2, LDE3, LDE4, LDE5 and DE-QI discussed in chapter 4 are used to solve the resulting optimization problem. For handling constraints, the mechanism described in Chapter 6 is used.]

8.1 Introduction

Electrical power system operates at various voltage levels from 415 V to 400kV or even more. This system can be divided into three parts: generation, transmission and utilization (load). Among these three, transmission of power is carried out by the electrical conductors, called transmission lines, placed in open. Therefore such lines more frequently undergo abnormalities than other parts in their life time due to various reasons: like faults (which create over-current), over load, over-voltage, under-frequency etc. One well known source for occurrence of over-voltage in such lines is lightning. These abnormalities cause interruption of the supply and may damage the equipments connected to the system, arising the need for protection. Over-current relay is the most commonly used protection scheme in the power system to protect the system from various faults.

Directional over-current relays (DOCRs) are good technical and economic alternative for the protection of interconnected subtransmission systems and secondary protection of transmission systems (Urdaneta et al, 1997). These relays are provided in electrical power systems to isolate only the fault lines in the event of the faults in the system. Relay is a logical element and issues a trip signal to the circuit breaker if a fault occurs within the relay jurisdiction and are placed at both ends of each line. Their coordination is an important aspect of the protection system design. Relay coordination problem is to determine the sequence of relay operations for each possible fault location so that faulted section is isolated, with sufficient coordination margins, without excessive time delays. This sequence selection is a function of power network topology, relay characteristics, and protection philosophy (Birla et al, 2006).

In DOCR protection scheme, two types of settings: current, which is referred as Plug Setting, and Time Dial Setting must be calculated. Optimization of these settings (main objective in this chapter) results in efficient coordination of relays that can be achieved and isolate the faulty transmission line, thus maintaining continuity of supply to healthy sections of the power systems. The above stated problem of coordinating each DOCR with one another in electrical power system is modelled as a non-linear constrained optimization problem. The two settings (PS and TDS) of each relay are considered as decision variables. Sum of the operating times of all the primary relays, which are expected to operate in order to clear the faults of their corresponding zones, is considered as objective function and the constraints of this problem are bounds on all decision variables, complexly interrelated times of the various relays (called selectivity constraints) and restrictions on each term of the objective function to be between certain limits.

The rest of the chapter is organized as follows: In section 8.2 literature review of the problem is given. The DOCR problem formulation is given in section 8.3. The general model of the DOCR coordination problem is stated in section 8.4. The optimization problem corresponding to IEEE 3-bus, IEEE 4-bus and IEEE 6-bus system are given in section 8.5, section 8.6 and section 8.7 respectively. The method of solution and discussion of results are given in section 8.8. Finally this chapter concludes with section 8.9.

8.2 Review on Previous Work

Several optimization techniques have been applied for coordinating directional over-current relays. Before applying optimization theory in these problems, trial and error approach was used but it has a well known drawback that slow rate of convergence due to the large number of iteration needed to reach a suitable relay setting. To overcome such disadvantage in trial and error method, many authors have assumed the value of DOCR settings based on expert's experience and solved these problems in linear environment (Irving and Elrafie, 1993; Chattopadhyay et al, 1996; Urdaneta et al, 1996; Urdaneta et al, 2001). But linear approach can't ensure correct settings of the relays (Laway and Gupta, 1993). They do not consider all possible operating conditions of the power system. Urdeneta et al (1988) was first to report the application of optimization theory in the coordination of DOCR. A detailed literature survey on this problem has been performed by Birla et al (2005). They have classified the previous works on DOCR coordination into three categories: curve fitting technique, graph theoretical technique and optimization technique.

Sparse Dual Revised Simplex method of linear programming has been used in (Irving and Elrafie, 1993) to optimize TDS settings for assumed non-linear PS settings. Some linear programming techniques applied in DOCR coordination problem include (Chattopadhyay et al, 1996; Urdaneta et al, 1996; Braga and Saraiva, 1996; Abyaneh and Keyhani 1995; Abdelaziz et al, 2002). Laway and Gupta (1993) applied Simplex and Rosenbrock - Hillclimb methods (non-linear programming technique) to optimize *TDS* and *PS* settings respectively, in a similar way, as used by Urdeneta et al (1988). The optimization of DOCR settings with Artificial Intelligence (AI) and Nature Inspired Algorithms (NIA) has received considerable attention recently. Some of the NIA algorithms, Evolutionary Programming (So and Li, 2000), Genetic Algorithm (GA)) (So et al, 1997; Farzad et al, 2008; Thakur, 2007), Modified Evolutionary Programming (So and Li 2000a; 2004), Particle Swarm Optimization (Mansour and Mekhamer, 2007; Zeineldin, 2006; Bansal and Deep, 2008), have been applied successfully in this problem. Self Organizing Migrating Algorithm (SOMA) and its hybridization with GA have been applied in (Dipti, 2007). Some of the AI methods, fuzzy logic (Abyane et al, 1997) and expert systems (Brown and Tyle, 1986; Lee et al, 1989; Hong et al, 1991; Jianping and Trecat, 1996) have also applied in this problem. Birla et al (2006) and Deep et al (2006) used Random Search Technique (RST2) to solve the relay coordination problem for IEEE 6-bus model and IEEE 3-bus, 4-bus models respectively.

8.3 Problem Formulation

An important characteristic of some types of protection in an electrical circuit is their capacity to determine the direction of the flow of power. Because of this characteristic they inhibit opening of the associated switch when the fault current flows in the opposite direction to the setting of the relays. Directional relays can tackle this situation when relays face fault currents in both directions because they operate only when fault current flows in specified tripping direction. Hence, directional over-current relays are used extensively for the protection of feeders having infeed from both the ends (e.g. loop systems, parallel feeders).

A *DOCR* consists of two units:

(i) An instantaneous unit

(ii) A time-delay unit

The instantaneous unit operates with no intentional time-delay when current is above a predefined threshold value, known as the instantaneous current setting. Time-delay unit is used for current, which is below the instantaneous current setting but exceeds the normal flow due to a fault. This unit operates at the occurrence of a fault with an intentional time-delay. Two settings are associated with the time-delay unit, which are as under:

- Time dial setting (*TDS*)
- Plug setting (*PS*) (e.g. tap setting)

The time dial setting adjusts time-delay before a relay operates whenever the fault current reaches a value equal to or greater than the pick-up current. Tap setting is a value that defines the pick-up current of the relay, and currents are expressed as multiple of this. These settings essentially specify the particular time-current characteristics from the family of available curves and the multiple of tap setting to be used to find the relay operating time for a given current flowing through the relay. Threshold" or "Pick-up current" is the minimum current for which the relay operates and is determined by selecting one of the plug settings taps available on the relay.

8.4 General Model of the Problem

The operating time (*T*) of a DOCR is non-linear function of the relay settings (Time Dial Settings (*TDS*) and Plug Settings (*PS*) and the fault current (*I*) seen by the relay. Therefore, Relay operating-time equation for a DOCR is given by a non-linear equation as given below

$$T = \frac{\alpha \times TDS}{\left(\frac{1}{PS \times CT_{pri\_rating}}\right)^{\beta} - \gamma} \tag{8.1}$$

Only TDS and PS are unknown variables in above equation. These are the "decision variables" of the problem. α, β and γ are the constants representing the behaviour of characteristic in a mathematical way, in which operating time of the DOCR varies and are given as 0.14, 0.02 and 1.0 respectively as per IEEE standard (1997). Value of CTpri_rating depends upon the number of turns in the equipment CT (Current Transformer). CT is used to reduce the level of the current so that relay can withstand it. With each relay one "Current Transformer" is used and thus, CTpri_rating is known in the problem. Value of I (Fault current passing through the relay) is also known, as it is a system dependent parameter and continuously measured by measuring instruments. Number of constraints for systems of bigger sizes will be dependent upon the number of lines in the system. Details of the number of lines in few larger systems are given in Table 8.1. In practice, in electrical engineering, power systems may be of even bigger sizes and there are other types of relays also besides DOCRs. Coordinating DOCRs with other types of relays generates even larger number of constraints are shown in Table 8.1. It is evident from Table 8.1 that simultaneous optimization of both the settings (TDS and PS) of each DOCR of the system is a complex problem

Table 8.1 The complexity of the DOCR problem as the bus size increases

	IEEE 3-bus	IEEE 4-bus	IEEE 6-bus
No. of lines	3	4	7
No. of DOCRs (relays)	6	8	14
No. of decision variables	12	16	28
No. of selectivity constraints	8	9	38
Constraints imposing restrictions on each term of objective function	24	32	104

Objective function and Constraints of the problem:

To solve any problem using optimization techniques an objective function subject to some criteria is minimized or maximized. The optimal coordination problem of DOCRs using optimization technique consists of minimizing an objective function (performance function) subject to certain coordination criteria and limits on problem variables. The relay, which is supposed to operate first to clear the fault, is called primary relay. A fault close to relay is known as the close-in fault for the relay and a fault at the other end of the line is known as a far-bus fault for this relay. Conventionally, objective function in coordination studies is constituted as the summation of operating-times of all primary relays, responding to clear all close-in and far-bus faults.

The ***objective function*** is as follows:

$$\text{Minimize } OBJ = \sum_{i=1}^{N_{cl}} T^{i}_{pri\_cl\_in} + \sum_{j=1}^{N_{far}} T^{j}_{pri\_far\_bus} \tag{8.2}$$

where,

N_{cl}	is number of relays responding for close-in fault.
N_{far}	is number of relays responding for far-bus fault.
$T_{pri\_cl\_in}$	is primary relay operating-time for close-in fault.
$T_{pri\_far\_bus}$	is primary relay operating-time for far-bus fault.

The ***constraints*** are as follows:

(1) Bounds on variables TDSs

$TDS^{i}_{min} \leq TDS^{i} \leq TDS^{i}_{max}$, where i varies from 1 to N_{cl}.

TDS^{i}_{min} is lower limit and TDS^{i}_{max} is upper limit of TDS^{i}. These limits are 0.05 and 1.1, respectively.

(2) Bounds on variables PSs

$PS^{j}_{min} \leq PS^{j} \leq PS^{j}_{max}$, where j varies from 1 to $N_{cl.}$

PS^{j}_{min} is lower limit and PS^{j}_{max} is upper limit of PS^{j}. These are 1.25 and 1.50, respectively.

(3) Limits on primary operation times

This constraint imposes constraint on each term of objective function to lie between 0.05 and 1.0.

(4) Selectivity constraints for all relay pairs:

$T_{backup} - T_{primary} - CTI \geq 0$

T_{backup} is operating time of backup relay and $T_{primary}$ is operating time of primary relay

8.5 The IEEE 3-bus Model

For the coordination problem of IEEE 3-bus model, value of each of N_{cl} and N_{far} is *6* (equal to number of relays or twice the lines). Accordingly, there are 12 decision variables (two for each relay) in this problem i. e. TDS^1 to TDS^6 and PS^1 to PS^6. The 3-bus system can be visualized as shown in Figure 8.1.

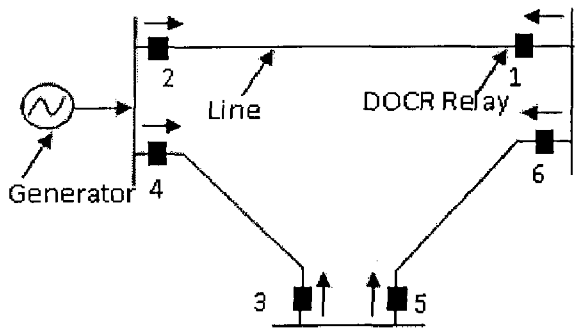

Figure 8.1 A typical IEEE 3-bus DOCR coordination problem model.

***Objective function** (OBJ) to be minimized as given by:*

$$OBJ = \sum_{i=1}^{6} T^{i}_{pri\_cl\_in} + \sum_{j=1}^{6} T^{j}_{pri\_far\_bus} \tag{8.3}$$

Where

$$T^{i}_{pri\_cl\_in} = \frac{0.14 \times TDS^{i}}{\left(\dfrac{a^{i}}{PS^{i} \times b^{i}}\right)^{0.02} - 1} \tag{8.4}$$

$$T^{i}_{pri\_far\_bus} = \frac{0.14 \times TDS^{j}}{\left(\dfrac{c^{i}}{PS^{j} \times d^{i}}\right)^{0.02} - 1} \tag{8.5}$$

The values of constants a^i, b^i, c^i and d^i are given in Table 8.2.

***Constraints for the model*:**

(1) Bounds on variables TDSs :

$$TDS^{i}_{min} \leq TDS^{i} \leq TDS^{i}_{max}$$, where, i varies from 1 to 6 (N_{cl})

(2) Bounds on variables PSs :

$$PS^{j}_{min} \leq PS^{j} \leq PS^{j}_{max}$$, where, j varies from 1 to 6 (N_{cl})

(3) Limits on primary operation times:

This constraint imposes constraint on each term of objective function to lie between 0.05 and 1.0.

(4) Selectivity constraints are:

$$T^{i}_{backup} - T^{i}_{primary} - CTI \geq 0 \tag{8.6}$$

T_{backup} is operating time of backup relay and $T_{primary}$ is operating time of primary relay. Value of *CTI* is 0.3. Here,

$$T^{i}_{backup} = \frac{0.14 \times TDS^{p}}{\left(\dfrac{e^{i}}{PS^{p} \times f^{i}}\right)^{0.02} - 1} \tag{8.7}$$

$$T^{i}_{primary} = \frac{0.14 \times TDS^{q}}{\left(\dfrac{g^{i}}{PS^{q} \times h^{i}}\right)^{0.02} - 1} \tag{8.8}$$

The values of constants e^i, f^i, g^i and h^i are given in the Table 8.3.

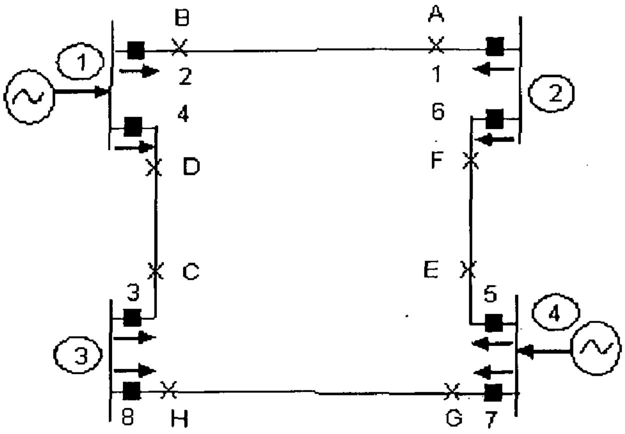

Figure 8.2 A typical IEEE 4-bus DOCR coordination problem model

Table 8.4 Values of constants a^i, b^i, c^i and d^i for IEEE 4-bus model

$T^i_{pri\_cl\_in}$			$T^i_{pri\_far\_bus}$		
TDS^i	a^i	b^i	TDS^j	c^i	d^i
TDS^1	20.32	0.48	TDS^2	23.75	0.48
TDS^2	88.85	0.48	TDS^1	12.48	0.48
TDS^3	13.61	1.1789	TDS^4	31.92	1.1789
TDS^4	116.81	1.1789	TDS^3	10.38	1.1789
TDS^5	116.7	1.5259	TDS^6	12.07	1.5259
TDS^6	16.67	1.5259	TDS^5	31.92	1.5259
TDS^7	71.7	1.2018	TDS^8	11	1.2018
TDS^8	19.27	1.2018	TDS^7	18.91	1.2018

Table 8.5 Values of constants e^i, f^i, g^i and h^i for IEEE 4-bus model

T^i_{backup}			$T^i_{primary}$		
p	e^i	f^i	q	g^i	h^i
5	20.32	1.5259	1	20.32	0.48
5	12.48	1.5259	1	12.48	0.48
7	13.61	1.2018	3	13.61	1.1789
7	10.38	1.2018	3	10.38	1.1789
1	1.16	0.48	4	116.81	1.1789
2	12.07	0.48	6	12.07	1.1789
2	16.67	0.48	6	16.67	1.5259
4	11	1.1789	8	11	1.2018
4	19.27	1.1789	8	19.27	1.2018

8.7 The IEEE 6-bus Model

The next coordination problem is IEEE 6-bus model, value of each of N_{cl} and N_{far} is *14* (equal to number of relays or twice the lines). Accordingly, there are 28 decision variables (two for each relay) in this problem i. e. TDS^1 to TDS^{14} and PS^1 to PS^{14}. The 6 bus system can be visualized as shown in Figure 8.3. The value of *CTI* for this model is 0.2. For the nominal state of the sample 6-bus model, 48 selectivity constraints are generated corresponding to all the possible near-end and far-end faults sensed by all the relays of the system. Based on the observation of Birla et al (2006a), ten constraints are relaxed.

The objective function and constraints for this model will be of same form as in the case of IEEE 3-bus problem with N_{cl} = 14. The values of constants a^i, b^i, c^i, d^i and e^i, f^i, g^i, h^i for 6-bus model are given in Table 8.6 and Table 8.7 respectively.

Table 8.6 Values of constants a^i, b^i, c^i and d^i for IEEE 6-bus model

$T^i_{pri\_cl\_in}$			$T^i_{pri\_far\_bus}$		
TDS^i	a^i	b^i	TDS^j	c^i	d^i
TDS^1	2.5311	0.2585	TDS2	5.9495	0.2585
TDS^2	2.7376	0.2585	TDS1	5.3752	0.2585
TDS^3	2.9723	0.4863	TDS4	6.6641	0.4863
TDS^4	4.1477	0.4863	TDS3	4.5897	0.4863
TDS^5	1.9545	0.7138	TDS6	6.2345	0.7138
TDS^6	2.7678	0.7138	TDS5	4.2573	0.7138
TDS^7	3.8423	1.746	TDS8	6.3694	1.746
TDS^8	5.618	1.746	TDS7	4.1783	1.746
TDS^9	4.6538	1.0424	TDS10	3.87	1.0424
TDS^{10}	3.5261	1.0424	TDS9	5.2696	1.0424
TDS^{11}	2.584	0.7729	TDS12	6.1144	0.7729
TDS^{12}	3.8006	0.7729	TDS11	3.9005	0.7729
TDS^{13}	2.4143	0.5879	TDS14	2.9011	0.5879
TDS^{14}	5.3541	0.5879	TDS13	4.335	0.5879

Table 8.7 Values of constants e^i, f^i, g^i and h^i for IEEE 6-bus model

T^i_{backup}			$T^i_{primary}$		
p	e^i	f^i	q	g^i	h^i
8	4.0909	1.746	1	5.3752	0.2585
11	1.2886	0.7729	1	5.3752	0.2585
8	2.9323	1.746	1	2.5311	0.2585
3	0.6213	0.4863	2	2.7376	0.2585
3	1.6658	0.4863	2	5.9495	0.2585
10	0.0923	1.0424	3	4.5897	0.4863
10	2.561	1.0424	3	2.9723	0.4863
13	1.4995	0.5879	3	4.5897	0.4863
1	0.8869	0.2585	4	4.1477	0.4863
1	1.5243	0.2585	4	6.6641	0.4863
12	2.5444	0.7729	5	4.2573	0.7138
12	1.4549	0.7729	5	1.9545	0.7138
14	1.7142	0.5879	5	4.2573	0.7138
3	1.4658	0.4863	6	6.2345	0.7138
3	1.1231	0.2585	6	6.2345	0.7138
11	2.1436	0.7729	7	4.1783	1.746
2	2.0355	0.2585	7	4.1783	1.746
11	1.9712	0.7729	7	3.8423	1.746
2	1.8718	0.2585	7	3.8423	1.746
13	1.8321	0.5879	9	5.2696	1.0424
4	3.4386	0.4863	9	5.2696	1.0424
13	1.618	0.5879	9	4.6538	1.0424
4	3.0368	0.4863	9	4.6538	1.0424
14	2.0871	0.5879	11	3.9005	0.7729
6	1.8138	0.7138	11	3.9005	0.7729
14	1.4744	0.5879	11	2.584	0.7729
6	1.1099	0.7138	11	2.584	0.7729
8	3.3286	1.746	12	3.8006	0.7729
2	0.4734	0.2585	12	3.8006	0.7729
8	4.5736	1.746	12	6.1144	0.7729
2	1.5432	0.2585	12	6.1144	0.7729
12	2.7269	0.7729	13	4.335	0.5879
6	1.6085	0.7138	13	4.335	0.5879
12	1.836	0.7729	13	2.4143	0.5879
10	2.026	1.0424	14	2.9011	0.5879
4	0.8757	0.4863	14	2.9011	0.5879
10	2.7784	1.0424	14	5.3541	0.5879
4	2.5823	0.4863	14	5.3541	0.5879

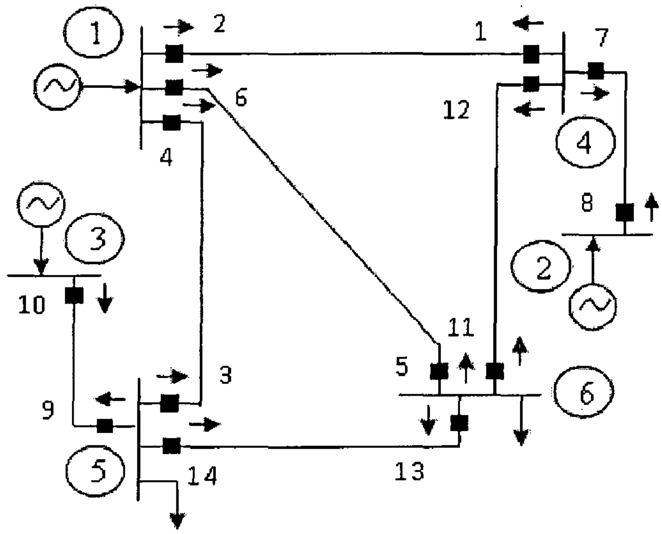

Figure 8.3 A typical IEEE 6-bus DOCR coordination problem model

8.8 Methods of Solution and Discussion of Results

All the three optimization models stated above are solved using the DE algorithms namely: DE, LDE1, LDE2, LDE3, LDE4, LDE5 and DE-QI. In order to make a fair comparison of all versions of DE algorithms, same seed for random number generation is fixed so that the initial population is same for all the algorithms. The population size is taken as 50. The crossover constant CR is set as 0. 5 and the scaling factor F is set as 0.5. For each algorithm, the stopping criteria is to terminate the search process when one of the following conditions is satisfied: (i) the maximum number of generations is reached (assumed 10000 generations), (ii) $|f_{max} - f_{min}| < 10^{-4}$ where f is the value of objective function. Constraint handling mechanism discussed in chapter 6 is used for handling constraints. A total of 30 runs for each experimental setting were conducted and the best solution throughout the run was recorded as global optimum. For comparison, previously quoted results by RST2 (Deep et al, 2006; Birla et al, 2006), GA, SOMA, SOMGA (Dipti, 2007), LX-POL and LX-PM (Thakur, 2007) are used. Figures 8.4 – 8.9 show the performance of DE and the proposed DE algorithms on IEEE 3-bus, 4-bus and 6-bus models.

The best solution obtained by DE and modified DE algorithms of IEEE 3–bus model interms of optimal decision variable values, objective function value and number of function evaluations are given in Table 8.8. From the numerical results, it can be seen that LDE4 gave better result than the other algorithms in terms of objective function value. On the other hand, in terms of comparisons of NFE, then the performance of LDE5 is better than all other compared algorithms. The experimental results of IEEE 4-bus and 6-bus models are given in Table 8.9 and 8.10 respectively. For the IEEE 4-bus model also, LDE4 performs better than other algorithms interms of best objective function value. Once again LDE5 gave better results in terms of NFE than other compared algorithms for 4-bus model. But the results of IEEE 6-bus model is entirely different from the results of previous two models. In this case DE-QI is a winner in terms of objective functive value and LDE2 is a winner interms of NFE. From the numerical results of Table 8.8, 8.9 and 8.10, we can see that all the modified versions of DE outperform the basic DE algorithm by a significant difference. In Table 8.11, the improvement (%) of modified DE algorithms in comparison with basic DE is given.

Also the numerical results of DE and the proposed DE algorithms for IEEE 3-bus, 4-bus and 6-bus models are compared with the results of some other algorithms in the literature; the corresponding results are given in Table 8.12. From the numerical results in Table 8.12, it can be seen that LDE4 and DE-QI algorithms are perform better than other algorithms for IEEE 3-bus and 6-bus models respectively; but in the case of 4-bus model, LX-POL, which is a modified version of real coded GA, gave better performance than other algorithms.

Table 8.8 Optimal design variables, objective function values and number of function of evaluations (NFE) of IEEE 3-bus model by DE and the proposed DE algorithms

	DE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI
TS^1	0.05	0.05	0.05	0.05	0.05	0.05	0.05
TS^2	0.219387	0.217774	0.197872	0.198761	0.197648	0.197648	0.197649
TS^3	0.05	0.05	0.05	0.05	0.05	0.05	0.05
TS^4	0.213499	0.209044	0.209474	0.209048	0.209037	0.209035	0.209034
TS^5	0.19498	0.181208	0.184715	0.181215	0.181208	0.181206	0.181206
TS^6	0.195307	0.180682	0.182734	0.180678	0.180677	0.180676	0.180677
PS^1	1.25	1.25	1.25	1.25	1.25	1.25	1.25002
PS^2	1.25	1.25	1.49996	1.48497	1.49999	1.5	1.5
PS^3	1.25001	1.25	1.25	1.25	1.25	1.25	1.25001
PS^4	1.46053	1.49988	1.49999	1.49985	1.49997	1.5	1.5
PS^5	1.25	1.5	1.43182	1.49982	1.49994	1.5	1.5
PS^6	1.25	1.4999	1.46195	1.49996	1.49997	1.5	1.5
F	4.84218	4.80699	4.78728	4.78227	4.78067	4.78068	4.78069
NFE	78360	72350	73350	97550	69270	38250	56700

Table 8.9 Optimal design variables, objective function values and number of function of evaluations (NFE) of IEEE 4-bus model by DE and the proposed DE algorithms

	DE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI
TS1	0.05	0.05	0.05	0.05	0.05	0.05	0.05
TS2	0.224898	0.212176	0.212356	0.21217	0.21217	0.212174	0.212183
TS3	0.05	0.050001	0.05	0.05	0.05	0.05	0.050001
TS4	0.151592	0.151593	0.151587	0.151596	0.151577	0.15158	0.151594
TS5	0.126413	0.126401	0.126401	0.126412	0.126236	0.126401	0.126244
TS6	0.05	0.05	0.05	0.05	0.05	0.050001	0.05
TS7	0.133788	0.133801	0.137135	0.133809	0.133799	0.133787	0.133791
TS8	0.050001	0.05	0.05	0.05	0.05	0.050001	0.050001
PS1	1.27344	1.27338	1.27334	1.27336	1.25	1.2734	1.25
PS2	1.25	1.49986	1.49598	1.5	1.5	1.49993	1.49983
PS3	1.25001	1.25001	1.25001	1.25	1.25	1.25002	1.25006
PS4	1.4997	1.49965	1.49975	1.49958	1.5	1.49997	1.49985
PS5	1.49976	1.5	1.5	1.49974	1.5	1.5	1.49983
PS6	1.25	1.25001	1.25	1.25	1.25	1.25001	1.25003
PS7	1.5	1.49975	1.42747	1.49952	1.49989	1.5	1.49999
PS8	1.25	1.25003	1.25	1.25	1.25	1.25	1.25
F	3.67744	3.66945	3.67349	3.66925	3.66749	3.66941	3.66757
NFE	95400	43400	67200	99700	55100	35330	70650

Table 8.10 Optimal design variables, objective function values and number of function of evaluations (NFE) of IEEE 6-bus model by DE and the proposed DE algorithms

	DE	LDE1	LDE2	LDE3	LDE4	LDE5	DE-QI
TS1	0.117325	0.117186	0.114991	0.103403	0.114487	0.102494	0.101411
TS2	0.208261	0.186646	0.203752	0.186301	0.18641	0.186341	0.186334
TS3	0.099714	0.096582	0.098299	0.096107	0.094739	0.094675	9.46E-02
TS4	0.112537	0.111923	0.103672	0.112567	0.10061	0.106796	0.100603
TS5	0.050005	0.050013	0.05	0.050001	0.050001	0.050002	5.00E-02
TS6	0.058011	0.050019	0.05	0.05	0.050008	0.05	0.05
TS7	0.050002	0.050001	0.050001	0.05	0.05	0.05	0.050001
TS8	0.050004	0.05	0.05	0.050003	0.050004	0.050002	0.050001
TS9	0.050005	0.050006	0.050001	0.05	0.05	0.05	0.050005
TS10	0.071966	0.070608	0.057507	0.070325	0.070155	0.056329	0.056263
TS11	0.064995	0.064998	0.066782	0.06499	0.064981	0.065005	0.064976
TS12	0.061796	0.061796	0.056615	0.050917	0.050917	0.055312	0.050903
TS13	0.050007	0.05	0.063515	0.05	0.050009	0.050005	0.050006
TS14	0.08566	0.086012	0.085904	0.085723	0.070928	0.070994	0.070851
PS1	1.25057	1.25153	1.26356	1.49956	1.26024	1.49911	1.49967
PS2	1.25009	1.49594	1.29936	1.49996	1.4987	1.49994	1.50E+00
PS3	1.25121	1.25258	1.26226	1.25754	1.27617	1.27716	1.27752
PS4	1.25151	1.26329	1.43227	1.25082	1.49924	1.36503	1.50E+00
PS5	1.25	1.25004	1.25	1.25	1.25002	1.25005	1.25001
PS6	1.25E+00	1.38225	1.38859	1.38102	1.38142	1.38181	1.38091
PS7	1.25005	1.25002	1.25083	1.25001	1.25	1.25005	1.25E+00
PS8	1.25E+00	1.25011	1.25	1.25008	1.25053	1.25003	1.25009
PS9	1.25022	1.25005	1.25147	1.25004	1.25	1.25	1.25
PS10	1.25023	1.25014	1.49707	1.25211	1.25	1.49961	1.49939
PS11	1.49987	1.49994	1.47597	1.49981	1.49998	1.49987	1.49998
PS12	1.25757	1.25297	1.47	1.49979	1.5	1.39319	1.5
PS13	1.48058	1.46644	1.27288	1.46474	1.46151	1.46134	1.4612
PS14	1.25577	1.25001	1.26242	1.25404	1.4979	1.49747	1.49936
F	10.6272	10.5067	10.6238	10.437	10.3812	10.3614	10.3287
NFE	212190	72960	18180	101580	100860	106200	163980

Table 8.11 Improvement(%) of proposed DE algorithms in comaprison with DE interms of objective function values

Algorithm	IEEE 3-bus	IEEE 4-bus	IEEE6-bus
LDE1	0.726739	0.217271	1.133883
LDE2	1.133787	0.107412	0.031993
LDE3	1.237253	0.222709	1.789747
LDE4	1.270296	0.270569	2.314815
LDE5	1.270089	0.218358	2.501129
DE-QI	1.269883	0.268393	2.80883

Table 8.12 Comparison results of IEEE 3-bus, 4-bus and 6-bus models: interms of objective function values

Algorithm	IEEE 3-bus	IEEE 4-bus	IEEE 6-bus
DE	4.84218	3.67744	10.6272
LDE1	4.80699	3.66945	10.5067
LDE2	4.78728	3.67349	10.6238
LDE3	4.78227	3.66925	10.437
LDE4	4.78067	3.66749	10.3812
LDE5	4.78068	3.66941	10.3614
DE-QI	4.78069	3.66757	10.3287
RST2	4.835427	3.705018	10.619223
GA	5.07616	3.85874	13.7996
SOMA	8.01016	3.78922	26.1495
SOMGA	4.78989	3.67453	10.3578
LX-POL	4.826506	3.574931	10.60281
LX-PM	4.828629	3.583045	10.62195

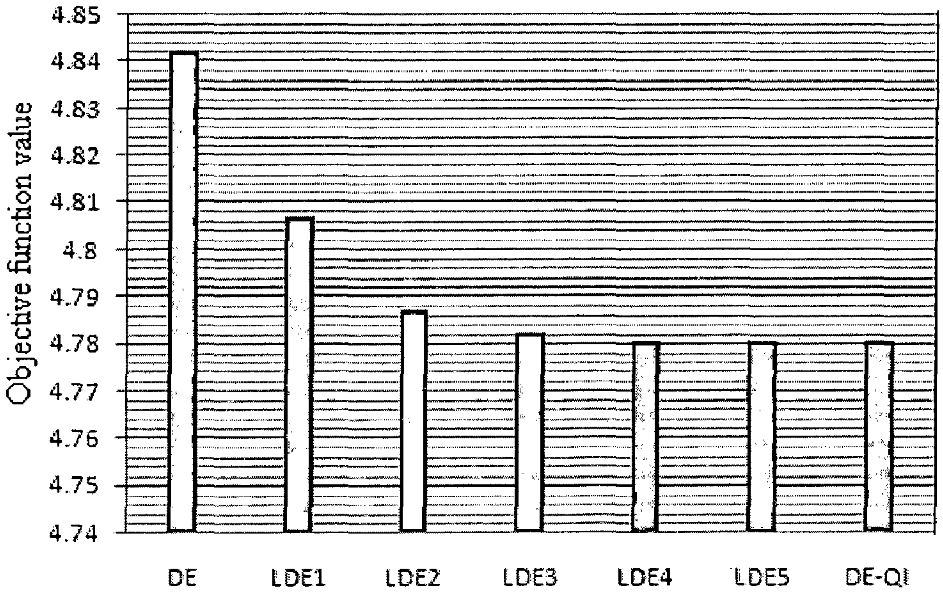

Figure 8.4 Comparison of DE and modified DE algorithms in terms of objective function values: IEEE 3-bus model

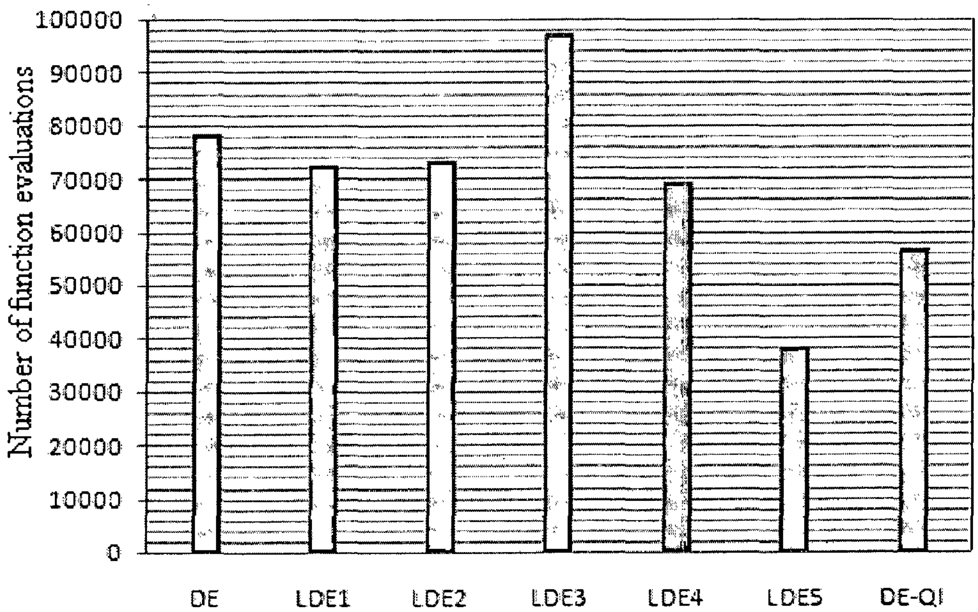

Figure 8.5 Comparison of DE and modified DE algorithms in terms of Number of function evaluations: IEEE 3-bus model

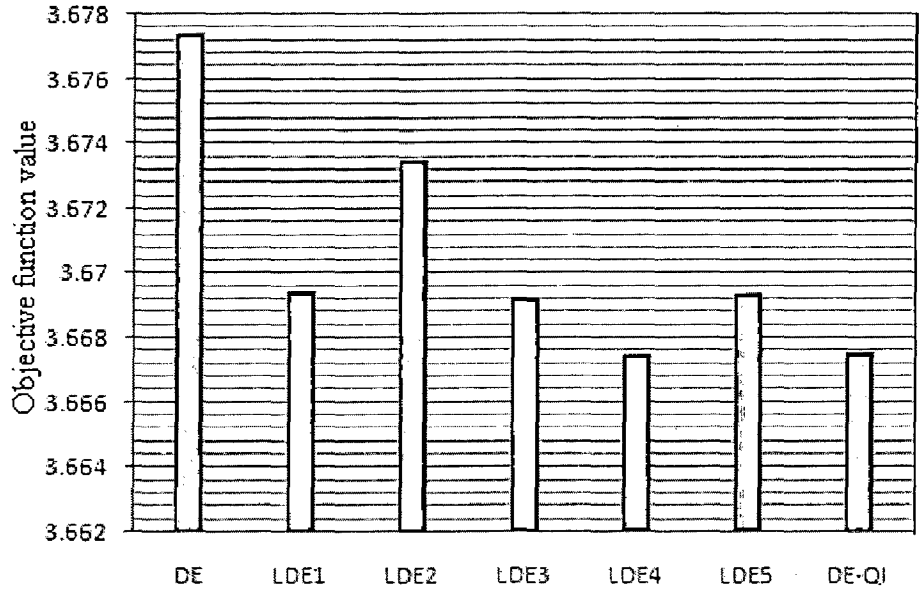

Figure 8.6 Comparison of DE and modified DE algorithms in terms of objective function values: IEEE 4-bus model

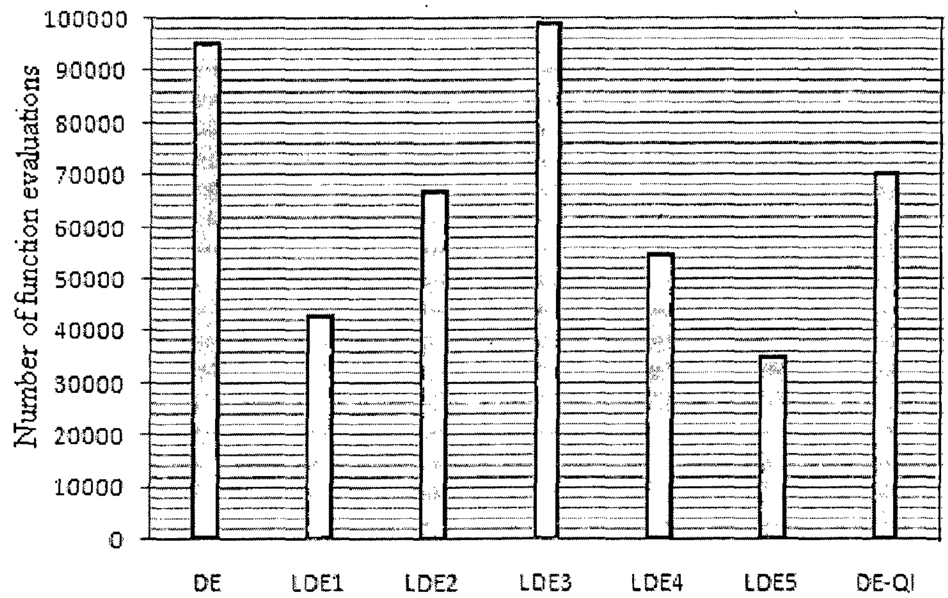

Figure 8.7 Comparison of DE and modified DE algorithms in terms of Number of function evaluations: IEEE 4-bus model

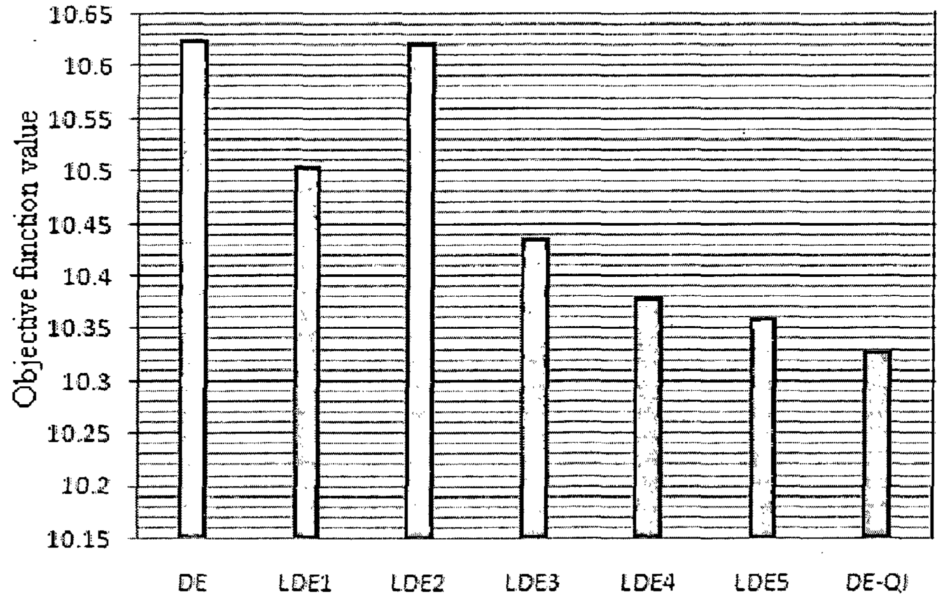

Figure 8.8 Comparison of DE and modified DE algorithms in terms of objective function values: IEEE 6-bus model

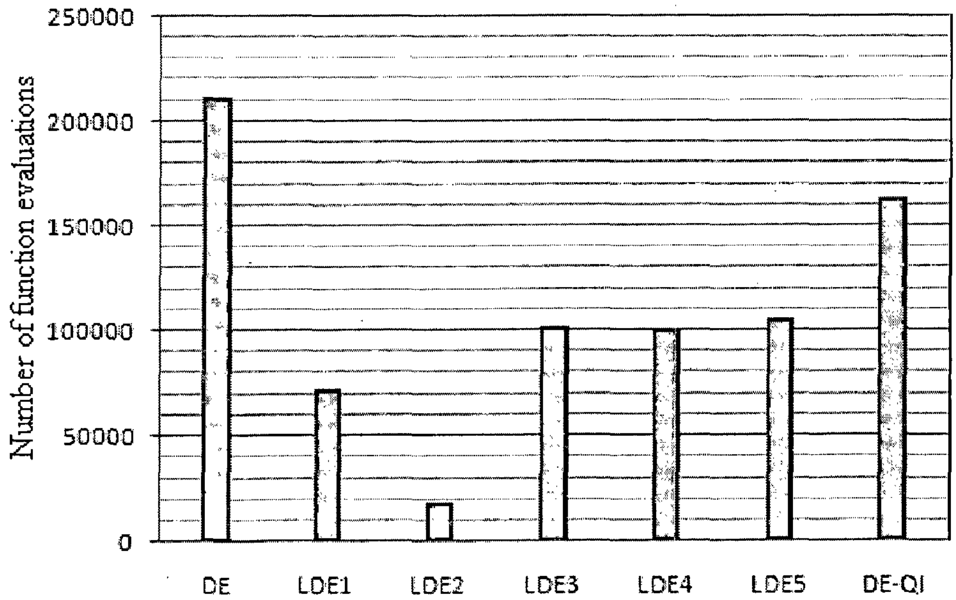

Figure 8.9 Comparison of DE and modified DE algorithms in terms of Number of function evaluations: IEEE 6-bus model

8.9 Conclusion

In this chapter, electrical engineering power system DOCR coordination problem, which is a constrained non-linear optimization problem, is solved by DE and six modified versions of DE namely LDE1, LDE2, LDE3, LDE4, LDE5 and DE-QI. The problem is to determine the optimal value of Time dial setting and Plug setting so that the relay time can be minimized. Three models of this problem namely IEEE 3- bus, IEEE 4-Bus and IEEE 6-bus were solved by using DE and its variants. The complexities of all the models are different due to different decision variables and constraints. The results obtained by modified DE algorithms on all models were superior with the basic DE algorithm. Also, the results obtained by DE and its variants were compared with RST2, GA, SOMA, SOMGA, LX-POL and LX-PM algorithms in the literature. In all the considered models, DE variants were superior or at par with all other algorithms; this variants are found to be a robust technique for solving such type of constrained nonlinear optimization problems.

Chapter 9

Optimization of Some Real Life Problems Using PSO and DE

[In this chapter, some real life problems, collected from various fields, are solved using the versions of PSO and DE presented in this thesis. These problems are: Static Power Scheduling problem, Dynamic Power Scheduling problem, Cost Optimization of Transformer Design, Weight Minimization of a Speed Reducer, Heat Exchanger Network Design, Gas Transmission Compressor Design, Optimal Design of a Industrial Refrigeration System, Optimization of Transistor Modeling, Optimal Capacity of Gas Production Facilities, Optimal Thermohydralic Performance of an Artificially Roughened Air Heater and Design of a Gear Train. The proposed algorithms discussed in chapter 2, 3, 4, 5 and 6 are used to solve the above mentioned real life problems.]

9.1 Introduction

Many engineering problems can be formulated as optimization problems. These problems when subjected to a suitable optimization algorithm help in improving the quality of solution. In particular there has been a focus on stochastic algorithms for obtaining the global optimum solution to the problem, because in many cases it is not only desirable but also necessary to obtain the global optimal solution. In order to further validate the efficiency of the proposed algorithms (discussed in chapters 2, 3, 4, 5 and 6), they were tested on several real life problems. These problems are segregated as constrained and unconstrained problems. The constrained problems are solved using ICDE and ICPSO algorithms. For unconstrained problems almost all the algorithms discussed in this thesis gave more or less similar results (better than basic versions of PSO and DE algorithms) however for the sake of brevity results are given for only a few chosen algorithms which gave marginally better results than the other algorithms developed in this thesis for solving unconstrained problems. The constrained problems are given in section 9.2 to 9.8, and section 9.9 onwards unconstrained problems are given. A brief description of the problems is given in the following subsections.

9.2 Static Power Scheduling Problem (Source: B-Biggs (1978))

In this problem the decision variables x_1 and x_2 are the real power outputs from two generators; x_3 and x_4 are the reactive power outputs; x_5, x_6 and x_7 are voltage magnitudes at three nodes of an electrical network and x_8 and x_9 are voltage phase angles at two of these nodes. The constraints other than the bounds are the real and reactive power balance equations, stating that the power flowing into a node must balance the power flowing out.

Mathematical model of Static Power Scheduling problem is given by:

Minimize $f(x) = 3000x_1 + 1000x_1^3 + 2000x_2 + 666.667x_2^3$

Subject to:

$$0.4 - x_1 + 2Cx_5^2 + x_5x_6[D\sin(-x_8) - C\cos(-x_8)] + x_5x_7[D\sin(-x_9) - C\cos(-x_9)] = 0$$

$$0.4 - x_2 + 2Cx_6^2 + x_5x_6[D\sin(x_8) - C\cos(x_8)] + x_6x_7[D\sin(x_8 - x_9) - C\cos(x_8 - x_9)] = 0$$

$$0.8 + 2Cx_7^2 + x_5x_7[D\sin(x_9) - C\cos(x_9)] + x_6x_7[D\sin(x_9 - x_8) - C\cos(x_9 - x_8)] = 0$$

$$0.2 - x_3 + 2Dx_5^2 - x_5x_6[C\sin(-x_8) + D\cos(-x_8)] - x_5x_7[C\sin(-x_9) + D\cos(-x_9)] = 0$$

$$0.2 - x_4 + 2Dx_6^2 - x_5x_6[C\sin(x_8) + D\cos(x_8)] - x_6x_7[C\sin(x_8 - x_9) + D\cos(x_8 - x_9)] = 0$$

$$-0.337 + 2Dx_7^2 - x_5x_7[C\sin(x_9) + D\cos(x_9)] - x_6x_7[C\sin(x_9 - x_8) + D\cos(x_9 - x_8)] = 0$$

$x_i \geq 0 \quad i = 1,2.$

$1.0909 \geq x_i \geq 0.90909 \quad i = 5,6,7.$

$C = \sin(0.25)48.4/50.176$

$D = \cos(0.25)48.4/50.176$

This problem is a constrained optimization problem; it has 9 decision variables and 6 equality constraints.

9.3 Dynamic Power Scheduling Problem (Source: B-Biggs (1978))

This problem is a representation of the problem of scheduling three generators to meet the demand for power over a period of time. The variable x_{3k+i} denotes the output from the i[th] generator at time $t^{(k)}$. The constraints in the problem are upper and lower limits on the power available from each generator, bounds on the amount by which the output from a generator can change from time $t^{(k)}$ to $t^{(k+1)}$, and the condition that the at each time $t^{(k)}$ the power generated must at least satisfy the demand.

Mathematical model of this problem is given by:

Minimize

$$f(x) = \sum_{k=0}^{4} (2.3x_{3k+1} + 0.0001x_{3k+1}^2 + 1.7x_{3k+2} + 0.0001x_{3k+2}^2 + 2.2x_{3k+3} + 0.00015x_{3k+3}^2)$$

Subject to:

$-7 \le x_1 - 15 \le 6$

$-7 \le x_{3k+1} - x_{3k-2} \le 6 \quad k = 1,.....,4$

$-7 \le x_2 - 50 \le 7$

$-7 \le x_{3k+2} - x_{3k-1} \le 7 \quad k = 1,.....,4$

$-7 \le x_3 - 10 \le 6$

$-7 \le x_{3k+3} - x_{3k} \le 6 \quad k = 1,.....,4$

$x_1 + x_2 + x_3 \ge 60$

$x_4 + x_5 + x_6 \ge 50$

$x_7 + x_8 + x_9 \ge 70$

$x_{10} + x_{11} + x_{12} \ge 85$

$x_{13} + x_{14} + x_{15} \ge 100$

$0 \le x_{3k+1} \le 90 \qquad k = 1,.....,4$

$0 \le x_{3k+2} \le 120 \qquad k = 1,.....,4$

$0 \le x_{3k+3} \le 60 \qquad k = 1,.....,4$

This problem is a constrained optimization problem; it has 15 decision variables, 35 inequality constraints and 30 boundary constraints.

9.4 Cost Optimization of a Transformer Design

(Source: B-Biggs (1978))

The objective function represents the worth of the transformer, including the operating cost, and the constraints refer to the rating of the transformer and the allowable transmission loss. The decision variables x_1, x_2, x_3 and x_4 are physical dimensions of winding and core and the variables x_5, x_6 are magnetic flux density and current density respectively.

The mathematical model of this problem is given by:

Minimize $f = 0.0204x_1x_4(x_1 + x_2 + x_3) + 0.0187x_2x_3(x_1 + 1.57x_2 + x_4) +$

$$0.0607x_1x_4x_5^2(x_1 + x_2 + x_3) + 0.0437x_2x_3x_6^2(x_1 + 1.57x_2 + x_4)$$

Subject to:

$x_1x_2x_3x_4x_5x_6 \ge 2.07 \times 10^3$

$1 - 0.00062x_1x_4x_5^2(x_1 + x_2 + x_3) - 0.00058x_2x_3x_6^2(x_1 + 1.57x_2 + x_4) \ge 0$

$x_i \ge 0 \ \ (i = 1,...,6)$.

This problem is a constrained optimization problem; it has 6 decision variables, 2 inequality constraints and 6 boundary constraints.

9.5 Weight Minimization of Speed Reducer

(Source: Floudas and Pardalos (1990))

The problem involves the design of a speed reducer for small aircraft engine.

The mathematical model of this problem is,

Minimize $f(x) = 0.7854x_1x_2^2(3.3333x_3^2 + 14.9334x_3 - 43.0934)$

$$-1.508x_1(x_6^2+x_7^2)+7.477(x_6^3+x_7^3)+0.7854(x_4x_6^2+x_5x_7^2)$$

Subject to:

$x_1x_2^2x_3 \ge 27$, $x_1x_2^2x_3^2 \ge 397.5$, $x_2x_6^4x_3x_4^{-3} \ge 1.93$,

$A_1B_1^{-1} \le 1100$

Where $A_1 = [(745x_4x_2^{-1}x_3^{-1})^2 + 16.9611^6]^{0.5}$, $B_1 = 0.1x_6^3$

$A_2B_2^{-1} \le 850$

Where $A_2 = [(745x_5x_2^{-1}x_3^{-1})^2 + 15.7510^6]^{0.5}$, $B_2 = 0.1x_7^3$

$x_2x_3 \le 40$, $x_1x_2^{-1} \ge 5$, $x_1x_2^{-1} \le 12$, $1.5x_6 - x_4 \le -1.9$, $1.5x_7 - x_5 \le -1.9$.

$2.6 \le x_1 \le 3.6$, $0.7 \le x_2 \le 0.8$, $17 \le x_3 \le 28$, $7.3 \le x_4 \le 8.3$, $7.3 \le x_5 \le 8.3$, $2.9 \le x_6 \le 3.9$,

$5 \le x_7 \le 5.5$

It is a constrained optimization problem having 7 decision variables, 10 inequality constraints and 14 boundary constraints.

9.6 Heat Exchanger Network Design (Source: Babu and Angira (2008))

This problem addresses the design of a heat exchanger network as shown in Figure 9.1. It has been taken from Babu and Angira (2008). Also, it has been solved by Adjiman et al. (1998) using αBB -Algorithm. One cold stream must be heated from 100 ^{0}F (37.78 ^{0}C) to 500 ^{0}F (260 ^{0}C) using three hot streams with different inlet temperatures. The goal is to minimize the overall heat exchange area.

The mathematical model of this problem is,

Minimize $f(x) = x_1 + x_2 + x_3$

Subject to:

$-1 + 0.0025(x_4 + x_6) \le 0$,

$-1 + 0.0025(x_5 + x_7 - x_4) \le 0$,

$-1 + 0.01(x_8 - x_5) \le 0$,

$-x_1x_6 + 833.33252x_4 + 100x1 - 83333.333 \le 0$,

$-x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \le 0$,

$-x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \le 0$,

$-100 \le x_1 \le 10000$, $1000 \le x_i \le 10000$ $(i = 2,3)$, $10 \le x_i \le 1000$ $(i = 4,\dots,8)$

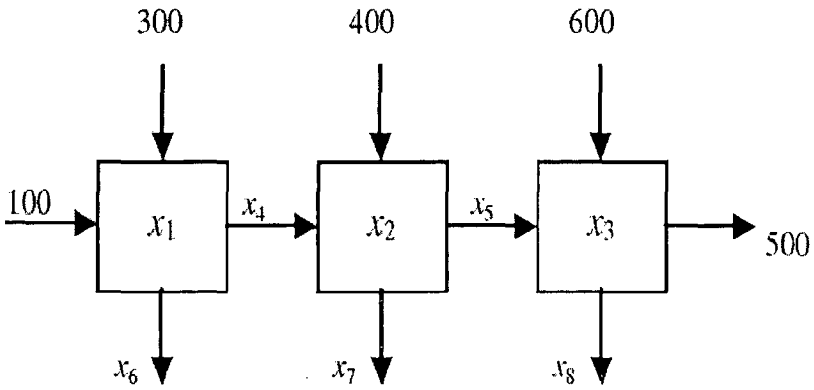

Figure 9.1 Heat exchanger network design problem

This problem is a constrained optimization problem; it has 8 decision variables, 6 inequality constraints and 16 boundary constraints.

9.7 Gas Transmission Compressor Design

(Source: Beightler and Phillips (1976))

In this problem the values of design parameters P1, x_1, x_2, x_3 are to be determined such that they deliver 100 million cu. Ft. of gas per day with minimum cost for a gas pipe line transmission system as shown below.

The mathematical model is,

$$\text{Minimize } f(x) = 8.61\times10^5 x_1^{1/2} x_2 x_3^{-2/3} x_4^{-1/2} + 3.69\times10^4 x_3 + 7.72\times10^8 x_1^{-1} x_2^{0.219} - 765.43\times10^6 x_1^{-1}$$

Subject to:

$x_4 x_2^{-2} + x_2^{-2} \le 1$

$20 \le x_1 \le 50$, $1 \le x_2 \le 10$, $20 \le x_3 \le 50$, $0.1 \le x_4 \le 60$

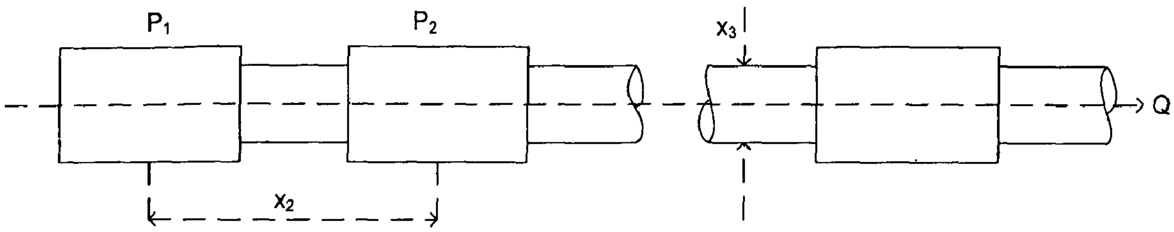

Figure 9.2 Gas transmission compressor

This problem is a constrained optimization problem; it has 4 decision variables, 1 inequality constraint and 8 boundary constraints.

9.8 Optimal Design of Industrial Refrigeration System

(Source: Paul and Tay (1987))

In this problem an individual refrigeration system is to be designed to meet the following requirements.

Refrigeration capacity	: 615.3 kW (175 tons)
Leaving chilled water temperature	: 6.7°C
Entering condenser water temperature	: 28°C
Condensing temperature	: 40°C
Evaporating temperature	: 5°C
Refrigerant	: R – 2°C

The various cost considered are evaporator fabrication cost, condenser fabrication cost, evaporator insulation cost and pumping cost. Apart from the above costs, physical size and heat transfer requirements have been incorporated in the formulation of the problem. The design is based on a standard vapour – condensation cycle. The condenser and evaporator are of the horizontal multi-pass, shell & tube type. The expansion device thermostatic expansion value and the compressor are off-the-shell items. The model is formulated based on some first principles of thermodynamics and according to certain standards set by ARI, ASTM, ASME and ASHRAE. The design presented here emulates a commercially available system.

The mathematical model is,

$$\text{Minimize } f(x) = 63098.88x_2x_4x_{12} + 5441.5x_2^2x_{12} + 115055.5x_2^{1.664}x_6$$
$$+ 6172.27x_2^2x_6 + 63098.88x_1x_3x_{11} + 5441.5x_1^2x_{11}$$

$$+115055.5x_1^{1.664}x_5 + 6172.27x_1^2x_5 + 140.53x_1x_{11}$$

$$+281.29x_3x_{11} + 70.26x1^2 + 281.29x_1x_3 + 281.29x_3^2$$

$$+14437x_8^{1.8812}x_{12}^{0.3424}x_{10}x_{14}^{-1}x_1^2x_7x_9^{-1} + 20470.2x_7^{2.893}x_{11}^{0.316}x_1^2$$

Subject to:

$1.524x_7^{-1} \le 1$, $1.524x_8^{-1} \le 1$, $0.07789x_1 - 2x_7^{-1}x_9 \le 1$,

$7.05305x_9^{-1}x_1^2x_{10}x_8^{-1}x_2^{-1}x_{14}^{-1} \le 1$, $0.0833x_{13}^{-1}x_{14} \le 1$,

$47.136x_2^{0.333}x_{10}^{-1}x_{12} - 1.333x_8x_{13}^{2.1195} + 62.08x_{13}^{2.1195}x_{12}^{-1}x_8^{0.2}x_{10}^{-1} \le 1$

$0.04771x_{10}x_8^{1.8812}x_{12}^{0.3424} \le 1$,

$0.0488x_9x_7^{1.893}x_{11}^{0.316} \le 1$, $0.0099x_1x_3^{-1} \le 1$, $0.0193x_2x_4^{-1} \le 1$, $0.0298x_1x_5^{-1} \le 1$,

$0.056x_2x_6^{-1} \le 1$, $2x_9^{-1} \le 1$, $2x_{10}^{-1} \le 1$, $x_{12}x_{11}^{-1} \le 1$,

$0.001 \le x_i \le 5$, $i = 1,\ldots,14$

This problem is a constrained optimization problem; it has 14 decision variables, 15 inequality constraints and 28 boundary constraints.

9.9 Optimization of Transistor Modeling (Source: Price (1978))

The objective function of this problem 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.

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

$$\alpha_k = (1 - x_1x_2)x_3\{\exp[x_5(g_{1k} - g_{3k}x_7 \times 10^{-3} - g_{5k}x_8 \times 10^{-3})] - 1\}g_{5k} + g_{4k}x_2$$

$$\beta_k = (1 - x_1x_2)x_4\{\exp[x_6(g_{1k} - g_{2k} - g_{3k}x_7 \times 10^{-3} + g_{4k}x_9 \times 10^{-3})] - 1\}g_{5k}x_1 + g_{4k}$$

$\gamma = x_1x_3 - x_2x_4$

Subject to:

$$x_i \geq 0,\ i = 1, 2, \ldots, 9$$

And 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 problem is an unconstrained optimization problem; it has 9 decision variables and 9 boundary constraints.

9.10 Optimal Capacity of Gas Production Facilities

(Source: Beightler and Phillips (1976))

This is the problem of determining the optimum capacity of production facilities that combine to make an oxygen producing and storing system. Oxygen for basic oxygen furnace is produced at a steady state level. The demand for oxygen is cyclic with a period of one hour, which is too short to allow an adjustment of level of production to the demand. Hence the manager of the plant has two alternatives.

(1) He can keep the production at the maximum demand level; excess production is lost in the atmosphere.

(2) He can keep the production at lower level; excess production is compressed and stored for use during the high demand period.

The mathematical model of this problem is given by:

$$\text{Minimize } f(x) = 61.8 + 5.72x_1 + 0.2623[(40 - x_1)\ln(\frac{x_2}{200})]^{-0.85}$$

$$+ 0.087(40 - x_1)\ln(\frac{x_2}{200}) + 700.23x_2^{-0.75}$$

Subject to: $x_1 \geq 17.5$, $x_2 \geq 200; 17.5 \leq x_1 \leq 40$, $300 \leq x_2 \leq 600$.

This problem is an unconstrained optimization problem; it has 2 decision variables and 4 boundary constraints.

9.11 Optimal Thermohydraulic Performance of an Artificially Roughened Air Heater (Source: Prasad and Saini (1991))

In this problem the optimal thermohydraulic performance of an artificially roughened solar air heater is considered. Optimization of the roughness and flow parameters (p/e, e/D, Re) is considered to maximize the heat transfer while keeping the friction losses to be minimum. The mathematical model of this problem is given by:

Maximize $L = 2.51 * \ln e^{+} + 5.5 - 0.1R_M - G_H$

Where $R_M = 0.95{x_2}^{0.53}$; $GH = 4.5(e^{+})^{0.28}(0.7)^{0.57}$; $e^{+} = x_1x_3(\bar{f}/2)^{1/2}$; $\bar{f} = (f_s + f_r)/2$;

$f_s = 0.079{x_3}^{-0.25}$; $f_r = 2(0.95{x_3}^{0.53} + 2.5 * \ln(1/2x_1)^2 - 3.75)^{-2}$;

Subject to:

$0.02 \le x_1 \le 0.8, 10 \le x_2 \le 40,\ 3000 \le x_3 \le 20000$

This problem is an unconstrained optimization problem; it has 3 decision variables and 6 boundary constraints.

9.12 Design of Gear Train (Source: Sandgren (1988))

This problem is to optimize the gear ratio for the compound gear train. This problem shown in Figure 9.3 was introduced by Sandgren (1988). 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,

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

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.

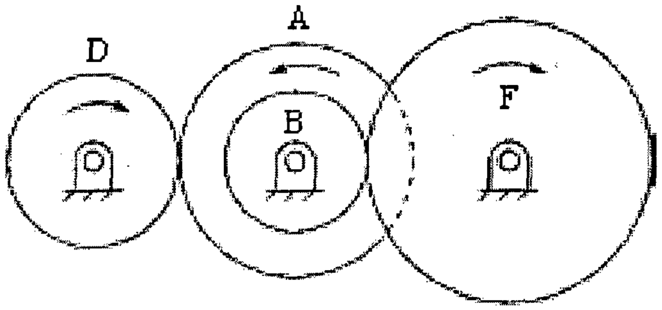

Figure 9.3 Compound gear train

This problem is an unconstrained optimization problem; it has 4 decision variables and 9 boundary constraints.

9.13 Methods of Solution and Results Discussion

All the constrained real life problems are solved by using the constraint handling algorithm (ICPSO and ICDE) discussed in chapter 6. For solving unconstrained real life problem, the algorithms which gave the best results in chapter 2 to 5 are used. In order to make a fair comparison of all algorithms, same seed for random number generation is fixed so that the initial population is same for all the algorithms. The population size is taken as 50. For DE versions, the crossover constant CR is set as 0. 5 and the scaling factor F is set as 0.5. For PSO algorithms, the acceleration coefficients c1 = c2 = 2.0 and the inertia weight w linearly decreases from 0.9 to 0.4. For each algorithm, the stopping criteria is to terminate the search process when one of the following conditions is satisfied: (i) the maximum number of generations is reached (assumed 2000 generations), (ii) $|f_{\max} - f_{\min}| < 10^{-4}$ where f is the value of objective function. A total of 50 runs for each experimental setting were conducted and the best solution throughout the run was recorded as global optimum. For comparison, previously quoted results in the literature are used. The numerical results of constrained and unconstrained real life problems are given in Table 9.1 and 9.2 respectively.

The first seven problems are constraint optimization problems. Out of this seven, ICDE gave a better performance in 3 cases; in one test case ICPSO is better than ICDE and the remaining 3 test cases both the algorithms perform the same. In comparison of the results of ICPSO and

ICDE with the quoted results in the literature, it can be seen that both the algorithms are better than the source results. Likewise, for unconstrained problems also PSO and DE versions performed well in all the test cases in comparison with the results of basic versions of PSO and DE and the quoted results in the literature.

Table 9.1 Results of constrained real life problems using ICDE and ICPSO algorithms

Algorithm	Best Fitness	Average Fitness	Worst Fitness	Standard deviation	NFE	Time (sec)	Source result
Static Power Scheduling Problem							
ICPSO	5048.46	5109.37	5136.04	37.77	2658	0.89	Time: 3.8 sec
ICDE	5046.69	5102.58	5175.51	53.29	3760	0.62	
Dynamic Power Scheduling Problem							
ICPSO	664.015	703.223	742.139	17.540	4212	1.68	Time: 40.7 sec
ICDE	661.719	721.045	730.654	18.2442	3978	1.44	
Transformer Design							
ICPSO	86.648	87.036	87.394	0.2696	41324	0.88	Time: 3.5 sec
ICDE	86.601	87.617	89.37	2.2033	53245	1.23	
Weight Minimization of a Speed Reducer							
ICPSO	2863.36	2863.36	2863.36	1.56e-05	3802	0.52	Fitness: 2994.47
ICDE	2863.36	2863.36	2863.36	1.84e-05	7458	1.08	
Heat Exchanger Network Design							
ICPSO	7049.25	7049.25	7049.25	6.17e-05	6316	0.18	Fitness: 7049.25
ICDE	7049.25	7049.25	7049.25	3.33e-05	7598	0.16	
Gas Transmission Compressor Design							
ICPSO	2.963e+06	2.963e+06	2.963e+06	8.79e-06	14634	0.56	Fitness: 2.99e+06
ICDE	2.963e+06	2.963e+06	2.963e+06	3.17 e-06	6640	0.28	
Optimal Design of Industrial Refrigeration System							
ICPSO	13646.5	13646.5	13646.5	7.82e-05	72312	8.34	Fitness: 19230.0
ICDE	13646.6	14282.5	15373.3	1.18	96749	9.83	

Table 9.2 Results of unconstrained real life problems using PSO, BTPSO, QIPSO3, DE, LDE4 and S-MDE algorithms

Transistor Modeling							
Item	PSO	BTPSO	QIPSO3	DE	LDE4	S-MDE	Source Result
x1	0.9010	0.9004	0.9019	0.9010	0.9010	0.9016	0.90
x2	0.8841	0.5224	0.8951	0.8856	0.6535	0.8770	0.45
x3	4.0386	1.0764	3.6675	4.0593	1.4207	3.5323	1.0
x4	4.1488	1.9494	3.6735	4.1728	2.0913	3.6724	2.0
x5	5.2436	7.8536	5.4421	5.2300	7.2996	5.5123	8.0
x6	9.9326	8.8364	11.2697	9.8842	10.00	10.802	8.0
x7	0.1009	4.7712	0.0979	0.0259	4.0985	0.5626	5.0
x8	1.0599	1.0074	1.1053	1.0625	1.0097	1.0746	1.0
x9	0.8066	1.8545	0.6799	0.8024	1.5988	0.7965	2.0
f(x)	0.0695	0.0113	0.0618	0.0673	0.0514	0.0657	NA
NFE	22195	17845	20743	21761	16423	19860	NA
Time	0.86	0.36	0.42	0.93	0.33	0.40	NA
Optimal Capacity of Gas Production Facilities							
x_1	17.5	17.5	17.5	17.5	17.5	17.5	17.5
x_2	600	600	600	600	600	600	465
f(x)	169.844	169.844	169.844	169.844	169.844	169.844	173.76
NFE	342	270	324	483	423	367	NA
Time	0.02	0.01	0.02	0.02	0.02	0.02	NA
Optimal Thermohydraulic Performance of an Artificially Roughened Air Heater							
x_1	0.05809	0.134009	0.032359	0.12469	0.15301	0.08508	0.052
x_2	10	10	10	10	10	10	10
x_3	10400.2	3000	16643.4	3811.07	3000	5935.45	10258
f(x)	4.21422	4.21422	4.21422	4.21422	4.21422	4.21422	4.182
NFE	6207	5190	4425	3652	3115	2947	NA
Time	0.3	0.3	0.3	0.2	0.2	0.2	NA
Design of Gear Train							
x1	13	16	19	16	19	19	18
x2	31	19	16	19	16	16	22
x3	57	49	43	49	43	43	45
x4	49	43	49	43	49	49	60
f(x)	9.98e-11	2.78e-12	2.78e-12	2.78e-12	2.78e-12	2.78e-12	5.7e-06
Gear ratio	0.14429	0.14428	0.14428	0.14428	0.14428	0.14428	0.14666
Error (%)	0.007398	0.000467	0.000467	0.000467	0.000467	0.000467	1.65
NFE	480	340	256	794	658	742	NA
Time	0.1	0.01	0.01	0.1	0.02	0.1	NA

9.14 Conclusion

This chapter investigated the performance of two popular; population based Evolutionary Algorithms Particle Swarm Optimization, Differential Evolution and their improved versions (presented in chapter 2, 3, 4, 5 and 6) on 11 real life problems taken from different fields. Out of these first 7 problems are constrained while the remaining four are unconstrained in nature. The constrained handling method described in Chapter 6 is used for solving the first 7 problems. For the unconstrained problems best results obtained by using the algorithms discussed in Chapters 2 to 5 are given. The simulation results show that, PSO, DE and their improved versions not only gave a better solution than the one quoted in the literature but they were also very time effective. Thus it may be concluded that PSO, DE and their improved versions can be used for solving engineering optimization problems.

Chapter 10

Conclusions and Future Scope

[In this chapter, the concluding observations based on this thesis are presented. In section 10.1, the conclusions of this study are stated and in section 10.2, salient features of the algorithms developed in this thesis are given. Finally in section 10.3, future research work in this direction is suggested.]

10.1 Conclusions

The objective of this study was to develop computational algorithms for obtaining the global optimal solutions of unconstrained and constrained nonlinear optimization problems. The aim was to solve not only benchmark test problems appearing in literature but also to solve challenging real life optimization problems arising in various disciplines. The focus of the present work is on the modifications of Particle Swarm Optimization and Differential Evolution algorithms primarily because of their popularity and wide applicability.

Various modifications were tested and implied on the basic structure of PSO and DE algorithms to further enhance their performance. These algorithms can be classified as:

➢ *Algorithms based on initial generation of random numbers*

Under this modification various distributions along with quasi random sequences were used to generate the initial population for PSO and DE algorithms. In all, 12 modified versions (GPSO, EPSO, BTPSO, GAPSO, VC-PSO, SO-PSO, GDE, EDE, BTDE, GADE, VC-DE and SO-DE) were proposed in which only the initial distribution of random numbers was changed from uniform to other probability distributions and quasi random sequences.

The simulation results showed that a significant improvement can be made in the performance of PSO and DE by simply changing the distribution of random numbers to other than uniform distribution and to quasi random sequence as the proposed algorithms outperformed the basic versions by a noticeable percentage.

➢ *Modified PSO algorithms*

The next focus was on the design and implementation of improved PSO algorithms based on the velocity vector, inertia weight, diversity, mutation and Quadratic Interpolation based crossover. For mutation the probability distributions Gaussian, Beta and Gamma and the low discrepancy Sobol sequence were used. Nineteen improved PSO algorithms were proposed: ATREPSO, GMPSO, BMPSO, GAMPSO, BGMPSO, QIPSO1, QIPSO2, QIPSO3, QIPSO4, SMPSO1, SMPSO2, GWPSO+GD, GWPSO+ED, GWPSO+UD, MPSO, Q-QPSO1, Q-QPSO2, SMQPSO1 and SMQPSO2.

- *Modified DE algorithms*

The next focus was on the design and implementation of improved DE algorithms based on the mutant vector, scale factor F and the crossover rate Cr. Two new mutant vectors based on the Laplace probability distribution (LDE) and on the concept of Quadratic Interpolation (DE-QI) were proposed. Five versions of LDE were proposed namely LDE1, LDE2, LDE3, LDE4 and LDE5. Also, an improved version of DE with adaptive control parameters (ACDE) was presented.

- *Hybridized algorithms*

One of the class of modified algorithms consists of the hybridization of algorithms, where the two algorithms are combined together to form a new algorithm. Three hybrid two phase global optimization algorithms namely DE-PSO, MDE and AMPSO algorithms were proposed. Based on the generation of initial population, three versions of MDE were given: U-MDE, G-MDE and S-MDE.

- *Constraint optimization algorithm*

A new constraint handling mechanism for solving constrained optimization problems was proposed. It is a simple approach for handing constraints and do not need any additional parameters. Based on the new constraint handling mechanism, two algorithms were presented namely ICPSO and ICDE. The performance of ICPSO and ICDE algorithms were validated on twenty constrained benchmark problems and compared with two other variants (constraint) of PSO and DE in the literature.

- *Real life problems*

The first real life problem is taken from the field of Electrical Engineering. The problem is to determine the In-Situ efficiency of Induction Motor without performing no-load test, which is not easily possible for the motors working in process industries where continuous operation is required. This problem was modeled as an unconstrained optimization problem and was framed by four different methods. The differences in the method were based on the number of input parameters used to the optimization algorithms and modifications in the equivalent circuit of the motor. Basic versions of PSO, DE and their six variants namely QPSO, ATREPSO, GMPSO, SMPSO1, LDE1 and DE-QI were used to solve this problem.

The second real life problem is also taken from the field of Electrical Engineering. The problem is to compute the values of the decision variables called relays, which control the act of isolation of faulty lines from the system without disturbing the healthy lines. This problem was modeled as a nonlinear constrained optimization problem, in which the objective function to be minimized is the sum of the operating times of all the relays, which are expected to operate in order to clear the faults of their corresponding zones. Three cases of the IEEE Bus system were considered namely, IEEE 3-bus, IEEE 4-bus and IEEE 6-bus system. This problem was solved by using the family of DE algorithms namely LDE1, LDE2, LDE3, LDE4, LDE5 and DE-QI.

Finally, a collection of eleven real life problems, taken from various fields of Science and Engineering, were given. Out of eleven problems, seven problems were constrained real life problems and four problems were unconstrained real life problems. All the problems were analyzed with both PSO and DE family of algorithms and were compared with the results in the literature. Empirical results showed that the families of proposed PSO and DE algorithms were quite competent for solving the considered real life problems.

10.2 Salient Features of the Algorithms Developed in the Thesis

This section describes the features of the proposed algorithms in this thesis.

- All the algorithms developed in the present work were tested on a wide range of test problems occurring in literature and were also compared with other versions of PSO and DE algorithms. The numerical analysis of results showed that even a simple modification like changing the initial generation can make a significant difference in the performance of the algorithm.
- While comparing the algorithms presented in the thesis with other versions available in literature, different parameter settings and performance measures were considered

according to the parameter settings and performance measures given in literature. This was done in order to have a fair comparison of the proposed algorithms with the available versions.

- The algorithms proposed in the present study are simple to imply and can be understood by even a person of non-mathematical background.
- Most of the modifications like different probability distributions for generation of random numbers or the selection and sorting rules for constrained optimization suggested in the present work are generic in nature and can be applied to any population based search technique.

10.3 Future Scope

The process of research is an everlasting and iterative process. This work is no exception to it. Several modifications can be incorporated in the present work. A few of them are listed below.

(1) The present study deals with only single objective optimization problems, work can be done in extending these algorithms to multi objective optimization problems as well.

(2) The algorithms are developed for the continuous optimization problems. However there are several real life optimization problems that have discrete variables. The algorithms developed in the present thesis can be suitably modified for discrete/ combinatorial cases.

(3) An extensive empirical analysis of numerical results has been done in the present work. It would be interesting to research on the theoretical analysis of the operators used while modifying various algorithms of the thesis.

Publications from this Work

Book Chapters

1. Millie Pant, Radha Thangaraj and Ajith Abraham, "Performance Tuning of Particle Swarm Optimization: An Investigation and Empirical Analysis", Foundations on Computational Intelligence, Vol. 3: Global Optimization: Theoretical Foundations and Applications, Studies in Computational Intelligence Series, Springer Verlag, Germany, ISBN: 978-3-642-01084-2, pp. 101 – 128, 2009.

Peer Reviewed International Journal Papers

2. Millie Pant, Radha Thangaraj and V. P. Singh, "A New Diversity Based Particle Swarm Optimization using Gaussian Mutation", Int. J. of Mathematical Modeling, Simulation and Applications, Vol. 1(1), pp. 47 – 60, 2008.
3. Millie Pant, Radha Thangaraj and V. P. Singh, "Efficiency Optimization of Electric motors: A Comparative Study of Stochastic Algorithms", World Journal of Modeling and Simulation, Vol. 4(2), pp.140 – 148, 2008.
4. Millie Pant, P. Sharma, T. Radha, R. S. Sangwan and U. Roy, "Nonlinear Optimization of Enzyme Kinetic Parameters", Journal of Biological Sciences, Vol. 8(8), pp. 1322 – 1327, 2008.
5. Millie Pant, Radha Thangaraj and V. P. Singh, "Particle Swarm Optimization with Crossover Operator and its Engineering Applications", Int. Journal of Computer Science, Vol. 36(2), pp. 112 – 121, 2009.
6. Millie Pant, Radha Thangaraj and V. P. Singh, "Sobol Mutated Quantum Particle Swarm Optimization", Int. Journal of Recent Trends in Engineering, Vol. 1(1), pp. 95 – 99, 2009.
7. Millie Pant, Radha Thangaraj and V. P. Singh, "Optimization of Mechanical Design Problems using Improved Differential Evolution Algorithm", Int. Journal of Recent Trends in Engineering, Vol. 1(5), pp. 21 – 25, 2009.

8. Millie Pant, Radha Thangaraj and Ajith Abraham, "DE-PSO: A New Hybrid Meta-Heuristic for solving Global Optimization Problems", New Mathematics and Natural Computation, World Scientific, 2009, Accepted.
9. Millie Pant, Radha Thangaraj and Ajith Abraham, "Low Discrepancy Initialized Particle Swarm Optimization for Solving Constrained Optimization Problems", Special Issue on Swarm Intelligence: Foundations and Applications, Fundamenta Informaticae, IOS Press, 2009, Accepted.

National Journal Papers

10. Millie Pant, Radha Thangaraj, Ajith Abraham and V. P. Singh, "Optimal Tuning of PI Speed Controller in PMSM: A Comparative Study of Evolutionary Algorithms", Journal of Electrical Engineering, Vol. 2(1), pp. 36 – 43, 2008.

Peer Reviewed International Conference Papers

11. Millie Pant, Radha Thangaraj and Ajith Abraham, "A New PSO Algorithm Incorporating Reproduction Operator for Solving Global Optimization Problems", 7th International Conference on Hybrid Intelligent Systems (HIS'07), Kaiserslautern, Germany, IEEE Computer Society press, USA, ISBN 07695-2662-4, pp. 144-149, 2007.
12. Millie Pant, Radha Thangaraj and V. P. Singh, "A New Particle Swarm Optimization with Quadratic Interpolation", Int. Conf. on Computational Intelligence and Multimedia Applications (ICCIMA'07), India, IEEE Computer Society Press, Vol. 1, pp. 55 – 60, 2007.
13. Millie Pant, Radha Thangaraj and V. P. Singh, "A Simple Diversity Guided Particle Swarm Optimization", IEEE Congress on Evolutionary Computation (CEC'07), Singapore, pp. 3294 – 3299, 2007.
14. Millie Pant, Radha Thangaraj and Ajith Abraham, "A New PSO Algorithm with Crossover Operator for Global Optimization Problems", HAIS'07: Advances in Soft computing Series, Springer Verlag, Germany, E. Corchado et al. (Eds.): Innovations in Hybrid Intelligent Systems, Vol. 44, pp. 215 - 222, 2007.

15. Millie Pant, Radha Thangaraj and V. P. Singh, "Particle Swarm Optimization: Experimenting the Distributions of Random Numbers", 3rd Indian Int. Conf. on Artificial Intelligence (IICAI'07), India, pp. 412 – 420, 2007.
16. Millie Pant, Radha Thangaraj and V. P. Singh, "A New Particle Swarm Optimization with Quadratic Crossover", Int. Conf. on Advanced Computing and Communications (ADCOM'07), India, IEEE Computer Society Press, pp. 81 – 86, 2007.
17. Millie Pant, Radha Thangaraj and V. P. Singh, "Particle Swarm Optimization using Gaussian Inertia Weight", Int. Conf. on Computational Intelligence and Multimedia Applications (ICCIMA'07), India, IEEE Computer Society Press, Vol. 1, pp. 97 – 102, 2007.
18. Millie Pant, Radha Thangaraj and Ajith Abraham, "Optimization of a Kraft Pulping System: Using Particle Swarm Optimization and Differential Evolution", 2nd Asia Int. Conf. on Modeling and Simulation (AMS'08), Malaysia, IEEE Computer Society Press, USA, pp. 637 – 641, 2008.
19. Millie Pant, Radha Thangaraj, Crina Grosan and Ajith Abraham, "Improved Particle Swarm Optimization with Low-discrepancy Sequences", IEEE Congress on Evolutionary Computation (CEC'08), Hong Kong, pp. 3016 – 3023, 2008.
20. Millie Pant, Radha Thangaraj, V. P. Singh and Ajith Abraham, "Particle Swarm Optimization Using Sobol Mutation", Int. Conf. on Emerging Trends in Engineering and Technology (ICETET'08), India, IEEE Computer Society Press, pp. 367 – 372, 2008.
21. Millie Pant, Radha Thangaraj, Deepti Rani, Ajith Abraham and Dinesh K. Srivastava, "Estimation using Differential Evolution for Optimal Crop Plan", HAIS'08: Lecture Notes in Computer Science: Hybrid Artificial Intelligent Systems, Springer Verlag, Germany, Vol. 5271, pp. 289 – 297, 2008.
22. Millie Pant, Radha Thangaraj and Ajith Abraham, "Particle Swarm Based Meta-heuristics for Function Optimization and Engineering Applications", 8th Int. Conference on Computer Information Systems and Industrial Management Applications (CISIM'08), Ostrava, Czech Republic, IEEE Computer Society Press, pp. 84 - 90, 2008.

23. Millie Pant, Radha Thangaraj and Ajith Abraham, "A New Quantum Behaved Particle Swarm Optimization", ACM SIGEVO Conference: Genetic and Evolutionary Computation Conference (GECCO'08), USA, 2008, pp. 87 - 94.
24. Millie Pant, Radha Thangaraj and Ajith Abraham, "Particle Swarm Optimization using Adaptive Mutation", 19th Int. Conf. on Database and Expert Systems Application (ETID'08), Italy, IEEE Computer Society Press, pp. 519 – 523, 2008.
25. Millie Pant, Radha Thangaraj and Ajith Abraham, "Optimal Tuning of PI Speed Controller using Nature Inspired Heuristics", 8th Int. Conf. on Intelligent Systems Design and Applications (ISDA'08), Taiwan, IEEE Computer Society Press, pp. 420 – 425, 2008.
26. Millie Pant, Radha Thangaraj and V. P. Singh, "Speed Optimization of Servo Motor in Optical Disc Systems Using Particle Swarm Optimization", POWERCON 2008 & 2008 IEEE Power India Conference, India, pp. 1 – 4, 2008.
27. Millie Pant, Radha Thangaraj, Crina Grosan and Ajith Abraham, "Hybrid Differential Evolution – Particle Swarm Optimization Algorithm for Solving Global Optimization Problems", 3rd IEEE Int. Conf. on Digital Information Management (ICDIM'08), UK, IEEE Computer Society Press, pp. 18 – 24, 2008.
28. Millie Pant, Radha Thangaraj and V. P. Singh, "A New Differential Evolution Algorithm for Solving Global Optimization Problems", Int. Conf. on Advanced Computer Control (ICACC'09), Singapore, IEEE Computer Society Press, pp. 388 – 392, 2009.
29. Millie Pant, Radha Thangaraj and V. P. Singh, "The Economic Optimization of Pulp and Paper Making Processes Using Computational Intelligence", Int. Conf. on Modeling of Engineering and Technical Problems (ICMETP'09), Agra, India.
30. Millie Pant, Radha Thangaraj, Ajith Abraham and C. Grosan, "Differential Evolution with Laplace Mutation Operator", IEEE Congress on Evolutionary Computation (CEC'09), Norway, pp. 2841-2849, 2009.
31. Radha Thangaraj, Millie Pant, Ajith Abraham and Youakim Badr, "Hybrid Evolutionary Algorithm for Solving Global Optimization Problems", HAIS'09: Lecture Notes in

Computer Science: Hybrid Artificial Intelligent Systems, Springer Verlag, Germany, Vol. 5572, pp. 310 – 318, 2009.

32. Radha Thangaraj, Millie Pant and Atulya K Nagar, "Maximization of Expected Target Damage Value Using Quantum Particle Swarm Optimization", Int. Conf. on Developments in eSystems Engineering (DeSE'09), Abu Dhabi, UAE, IEEE Computer Society Press, 2009, Accepted.
33. Radha Thangaraj, Millie Pant and Ajith Abraham, "A Simple Adaptive Differential Evolution Algorithm", World Congress on Nature and Biologically Inspired Computing (NaBIC'09), India, IEEE Computer Society Press, 2009, Accepted.
34. Radha Thangaraj, Millie Pant and Ajith Abraham, "A New Diversity Guided Particle Swarm Optimization with Mutation", World Congress on Nature and Biologically Inspired Computing (NaBIC'09), Special Session on Recent Advances in the Development of Particle Swarm Optimization for Global Optimization Problems, India, IEEE Computer Society Press, 2009, Accepted.
35. Radha Thangaraj, Millie Pant and Ajith Abraham, "Evolutionary Algorithms based Speed Optimization of Servo Motor in Optical Disc systems", 8^{th} Int. Conference on Computer Information Systems and Industrial Management Applications (CISIM'08), India, IEEE Computer Society Press, 2009, Accepted.
36. Radha Thangaraj and Millie Pant, "Optimization of Directional Overcurrent Relay Setting by Differential Evolution Algorithm", International Conference on Power Systems (ICPS'09), IIT Kharagpur, 2009, Accepted.

National Conferences

37. Millie Pant, Radha Thangaraj and V. P. Singh, "A Comparison of Three Stochastic Algorithms for Solving Global Optimization Problems", National Conference on Mathematical Modeling, Optimization and Their Application (OPTIMA'07), India, 2007.
38. Millie Pant, Radha Thangaraj and V. P. Singh, "Industrial Application of Particle Swarm Optimization: An Example of Pulp and Paper Industry", 40th Annual Convention of ORSI, India, 2007.

39. Millie Pant, Radha Thangaraj and V. P. Singh, "Application of Particle Swarm Optimization in Electrical Engineering", 11th Panjab National Congress, India, 2008.

Communicated Journal Papers

40. Millie Pant, Radha Thangaraj, V. P. Singh and Ajith Abraham, "Particle Swarm Optimization Using Sobol Mutation", Submitted to Int. Journal of Simulation Systems, Science and Technology, 2008.
41. Millie Pant, Radha Thangaraj, Deepti Rani, Ajith Abraham and Dinesh K. Srivastava, "Evolutionary Algorithms for Optimal Crop Plan Estimation", Submitted to Water Resources Management, Springer Verlag, 2009.
42. Millie Pant, Radha Thangaraj and Atulya K. Nagar, "Particle Swarm Optimization and its Engineering Applications", Submitted to Int. J. Artificial Intelligence and Soft Computing, Inderscience, 2008.
43. Radha Thangaraj, Millie Pant, Thanga Raj C and Atulya K Nagar, "In-Situ Efficiency Determination of Induction Motor: A Comparative Study of Evolutionary Techniques", Submitted to Applied Artificial Intelligence An International Journal, Taylor and Francis, 2009.
44. Radha Thangaraj, Thanga Raj C, Millie Pant and Ajith Abraham, "Optimal Gain-Tuning of PI Speed Controller in Induction Motor Drives Using Particle Swarm Optimization", Submitted to Neural Network World, 2009.
45. Radha Thangaraj, Millie Pant and Kusum Deep, "Optimal Coordination of Overcurrent Relays using Modified Differential Evolution Algorithms", Submitted to Special Issue: Advances in Metaheuristics, Engineering Applications of Artificial Intelligence, Elsevier Science, 2009.
46. Radha Thangaraj, Millie Pant and Ajith Abraham, "New Mutation Schemes for Differential Evolution Algorithm and their application to the Optimization of Directional Overcurrent Relay Settings", Submitted to Applied Mathematics and Computation, Elsevier Science, 2009.

47. Radha Thangaraj, Millie Pant and Ajith Abraham, "Modified Differential Evolution Algorithm with Laplace Mutation", Submitted to Int. J. of Innovative Computing, Information and Control, 2009.
48. Radha Thangaraj and Millie Pant, "Modified Particle Swarm Optimization with Time Varying Velocity Vector", Submitted to Int. J. of Computational Intelligence Studies, Inderscience, 2009.
49. Radha Thangaraj, Millie Pant and Ajith Abraham, "Modified Differential Evolution Algorithm for Constrained Global Optimization", Submitted to Special Issue: Learning and Intelligent Optimization, Annals of Mathematics and Artificial Intelligence, Springer Verlag, 2009.

Communicated International Conference Papers

50. Radha Thangaraj, Millie Pant and Kusum Deep, "Initializing PSO with Probability Distributions and Low-discrepancy Sequences: The Comparative Results", Submitted to Bio Inspired Computing and Applications (BICA'09), India, 2009.

Bibliography

Abbass, H. and Sarker, R. (2002): The Pareto Differential Evolution Algorithm, International Journal of Artificial Intelligence Tools, World Scientific Publishing Company, Vol. 11(4), pp. 531 - 552.

Abdelaziz, A. Y., Talaat, H. E. A., Nosseir, A. I. and Hajjar, A. A. (2002): An Adaptive Protection Scheme for Optimal Coordination of Overcurrent Relays, Electric Power Systems Research, Vol. 61 (1), pp. 1 - 9.

Abido, M. A. (2002): Optimal Design of Power System Stabilizers using Particle Swarm Optimization, IEEE Transactions on Energy Conversion, Vol. 17(3), pp. 406 - 413.

Abyane, H. A., Faez, K. and Karegar, H. K. (1997): A New Method for Overcurrent Relay (O/C) using Neural Network and Fuzzy Logic, In Proc. IEEE TENCON, Speed and Image Technologies for Computing and Telecommunications, pp. 407 - 410.

Abyaneh, H. A. and Keyhani, R. (1995): Optimal Coordination of Overcurrent Relays in Power System by Dual Simplex Method, In Proceedings of AUPEC Conference, Perth, Australia, pp. 440 - 445.

Ackley, D. H. (1987): A Connectionist Machine for Menetic Hillclimbing, Boston: Kluwer Academic Publishers.

Adjiman, C. S., Dallwig, S., Floudas, C.A. and Neumaier, A. (1998): A Global Optimization Method, QBB, for General Twice – Differentiable Constrained NLPs – I. Theoretical Advances, Computers and Chemical Engineering, Vol. 22, pp. 1137 – 1158.

Ali, M. M and Torn, A. (2003): Population Set Based Global Optimization Algorithms: Some Modifications and Numerical Studies, www.ima.umn.edu/preprints/.

Ali, M. M. (2007): Differential Evolution with Preferential Crossover, European Journal of Operational Research, Vol. 181, pp. 1137 – 1147.

Ali, M. M. and Torn, A. (2004): Population Set-based Global Optimization Algorithms: Some Modifications and Numerical Studies, Computers & Operations Research, Vol. 31(10), pp. 1703 – 1725.

Angel, E. M - Z, Arturo, H – A. and Enrique, R. Villa-Diharce and Salvador, B – R. (2006): PESO+ for Constrained Optimization, IEEE Congress on Evolutionary Computation, pp. 231 – 238.

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

Babu, B. V. and Angira, R. (2008): Optimization of Industrial Processes Using Improved and Modified Differential Evolution, Studies in Fuzziness and Soft Computing, Vol. 226, pp. 1 – 22.

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 Computation, Vol. 7 (1), pp. 109 – 124.

Bansal, J. C. and Deep, K. (2008): Optimization of Directional Overcurrent Relay Times by Particle Swarm Optimization, In Proc. of IEEE Swarm Intellignt Symposium, USA, pp. 1 - 7.

Baskar, S. and Suganthan, P. N. (2004): A Novel Concurrent Particle Swarm Optimization, In Proc. of IEEE Congress on Evolutionary Computation, Vol. 1, pp. 792–796.

B-Biggs, M. C. (1978): A Numerical Comparison between Two Approaches to the Nonlinear Programming Problem, In: L. C. W. Dixon and G. P. Szego, eds., Towards Global Optimization 2, Amsterdam, Holland: North Holland Publishing Company, pp. 293 – 312.

Bector, C. R., Chandra, S. and Dutta, J. (2005): Principles of Optimization Theory, Narosa Publishing House, New Delhi, India.

Beightler, C. S and Phillips D. T. (1976): Applied Geometric Programming, Jhon Wiley & Sons, New York.

Benaidja, N. and Khenfer, N. (2006): Identification of Asynchronous Machine Parameters by Evolutionary Techniques, Electric Power Components and systems, Vol. 34, pp. 1359-1376.

Bergh, F. and Engelbrecht, A. (2004): A Cooperative Approach to Particle Swarm Optimization, IEEE Transaction on Evolutionary Computation, Vol. 8 (3), pp. 225–239.

Birla, D., Maheshwari, R. P. and Gupta, H. O. (2005): Time-overcurrent relay coordination: a review, Emerging Electric Power Systems, Vol. 2 (2), pp. 1 - 13.

Birla, D., Maheshwari, R. P. and Gupta, H. O. (2006): A New Nonlinear Directional Overcurrent Relay Coordination Technique, and Banes and Boons of Near-End Faults Based Approach, IEEE Transaction on Power Delivery, Vol. 21 (3), pp. 1176 – 1182.

Birla, D., Maheswari R. P., Gupta, H. O., Deep, K. and Thakur, M. (2006a): Application of Random Search Technique in Overcurrent Relay Coordination, Emerging Electric Power Systems, Vol. 7 (1), pp. 1 – 14.

Blackwell, T.M. and Bentley, P. J. (2002): Dynamic Search with Charged Swarms, In Proc. of the genetic and Evolutionary Computation Conference, pp. 19 – 26.

Bounekhla, M., Zaim, M. E. and Rezzoug, A. (2005): Comparative study of three minimization methods applied to the induction machine parameters identification using transient stator current, Electric Power Components and systems, Vol. 33, pp. 913-930.

Bracken, J. and McCormick, G. P. (1968): Selected Applications of Non – Linear Programming, Springer Verlag.

Braga, A. S. and Saraiva J. T. (1996): Co-ordination of Directional Overcurrent Relays in Meshed Networks using Simplex Method, In Proc. of the IEEE MELECON Conference, pp. 1535 - 1538.

Branin, F. K. (1972): A Widely Convergent Method for finding Multiple Solutions of Simultaneous Non Linear Equations, IBM J. Res. Develop., pp. 504 – 522.

Brest, J., Boˇskovi´c, B., Greiner, S., ˇZ umer, V. and Mauˇcec, M. S. (2006): Performance Comparison of Self-Adaptive and Adaptive Differential Evolution Algorithms, Technical Report #2006-1-LABRAJ, University of Maribor, Faculty of Electrical Engineering and Computer Science, Slovenia, http://marcel.uni-mb.si/janez/brest-TR1.html.

Brest, J., Greiner, S., Boˇskovi´c, B., Mernik, M. and ˇZumer, V. (2006a): Self- Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems, IEEE Transactions on Evolutionary Computation, Vol. 10(6), pp. 646 – 657.

Brest, J., Zumer, V. and Mausec, M. S. (2006b): Self-Adaptive Differential Evolution Algorithm in Constrained Real-Parameter Optimization, IEEE Congress on Evolutionary Computation, pp. 216 – 222.

Brits, R. and Engelbrecht, A.P. and van den Bergh, F. (2002): A Niching Particle Swarm Optimizater, In proc. of the fourth Asia Pacific Conference on Simulated Evolution and learning, pp 692 – 696.

Brits, R. and Engelbrecht, A.P. and van den Bergh, F. (2002a): 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.

Brown, K. and Tyle, N. (1986): An Expert System for Overcurrent Protective Device Coordination, Mc Graw-Edison Power System Division, In Proc. of IEEE Rural Electric Power System Conference, Apr. 20 - 22.

Bulger, D., Baritompa, W. P. and Wood, G. R. (2003): Implementing pure adaptive search with Grover's quantum algorithm, *J. of Optimization: Theory and Application*, Vol. 116(3), pp. 517 – 529.

Carlisle, A. and Dozier, G. (2001): An off-the-shelf PSO, In Proceedings of the workshop on Particle Swarm Optimization, Indianapolis, USA, pp. 1 – 6.

Chakraborty, U. K. (2008): Advances in Differential Evolution, (Ed.) Springer-Verlag, Heidelberg.

Chattopadhyay, B., Sachdev, M. S. and Sidhu, T. S. (1996): An On-Line Relay Coordination Algorithm for Adaptive protection using Linear Programming Technique, IEEE Transaction on Power Delivery, Vol. 11 (1), pp. 165 - 173.

Chi, H. M., Beerli, P., Evans, D. W. and Mascagni, M. (1999): On the Scrambled Sobol Sequence, In Proc. of Workshop on Parellel Monte Carlo Algorithms for Diverse Applications in a Distributed Setting, LNCS 3516, Springer Verlog, pp. 775 – 782.

Clerc, M. (1999): 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.

Clerc, M. (2001): Think Locally, Act Locally: The Way of Life of Cheap-PSO, an Adaptive PSO, Technical Report, http: // clerc.maurice.free.fr/pso.

Clerc, M. and Kennedy, J. (2002): The Particle Swarm-Explosion, Stability, and Convergence in a Multimodal Complex Space, IEEE Trans. on Evolutionary Computation, Vol. 6(1), pp. 58 – 73.

Coelho, L. S. (2006): A Quantum Particle Swarm Optimizer with Chaotic Mutation Operator, Chaos, Solitons and Fractals, Vol. 37(5), pp. 1409 – 1418.

Coelho, L. S., Nedjah, N. and Mourelle, L. M. (2008): Gaussian Quantum-Behaved Particle Swarm Optimization Applied to Fuzzy PID Controller Design, Studies in Computational Intelligence, Vol. 121, pp. 1 – 15.

Das, S. and Konar, A. (2006): Design of Two Dimensional IIR Filters with Modern Search Heuristics: A Comparative Study", International Journal of Computational Intelligence and Applications, World Scientific Press, Vol. 6(3).

Das, S., Abraham, A. and Konar, A. (2008): Adaptive Clustering using Improved Differential Evolution Algorithm, IEEE Transactions on Systems, Man and Cybernetics – Part A, IEEE Press, USA, Vol. 38(1), pp. 218 - 237.

Das, S., Abraham, A. and Konar, A. (2008): Particle Swarm Optimization and Differential Evolution Algorithms: Technical Analysis, Applications and Hybridization Perspectives, Studies in Computational Intelligence: Advances of Computational Intelligence in Industrial Systems, Vol. 118, pp. 1 – 38.

Das, S., Konar, A. and Chakraborty, U. K. (2005): Two Improved Differential Evolution Schemes for Faster Global Search, ACM-SIGEVO Proceedings of GECCO, Washington D.C., pp. 991-998.

Deep, K., Birla, D., Maheshwari, R. P., Gupta, H.O. and Takur, M. (2006): A Population Based Heuristic Algorithm for Optimal Relay Operating Time, World Journal of Modelling and Simulation, Vol. 3 (3), pp. 167 - 176.

DeJong, K. (1975): An Analysis of the Behaviour of a Class of Genetic Adaptive Systems, Ph.D thesis, University of Michigan, Dissertation Abstracts International, Vlo. 36(10), 5140B, Univarsity Microfilms No. 76-9381.

Dipti (2007): Hybrid Genetic Algorithms and Their Applications, Ph.D. Thesis, Dept. of Mathematics, Indian Institute of Technology Roorkee, Roorkee, India.

Dixon, L. C. W. and Szego, G. P. (1978): The Optimization Problem: An Introduction , In Dixon, L. C. W. and Szego, G. P. (eds): Towards Global Optimization II, New York: North Holland.

Dong, D., Jie, J., Zeng, J. and Wang, M. (2008): Chaos-Mutation-Based Particle Swarm Optimizer for Dynamic Environment, In Proc. of the 3rd Int. Conf. on Intelligent System and Knowledge Engineering, pp. 1032 – 1037.

Eberhart, R. C. and Hu, X. (1999): Human Tremor Analysis using Particle Swarm Optimization, In Proc. of the IEEE Congress on Evolutionary Computation, pp. 1927 - 1930.

Eberhart, R. C. and Shi, Y. (1998): Comparison between Genetic Algorithms and Particle Swarm Optimization, In Proc. of 7th Annual Conf. on Evolutionary Programming, pp. 611 – 616.

Eberhart, R. C. and Shi, Y. (2000): Comparing Inertia Weights and Constriction Factors in Particle Swarm Optimization", *IEEE Congress on Evolutionary Computation*, Vol. 1, pp. 84 – 88.

Eberhart, R. C. and Shi, Y. (2001): Particle Swarm Optimization: Developments, Applications and Resources, in Proc. of IEEE Congress of Evolutionary Computation, Vol. 1, pp. 27 - 30.

Engelbrecht, A. P. (2005): Fundamentals of Computational Swarm Intelligence, England: John Wiley & Sons Ltd.

Esquivel, S. C and Coello Coello, C. A. (2003): On the Use of Particle Swarm Optimization with Multimodal Functions, In Proceedings of the 2003 Congress on Evolutionary Computation, IEEE Press, pp. 1130– 1136.

Fan, S.K.S. and Zahara, E. (2007): A Hybrid Simplex Search and Particle Swarm Optimization for Unconstrained Optimization, Eur. J. Operations Research, Vol. 181(2), pp.527-548.

Fan, S-K. S. and Chiu, Y-Y. (2007): A Decreasing Inertia weight Particle Swarm Optimizer, Engineering Optimization, Vol. 39(2), pp. 203 – 228.

Fan. S-K. S. and Chang, J-M. (2007): A Modified Particle Swarm optimizer Using an Adaptive Dynamic Inertia Weight, LNCS: Digital Human Modeling, Vol. 4561, pp. 56 – 65.

Farzad, R., Hossein A. A., Majid Al-D. (2008): A New Comprehensive Genetic Algorithm Method for Optimal Overcurrent Relays Coordination, Electrical Power System Research, Vol. 78, pp. 713 - 720.

Feng, B. and Xu, W. (2004): Adaptive Particle Swarm Optimization Based on Quantum Oscillator Model, In Proc. of the 2004 IEEE Conf. on Cybernetics and Intelligent Systems, Singapore, pp. 291 – 294.

Floudas, C. A., Pardalos, P. M., (1990): A collection of test problems for constrained global optimization algorithms, LNCS. Springer, Germany.

Floundas, C. (1999): Handbook of Test Problems in Local and Global Optimization, Nonconvex Optimization and its Applications, Kluwer Academic Publishers, The Netherlands, 1999.

Floundas, C. and Pardalos, P. (1987): A Collection of Test Problems for Constrained Global Optimization, volume 455 of Lecture Notes in Computer Science. Springer-Verlag, Berlin, Germany.

Fogel, L. J., Owens, A. J. and Walsh, M. J. (1965): Artificial intelligence through a simulation of evolution, In M. Maxfield, A. Callahan and L. J. Fogel, editors, Biophysics and Cybernetic systems, In Proc. of the 2nd Cybernetic Sciences Symposium, pp. 131 – 155.

Fred, G. (1986): Future Paths for Integer Programming and Links to Artificial Intelligence, Computers & Operations Research, Vol. 13 (5), pp. 533-549.

Gämperle, R., Müller, S. D. and Koumoutsakos, P. (2002): A Parameter Study for Differential Evolution, WSEAS NNA-FSFS-EC 2002. Interlaken, Switzerland, WSEAS, pp. 293 – 298.

Gehlhaar and Fogel (1996): Tuning Evolutionary Programming for Conformationally Flexible Molecular Docking, In Proc. of the fifth Annual Conference on Evolutionary Programming, pp. 419 – 429.

Gentle, E. J. (1998): RandomNumber Generation and Monte Carlo Methods, Springer-Verlog, Germany.

Glover, F. (1986): Future Paths for Integer Programming and Links to Artificial Intelligence, Computers and Operations Research, Vol. 13(5), pp. 533 – 549.

Goldberg, D. (1986): Genetic Algorithms in Search Optimization and Machine Learning, Addison Wesley Publishing Company, Reading, Massachutes.

Goldstein, A. A and Price, I. F. (1971): On Decent from Local Minima, Math. Comp., Vol. 25(115).

Griewank, A.O. (1981): Generalized Descend for Global Optimization, Journal of Optimization Theory and Applications, Vol. 34, pp. 11–39.

Grosan, C., Abraham, A. and Nicoara, M. (2005): Search Optimization Using Hybrid Particle Sub-Swarms and Evolutionary Algorithms, International Journal of Simulation Systems, Science & Technology, UK, Vol. 6(10, 11), pp. 60-79.

Halton, J. (1960): On the Efficiency of Certain Quasi-random Sequences of Points in Evaluating Multi-dimensional Integrals, Numerische Mathematik, Vol. 2, pp. 84 – 90.

Hao, Z-F., Guo, G-H. and Huang, H. (2007): A Particle Swarm Optimization Algorithm with Differential Evolution, In Proc. of the 6th Int. Conf. on Machine Learning and Cybernetics, pp. 1031 – 1035.

Hao, Z-F., Wang, Z-G. and Huang, H. (2007a): A Particle Swarm Optimization Algorithm with Crossover Operator, In Proc. of Int. Conf. on Machine Learning and Cybernetics, pp. 1036 – 1040.

Hassanien, A. E., Abraham, A., Kacprzyk, J. and Peters, J. (2008): Computational Intelligence in Multimedia Processing: Foundations and Trends, Computational Intelligence in Multimedia Processing, Aboul-Ella Hassanien et al. (Eds.), Springer Verlag, Germany, ISBN: 978-3-540-76826-5, pp. 3-49.

Hassanien, A. E., Milanova, M., Smolinski, T. and Abraham, A. (2008a): Computational Intelligence in Solving Bioinformatics Problems: Reviews, Perspectives and Challenges, Computational Intelligence in Biomedicine and Bioinformatics, Studies in Computational Intelligence, SCI 151, Springer Verlag, Germany, ISBN 978-3-540-70776-9, pp. 3-47, 2008.

He, S., Wen, J., Prempain, E., Wu, Q., Fitch, J. and Mann, S. (2004): An Improved Particle Swam Optimization for Optimal Power Flow, In Proc. of Int. Conf. Power Syst. Technol., pp. 1633–1637.

He, Z., Wei, C., Yang, L., Gao, X., Yao, S., Eberhart, R. C. and Shi, Y. (1998): Extracting Rules from Fuzzy Neural Network by Particle Swarm Optimization, In Proc. of IEEE Congress on Evolutionary Computation, Anchorage, Alaska, USA.

Hendtlass T. (2001): A Combined Swarm Differential Evolution Algorithm for Optimization Problems, In Proc. of 14th Int. conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems, Lecture Notes in Computer Science, Springer Verlag, Vol. 2070, pp. 11 – 18.

Higashi, H. and Iba, H. (2003): Particle Swarm Optimization with Gaussian Mutation, In Proc. of the IEEE Swarm Intelligence Symposium, pp. 72 – 79.

Himmelblau, D. M. (1972): Applied Non-Linear programming, New York: McGraw-Hill.

Ho, S.L., Yang, S.Y., Ni, G.Z: and Wong, K.F. (2007): An improved PSO method with application to multimodal functions of inverse problems, IEEE Trans. on Magnetics, Vol. 43(4):1597-1600.

Hock, W. and Schittkowski, K. (1981): Test Examples for Nonlinear Programming Codes, Lecture Notes in Economics and Mathematical Systems. Springer-Verlag, Berlin, Germany.

Hogg, T. and Portnov, D. S. (2000): Quantum Optimization, Information Sciences, Vol. 128, pp. 181 – 197.

Hong, H. W., Sun, C. T., Mesa, V. M. and Ng, S. (1991): Protective Device Coordination Expert System, IEEE Transactions on Power Delivery, Vol. 6 (1), pp. 359 - 365. http://www.ntu.edu.sg/home/EPNSugan

Hu, X., Eberhart, R. C. and Shi, Y. (2003): Swarm Intelligence for Permutation Optimization: A Case Study on n-Queens problem, In Proc. of IEEE Swarm Intelligence Symposium, pp. 243 – 246.

IEC Standard (1972): Rotating Electrical Machines—Part 2: Methods for Determining Losses and Efficiency of Electrical Machinery From Tests (Excluding Machines for Traction Vehicles) With Amendments 1: 1995 and 2: 1996, IEC Std 60 034-2.

IEEE Standard (2004): IEEE Power Engineering Society, IEEE Standard Test Procedure for Poly phase Induction Motors and Generators, IEEE Std: 112-2004, November 4, 2004.

Indian Standard (2002): Indian standard, Guide for Testing Three Phase Induction Motor, IS Std: 4029.

Irving, M. R. and Elrafie, H. B. (1993): Linear Programming for Directional Overcurrent Relay Coordination in Interconnected Power Systems with Constraint Relaxation, Electric Power Systems Research, Vol. 27 (3), pp. 209 - 216.

Jiang, W., Zhang, Y. and Xie, J. (2008): A Particle Swarm Optimization Algorithm Based on Diffusion-Repulsion and Application to Portfolio Selection, In Proc. of Int. Symposium on Information Science and Engineeing, pp. 498 – 501.

Jianping, W. and Trecat, J. (1996): RSVIES - a Relay Setting Value Identification Expert System, Electric Power Systems Research, Vol. 37, pp. 153 - 158.

Jiao, B., Lian, Z. and Gu, X. (2008): A Dynamic Inertia Weight Particle Swarm Optimization Algorithm, Chaos, Solitons and Fractals, Vol. 37, pp. 698 – 705.

Joshi, R. and Sanderson, A. C. (1999): Minimal Representation Multi-sensor Fusion using Differential Evolution, IEEE Trans. Systems, Man, and Cybernetics, Part A, Vol. 29(1), pp. 63 -76.

Juang, C - F. (2003): A hybrid of genetic Algorithm and Particle Swarm Optimization for Recurrent Network Design, IEEE Trans. On Systems, Man, and Cybernetics – Part B: Cybernetics, Vol. 34(2), pp. 997 – 1006.

Kannan, B. K. and Kramer, S. N. (1994): An Augmented Lagrange Multiplier Based Method for Mixed Integer Discrete Continuous Optimization and its Applications to Mechanical Design, J. of Mechanical Design, 116/405.

Kannan, S., Slochanal, S. M. R., Subbaraj, P., Padhy, N. P. (2004): Applications of Particle Swarm Optimization Techniques and its Variants to Generation Expansion Planning, Electric Power Systems Research, Vol. 70 (3), pp. 203 - 210.

Kapur, P. K. (1996): Operations Research – Theory and Practice, (Edited) Spaniel Publishers, India, 1996.

Kapur, P. K., Bardhan, A.K. and Jha, P. C. (2003): Optimal Reliability Allocation Problem for a Modular Software System, OPSEARCH, Vol. 40(2), pp. 138 - 148.

Kapur, P. K., Jha, P. C. and Bardhan, A. K. (2003a): Dynamic Programming Approach to Testing Resource Allocation Problem for Modular Software, Journal Ratio Mathematica, Vol. 14, pp. 27 - 40.

Kapur, P. K., Jha, P. C. and Bardhan, A. K. (2004): Optimal Allocation of Testing Resource for Modular Software, Asia Pacific Journal of Operational Research, Vol. 21(3), pp. 333 - 354.

Kennedy, J. (1997): The Particle Swarm: Social Adaptation of Knowledge, In Proc. of the IEEE Int. Conf. on Evolutionary Computation, pp. 303 – 308.

Kennedy, J. (1999): 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.

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

Kennedy, J. and Eberhart, R.C. (2001): Swarm Intelligence. Morgan Kaufmann Publishers, Inc., San Francisco, CA.

Kennedy, J. and Mendes, R. (2003): Neighborhood Topologies in Fully-Informed and Best-of-Neighborhood Particle Swarms, In Proc. of the IEEE Int. Workshop on Soft Computing in Industrial Applications, pp. 45 – 50.

Kimura, S. and Matsumura, K. (2005): Genetic Algorithms using Low Discrepancy Sequences, In Proc. of GEECO 2005, pp. 1341 – 1346.

Kirkpatrick, S., Gelatt, C. D. and Vecchi, M. P. (1983): Optimization by Simulated Annealing, *Science*. New Series, Vol. 220 (4598), pp. 671-680. http://www.jstor.org/stable/1690046

Knuth, D. E. (1998): The Art of Computer Programming, Semi numerical Algorithms, Vol. 2, Addison-Wesley.

Koziel, S. and Michalewicz, Z. (1999): Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization, Evolutionary Computation, Vol. 7(1), pp. 19-44.

Krink, T., Vestrestrom, J. S. and Riget, J. (2002): Particle Swarm Optimization with Spatial Partial Extension, In Proc. of the IEEE Congress on Evolutionary Computation, Vol. 2, pp. 1474 – 1479.

Krohling, R. A. (2005): Gaussian Particle Swarm with Jumps, In. Proc. of the IEEE Congress on Evolutionary Computation, Edinburgh, UK, pp. 1226-1231.

Krohling, R. A. and Coelho, L. S. (2006): PSO-E: Particle Swarm Optimization with Exponential Distribution, In Proc. of IEEE Congress on Evolutionary Computation, pp. 1428 – 1433.

Kuo, H - C., Chang, J – R. and Liu, C – H. (2006): Particle Swarm Optimization for Global Optimization Problems, Journal of Marine Science and Technology, Vol. 14(3), pp. 170 – 181.

Lampinen, J. (1999): A Bibliography of Differential Evolution Algorithm, Technical Report, Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing. http://www.lut.fi/~jlampine/debiblio.htm.

Lampinen, J. and Zelinka, I. (2000): On stagnation of the Differential Evolution Algorithm, In: Pavel Ošmera, (ed.) Proc. of MENDEL 2000, 6th International Mendel Conference on Soft Computing, pp. 76 – 83.

Laway, N. A. and Gupta, H. O. (1993): A Method for Adaptive Coordination of Directional Relays in an Interconnected Power System, In Proc. of IEE conference, Developments in power system protection, pp. 240 – 243

Lee, S. F., Yoon, S. H. and Jang, J. (1989): An Expert System for Protective Relay Setting of Transmission Systems, Proceedings of IEEE Power Industry Computer Application (PICA) Conference, Seatle, WA, pp. 296-302.

Lee, T.Y. and Chen, C.L. (2007): Iteration Particle Swarm Optimization for Contract Capacities Selection of Time-of-use Rates Industrial Customers, Energy Conv. Manag., Vol. 48(4), pp. 1120-1131.

Levy, A.V. and Montalvo, A. (1985): The Tunneling Algorithm for the Global Minimization of Functions, Society for Industrial and Applied Mathematics, Vol. 6, pp. 15–29.

Li, X. (2004): Adaptively Choosing Neighborhood Bests using Species in a Particle Swarm Optimizer for Multimodal Function Optimization, In Proc. of Genetic and Evolutionary Computation Conference, pp. 105–116.

Liang, J. J., Runarsson, T. P., Mezura-Montes, E., M. Clerc, M., Suganthan, N., Coello, C. A. C and Deb, K. (2006): Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization, Technical Report Report #2006005, Nanyang Technological University, Singapore and et al., Dec, 2006.

Liu, H. and Abraham, A. (2007): An Hybrid Fuzzy Variable Neighborhood Particle Swarm Optimization Algorithm for Solving Quadratic Assignment Problems, Journal of Universal Computer Science, Vol. 13(7), pp. 1032-1054.

Liu, H., Abraham, A. and Hassanien, A. E. (2009): Scheduling Jobs on Computational Grids Using Fuzzy Particle Swarm Algorithm, Future Generation Computing Systems, Elsevier Science, Netherlands, Article in Press, doi:10.1016/j.future.2009.05.022.

Liu, H., Abraham, A. and Zhang, W. (2007): A Fuzzy Adaptive Turbulent Particle Swarm Optimization, International Journal of Innovative Computing and Applications, Vol. 1(1), pp. 39 - 47.

Liu, J. and Lampinen, J. (2002): On Setting the Control Parameter of the Differential Evolution Method, In Proc. of 8th Int. Conf. on Soft Computing (MENDEL 2002), pp. 11–18.

Liu, J. and Lampinen, J. (2005): A Fuzzy Adaptive Differential Evolution Algorithm, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 9(6), pp. 448 – 462.

Liu, J., Sun, J., and Xu, W. (2006): Quantum-Behaved Particle Swarm Optimization with Adaptive Mutation Operator, *ICNC 2006*, Part I, Springer-Verlag: pp. 959 – 967.

Liu, J., Sun, J., and Xu, W. (2006a): Improving Quantum-Behaved Particle Swarm Optimization by Simulated Annealing, LNCS: Computational Intelligence and Bioinformatis, Vol. 4115, pp. 130 – 136.

Liu, J., Sun, J., and Xu, W. (2006b): Quantum-Behaved Particle Swarm Optimization with Immune Operator, LNCS: Foundations of Intelligent Systems, Vol. 4203, pp. 77 – 83.

Liu, J., Xu, W., and Sun, J. (2005): Quantum-Behaved Particle Swarm Optimization with Mutation Operator, In Proc. of the 17th IEEE Int. Conf. on Tools with Artificial Intelligence, Hong Kong (China), pp. 237 – 240.

Mansour, M. M. and Mekhamer, S. F. (2007): A Modified Particle Swarm Optimizer for the Coordination of Directional Overcurrent Relays. IEEE Trans. on Power Delivery, Vol. 20 (3), 1400–1410.

McCormick, G.P. (1982): Applied Nonlinear Programming, Theory, Algorithms, and Applications, John Wiley and Sons, New York.

Michalewicz, Z. (1992): Genetic Algorithms + Data Structures = Evolution Programs, Berlin, Heidelberg, New York: Springer Verlog.

Michalewicz, Z. (1996): Genetic Algorithms + Data Structures = Evolution Programs, Second, Extended Edition, Berlin, Heidelberg, New York: Springer Verlog.

Michalewicz, Z., Nazhiyath, G. and Michalewicz, M. (1996): A Note on Usefulness of Geometrical Crossover for Numerical Optimization Problems, In L.J. Fogel, P.J. Angeline, and T. BÄack, editors, In Proc. of the 5th Annual Conference on Evolutionary Programming, pp. 305 - 312.

Miranda, V. and Fonseca, N. (2002): EPSO: Best of Two Worlds Meta-heuristic Applied to Power System Problems, In Proc. IEEE Congress Evolutionary Computation, Vol. 2, pp. 1080–1085.

Miranda, V. and Fonseca, N. (2002a): EPSO: Evolutionary Particle Swarm Optimization, a New Algorithm with Applications in Power Systems, In IEEE/PES Transmission and Distribution Conference Exhibition Asia Pacific, Vol. 2, pp. 745–750.

Mishra, S. K. (2006): Global Optimization by Differential Evolution and Particle Swarm Methods: Evaluation on Some Benchmark Functions, Munich Research Papers in Economics, MPRA Paper 1005, Munich.

Mohan, C. and Nguyen H. T. (1999) 'A Controlled Random Search Technique Incorporating the Simulated Annealing Concept for Solving Integer and Mixed Integer Global Optimization Problems, Computational Optimization and Applications, Vol. 14, pp. 103 – 132.

Mohan, C. and Shanker, K. (1994): A Controlled Random Search Technique for Global Optimization Using Quadratic Approximation, Asia-Pacific Journal of Operational Research, Vol. 11, pp. 93 - 101.

Naka, S., Genji, T., Yura, T. and Fukuyama, Y. (2001): Practical Distribution State Estimation using Hybrid Particle Swarm Optimization, In IEEE Power Engineering Society Winter Meeting, Vol. 2, pp. 815-820.

Nangsue, P., Pillay, P. and Conry, S. E. (1999): Evolutionary Algorithms for Induction Motor Parameter Determination, IEEE Transaction on Energy Conversion, Vol. 14(3), pp. 447-453.

Nguyen X. H., Nguyen Q. Uy., Mckay, R. I. and Tuan, P. M. (2007): Initializing PSO with Randomized Low-Discrepancy Sequences: The Comparative Results, In Proc. of IEEE Congress on Evolutionary Algorithms, pp. 1985 – 1992.

Niu, B., Zhu, Y., He, X. and Zeng, X. (2006): An Improved Particle Swarm Optimization Based on Bacterial Chemotaxis, In Proc. of the 6^{th} World Congress on Intelligent Control and Automation, pp. 3193 – 3197.

Niu, Q. and Gu, X. (2006): An Improved Genetic-Based Particle Swarm Optimization for No-Idle Permutation Flow Shops with Fuzzy Processing Time, LNAI: PRICAI 2006, Vol. 4099, pp. 757 – 766.

Noman, N. and Iba, H. (2005): Enhancing Differential Evolution Performance with Local Search for high dimensional function optimization, In Proc. of the 2005 Conference on Genetic and Evolutionary Computation, pp. 967–974.

Noman, N. and Iba, H. (2008): Accelerating Differential Evolution Using an Adaptive Local Search, IEEE Transactions on Evolutionary Computation, Vol. 12 (1), pp. 107 – 125.

Olorunda, O. and Engelbrecht, A. P. (2008): Measuring Exploration/Exploitation in Particle Swarms using Swarm Diversity, In Proc. of IEEE Congress on Evolutionary Computation, pp. 1128 -1134.

Omran, M., Engelbrecht, A. P. and Salman, A. (2005): Differential Evolution Methods for Unsupervised Image Classification, In Proc. of Seventh Congress on Evolutionary Computation, Vol. 2, pp. 966 - 973.

Omran, Mohd. G. H., Engelbrecht, A. P. and Salman, A. (2007): Differential Evolution based Particle Swarm Optimization, In Proc. of IEEE Swarm Intelligence Symposium, pp. 112 – 119.

Ontario Hydro Report (1990): In-Plant Electric Motor Loading and Efficiency Techniques, Ontario Hydro Report TSDD-90-043.

Onwubolu, G. C. and Babu, B. V. (2004): New Optimization Techniques in Engineering, Springer Verlag, Germany.

Otaduy, P. (1996): Oak Ridge Motor Efficiency and Load - A Computer Program for In-Service Estimation of Motor Efficiency and Load with Minimum Intrusion: Users Guide, ORNL/MC-ORMEL1.

Palanichamy, C., Nadarajan, C., Naveen, P., Babu, N. S. and Dhanalakshmi (2001): Budjet constrained energy conservation- An experience with a textile industry, IEEE Transaction on Energy Conversion, Vol. 16(4), pp. 340-345.

Pang XF. (2005): Quantum mechanics in nonlinear systems, World Scientific Publishing Company, River Edge (NJ, USA).

Pant, M., Thangaraj R., Abraham A. and Singh, V. P. (2008a): Optimal Tuning of PI Speed Controller in PMSM: A Comparative Study of Evolutionary Algorithms, Journal of Electrical Engineering, Vol. 2(1), pp. 36 – 43.

Pant, M., Thangaraj R., and Singh, V. P. (2008): Efficiency Optimization of Electric motors: A Comparative Study of Stochastic Algorithms, World Journal of Modeling and Simulation, Vol. 4(2), pp.140 – 148.

Paquet, U. and Engelbrecht, A. P. (2003): A New Particle Swarm Optimizer for Linearly Constrained Optimization, In Proc. of the IEEE Congress on Evolutionary Computation, Vol. 1, pp. 227 – 233.

Parsopoulos, K. E. and Vrahatis, M. N. (2002): Recent Approaches to Global Optimization Problems through Particle Swarm Optimization, Natural Computing, Vol. 1, pp. 235–306.

Parsopoulos, K. E. and Vrahatis, M. N. (2002a): Particle Swarm Optimization in noisy and continuously changing environments, In Proc. of International Conference on Artificial Intelligence and soft computing, pp. 289-294.

Parsopoulos, K. E., Plagianakos, V. P., Magoulus, G. D. and Vrahatis, M. N. (2001): Objective Function "Strectching" to Alleviate Convergence to Local Minima, Nonlinear Analysis, Theory, Methods and Applications. Vol. 47(5), pp. 3419 – 3424.

Paterlini, S. and Krink, T. (2006): Differential Evolution and Particle Swarm Optimization in Partitional Clustering, Computational Statistics and Data Analysis, Vol. 50, PP. 1220 – 1247.

Paul, H and Tay, A. O. (1987): Optimal Design of an Industrial Refrigeration System, In Proc. of Int. Conf. on Optimization Techniques and Applications, pp. 427 – 435.

Peram, T., Veeramachaneni, K. and Mohan, C. K. (2003): Fitness-Distance-Ratio based Particle Swarm Optimization, In Proc. of the IEEE Swarm Intelligence Symposium, pp. 174-181.

Pillay, P., Levin, V., Otaduy, P., and Kueck, J. (1998): In-Situ Induction Motor Efficiency Determination Using Genetic Algorithm, IEEE Transaction on Energy Conversion, Vol. 13(4), pp. 326-333.

Poli, R., Langdon, W. B. and Holland, O. (2005): Extending Particle Swarm Optimization via Genetic Programming, Lecturer Notes in Computer Science, Genetic Programming, pp. 291 – 300.

Prasad, B. N. and Saini, J. S. (1991): Optimal Thermo Hydraulic Performance of Artificially Roughened Solar Air Heaters, J. Solar Energy, pp. 91 – 96.

Price, K. and Storn, R. (1995): Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization Over Continuous Spaces, Technical Report, International Computer Science Institute, Berkley.

Price, W. L. (1978): A Controlled Random Search Procedure for Global Optimization, In: L. C. W. Dixon and G. P. Szego eds., Towards Global Optimization 2, Amsterdam, Holland: North Holland Publishing Company, pp. 71 – 84.

Protopescu, V. and Barhen, J. (2002): Solving a Class of Continuous Global Optimization Problems Using Quantum Algorithms, Physics Letters A, Vol. 296 (1), pp. 9 – 14.

Qin, A. K., Huang, V. L. and Suganthan, P. N. (2009): Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization, IEEE Transactions on Evolutionary Computations, Vol. 13 (2), pp. 398 – 417.

Rahnamayan, S., Tizhoosh, H. R. and Salama, M. M. A. (2008): Opposition-Based Differential Evolution, IEEE Trans. on Evolutionary Compuation, Vol. 12(1), pp. 64 – 79.

Ramji, K., Saran, V. H., Goel, V. K., and Deep, K. (2003): Optimum Design of Suspension System for Three-wheeled Motor Vehicle - Using Random Search Optimization Technique, 18th I.A.V.S.D. Symposium on Dynamics of Vehicles on Roads and on Tracks, Kanagawa, Japan.

Rastringin, L.A. (1968): Statistical Method of Search, (Nanka, Moscow), 376 (in Russian), (4, 8).

Ratnaweera, A., Halgamuge, S. K., and Watson, H. C. (2004): Self-organizing Hierarchical Particle Swarm Optimizer with Time-varying Acceleration Coefficients, IEEE Transactions on Evolutionary Computation, Vol. 8 (3), pp. 240 – 255.

Rechenberg, I. (1973): Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution, Fromman-Holzboog.

Riget, J. and Vestrestrom, J. S. (2002): A Diversity Guided Particle Swarm Optimizer – The arPSO, Technical Report, Department of Computer Science, University of Aarhus.

Robinson, J., Sinton, S. and Rahmat-Samii, Y. (2002): Particle Swarm, Genetic Algorithm, and Their Hybrids: Optimization of a Profiled Corrugated Horn Antenna, In Proc. of the IEEE Antennas and Propagation Society International Symposium and URSI national radio Science meeting, Vol. 1, pp. 314 – 317.

Rogalsky, T., Derksen, R. W. and Kocabiyik, S. (1999): Differential Evolution in Aerodynamic Optimization, In Proc. of 46th Annual Conf. of Canadian Aeronautics and Space Institute, pp. 29 – 36.

Salerno, J. (1997): Using the Particle Swarm Optimization Technique to Train a Recurrent Neural Model, In Proc. of IEEE International Conference on Tools with Artificial Intelligence, pp. 45 - 49.

Sandgren, E. (1988): Nonlinear Integer and Discrete Programming in Mechanical Design, In Proc. of the ASME Design Technology Conference, Kissimme, Fl, pp. 95 – 105.

Schwefel, H-P. (1981): Numerical Optimization of Computer Models, Chichester: Wiley & Sons.

Secrest, B. R. and Lamont, G. B. (2003): Visualizing Particle Swarm Optimization – Gaussian Particle Swarm Optimization, In Proceedings of the IEEE Swarm Intelligence Symposium, pp. 198 – 204.

Shen, X., Wei, K., Wu, D., Tong, Y. and Li, Y. (2007): An Dynamic Adaptive Dissipative Particle Swarm Optimization with Mutation Operation, In Proc. of IEEE Int. Conf. of Control and Automation, pp. 586 – 589.

Shi, X., Hao, J., Zhou, J. and Liang, Y. (2003): Hybrid Evolutionary Algorithms Based on PSO and GA, In Proc. of the IEEE Congrss on Evolutionary Computation, Vol. 4, pp. 2393 – 2399.

Shi, Y. and Eberhart, R. C. (2001): Fuzzy Adaptive particle Swarm Optimization, In Proc. of the IEEE Congress on Evolutionary Computation, Vol. 1, pp. 101-106.

Shi, Y. and Eberhart, R. C. (1998): A Modified Particle Swarm Optimizer, In Proc. of IEEE Congress of Evolutionary Computation, pp. 69 -73.

Shi, Y. and Eberhart, R. C. (2008): Population Diversity of Particle Swarms, IEEE Congress on Evolutionary Computation, pp. 1063 -1067.

Shi, Y. and Krohling, R. A. (2002): Co-Evolutionary Particle Swarm Optimization to Solve Min-Max Problems, In Proc. of the IEEE Congress on Evolutionary Computation, Vol.2, pp. 1682 – 1687.

So, C. W. and Li, K. K. (2000): Overcurrent Relay Coordination by Evolutionary Programming, Electric Power Systems Research, Vol. 53, pp. 83 - 90.

So, C. W. and Li, K. K. (2000a): Time coordination method for power system protection by evolutionary algorithm, IEEE Trans. Ind. Appl., Vol. 36 (5), pp. 1235 – 1240.

So, C. W. and Li, K. K. (2004): Intelligent method for protection coordination, In Proc. IEEE International Conference of Electric Utility Deregulation Restructuring and Power Technology, Hong Kong, Vol. 1, pp. 378 - 382.

So, C.W., Li, K. K., Lai, K.T. and Fung, K.Y. (1997): Application of genetic algorithm for overcurrent relay coordination, In Proc. of IEE Conf. Developments in Power System Protection, 66–69.

Sobol I. M. (1967): On the distribution of points in a cube and the approximate evaluation of integrals, USSR Computational Mathematics and Mathematical Physics, Vol. 7, pp. 86 – 112.

Sobol I. M. (1976): Uniformly distributed sequences with an additional uniform property, USSR Computational Mathematics and Mathematical Physics, Vol. 16, pp. 236 – 242.

Sriyanyong, P. (2008): Solving Economic Dispatch Using Particle Swarm Optimization Combined with Gaussian Mutation, In Proc. of ECTI-CON, pp. 885 – 888.

Stacey, A., Jancic, M. and Grundy, I. (2003): Particle Swarm Optimization with Mutation, In Proc. of the IEEE Congress on Evolutionary Computation, pp. 1425 – 1430.

Storn, R. (1999): System Design by Constraint Adaptation and Differential Evolution, IEEE Transactions on Evolutionary Computation, Vol. 3, pp. 22-34.

Storn, R. and Price, K. (1997): Differential Evolution – a Simple and Efficient Heuristic for Global Optimization Over Continuous Spaces, Journal Global Optimization, Vol. 11, pp. 341 – 359.

Sun, J., Feng, B. and Xu, W. (2004a): Particle Swarm Optimization with Particles having Quantum Behavior, In Proc. of Congress on Evolutionary Computation, Portland (OR, USA), pp. 325 – 331.

Sun, J., Lai, C. H., Xu, W. and Chai, Z. (2007b): A Novel and More Efficient Search Strategy of Quantum-Behaved Particle Swarm Optimization, LNCS: Adaptive and Natural Computing Algorithms, Vol. 4431, pp. 394 - 403.

Sun, J., Lai, C-H., Xu, W., Ding, Y. and Chai, Z. (2007): A Modified Quantum-Behaved Particle Swarm Optimization, LNCS: Computational Science – ICCS 2007, Vol. 4487, pp. 294 – 301.

Sun, J., Xu, W. and Fang, W. (2006): Quantum-Behaved Particle Swarm Optimization Algorithm with Controlled Diversity, *LNCS, ICCS 2006*, pp. 847 – 854.

Sun, J., Xu, W. and Fang, W. (2006b): A Diversity-Guided Quantum-Behaved Particle Swarm Optimization Algortihm, LNCS: Simulated Evolution and Learning, Vol. 4247, pp. 497 – 504.

Sun, J., Xu, W. and Fang, W. (2006c): Enhancing Global Search Ability of Quantum-Behaved Particle Swarm Optimization by Maintaining Diversity of the Swarm, LNCS: Rough Sets and Current Trends in Computing, Vol. 4259, pp. 736 – 745.

Sun, J., Xu, W. and Fang, W. (2006d): Quantum-Behaved Particle Swarm Optimization with a Hybrid Probability Distribution, LNCS: PRICAI 2006: Trends in Artificial Intelligence, Vol. 4099, pp. 737 – 746.

Sun, J., Xu, W. and Feng, B. (2004b): A Global Search Strategy of Quantum-Behaved Particle Swarm Optimization, *In Proc. of the 2004 IEEE Conf. on Cybernetics and Intelligent Systems*, Singapore, pp. 111 – 116.

Sun, J., Xu, W. and Ye, B. (2006a): Quantum-Behaved Particle Swarm Optimization Clustering Algorithm, LNCS: Advanced Data Mining and Applications, Vol. 4093, pp. 340 – 347.

Sun, J., Xu, W., Fang, W. and Chai, Z. (2007a): Quantum-Behaved Particle Swarm Optimization with Binary Encoding, LNCS: Adaptive and Natural Computing Algorithms, Vol. 4431, pp. 376 – 385.

Sun, J., Zhang, Q. and Tsang, E. P. K. (2004): DE/EDA: A New Evolutionary Algorithm for Global Optimization, Information Sciences, Vol.169, pp. 249– 262.

Talibi, H., and Bautouche (2004): Hybrid Particle Swarm with Differential Evolution for Multimodal Image Regression, In Proc. of IEEE International Conference on Industrial Technology, Vol. 3, pp. 1567-1573.

Teo, J. (2006): Exploring Dynamic Self-adaptive Populations in Differential Evolution, Soft Computing - A Fusion of Foundations, Methodologies and Applications, Vol. 10 (8), pp. 673 – 686.

Thakur, M. (2007): New Real Coded Genetic Algorithms for Global Optimization, Ph. D thesis, Department of Mathematics, Indian Institute of Technology Roorkee, India.

Thanga Raj, C., Agarwal, P. and Srivastava, S. P. (2007): Indian Standard IS 325; A Review, Engineering Advances Magazine, Vol. 19(4), pp. 24-32.

Thompson, B. B., Marks, R. J., El-Sharkawi, M. A., Fox, W. J. and Miyamoto, R. T. (2003): Inversion of Neural Network Underwater Acoustic Model for Estimation of Bottom Parameters Using Modified Particle Swarm Optimizer, In Proc. of the Int. Joint Conference on Neural Networks, Vol. 2, pp. 1301 – 1306.

Ting, T-O., Rao, M. V. C., Loo, C. K. and Ngu, S-S. (2003): A New Class of Operators to Accelerate Particle Swarm Optimization, In Proc. of the IEEE Congress on Evolutionary Computation, Vol. 4, pp. 2406 – 2410.

Tsoulas, I. G. (2008): Modifications of Real Coded Genetic Algorithm for Global Optimization, Applied Mathematics and Compatuation, Vol. 203(2), pp. 598 – 607.

Urdaneta, A. J., Nadira, R. and Perez, L. (1988): Optimal Coordination of Directional Overcurrent Relay in Interconnected Power Systems, IEEE Transaction on Power Delivery, Vol. 3, pp. 903 – 911.

Urdaneta, A. J., Perez, L. G. and Restrepo, H. (1997): Optimal Coordination of Directional Overcurrent Relays Considering Dynamic Changes in the Network Topology, IEEE Transaction on Power Delivery, Vol. 12 (5), pp. 1458–1463.

Urdaneta, A. J., Perez, L. G., Gomez, J. F., Feijoo, B. and Gonzalez, M. (2001): Presolve Analysis and Interior Point Solutions of the Linear Programming Coordination Problem of Directional Overcurrent Relays, International Journal of Electrical Power and Energy Systems, Vol. 23 (8), pp. 819 - 825.

Urdaneta, A. J., Resterbo, H., Sanchez, J. and Fajardo, J. (1996): Coordination of Directional Overcurrent Relays Timing Using Linear Programming, IEEE Transaction on Power Delivery, Vol. 11(1), pp. 122 - 129.

Ursem, R. K. and Vadstrup, P. (2003): Parameter Identification of Induction Motors using Differential Evolution, In Proc. IEEE congress on Evolutionary Computation (CEC 2003), 2003, pp. 790-796.

Ursem, R. K. and Vadstrup, P. (2004): Parameter identification of induction motors using stochastic optimization algorithms, Applied Soft Computing, Vol.4 (1), pp. 49-64.

van den Bergh F (1999): Particle Swarm Weight Initialization in Multi-layer Perception Artificial Neural Networks, Development and Practice of Artificial Intelligence Techniques, pp. 41 -4 5.

Van der Corput, J. G. (1935): Verteilungsfunktionen. Proc. Ned. Akad. v. Wet., 38: pp. 813–821.

Vesterstroem, J. and Thomsen, R. (2004): A Comparative Study of Differential Evolution, Particle Swarm Optimization, and Evolutionary Algorithms on Numerical Benchmark Problems, In Proc. IEEE Congress on Evolutionary Computation, Portland, OR, pp. 1980–1987.

Vestrestrom, J. S., Riget, J. and Krink, T. (2002): Division of Labor in Particle Swarm Optimization, In Proc. of the IEEE Congress on Evolutionary Computation, pp. 1570 - 1575.

Wachowiak, M. P., Smolíková, R., Zheng, Y., Zurada, M. J., Elmaghraby, A. S. (2004): An Approach to Multimodal Biomedical Image Registration Utilizing Particle Swarm Optimization, IEEE Transactions on Evolutionary Computation, Vol. 8(3), 289- 301.

Wang, F-S. and Jang, H-J. (2000): Parameter Estimation of a Bio-reaction Model by Hybrid Differential Evolution, In Proc. of the IEEE Congress on Evolutionary Computation, Vol.1, pp. 410 - 417.

Wang, H. and Qian, F. (2008): Improved PSO-based Multi Objective Optimization Using Inertia Weight and Acceleration Coefficients Dynamic Changing, Crowding and Mutation, In Proc. of the 7^{th} World Congress on Intelligent Control and Automation, pp. 4479 – 4484.

Wang, H., Liu, H., Zeng, S., Li, H. and Li, C. (2007a): Opposition-based Particle Swarm Algorithm with Cauchy Mutation, In Proc. of IEEE Congress on Evolutionary Computation pp. 4750 – 4756.

Wang, H., Wu, Z., Liu, Y. and Zeng, S. (2008): Particle Swarm Optimization with a Novel Multi-parent Crossover Operator, In Proc. of Int. Conf. on Natural Computation, pp. 664 – 668.

Wang, H., Zeng, S., Liu, Y. and Wang, W. (2007): Re-diversification Based Particle Swarm Algortihm with Cauchy Mutation, LNCS: Advances in Computation and Intelligence, Vol. 4683, pp. 362 -271.

Wang, J and Zhou, Y. (2007): Quantum Behaved Particle Swarm Optimization with Generalized Local Search Operator for Global Optimization, LNCS: Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, Vol. 4682, pp. 851 – 860.

Watts, D. J. and Strogatz, S. H. (1998): Collective Dynamics of 'Small-World' Networks, Nature, Vol. 393(6684), pp. 440–442.

Wei, C., He, Z., Zheng, Y. and Pi, W. (2002): Swarm Directions Embedded in Past Evolutionary Programming, In Proc. of IEEE Congress on Evolutionary Computation, Vol. 2, pp. 1278 – 1283.

Wei, J., Guangbin, L. and Dong, L. (2008): Elite Particle Swarm Optimization with Mutation, In Proc. of 7th Int. Conf. on Systems Simulation and Scientific Computing, pp. 800 – 803.

Wei, J., Yuncan, X. and Jixin, Q. (2004): An Improved Particle Swarm Optimization Algorithm with Disturbance, In proc. of IEEE Int. Conference on Systems, Man and Cybernetics, pp. 5900 – 5904.

Xie, X-F., Zhang, W-J. and Yang, Z-L. (2002): A Dissipative Particle Swarm Optimization, In Proc. of IEEE Congress on Evolutionary Computation, pp. 1456 – 1461.

Xu, W. and Sun, J. (2005): Adaptive Parameter Selection of QPSO on Global Level, LNCS: Advanced in Intelligent Computing, Vol. 3644, pp. 420 – 428.

Yang, B., Chen, Y. and Zhao, Z. (2007a): A Hybrid Evolutionary Algorithm by Combination of PSO and GA for Unconstrained and Constrained optimization Problems, In Proc. of the IEEE Int. Conf. on Control and Automation, pp. 166 – 170.

Yang, C. and Simon, D. (2005): A New Particle Swarm Optimization Technique, In Proc. of 18th Int. Conference on Systems Engineering, pp. 164 – 169.

Yang, M., Huang, H. and Xiao, G. (2009): A Novel Dynamic Particle Swarm Optimization Algorithm Based on Chaotic Mutation, Int. Workshop on Knowledge Discovery and Data Mining, pp. 656 – 659.

Yang, X, Yuan, J., Jiangye, Y. and Mao, H. (2007b): A Modified Particle Swarm Optimizer with Dynamic Adaptation, Applied Mathematics and Computation, Vol. 189, pp. 1205 – 1213.

Yang, Z., He, J. and Yao, X. (2007): Making a Difference to Differential Evolution, Advances in Meta-heuristics for Hard Optimization, Z. Michalewicz and P. Siarry (Eds.), pp 397 - 414, Springer Verlog.

Yang, Z., Tang, K. and Yao, X. (2008): Large Scale Evolutionary Optimization Using Cooperative Coevolution, Information Sciences, Vol. 178 (15), pp. 2985-2999.

Yang, Z., Tang, K. and Yao, X. (2008a): Self-adaptive Differential Evolution with Neighborhood Search, In Proc. IEEE Congress on Evolutionary Computation, Hong Kong, pp. 1110-1116.

Yao, X. and Liu, Y. (1996): Fast Evolutionary Programming, In Proc. of 5th Ann. Conf. on Evolutionary Programming, pp. 451 – 460.

Yao, X., Yong, L. and Guangming, L. (1999): Evolutionary Programming Made Faster, IEEE Trans. on Evoltuionary Computation, Vol. 3(2), pp. 82 – 102.

Yoshida, H., Fukuyama, Y., Takayama, S. and Nakanishi, Y. (1999): A Particle Swarm Optimization for Reactive Power and Voltage Control in Electric Power Systems Considering Voltage Security Assessment, In Proc. of the IEEE Int. Conf. on Systems, Man, and Cybernetics, Vol. 6, pp. 497-502.

Yoshida, H., Kawata, K., Fukuyama, Y., Takayama, S., and Nakanishi, Y., (2000): A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment, IEEE Transactions on Power Systems, Vol. 15(4), pp. 1232 - 1239.

Yue-lin, G., Xiao-hui, A. and Jun-min, L. (2008): A Particle Swarm optimization Algorithm with Logarithm Decreasing Inertia Weight and Chaos Mutation, In Proc. of Int. Conference on Computational Intelligence and Security, pp. 61 – 65.

Zeineldin, H. H., El-Saadany, E. F. and Salama, M. M. A. (2006): Optimal Coordination of Overcurrent Relays Using a Modified Particle Swarm Optimization, Electrical Power System Research, Vol. 76, pp. 988 - 995.

Zhang, L.P., Yu, H.J. and Hu, S.X. (2005): Optimal Choice of Parameters for Particle Swarm Optimization, J. Zhejiang Univ. Sci., Vol. 6A (6), pp. 528 - 534.

Zhang, Q., Li, C., Liu, Y. and Kang, L. (2007): Fast Multi-swarm Optimization with Cauchy Mutation and Crossover Operation, LNCS: Advances in Computation and Intelligence, Vol. 4683, pp. 344 – 352.

Zhang, W., Liu, Y. and Clerc, M. (2003): An Adaptive PSO Algorithm for Reactive Power Optimization, In Proc. 6th Int. Conf. Advances in Power System Control, Operation and Management, pp. 302 – 307.

Zhang, W-T and Xie, X-F. (2003): DEPSO: Hybrid Particle Swarm with Differential Evolution Operator, In Proc. of IEEE Int. Conf. on Systems Man and Cybernetics, Vol. 4, pp. 3816 – 3821.

Zhong, W-M., Li, S-J. and Qian, F. (2008): θ-PSO: a New Strategy of Particle Swarm Optimization, Journal of Zhejiang University SCIENCEA, Vol. 9(6), pp. 786 – 790.

Zielinski, K. and Rainer, L. (2006): Constrained Single-Objective Optimization Using Particle Swarm Optimization, IEEE Congress on Evolutionary Computation, pp. 443 – 450.

Zielinski, K. and Rainer, L. (2006a): Constrained Single-Objective Optimization Using Differential Evolution, IEEE Congress on Evolutionary Computation, pp. 223 – 230.

[site] http://www.geatbx.com/docu/fcnindex-01.html#P89_3090

Appendix I
List of Unconstrained Test Problems

Table I.1 Name of unconstrained test problems, assigned codes and characteristics

Sl. No.	Name of the function	Function code	Characteristics		
			UM/MM	SL/NSL	SP/NSP
1	Ackley's path function	ACK	MM	SL	NSP
2	Alphine function	ALP	MM	SL	SP
3	Axis parallel hyperellipsoid	APH	UM	SL	SP
4	Branin function	BR	MM	NSL	NSP
5	Colvillie function	CLV	UM	NSL	NSP
6	Dejong's function	DeJ	UM	SL	SP
7	Dejong's function1 (no noise)	DeJ1	UM	SL	SP
8	Dejong's function with noise	DeJ-N	UM	SL	SP
9	Generalized penalized function 1	GP1	MM	SL	NSP
10	Generalized penalized function 2	GP2	MM	SL	NSP
11	Goldstein and price problem	GP	MM	NSL	NSP
12	Griewank function	GR	MM	SL	SP
13	Hartmann function 1	HM1	MM	NSL	NSP
14	Hartmann function 2	HM2	MM	NSL	NSP
15	Levy and Mantalvo function	LM	MM	SL	NSP
16	Matyas function	MT	MM	NSL	NSP
17	Mccormic function	MC	MM	NSL	NSP
18	Michalewicz function	Mic	MM	SL	NSP
19	Modified Himmelblau function	MH	MM	NSL	NSP
20	Ratringin function	RS	MM	SL	SP
21	Rosenbrock function	RB	UM	SL	NSP

Table I.1 contd...

21	Rosenbrock function	RB	UM	SL	NSP
22	Schwefel function	SWF	MM	SL	SP
23	Schwefel's function 1.2	SWF1.2	UM	SL	SP
24	Schwefel's function 2.21	SWF2.21	UM	SL	NSP
25	Schwefel's function 2.22	SWF2.22	UM	SL	NSP
26	Shaffer's function 6	SF6	MM	NSL	NSP
27	Shaffer's function 7	SF7	MM	SL	NSP
28	Shekel's Foxholes function	SK	MM	NSL	NSP
29	Shubert function 1	SB1	MM	NSL	NSP
30	Shubert function 2	SB2	MM	SL	SP
31	Six hump camel back function	CB6	MM	NSL	NSP
32	Step function	ST	UM	SL	SP
33	Sum of different power	SDP	UM	SL	SP
34	Test2N function	T2N	MM	SL	SP
35	Zhakarov	ZK	UM	SL	NSP

UM – Unimodal

MM – Multimodal

SL – Scalable

NSL – Nonscalable

SP – Separable

NSP – Nonseperable

1. *Ackley's path function (ACK) (Ackley, 1987)*

$$\min_x f(x) = 20 + e - 20\exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)),$$

$-32 \le x_i \le 32$, $x^* = (0,0,\ldots,0)$, $f(x^*) = 0$.

This function is widely used multimodal function. The number of local minima is not known, but the global minimum is located at the origin.

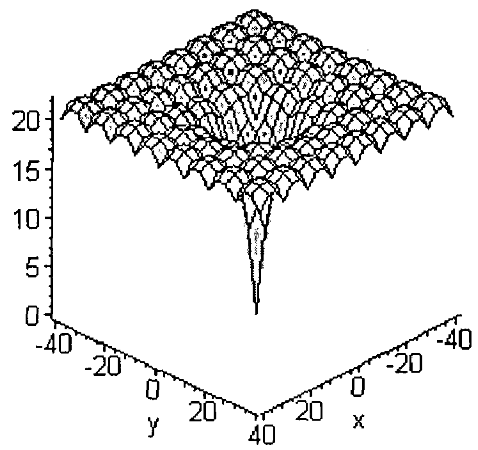

Figure I.1 3D plot of Ackley's path function

2. *Alphine function (ALP) (Rahnamayan et al., 2008)*

$$\min_x f(x) = \sum_{i=1}^{n} | x_i \sin(x_i) + 0.1 x_i |,$$

$-10 \le x_i \le 10$, $x^* = (0,0,0\ldots,0)$, $f(x^*) = 0$

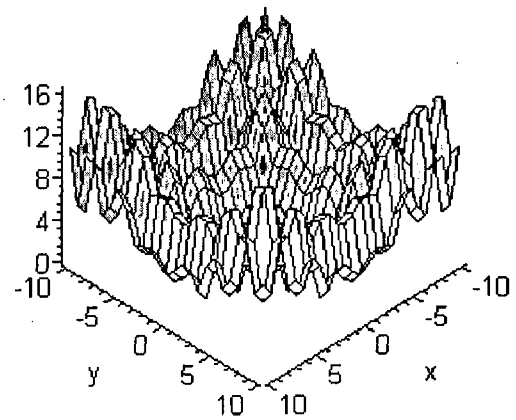

Figure I.2 3D plot of Alphine function

3. *Axis parallel hyper ellipsoid (APH) [site]*

$$\min_x f(x) = \sum_{i=1}^{n} i x_i^2 ,$$

$-5.12 \le x_i \le 5.12$, $x^* = (0,0,...,0)$, $f(x^*) = 0$

This problem is similar to DeJong's function. It is also known as the weighted sphere model. It is continuous, convex and unimodal.

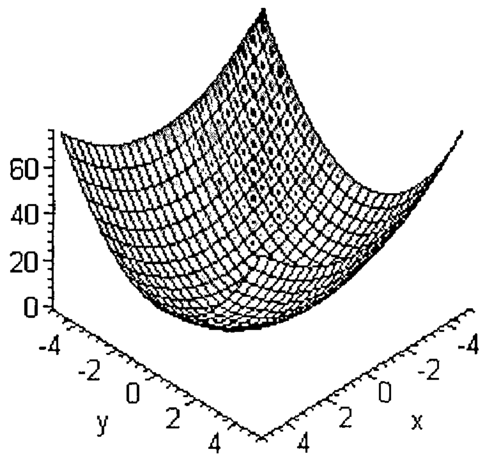

Figure I.3 3D plot of Axis parallel hyper ellipsoid function

4. *Branin function (BR) (Branin, 1972)*

$$\min_x f(x) = (x_1 - \frac{5.1}{4\pi^2} x_0^2 + \frac{5}{\pi} x_0 - 6)^2 + 10(1 - \frac{1}{8\pi})\cos(x_0) + 10 ,$$

$-10 \le x_i \le 10$, $x^* = (9.42, 2.47)$, $f(x^*) = 0.397886$

This function has three global minima.

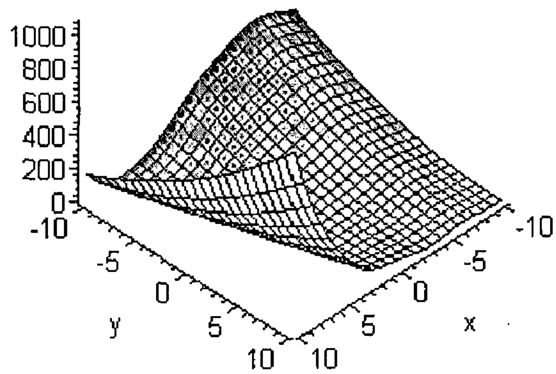

Figure I.4 3D plot of Branin function

5. *Colvillie function (CLV) (Michalewicz, 1996)*

$$\min_x f(x) = 100(x_2 - x_1^2)^2 + (1-x_1)^2 + 90(x_4 - x_3^2)^2 + (1-x_3)^2 + 10.1((x_2-1)^2 + (x_4-1)^2) + 19.8(x_2-1)(x_4-1),$$

$-10 \le x_i \le 10$, $x^* = (1,1,1,1)$, $f(x^*) = 0$

This function has a saddle point near (1,1,1,1). The only minimum is located at (1,1,1,1) with the minimum value zero.

6. *Dejong's function (DeJ) (De Jong, 1975)*

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2,$$

$-5.12 \le x_i \le 5.12$, $x^* = (0,0,\ldots,0)$, $f(x^*) = 0$.

This function is the dream of every optimization algorithm. It is also called the sphere model. It is smooth and symmetric. Also this function is continuous, convex and unimodal.

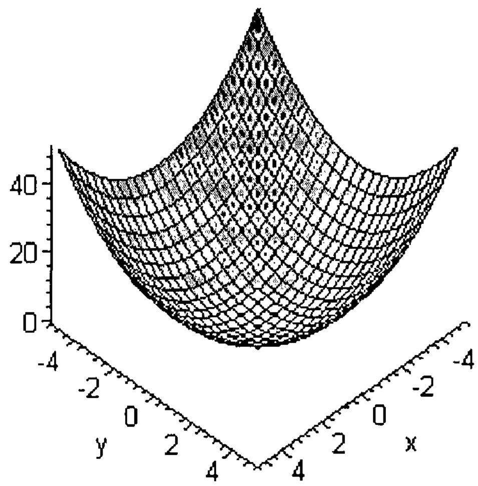

Figure I.5 3D plot of Dejong's function

7. *Dejong's function1 (no noise) (DeJ1) (Rahnamayan et al., 2008)*

$$\min_x f(x) = \sum_{i=0}^{n-1} (i+1)x_i^4,$$

$-1.28 \le x_i \le 1.28$, $x^* = (0,0,\ldots,0)$, $f(x^*) = 0$.

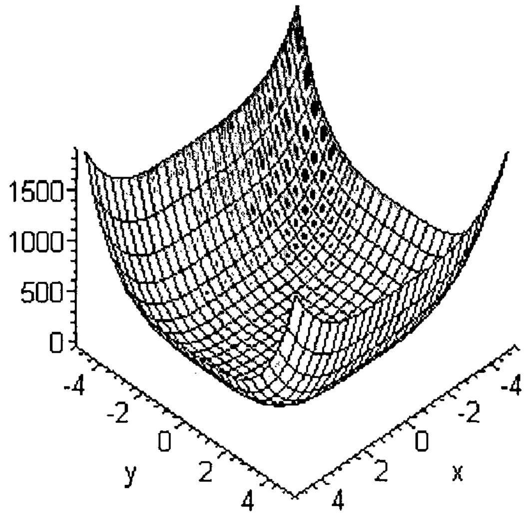

Figure I.6 3D plot of Dejong's function1

8. Dejong's function with noise (DeJ-N) (De Jong, 1975)

$$\min_x f(x) = (\sum_{i=0}^{n-1} (i+1)x_i^{\,4}) + rand[0,1],$$

$-1.28 \le x_i \le 1.28$, $x^* = (0,0,...,0)$, $f(x^*) = 0$.

This function is a simple unimodal function padded with noise. Algorithms that do not do well on this test function will do poorly on noisy data.

9. Generalized penalized function 1 (GP1) (Yao et al., 1999)

$$\min_x f(x) = \frac{\pi}{n}\{10\sin^2(\pi y_1) + \sum_{i=1}^{n-1}(y_i - 1)^2[1 + 10\sin^2(y_{i+1}\pi)]$$

$$+(y_n - 1)^2\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4),$$

Where $y_i = 1 + \frac{1}{4}(x_i + 1)$, $-50 \le x_i \le 50$, $x^* = (0,0,...,0)$, $f(x^*) = 0$

This function is a multimodal function where the number of local minima increases exponentially with the problem dimension. It appear to be the most difficult class of problems for many optimization algorithms.

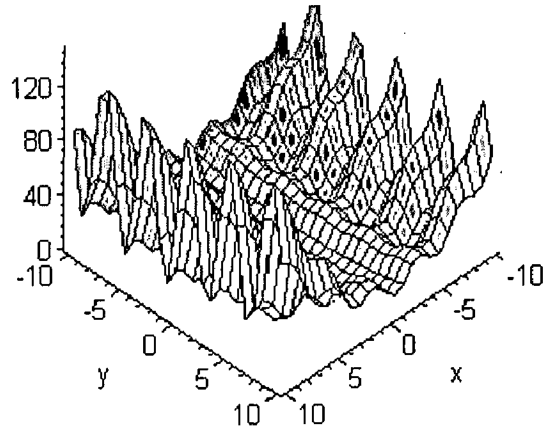

Figure I.7 3D plot of generalized penalized function 1

10. Generalized penalized function 2 (GP2) (Yao et al., 1999)

$$\min_x f(x) = (0.1)\{\sin^2(3\pi x_1) + \sum_{i=1}^{n-1}((x_i - 1)^2(1 + \sin^2(3\pi x_{i+1})))$$

$$+ (x_n - 1)(1 + \sin^2(2\pi x_n))\} + \sum_{i=0}^{n-1} u(x_i, 5, 100, 4),$$

$-50 \le x_i \le 50$, $x^* = (1,1,\ldots,-4.76)$, $f(x^*) = -1.1428$

This function is similar to Generalized penalized function 1. It is a multimodal function where the number of local minima increases exponentially with the problem dimension. It appear to be the most difficult class of problems for many optimization algorithms.

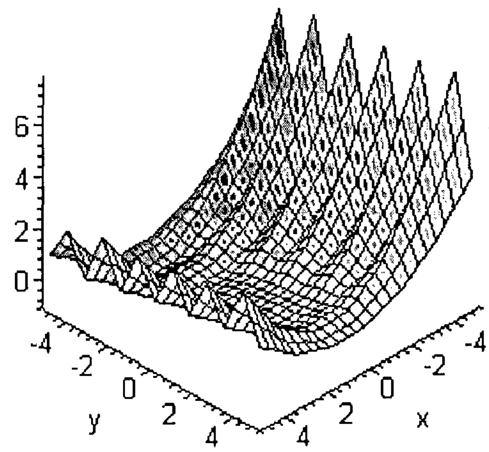

Figure I.8 3D plot of generalized penalized function 2

In problem 9 and 10,

$u(x, a, b, c) = b(x-a)^c \quad \text{if } x > a,$

$u(x, a, b, c) = b(-x-a)^c \quad \text{if } x < -a,$

$u(x, a, b, c) = 0 \quad \text{if } -a \le x \le a.$

11. Goldstein and price problem (GP) (Goldstein and Price, 1971)

$$\min_x f(x) = \{1 + (x_0 + x_1 + 1)^2 (19 - 14x_0 + 3x_0^2 - 14x_1 + 6x_0x_1 + 3x_1^2)\}$$
$$\{30 + (2x_0 - 3x_1)^2 (18 - 32x_0 + 12x_0^2 + 48x_1 - 36x_0x_1 + 27x_1^2)\},$$

$-2 \le x_i \le 2$, $x^* = (0,1)$, $f(x^*) = 3$

This problem has four local minima and one global minima.

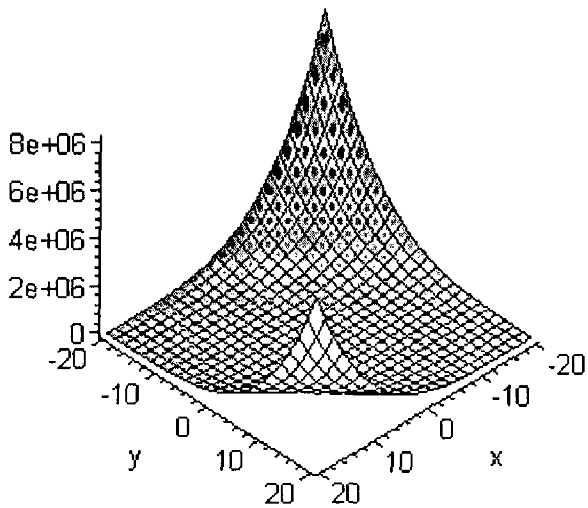

Figure I.9 3D plot of Goldstein and Price problem

12. Griewank function (GR) (Griewank, 1981)

$$\min_x f(x) = \frac{1}{4000} \sum_{i=0}^{n-1} x_i^2 - \sum_{i=0}^{n-1} \cos\left(\frac{x_i}{\sqrt{i+1}}\right) + 1, \; -600 \le x_i \le 600, \; x^* = (0,0,...,0), \; f(x^*) = 0.$$

This test problem is similar to Rastringin function. It has thousands of local minima and is used widely. However the locations of minima are regularly distributed.

20. Ratringin function (RS) (Rastringin, 1968)

$$\min_x f(x) = \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i) + 10),$$

$-5.12 \le x_i \le 5.12$, $x^* = (0,0,...,0)$, $f(x^*) = 0$.

This function is highly multimodal with regularly distributed many local minima. The total number of minima for this function is not exactly known but the global minimum is located at the origin. For 2 dimension, it has about 50 local minimas arranged in a lattice like configuration.

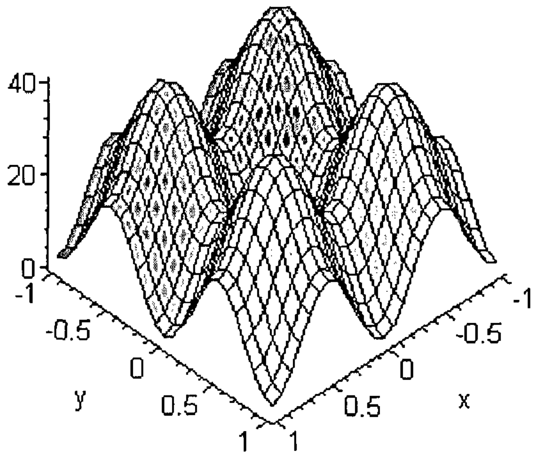

Figure I.16 3D plot of Rastringin function

21. Rosenbrock function (RB) (DeJong, 1975)

$$\min_x f(x) = \sum_{i=0}^{n-1} 100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2, \; -30 \le x_i \le 30, \; x^* = (1,1,...,1), \; f(x^*) = 0.$$

It is a classic optimization problem with a narrow global optimum hidden inside a long, narrow, curved flat valley. It is unimodal, yet due to a saddle point it is very difficult to locate the minimum. This function is also known as banana valley function.

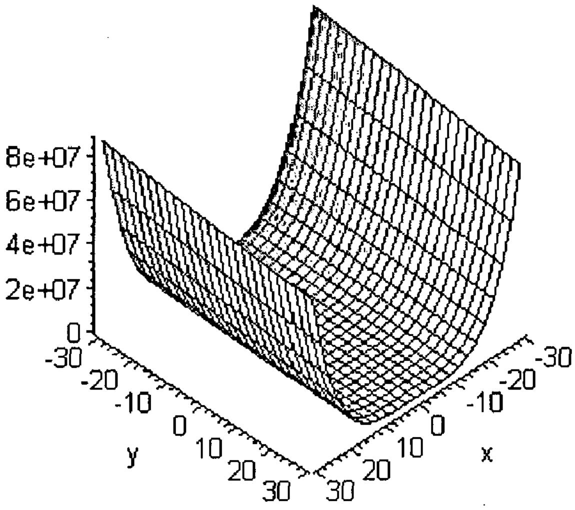

Figure I.17 3D plot of Rosenbrock function

22. Schwefel function (SWF) (Schwefel, 1981)

$$\min_{x} f(x) = -\sum_{i=1}^{n} x_i \sin(\sqrt{|x_i|}),$$

$$-500 \le x_i \le 500,\ x^* = (420.97, 420.947, \ldots, 420.947),\ f(x^*) = -418.9829 * n$$

This function is deceptive in that the global minimum is geometrically distant, over the parameter space, from the next best global minima. Therefore the search algorithms are prone to converge in wrong direction.

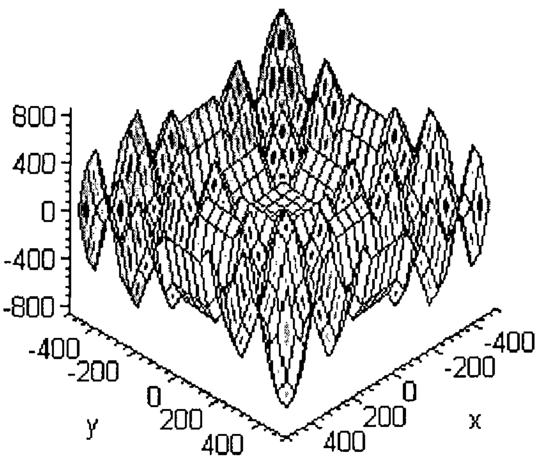

Figure I.18 3D plot of Schwefel function

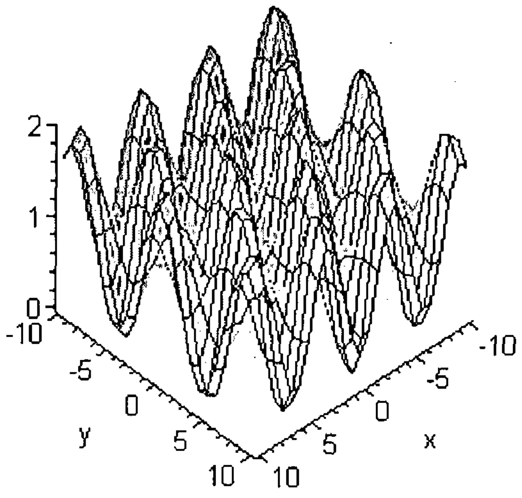

Figure I.10 3D plot of Griewank function

13. Hartmann function 1 (HM1) (Dixon and Szego, 1978)

$$\min_x f(x) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{3} A_{ij}(x_j - P_{ij})^2\right),$$

$0 \le x_i \le 1$, $x^* = (0.114614, 0.555649, 0.852547)$, $f(x^*) = -3.86278$

Where, $\alpha = \begin{bmatrix} 1 & 1.2 & 3 & 3.2 \end{bmatrix}$, $A = \begin{bmatrix} 3 & 10 & 30 \\ 0.1 & 10 & 35 \\ 3 & 10 & 30 \\ 0.1 & 10 & 35 \end{bmatrix}$, $P = \begin{bmatrix} 0.3689 & 0.117 & 0.2673 \\ 0.4699 & 0.4387 & 0.747 \\ 0.1091 & 0.8732 & 0.5547 \\ 0.03815 & 0.5743 & 0.8828 \end{bmatrix}$

This function has four local minima and one global minima.

14. Hartmann function 2 (HM2) (Dixon and Szego, 1978)

$$\min_x f(x) = -\sum_{i=1}^{4} \alpha_i \exp\left(-\sum_{j=1}^{6} B_{ij}(x_j - Q_{ij})^2\right),$$

$0 \le x_i \le 1$, $x^* = (0.20169, 0.150011, 0.476874, 0.275332, 0.311652, 0.6573)$, $f(x^*) = -3.32237$

Where $\alpha = \begin{bmatrix} 1 & 1.2 & 3 & 3.2 \end{bmatrix}$, $B = \begin{bmatrix} 10 & 3 & 17 & 3.05 & 1.7 & 8 \\ 0.05 & 10 & 17 & 0.1 & 8 & 14 \\ 3 & 3.5 & 1.7 & 10 & 17 & 8 \\ 17 & 8 & 0.05 & 10 & 0.1 & 14 \end{bmatrix}$,

$$Q = \begin{bmatrix} 0.1312 & 0.1696 & 0.5569 & 0.0124 & 0.8283 & 0.5886 \\ 0.2329 & 0.4135 & 0.8307 & 0.3736 & 0.1004 & 0.9991 \\ 0.2348 & 0.1451 & 0.3522 & 0.2883 & 0.3047 & 0.6650 \\ 0.4047 & 0.8828 & 0.8732 & 0.5743 & 0.1091 & 0.0381 \end{bmatrix}$$

This function has four local minima and one global minima.

15. Levy and Mantolva function (LM) (Rahnamayan et al., 2008)

$$\min_x f(x) = \sin^2(3\pi x_1) + \sum_{i=1}^{n-1} (x_i - 1)^2 (1 + \sin^2(3\pi x_{i+1})) + (x_n - 1)(1 + \sin^2(2\pi x_n)),$$

$-10 \le x_i \le 10$, $x^* = (1,1,\ldots,1,-9.7523)$, $f(x^*) = -21.5023$

This problem has several local minima.

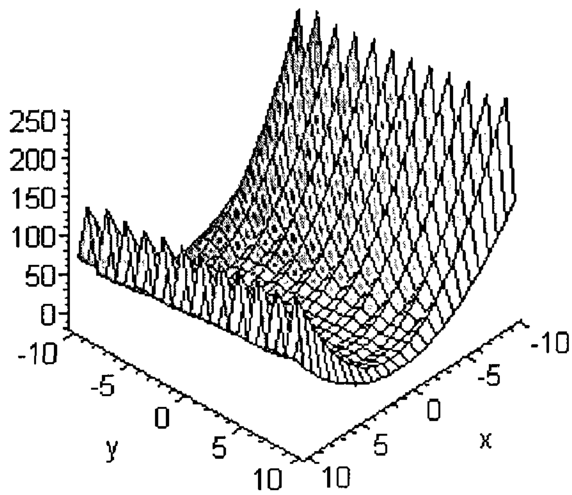

Figure I.11 3D plot of Levy and Mantolva function

16. Matyas function (MT) (Rahnamayan et al., 2008)

$$\min_x f(x) = 0.26(x_1^2 + x_2^2) - 0.48 x_1 x_2,$$

$-10 \le x_i \le 10$, $x^* = (0,0)$, $f(x^*) = 0$

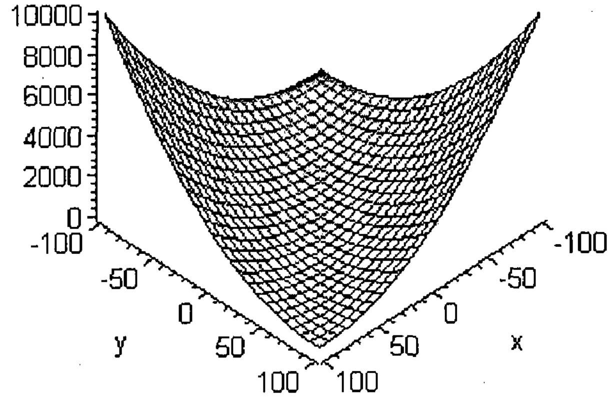

Figure I.12 3D plot of Matyas function

17. Mccormic function (MC) (McCormic, 1982)

$$\min_x f(x) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1,$$

$-2 \le x_i \le 2$, $x^* = (-0.5471, -1.5473)$, $f(x^*) = -1.9132$

This problem has one local minima and one global minima. The local minima is at (2.59, 1.59) and the global minima is at (-0.5471, -1.5473).

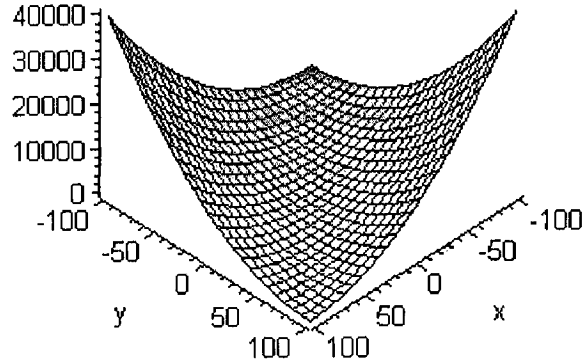

Figure I.13 3D plot of Mccormic function

18. Michalewicz function (Mic) (Michalewicz, 1992)

$$\min_x f(x) = -\sum_{i=1}^{n} \sin(x_i)(\sin(i\frac{x_i^2}{\pi}))^{2m} \quad , \ m = 10, \ -\pi \le x_i \le \pi,$$

$f(x^*) = -1.8013$, If n = 2,

$f(x^*) = -4.6876$, If n = 5,

$f(x^*) = -9.66015$, If n = 10.

This function is a highly multimodal, nonlinear, nonseperable test problem. It has n! local optima. The parameter m defines the steepness of the valleys or edges. Larger m leads to more difficult search. For every large m the function behaves like a needle in the haystack since the function values for points in the space outside the narrow peaks give very little information on the location of the global optimum.

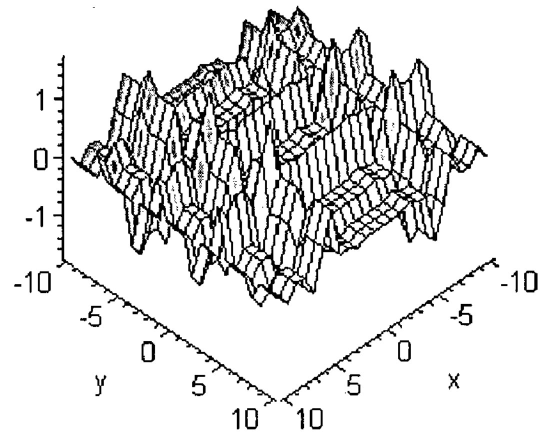

Figure I.14 3D plot of Michalewicz function

19. Modified Himmelblau function (MH) (Kuo et al, 2006)

$$\min_{x} f(x) = (x_2 + x_1^2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + x_1,$$

$-5 \le x_i \le 5$, $x^* = (-3.788, -3.286)$, $f(x^*) = -3.7839$

It has three local minima and one global minimum.

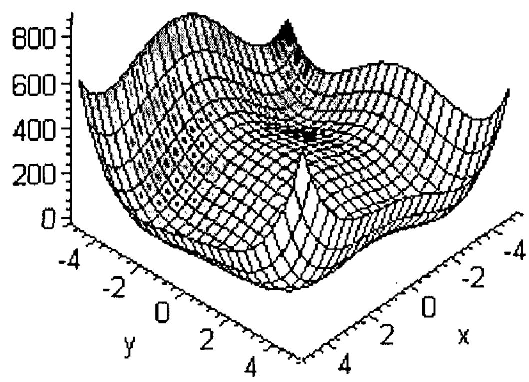

Figure I.15 3D plot of Modified Himmelblau function

25. Schwefel's function 2.22 (SWF2.22) (Yao et al., 1999)

$$\min_x f(x) = \sum_{i=0}^{n-1} |x_i| + \prod_{i=0}^{n-1} |x_i|, \; -10 \le x_i \le 10, \; x^* = (0,0,...,0), \; f(x^*) = 0$$

This function is a unimodal function.

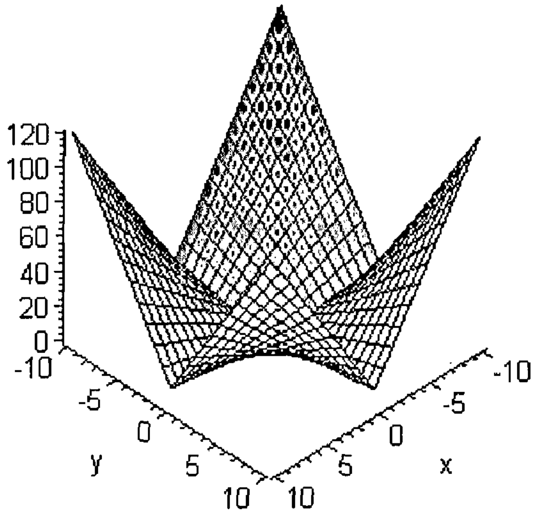

Figure I.21 3D plot of Schwefel function 2.22

26. Shaffer's function 6 (SF6) (Michalewicz, 1996)

$$\min_x f(x) = 0.5 + \frac{\sin^2 \sqrt{(x_1^2 + x_2^2)} - 0.5}{1 + 0.01(x_1^2 + x_2^2)^2}, \; -10 \le x_i \le 10, \; x^* = (0,0), \; f(x^*) = 0$$

This function contains "minimum rings" around the global minima with almost the same fitness as the global minima. The number of local minima is not known but the global minima is at the origin.

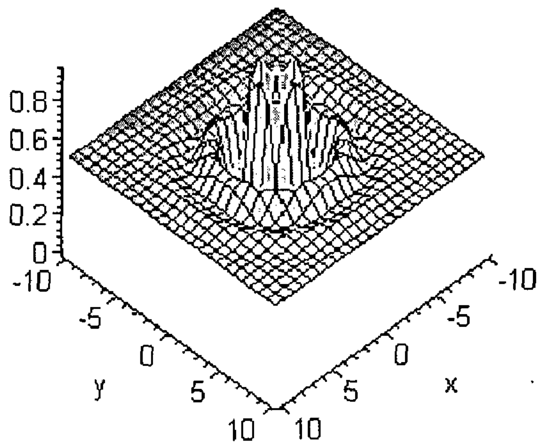

Figure I.22 3D plot of Shaffer's function 6

23. Schwefel's function 1.2 (SWF1.2) (Yao et al., 1999)

$$\min_x f(x) = \sum_{i=0}^{n-1} \left(\sum_{j=0}^{i} x_j \right)^2, \; -100 \le x_i \le 100, \; x^* = (0,0,...,0), \; f(x^*) = 0.$$

This function is a unimodal function.

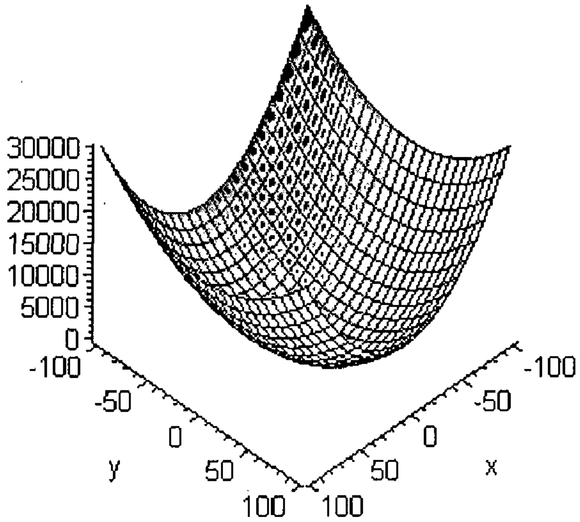

Figure I.19 3D plot of Schwefel function 1.2

24. Schwefel's function 2.21 (SWF2.21) (Yao et al., 1999)

$$\min_x f(x) = \max |x_i|, \quad 0 \le i < n,$$

$$-100 \le x_i \le 100, \; x^* = (0,0,0...,0), \; f(x^*) = 0$$

This function is a unimodal function.

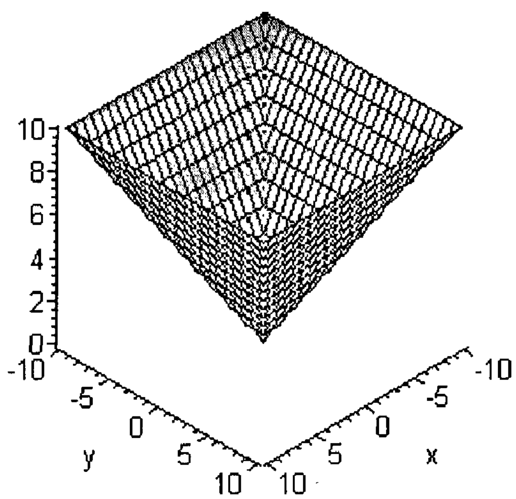

Figure I.20 3D plot of Schwefel function 2.21

27. Shaffer's function 7 (SF7) (Michalewicz, 1996)

$$\min_x f(x) = (\sum_{i=1}^{n} x_i^2)^{1/4}[\sin^2(50(\sum_{i=1}^{n} x_i^2)^{1/10}) + 1.0],$$

$-32.767 \le x_i \le 32.767$, $x^* = (0,0,0...,0)$, $f(x^*) = 0$

The number of local minima of this function is not known, but the global minima is located at the origin.

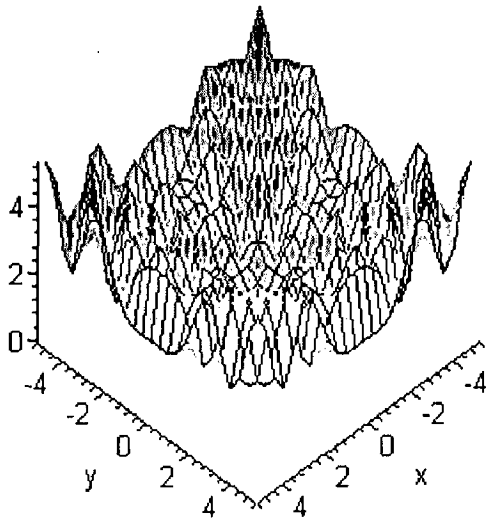

Figure I.23 3D plot of Shaffer's function 7

28. Shekel's Foxholes function (SK) (Yao et al., 1999)

$$\min_x f(x) = (\frac{1}{500} + \sum_{j=0}^{24} (j+1+\sum_{i=0}^{1} (x_i - a_{ij})^6)^{-1})^{-1},$$

$-65.54 \le x_i \le 65.54$, $x^* = (-31.95,-31.95)$, $f(x^*) = 1$

Where $a = \begin{pmatrix} -32,-16,0,16,32,...,-32,-16,0,16,32 \\ -32,...,-16,...,0,...,16,....,32,... \end{pmatrix}$

This function is a low dimensional function; it has only a few local minima.

29. Shubert function 1 (SB1) (Levy and Montalvo, 1985)

$$\min_x f(x) = \sum_{j=1}^{5} j\cos((j+1)x_1 + j) \sum_{j=1}^{5} j\cos((j+1)x_2 + j),$$

$-10 \le x_i \le 10$, $f(x^*) = -186.7309$

The function has 760 local minima, 18 of which are global minima with the minimum of -186.7309.

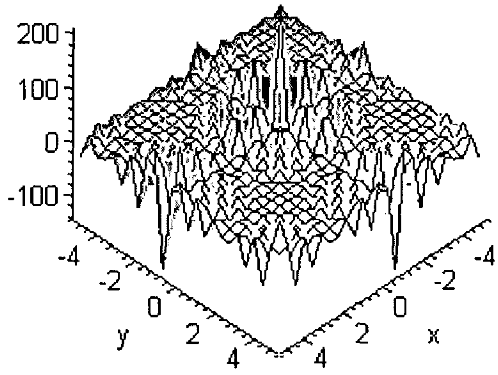

Figure I.24 3D plot of Shubert function 1

30. Shubert function 2 (SB2) (I. G. Tsoulos, 2008)

$$\min_x f(x) = -\sum_{i=1}^{n} \sum_{j=1}^{5} j \sin((j+1)x_i + j),\ -10 \le x_i \le 10,\ f(x^*) = -24.06249 \text{ for n = 2.}$$

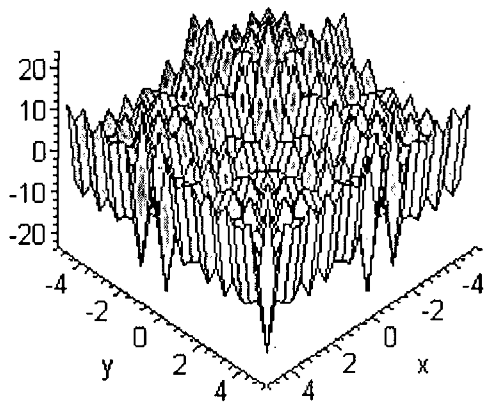

Figure I.25 3D plot of Shubert function 2

31. Six hump camel back function (CB6) (Dixon and Szego, 1978)

$$\min_x f(x) = 4x_0^2 - 2.1x_0^4 + \frac{1}{3}x_0^6 + x_0x_1 - 4x_1^2 + 4x_1^4,$$

$-5 \le x_i \le 5$, $x^* = (0.09,-0.71)$, $f(x^*) = -1.03163$

This function has two global minima with the minimum of -1.03163.

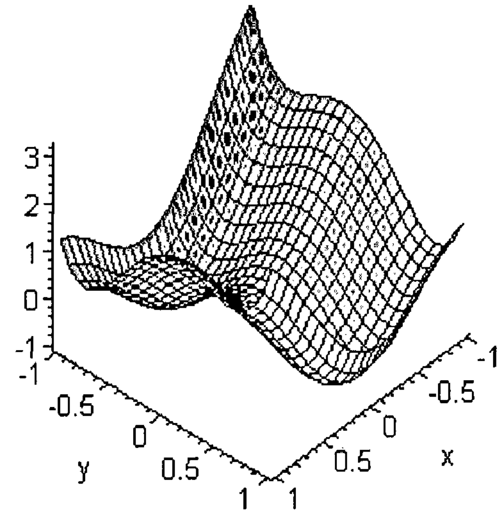

Figure I.26 3D plot of Six hump camel back function

32. Step function (ST) (DeJong, 1975)

$$\min_x f(x) = \sum_{i=0}^{n-1} \lfloor x_i + 1/2 \rfloor^2,\ -100 \le x_i \le 100,\ x^* = (0,0,\ldots,0),\ f(x^*) = 0.$$

It is the representative of the problem of flat surfaces. Flat surfaces are obstacles for optimization algorithms, because they do not give any information as to which direction is favorable. Unless an algorithm has variable step sizes, it can get stuck on one of the flat plateans. It has one global minimum and is discontinuous.

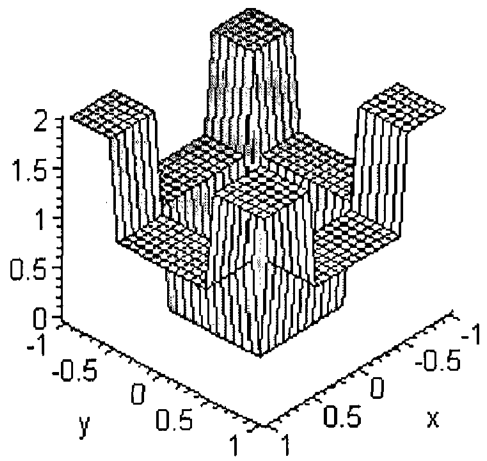

Figure I.27 3D plot of Step function

33. Sum of different power (SDP) [site]

$$\min_{x} f(x) = \sum_{i=1}^{n} |x_i|^{(i+1)}, \ -1 \le x_i \le 1, \ x^* = (0,0,0...,0), \ f(x^*) = 0$$

This function is a commonly used unimodal function.

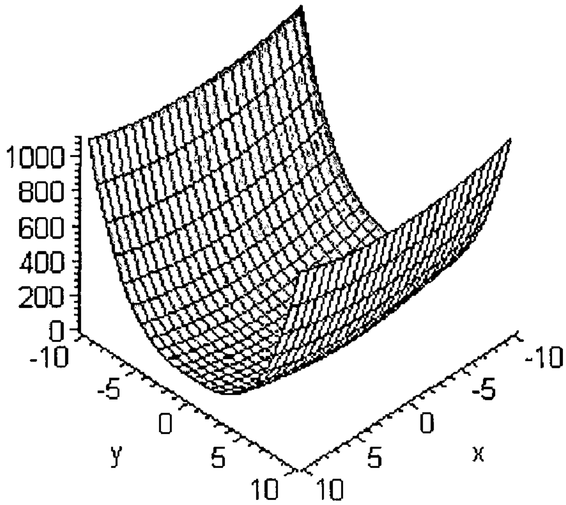

Figure I.28 3D plot of Sum of different power function

34. Test2N function (T2N) (I. G. Tsoulos, 2008)

$$\min_{x} f(x) = \frac{1}{n} \sum_{i=1}^{n} (x_i^4 - 16x_i^2 + 5x_i),$$

$$-5 \le x_i \le 5, \ x^* = (-2.903, -2.903, \ldots, -2.903), \ f(x^*) = -78.3323$$

This function has 2^n local minima in the specified range.

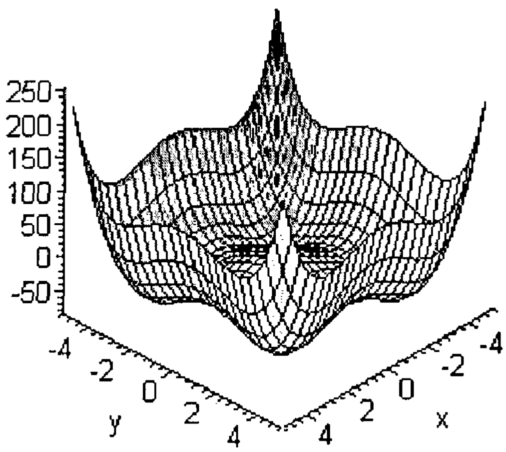

Figure I.29 3D plot of Test2N function

35. Zhakarov function (ZK) (Hedar Home page)

$$\min_x f(x) = \sum_{i=1}^{n} x_i^2 + (\sum_{i=1}^{n} 0.5ix_i)^2 + (\sum_{i=1}^{n} 0.5ix_i)^4, \ -10 \le x_i \le 10, \ x^* = (0,0,0...,0), \ f(x^*) = 0$$

This function has no local minima, it has one global minima at the origin.

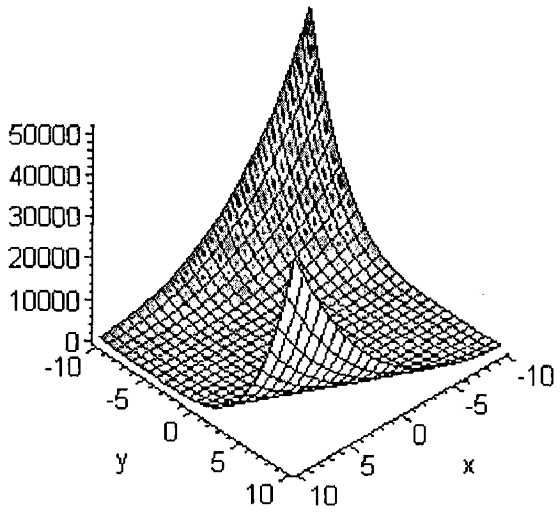

Figure I.30 3D plot of Zhakarov function

Appendix II
List of Constrained Test Problems

Table II.1 Name of constrained test problems, assigned codes and characteristics

Sl. No.	Function	Code	N_v	N_{EQ}	N_{IEQ}	N_a	Type of function
1	Problem 1	g01	13	0	9	6	quadratic
2	Problem 2	g02	20	0	2	1	nonlinear
3	Problem 3	g03	10	1	0	1	polynomial
4	Problem 4	g04	5	0	6	2	quadratic
5	Problem 5	g05	4	3	2	3	cubic
6	Problem 6	g06	2	0	2	2	cubic
7	Problem 7	g07	10	0	8	6	quadratic
8	Problem 8	g08	2	0	2	0	nonlinear
9	Problem 9	g09	7	0	4	2	polynomial
10	Problem 10	g10	8	0	6	6	linear
11	Problem 11	g11	2	1	0	1	quadratic
12	Problem 12	g12	3	0	1	0	quadratic
13	Problem 13	g13	5	3	0	3	nonlinear
14	Problem 14	g14	10	3	0	3	nonlinear
15	Problem 15	g15	3	2	0	2	quadratic
16	Problem 16	g16	5	0	38	4	nonlinear
17	Problem 17	g17	6	4	0	4	nonlinear
18	Problem 18	g18	9	0	13	6	quadratic
19	Problem 19	g19	2	0	2	2	linear
20	Problem 20	g20	7	5	1	6	linear

N_v – Number of variables

N_{EQ} – Number of equality constraints

N_{IEQ} – Number of inequality constraints

N_a – Number of active constraints

1. *Problem 1 (g01) (Floudas et al, 1987)*

   Minimize $f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$

   Subject to:

   $$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0$$

   $$g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0$$

   $$g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0$$

   $$g_4(x) = -8x_1 + x_{10} \le 0$$

   $$g_5(x) = -8x_2 + x_{11} \le 0$$

   $$g_6(x) = -8x_3 + x_{12} \le 0$$

   $$g_7(x) = -2x_4 - x_5 + x_{10} \le 0$$

   $$g_8(x) = -2x_6 - x_7 + x_{11} \le 0$$

   $$g_9(x) = -2x_8 - x_9 + x_{12} \le 0$$

   $$0 \le x_i \le 1 \ (i = 1,2,...,9),\ 0 \le x_i \le 100 \ (i = 10,11,12),\ 0 \le x_{13} \le 1$$

   The optimum value is $f(x^*) = -15$ at $x^* = (1,1,1,1,1,1,1,1,1,3,3,3,1)$

   Constraints g_1, g_2, g_3, g_7, g_8, g_9 are active.

2. *Problem 2 (g02) (Koziel et al, 1999)*

   Maximize $f(x) = \left| \dfrac{\Sigma_{i=1}^{n} \cos^4(x_i) - 2\Pi_{i=1}^{n} \cos^2(x_i)}{\Sigma_{i=1}^{n} i x_i^2} \right|$

   Subject to:

   $$g_1(x) = 0.75 - \Pi_{i=1}^{n} x_i \le 0$$

   $$g_2(x) = \Sigma_{i=1}^{n} x_i - 7.5n \le 0$$

   $$0 \le x_i \le 10 \ (i = 1,2,..., n)\ , n = 20$$

   The optimum value is unknown. The known best value is $f(x^*) = 0.803619$

   Constraint g_1 is active.

3. *Problem 3 (g03) (Michalewicz et al, 1996)*

   Minimize $f(x) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i$

   Subject to:

   $$h_1(x) = \Sigma_{i=1}^{n} x_i^{\;2} - 1 = 0$$

   $$0 \le x_i \le 10 \quad (i = 1,2,\ldots, n)$$

   The optimum value is $f(x^*) = -1$ at $x^* = (1/\sqrt{n})$, $n = 10$.

4. *Problem 4 (g04) (Himmelblau, 1972)*

   Minimize $f(x) = 5.3578547x_3^{\;2} + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$

   Subject to:

   $$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5$$

   $$g_2(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^{\;2}$$

   $$g_3(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4$$

   $$0 \le g_1(x) \le 92$$

   $$90 \le g_2(x) \le 110$$

   $$20 \le g_3(x) \le 25$$

   $$78 \le x_1 \le 102,\ 33 \le x_2 \le 45,\ 27 \le x_i \le 45\ (i = 3,4,5).$$

   The optimum value is $f(x^*) = -30665.539$ at

   $x^* = (78,33,29.995256025682,45,36.775812905788)$

5. *Problem 5 (g05) (Hock et al, 1981)*

   Minimize $f(x) = 3x_1 + 0.000001x_1^{\;3} + 2x_2 + (0.000002/3)x_2^{\;3}$

   Subject to:

   $$g_1(x) = -x_4 + x_3 - 0.55 \le 0$$

   $$g_2(x) = -x_3 + x_4 - 0.55 \le 0$$

$$h_3(x) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(x) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(x) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

$$0 \le x_i \le 1200 \ (i = 1,2), \ -0.55 \le x_2 \le 0.55 \ (i = 3,4).$$

The optimum value is $f(x^*) = 5126.4981$ at

$x^* = (679.9463, 1026.067, 0.1188764, -0.3962336)$.

6. *Problem 6 (g06) (Floudas et al, 1987)*

Minimize $f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$

Subject to:

$$g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \le 0$$

$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \le 0$$

$$13 \le x_1 \le 100, \ 0 \le x_2 \le 100$$

The optimum value is $f(x^*) = -6961.81388$ at $x^* = (14.095, 0.84296)$.

7. *Problem 7 (g07 (Hock et al, 1981)*

Minimize $f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2 + (x_5 - 5)^2$

$$+ 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$

Subject to:

$$g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0$$

$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0$$

$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0$$

$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \le 0$$

$$g_5(x) = 5x_1^2 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \le 0$$

$$g_6(x) = x_1^2 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0$$

$$g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \le 0$$

$$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \le 0$$

$$-10 \le x_i \le 10 \ (i = 1,2,\dots,10)$$

The optimum value is $f(x^*) = 24.3062091$ at

$x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574,$
$1.321644, 9.828726, 8.280092, 8.375927)$

Constraints g_1, g_2, g_3, g_4, g_5 and g_6 are active.

8. *Problem 8 (g08) (Koziel et al, 1999)*

Maximize $f(x) = \dfrac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1 + x_2)}$

Subject to:

$$g_1(x) = x_1^2 - x_2 + 1 \le 0$$

$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \le 0$$

$$0 \le x_i \le 10 \ (i = 1,2).$$

The optimum value is $f(x^*) = 0.095825$ at $x^* = (1.2279713, 4.2453733)$.

9. *Problem 9 (g09) (Hock et al, 1981)*

Minimize

$$f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 +$$
$$10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

Subject to:

$$g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0$$

$$g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0$$

$$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0$$

$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

$$-10 \leq x_i \leq 10 \quad (i = 1,2,...,7)$$

The optimum value is $f(x^*) = 680.6300573$ at

$x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.624487, 1.038131, 1.5942270)$.

10. Problem 10 (g10) (Hock et al, 1981)

Minimize $f(x) = x_1 + x_2 + x_3$

Subject to:

$$g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$$

$$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$

$$g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$$

$$g_4(x) = -x_1x_6 + 833.33252x_4 + 100x1 - 83333.333 \leq 0$$

$$g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$

$$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$

$$-100 \leq x_1 \leq 10000,\ 1000 \leq x_i \leq 10000\ (i = 2,3),\ 10 \leq x_i \leq 1000\ (i = 4,...,8).$$

The optimum value is $f(x^*) = 7049.25$ at

$x^* = (579.19, 1360.13, 5109.5979, 182.0174, 295.5985, 217.9799, 286.40, 395.5979)$.

11. Problem 11 (g11) (Koziel et al, 1999)

Minimize $f(x) = x_1^2 + (x_2 - 1)^2$

Subject to:

$$h_1(x) = x_2 - x_1^2 = 0$$

$$-1 \leq x_i \leq 1 \quad (i = 1,2)$$

The optimum value is $f(x^*) = 0.75$ at $x^* = (\pm 1/\sqrt{2}, 1/2)$.

12. *Problem 12 (g12) (Koziel et al, 1999)*

Minimize $f(x) = -(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$

Subject to:

$$g(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \le 0$$

$$0 \le x_i \le 10,\ i = 1,2,3,\ p,q,r = 1,2,\dots,9$$

The optimum value is $f(x^*) = -1$ at $x^* = (5,5,5)$.

13. *Problem 13 (g13) (Hock et al, 1981)*

Minimize $f(x) = e^{x_1 x_2 x_3 x_4 x_5}$

Subject to:

$$h_1(x) = x_1{}^2 + x_2{}^2 + x_3{}^2 + x_4{}^2 + x_5{}^2 - 10 = 0$$

$$h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0$$

$$h_3(x) = x_1{}^3 + x_2{}^3 + 1 = 0$$

$$-2.3 \le x_i \le 2.3,\ i = 1,2$$

$$-3.2 \le x_i \le 3.2,\ i = 3,4,5$$

The optimum value is $f(x^*) = 0.0539$ at

$x^* = (-1.7171, 1.5957, 1.8272, -0.7636, -0.7636)$.

14. *Problem 14 (g14) (Himmelblau, 1972)*

Minimize $$f(x) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$$

Subject to:

$$h_1(x) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0$$

$$h_2(x) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0$$

$$h_3(x) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0$$

$$0 < x_i \le 10\,,\; i = 1,\ldots,10$$

Where $c_1 = -6.089, c_2 = -17.164, c_3 = -34.054,$

$c_4 = -5.914, c_5 = -24.721, c_6 = -14.986,$

$c_7 = -24.1, c_8 = -10.708, c_9 = -26.662, c_{10} = -22.179$

The optimum value is $f(x^*) = -47.7648$ at

$x^* = (0.04066, 0.14772, 0.78320, 0.00141, 0.48529,$

$0.00069, 0.02740, 0.017950, 0.03732, 0.09688)$

15. Problem 15 (g15) (Himmelblau, 1972)

Minimize $f(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$

Subject to:

$$h_1(x) = x_1^2 + x_2^2 + x_3^2 - 25 = 0$$

$$h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0$$

$$0 \le x_i \le 10\,,\; i = 1,2,3$$

The optimum value is $f(x^*) = 961.7150$ at $x^* = (3.5121, 0.2169, 3.5521)$

16. Problem 16 (g16) (Himmelblau, 1972)

Minimize $f(x) = 0.00011y_{14} + 0.1365 + 0.00002358y_{13}$

$$+ 0.000001502y_{16} + 0.0321y_{12} + 0.004324y_5$$

$$+ 0.0001\frac{c_{15}}{c_{16}} + 37.48\frac{y_2}{c_{12}} - 0.0000005843y_{17}$$

Subject to:

$$g_1 = \frac{0.28}{0.72}y_5 - y_4 \le 0, \quad g_2 = x_3 - 1.5x_2 \le 0,$$

$$g_3 = 3496\frac{y_2}{c_{12}} - 21 \le 0, \quad g_4 = 110.6 + y_1 - \frac{62{,}212}{c_{17}} \le 0, \quad g_5 = y_1 - 405.23 \le 0,$$

$$g_6 = 213.1 - y_1 \le 0\,, \quad g_7 = y_2 - 1053.6667 \le 0, \quad g_8 = 17.505 - y_2 \le 0$$

$g_9 = y_3 - 35.03 \le 0$, $g_{10} = 11.275 - y_3 \le 0$, $g_{11} = y_4 - 665.585 \le 0$,

$g_{12} = 214.228 - y_4 \le 0$, $g_{13} = y_5 - 584.463 \le 0$, $g_{14} = 7.458 - y_5 \le 0$,

$g_{15} = y_6 - 265.916 \le 0$, $g_{16} = 0.961 - y_6 \le 0$, $g_{17} = y_7 - 7.046 \le 0$,

$g_{18} = 1.612 - y_7 \le 0$, $g_{19} = y_8 - 0.222 \le 0$, $g_{20} = 0.146 - y_8 \le 0$,

$g_{21} = y_9 - 273.366 \le 0$, $g_{22} = 107.99 - y_9 \le 0$, $g_{23} = y_{10} - 1286.105 \le 0$,

$g_{24} = 922.693 - y_{10} \le 0$, $g_{25} = y_{11} - 1444.046 \le 0$, $g_{26} = 926.832 - y_{11} \le 0$,

$g_{27} = y_{12} - 537.141 \le 0$, $g_{28} = 18.766 - y_{12} \le 0$, $g_{29} = y_{13} - 3247.039 \le 0$,

$g_{30} = 1072.163 - y_{13} \le 0$, $g_{31} = y_{14} - 26844.086 \le 0$, $g_{32} = 8961.448 - y_{14} \le 0$,

$g_{33} = y_{15} - 0.386 \le 0$, $g_{34} = 0.063 - y_{15} \le 0$, $g_{35} = y_{16} - 140{,}000 \le 0$,

$g_{36} = 71{,}084.33 - y_{16} \le 0$, $g_{37} = y_{17} - 12{,}146{,}108 \le 0$, $g_{38} = 2{,}802{,}713 - y_{17} \le 0$,

Calculations:

$y_1 = x_2 + x_3 + 41.6$, $c_1 = 0.024x_4 - 4.62$, $y_2 = \frac{12.5}{c_1} + 12$,

$c_2 = 0.0003535{x_1}^2 + 0.5311x_1 + 0.08705y_2x_1$, $c_3 = 0.052x_1 + 78 + 0.002377y_2x_1$,

$y_3 = \frac{c_2}{c_3}$, $y_4 = 19y_3$,

$$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3,$$

$c_5 = 100x_2$, $c_6 = x_1 - y_3 - y_4$, $c_7 = 0.95 - \frac{c_4}{c_5}$, $y_5 = c_6c_7$,

$y_6 = x_1 - y_5 - y_4 - y_3$, $c_8 = (y_5 + y_4)0.995$, $y_7 = \frac{c_8}{y_1}$, $y_8 = \frac{c_8}{3798}$,

$c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153$, $y_9 = \frac{96.82}{c_9} + 0.321y_1$,

$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6$, $y_{11} = 1.71x_1 - 0.452y_4 + 0.58y_3$

$c_{10} = \frac{12.3}{752.3}$, $c_{11} = (1.75y_2)(0.995x_1)$, $c_{12} = 0.995y_{10} + 1998$, $y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$

$$y_{13} = c_{12} - 1.75y_2 \quad , \quad y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5}$$

$$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095 \quad , \quad y_{15} = \frac{y_{13}}{c_{13}}$$

$$y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13} \quad , \quad c_{14} = 2324y_{10} - 28740000y_2$$

$$y_{17} = 14130{,}000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}} \quad , \quad c_{15} = \frac{y_{13}}{y_{15}} - \frac{y_{13}}{0.52}$$

$$c_{16} = 1.104 - 0.72y_{15} \quad , \quad c_{17} = y_9 + x_5 \ .$$

$704.4148 \le x_1 \le 906.3855$, $68.6 \le x_2 \le 288.88$, $0 \le x_3 \le 134.75$, $193 \le x_4 \le 287.0966$, $25 \le x_5 \le 84.1988$.

The optimum value is $f(x^*) = -1.905155$ at

$x^* = (-0.65777, -0.15341, 0.32341, -0.94625, -0.65777,$

17. *Problem 17 (g17) (Himmelblau, 1972)*

Minimize $f(x) = f_1(x_1) + f_2(x_2)$

Constraints:

$$f_1(x_1) = \begin{cases} 31x_1 & 0 \le x_1 \le 300 \\ 30x_1 & 300 \le x_1 \le 400 \end{cases}$$

$$f_2(x_2) = \begin{cases} 28x_2 & 0 \le x_2 \le 100 \\ 29x_2 & 100 \le x_2 \le 200 \\ 30x_2 & 200 \le x_2 \le 1000 \end{cases}$$

$$x_1 = 300 - \frac{x_3x_4}{131.078}\cos(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\cos(1.47588)$$

$$x_2 = -\frac{x_3x_4}{131.078}\cos(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\cos(1.47588)$$

$$x_5 = -\frac{x_3x_4}{131.078}\sin(1.48477 + x_6) + \frac{0.90798x_4^2}{131.078}\sin(1.47588)$$

$$200 - \frac{x_3x_4}{131.078}\sin(1.48477 - x_6) + \frac{0.90798x_3^2}{131.078}\sin(1.47588) = 0$$

$0 \le x_1 \le 400$

$0 \le x_2 \le 1000$

$340 \le x_3 \le 420$

$340 \le x_4 \le 420$

$-1000 \le x_5 \le 1000$

$0 \le x_6 \le 0.5236$

18. *Problem 18 (g18) (Himmelblau, 1972)*

Minimize $f(x) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$

Subject to:

$g_1(x) = x_3^2 + x_4^2 - 1 \le 0$

$g_2(x) = x_9^2 - 1 \le 0$

$g_3(x) = x_5^2 + x_6^2 - 1 \le 0$

$g_4(x) = x_1^2 + (x_2 - x_9)^2 - 1 \le 0$

$g_5(x) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \le 0$

$g_6(x) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \le 0$

$g_7(x) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \le 0$

$g_8(x) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \le 0$

$g_9(x) = x_7^2 + (x_8 - x_9)^2 - 1 \le 0$

$g_{10}(x) = x_2x_3 - x_1x_5 \le 0$

$g_{11}(x) = -x_3x_9 \le 0$

$g_{12}(x) = x_5x_9 \le 0$

$g_{13}(x) = x_6x_7 - x_5x_8 \le 0$

$-10 \le x_i \le 10$, $i = 1,\ldots,8$, $0 \le x_9 \le 10$

The optimum value is $f(x^*) = -0.8660$ at

$$x^* = (-0.65777, -0.15341, 0.32341, -0.94625, -0.65777, -0.75321, 0.32341, -0.34646, 0.59979)$$

19. Problem 19 (g19) (Floudas, 1999)

Minimize $f(x) = -x_1 - x_2$

Subject to:

$$g_1(x) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \le 0,$$

$$g_2(x) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 2 \le 0$$

$$0 \le x_1 \le 3,\ 0 \le x_2 \le 4.$$

The optimum value is $f(x^*) = -5.50801$ at $x^* = (2.32952, 3.17849)$

20. Problem 20 (g20) (Himmelblau, 1972)

Minimize $f(x) = (x_1 - 2)^2 + (x_2 - 1)^2$

Subject to:

$$h_1(x) = x_1 - 2x_2 + 1 = 0 \quad , \quad g_2(x) = -\frac{x_1^2}{4} - x_2^2 + 1 \ge 0$$

The optimum value is $f(x^*) = 1.393$ at $x^* = (0.823, 0.911)$