PARTICLE SWARM OPTIMIZATION TECHNIQUES FOR SOLVING NUMERICAL AND ENGINEERING OPTIMIZATION PROBLEMS

A Thesis submitted in the partial fulfillment of the requirements for the award of degree of

DOCTOR OF PHILOSOPHY

in

Electronics Science

by

LAYAK ALI

School of Physics
University of Hyderabad
Hyderabad - 500046
INDIA

November 4, 2011

School of Physics
University of Hyderabad
Hyderabad, India

CERTIFICATE

This is to certify that the work described in this thesis entitled "**Particle Swarm Optimization techniques for solving numerical and engineering optimization problems**" has been carried out by **Layak Ali** bearing the Reg. No. 06PHPE02, under my direct supervision and this has not been submitted for any degree or diploma at this or any other University.

Dr. Samrat L. Sabat
Reader
School of Physics
University of Hyderabad

Prof. C. Bansal
Dean
School of Physics
University of Hyderabad

School of Physics
University of Hyderabad
Hyderabad, India

DECLARATION

I, Mr. Layak Ali here by declare that the work embodied in this dissertation entitled "**Particle Swarm Optimization techniques for solving numerical and engineering optimization problems**" submitted to the University Of Hyderabad, Hyderabad, for partial fulfillment of the degree of Doctor of Philosophy in Electronics Science has been carried out by me under the supervision of Dr.Samrat L.Sabat, School of Physics, University Of Hyderabad. To the best of my knowledge, this work has not been submitted for any other degree in any University.

Layak Ali
Ph. D. (Electronics Science)
Reg. No: 06PHPE02

To

My Parents

Acknowledgements

This work would have never been accomplished without God Almighty with his blessings and his power that work within me.

I would like to express my sincere gratitude to my research supervisor **Dr. Samrat L. Sabat** for his constant support, encouragement and helping me in selecting my thesis topic. I have greatly benefited by his valuable suggestions and ideas. It is with great pleasure that I express my deep sense of gratitude to him for his able guidance, constant encouragement and patience throughout this work. The research has been a learning and growing experience for me.

I take this opportunity to thank Dr. Siba K. Udgata, Dr. K. P. N. Murthy, Prof. G. Rajaram and Dr. M Ghanashyam Krishna for their valuable suggestions.

I express my gratitude to Prof. Allah Baksh for allowing me to use the resources available in the college and regular encouragement. I am thankful to all teaching and non-teaching staff of the college, and all other supporting members.

I am very thankful to Lina, who has been morale boosters to me and encouraged personally in many situations throughout my research work.

I am thankful to Rangababu, Kiran, Shravan, Srinu and Vijay Raju for their co-operation and suggestions. I wish to take this opportunity to thank my friends who encouraged and helped me throughout my research work.

I am thankful to Mr. Rahul Kashyap for his kind cooperation.

I would be failing in my duties if I do not make a mention of my family mem-

bers including my mother, my wife and my children for providing moral support, without which this work would not have been completed.

I would like to thank everyone who has helped me knowingly or unknowingly during my whole research work.

Layak Ali

Abstract

The success of technology is mostly dependent on how well the real world applications or problems are formulated, controlled and optimized. The complexities associated with real world problems are increasing day by day. The real world problems are often characterized by noisy, incomplete data or multimodality due to their inflexible construction. This demands a robust and efficient optimization or computational paradigm. Since the conventional optimization algorithms do not provide good solutions while optimizing nondifferentiable, nonseparable, discontinuous, discrete and multimodal problems, nature inspired algorithms are the most sorted out paradigms for handling such real world problems. The nature inspired algorithms have evolved over decades and often contain many simple individuals which when work together, produce complex emergent behavior and can be used to solve complex optimization problems. Among many nature inspired algorithms, the Swarm Intelligence (SI) is widely used over decades. SI is an innovative, distributed and intelligent paradigm for solving optimization problems, developed from the inspiration of biological phenomena such as swarming, flocking and herding of different entities. One of the very simple SI based computational algorithm is Particle Swarm Optimization (PSO). It is a decade old concept in the optimization domain introduced in 1995 by Kennedy and Eberhart. Being a stochastic algorithm it exhibits many similarities with the other evolutionary algorithms. PSO essentially imitates the food foraging behavior of swarm of birds or school of fish. The main source of swarm's search capability is the interaction among the individuals and the reaction to others experience in reaching the goal.

Almost all the real world optimization problems can be modeled as any of the four types of optimization problems; i) Single Objective Unconstrained, ii) Single Objective Constrained, iii) Multi Objective Unconstrained and iv) Multi Objective Constrained problem. Though many variants of PSO are being developed to solve these kind of problems, it still suffers with major problems like premature

convergence, poor quality of solution and uncertainty in solution.

- Premature convergence: PSO ends up searching on early best solutions. This is predominant in multimodal functions.
- Convergence speed: PSO explores the good solution in its early stage of search but get stagnated for exploiting the global solution.
- Quality of solution. PSO produces low quality solution due to inherent complexity, discontinuity and multimodality of the problem.
- Uncertainty in solution. Due to stochastic nature, PSO produces different solutions in different runs.
- Instability: Instability caused by particles accelerating out of the solution space.
- PSO neither internally nor externally has the mechanism to handle constraints.
- The associated constraints may be linear and or non-linear; this makes the problem more complex.
- Due to complex solution space of the problems, PSO gets stuck in local optima.
- PSO again neither internally nor externally has the mechanism to handle multiple objectives and multiple objectives with associated constraints.

The work presented in this thesis concentrates on developing efficient and robust PSO variants for solving complex optimization problems. The thesis is presented in the following manner: Chapter 1 presents introduction to optimization. Single Objective Unconstrained Optimization concept, challenges and research contribution are presented in chapter 2. Single Objective Constrained Optimization concept, challenges and research contribution are presented in chapter 3. Multi Objective Unconstrained Optimization concept, challenges and research contribution are presented in chapter 4. Multi Objective Constrained Optimization concept, challenges and research contribution are presented in chapter 5. The

chapter 6 presents the graphical user interface tool developed for solving different types of optimization problems. The thesis is concluded with future research possibilities.

Chapter 1: Introduction

This chapter presents an introduction and theoretical background of optimization and swarm intelligence techniques. It also deals with detailed description of Particle Swarm Optimization. Further this chapter also presents the mathematical model of PSO.

Chapter 2: Single Objective Unconstrained Optimization

In Single Objective Unconstrained Optimization problems, there will be only one objective function without any constraints for optimization. The characteristics of the problems in this domain were investigated. It was observed that the objective functions with multiple optima, discontinuity in curvature and non separability of variables leads to divergence and were responsible for poor quality of solution. Further the experiments were conducted on the PSO dynamics and it was found that the basic PSO was inefficient in handling complex multimodal problems. The major problems with basic PSO were 1) Premature convergence (PSO ends up searching on early best solutions), 2) Convergence speed (PSO explores the good solution in its early stage of search but gets stagnated in the process of exploiting the global solution), 3) Poor quality of solution (PSO produced low quality of solution) and 4) Uncertainty in solutions (Due to stochastic nature, PSO produces different solutions in different runs). The PSO state of the art was implemented and tested on the standard benchmark problems. It was observed that the same problems persist with different popular PSO variants, though little bit improved but not completely resolved. In this thesis the research work was carried out to address the problems persisting with PSO. The first enhancement made to PSO was to overcome the premature convergence and to produce good quality of solutions. It was found that some of the particles in PSO were stuck in local optima during the search process and these were the particles that causes premature convergence and results into poor solution. To address these problems, Accelerated Exploration Particle Swarm Optimizer (AEPSO) was proposed, where the particles responsible for premature convergence are identified and accelerated as diverged particles. The diverged particles were selected based on their fitness value and the Euclidian distance from the global solution. For minimization problems, lower the fitness value better is the particle and hence

particles which shows higher fitness value were selected as diverged particles. The particles that were far away from the global solution are also selected as diverged particles (based on their Euclidean distance). These diverged particles were then accelerated in the direction of best solution. Only twenty percent of the diverged particles were selected for acceleration. Experiments were conducted, by selecting diverged particles from one percent to ninety nine percent for acceleration and not much noticeable improvement was seen. Twenty percent of the diverged particles were selected for acceleration and rest of the particles were allowed to search randomly in the solution space. During acceleration, it was determined that complete swarm collapses if diverged particles are re-initialized with best solution. Hence in AEPSO the diverged particles were forced to search around the best solution with a search patch of fixed size instead of re-initializing to the best solution. The diverged particles are accelerated at the end of every $\mathbf{10}^{\boldsymbol{th}}$ iteration (ex. in 1000 iterations and 10 refresh rate, acceleration will be only for 100 times). The experiments were conducted on different values of refresh rate, and we observed that refresh rate of ten was optimum for getting good solution. The comprehensive analysis of AEPSO was carried out on complex and multimodal CEC 2005 numerical functions of dimension 10, 30, 50 and 100. The objective functions were further rotated and shifted to test the effectiveness of the algorithm. The comprehensive analysis of the developed algorithm and with the state of the art was carried out using two different kinds of metrices, i) conventional (convergence, mean results and standard deviation) and ii) statistical significance tests (Friedman′s test followed by Dunn′s multiple comparison test). The statistical results reveal that the proposed algorithm was competitive and in some cases outperforms the state of the art. The proposed algorithm produced prominent results on normal and rotated problems of dimensions 10, 30, 50 and 100. The performance of AEPSO does not degrade with increased or higher dimension of the problems. Further experiments were conducted on PSO tuning parameter and update strategy with respect to coordinate system. In literature almost all the tuning parameters of PSO are updated in cartesian coordinate system, and it was found that we can have better control over these parameters if they are updated in hyperspherical coordinate system. The premature convergence in PSO is mainly caused due to particles stuck in local minima. This problem can be resolved if the particles are sufficiently intelligent to make the variation in their coordinates minutely. Small changes can be made efficiently with information about angle.

Since hyperspherical coordinate system gives information about the angle, the particles position and velocity are updated in hyperspherical coordinates. Due to this, the magnitudes and the directions of the particles are changed in a controlled manner and hence particles come out of the deep local minima to search for better solution. Thus the concept of AEPSO was further integrated with hyperspherical updates and proposed a new algorithm namely Hyperspherical Acceleration Effect Particle Swarm Optimization (HAEPSO). The comprehensive analysis of this algorithm was also carried out on the same set of CEC 2005 benchmark functions (dimension 10, 30, 50 and 100) and two different kinds of metrices, i) conventional (convergence, mean results and standard deviation) and ii) statistical significance tests (Friedman′s test followed by Dunn′s multiple comparison test). HAEPSO showed superior performance over its counterparts in terms of quality of solution on normal and rotated problems with increased dimensions. We also studied learning strategy of PSO and Comprehensive Learning Particle Swarm Optimizer (CLPSO). In comprehensive learning the particles learn and update their position based on other particles of different dimension and at different time. This learning strategy was integrated with our proposed HAEPSO, to evolve Integrated Learning Particle Swarm Optimizer (ILPSO). The ILPSO also has shown the superior performance over the other algorithms including HAEPSO. The performance of ILPSO was not effected with the increase in the dimensions of the problems (evident from statistical results), especially on normal and rotated problems. Further the research was conducted on search patch size of AEPSO and we found that the fixed and unit radius search patch was not an efficient method for better searching and hence proposed an adaptive search patch. Thus AEPSO was extended with adaptive search patch and we called it as Adaptive Accelerated Exploration Particle Swarm Optimizer (AAEPSO). In AAEPSO, the search patch around best solution was adapted with iterations. In the beginning of the search process, size of the search patch was kept as per search range of solution space. With iterations the size of this search patch was decreased exponentially. This strategy helped the PSO for more exploration during the initial stage and more exploitation during final search stage. Further in AEPSO, number of diverged particles selected for acceleration were fixed (twenty percent of the diverged particles) and this was investigated again. This strategy of considering fixed number of particles has restricted the population diversity. Thus in AAEPSO, we used random number of particles from the pool of identified diverged particles to search around the best

solution and remaining diverged particles were initialized randomly in the search space. These above two strategies has elevated the performance to a great extent. The comprehensive analysis of AAEPSO was carried out on complex test suite. The effectiveness of the developed algorithm AAEPSO was established by the statistical results with its counterparts. The AAEPSO has shown excellent convergence and quality of solution on lower dimension of normal and rotated problems, but as dimensions of the problem increases, the AAEPSO failed to achieve good results. The major feature of AAEPSO has been proved on shifted and shifted rotated problems.

Chapter 3: Single Objective Constrained Optimization

Further research was carried on Single Objective Constrained Optimization problems, that has only one objective function having one or more constraints associated with it. It was found that the presence of constraints had prevented the PSO from reaching the global solution. The PSO does not have either internal or external mechanism to handle constraints. We conducted many experiments on different constraint handling mechanism with PSO. Though penalty method is a popular approach, quality of solution mainly depends on the choice of penalty factor. It was found that PSO along with Stochastic Ranking (SR) method produced acceptable results. The SR mechanism is computationally efficient, does not depend on any external tuning parameters and also independent of problem characteristics. This motivated us to integrate SR with AEPSO for a constrained optimization problem. The developed algorithm is named Stochastic Ranking Particle Swarm Optimization (SRPSO). The SR mechanism uses a simple bubble sort algorithm to rank the individuals based on the constraints violations. In SR, a probability $\boldsymbol{P_f}$ was introduced to rank the individuals. The two adjacent individuals were used for ranking. If both were in feasible space, the individual with smaller objective values gets higher rank. If both adjacent individuals were in infeasible space, the individual with smaller objective value occupies the rank with the probability $\boldsymbol{P_f}$. If one particle is in feasible space and the other one in infeasible space, then the particle in feasible space gets the higher rank. The lower ranked particles were treated as diverged particles and they were accelerated towards higher ranked particles. The performance comparison of developed algorithm SRPSO was carried out on standard CEC 2006 numerical benchmark and engineering design problems with two performance indicators viz. conventional and statistically significance metrices. The results have shown that the SRPSO

performed well on most of the numerical benchmark problems and has shown superior performance on complex engineering design problems.

Chapter 4: Multi Objective Unconstrained Optimization

Further we addressed Multi Objective Unconstrained Optimization problems. These problems will have multiple objectives to be optimized simultaneously without any constraints. PSO has neither internal nor external mechanisms to optimize multiple objectives simultaneously, and the conflicting objectives deteriorate the performance of the optimizing algorithms. Multi objective unconstrained optimization problems have multiple objectives and multiple solutions, thus they are explained using Pareto optimality concept. The Pareto optimality denote a set of trade off solutions. AAEPSO algorithm proposed for single objective unconstrained optimization exhibited noticeable performance improvement which became the main motivation for applying it to the multiobjective problems. Thus we proposed Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (MOAAEPSO) as a variant of AAEPSO. While solving multi objective unconstrained optimization problems there are multiple solutions produced by the particles, and the desired ones are called nondominated and undesired ones are called dominated. The dominated set of particles causes stagnation and poor quality of solution. Identification and strategic updates of these dominated solutions are required to improve quality of solution. Thus in MOAAEPSO the dominated particles were identified and accelerated towards the nondominated particles. Similar to AAEPSO, the adaptive search patch around the nondominated particle was maintained. Since PSO does not have an internal memory to store the nondominated solutions produced, a bounded external archive was maintained. If the nondominated solutions produced were more than the size of the archive, then crowding distance was used to eliminate the members from the archive. The proposed algorithm was validated on a set of complex multi objective benchmark problems chosen from CEC 2009. Different performance indicators such as conventional, statistical and graphical (pareto front) used for performance comparison of the algorithms. Conventional performance indicators include convergence metric, diversity metric, spacing and inverted generational distance (IGD), similarly statistically significant indicators include hypervolume indicator and ANOVA followed by Pareto front graphs. The statistical results revealed that proposed MOAAEPSO algorithm gives competitive results.

Chapter 5: Multi Objective Constrained Optimization

Many science and engineering design optimization problems can be converted to multiobjective constrained optimization problem. Thus we have further extended our work to multiobjective constrained optimization problems. These problems have multiple objectives to be simultaneously optimized along with one or more constraints. PSO does not have the mechanism to handle multiple objectives and their associated constraints. To address these problems we have divided the solution space into four categories. Based on constraints, the solution will be either feasible or infeasible. The feasible solutions are always desired. Based on multiple objectives, the solution will be either dominated or nondominated and nondominated are desired. In constrained multiple objective (MOCO) problem, the solutions will be either of the four types a) feasible − dominated, b) feasible − nondominated, c) infeasible − dominated and d) infeasible − nondominated. In case of MOCO feasible − nondominated solutions are the desired solutions and rest are undesired solutions. The particles having undesired solutions are called as diverged particles. To solve MOCO problems, we have proposed a new algorithm namely Constrained Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (CMOAAEPSO). In this algorithm the process of selection and acceleration of diverged particles are same as MOAAEPSO, where the undesired particles are accelerated in the direction of feasible − nondominated particles. To maintain the diversity and to prevent the swam from collapse, the diverged particles neither made independent nor allowed to clone the feasible − nondominated, but are accelerated adaptively around the feasible − nondominated particles. Further this adaptive acceleration is done for every 10 iteration (refreshing rate) instead of every iteration. This strategy helped the particles to search other places for finding better Pareto optimal solutions. CMOAAEPSO maintained a bounded external archive to retain feasible − nondominated particles that were produced during search process. If the number of feasible − nondominated solutions exceed the limit of the archive, crowding distance was used to delete the less feasible and less nondominated particles from the archive. We have used two different set of performance metrics; conventional (igd, convergence and robustness), statistically significant (hypervolume indicator and Wilcoxon) and Pareto front graphs for validation of proposed algorithm with current state of the art. The results have revealed that proposed CMOAAEPSO algorithm gives competitive results on benchmark functions and engineering design problems.

Chapter 6: Particle Swarm Optimization Universal Solver

Since the work has been carried out on all the possible domains of the optimization and reported a good algorithm in every domain, we are motivated to develop a universal PSO tool. This is called Particle Swarm Optimization Universal Solver (PSOUS) with user friendly interface. The PSOUS is a Graphical User Interface (GUI) developed in MATLAB environment and possesses the essential attributes of being interactive. PSOUS has the option for selecting type of problem, to enter problem specific parameters. After taking user inputs, the particular solver will be invoked for optimization. To the best of our knowledge this tool is the first of its kind in the optimization domain. There are tools based on different algorithms, but they are either for single objective unconstrained and constrained or multi objective unconstrained and constrained. There hardly exist tools that can span all the four types of optimization domain. This tool will give the researchers across the globe to carry out further research and validate the results. This tool will be helpful even to industrialist and design engineers.

Chapter 7: Conclusions and Future scopes

This chapter presents the over all conclusions of research carried out on all the possible domains of optimization. The chapter is concluded with future scopes of research.

Contents

List of Figures

List of Tables

Abbrevations

AAEPSO - Adaptive Accelerated Exploration Particle Swarm Optimizer
AEPSO - Accelerated Effect Particle Swarm Optimizer
CCEA - Cooperative Co-Evolutionary Algorithm
CDE - Co-evolutionary Differential Evolution
CEC - Congress on Evolutionary Computation
CM - Convergence Metric
CMOAAEPSO - Constrained Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer
CPSA - Constrained Pareto Simulated Annealing
CPSO - Cooperative Approach to Particle Swarm Optimization
CRGA - Changing Range Genetic Algorithm
DM - Diversity Metric
DMOEA - Dynamic Multi Objective Evolutionary Algorithm
DMOEADD - MultiObjective Evolutionary Algorithm with Domain Decomposition
EDWA - Evolutionary Dynamic Weighted Aggregation
EMMOPSO - Elitist Mutated MultiObjective Particle Swarm Optimization
FDRPSO - Fitness-Distance Ratio Based Particle Swarm Optimization
FIPSO - Fully Informed Particle Swarm Optimization
GA - Genetic Algorithm
GDE3 - Generalized Differential Evolution 3
GEATbx - Genetic and Evolutionary Algorithm Toolbox
GPSO - Gregarious Particle Swarm Optimization
GUI - Graphical User Interface
HAEPSO - Hyperspherical Acceleration Effect Particle Swarm Optimization
HPSO –Hierarchical Particle Swarm Optimizer
IEMO - Interactive Evolutionary Multi-objective Optimization tool

IGD - Inverted Generational Distance
ILPSO - Integrated Learning Particle Swarm Optimizer
ITCEMOP - ITerative Co-Evolutionary Multiojective Optimization
KEA - kit for evolutionary algorithms LE - Linear Equality constrained
LI - Linear Inequality constrained
MOAAEPSO - Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer
MOCO - Multi Objective Constrained Optimization
MOEA - Multi Objective Evolutionary Algorithms
MOES - MultiObjective Evolution Strategy
MOGA - Multi-Objective Genetic Algorithms
MOSaDE - MultiObjective Self adaptive Differential Evolution
MOUO - Multi Objective Unconstrained Optimization
MTS - Multiple Trajectory Search
NE - Nonlinear Equality constrained
NFL - No Free Lunch theorem
NI - Nonlinear Inequality constrained
NNIA - Nondominated Neighbor Immune Algorithm
NSGA - Nondominated Sorting Genetic Algorithm
OEGADO - objective exchange genetic algorithm for design optimization
PSO - Particle Swarm Optimization
PSOUS - Particle Swarm Optimizer Universal Solver
S - Spacing
SAPF - Self Adaptive Penalty Function
SI - Swarm Intelligence
SMES - Simple Multi-membered Evolution Strategy
SOCO - Single objective Constrained Optimization
SOUO - Single Objective Unconstrained Optimization
SPEA - Strength Pareto Evolutionary Algorithm
SR - Stochastic Ranking
SRPSO - Stochastic Ranking Particle Swarm Optimization
TVAC - Time Varying Acceleration Coefficients
VIDEO - Visually Interactive Decision-making Evolutionary Optimization

Chapter 1

INTRODUCTION

This is an introductory chapter of the thesis. It presents basics of optimization, different types of optimization problem and techniques for solving the problems. It also introduces Swarm Intelligence and Particle Swarm Optimization algorithm in detail.

1.1 What is Optimization

Optimization is the branch of computational science that deals with methods to obtain best possible answer for a given set of problem. The quality of solution can be expressed in terms of a numerical value and is problem dependant. Thus the aim of optimization is to select the best possible decision for a given set of circumstances. In recent years the subject of computational science has matured and is widely used in innumerable scientific and engineering applications. The applications include and are not limited to the following domains; optimal petroleum refining, aircrafts designs with minimum weight, optimal missile trajectories, profitable business activities, physical, chemical and biological sciences, engineering, architecture, economics and management. There are many algorithms or techniques available to solve optimization problems. The history of optimization goes back many centuries; Euclid (300 B.C.) and Heron (100 B.C.) solved the problem of light traveling between two points and found the shortest path between the points. The real valued non-linear unconstrained or constrained functions were optimized by Gottfried Leibniz, Isaac Newton, Leonhard Euler and Joseph Lagrange. They assumed that the objective and constraint functions are differen-

tiable. The constrained nonlinear function optimization was addressed by Harold Kuhn(1951), Albert Tucker (1951) and Karush. The optimization was basically based on Leibniz−Newton principles and therefore could not easily escape local optima. The Monte Carlo method of simulation was studied and implemented in around 1940. The evolutionary method for nonlinear optimization was first introduced by George Box in 1957. Box et. al. in 1965 developed a complex method by exploiting the entire search space for generating random numbers. This method had a great potential to escape local optima and locate the global optima while optimizing nonlinear functions. The John Nelder and Roger Mead in 1964 has introduced simplex method. They incorporated the ability to learn from its earlier search experience and the algorithm was adapted to the topography of the surface of the optima and function. The major break through in evolutionary computational science was seen by the noticeable work of John Holland in 1975. He introduced and implemented the popular algorithm namely Genetic Algorithm (GA). The Simulated Annealing method was proposed to mimic the annealing process in metallurgy by Kirkpatrick et al., in 1983 and Cerny in 1985. In 1992 Marco Dorigo introduced Ant Colony Optimization (ACO) that mimics the social behavior of ant. James Kennedy and Russell Eberhart introduced Particle Swarm Optimization (PSO) in 1995. PSO mimics the food foraging behavior of swarm of birds or insects or a school of fish etc.

Mathematically an optimization problem can be formulated as

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ & f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x}) \\ subject\ to: \quad & g_i(\overrightarrow{x}) < 0, i = 1, 2, \ldots, q \\ & h_i(\overrightarrow{x}) = 0, i = q+1, q+2, \ldots, m \\ Where\ & l_i \leq x_i \leq u_i, i = 1, 2, \ldots, D \end{aligned} \tag{1.1}$$

Here $\overrightarrow{x}$ is a solution vector, D is the dimension of problem, k refers to the number of objective functions need to be simultaneously optimized. The f_i, $\forall$ i $\in$ k is termed as fitness of objective function to be optimized. The g_i, and h_i are the inequality and equality constraints respectively. The values l_i and u_i, $\forall$ i $\in$ D are the lower and upper bounds defining the search space.

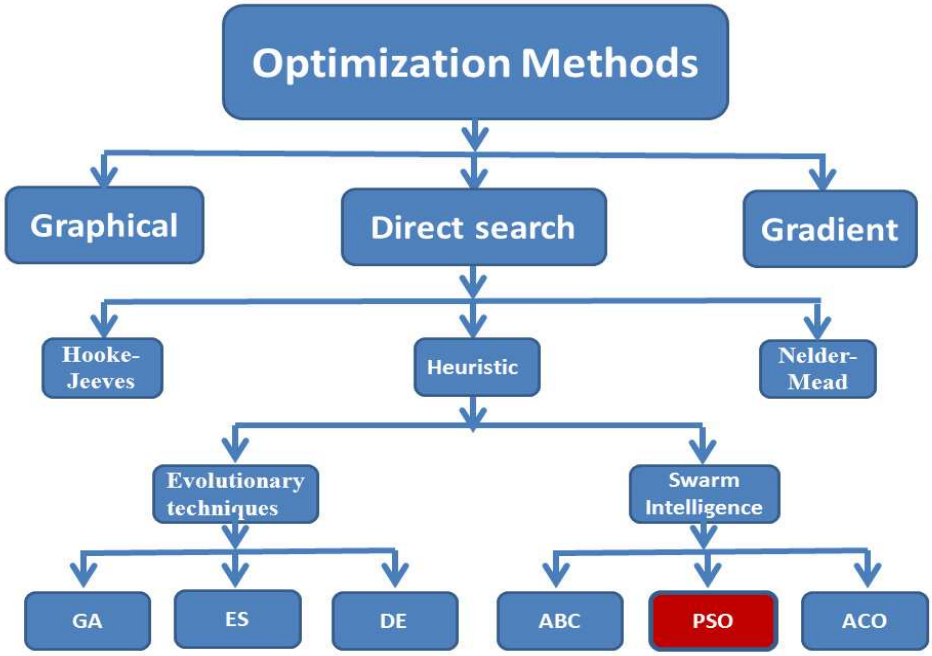

Figure 1.1: Diagram illustrating Optimization methods

The optimization process addresses the problem of determining the values of the independent variables that do not violate the constraints and at the same time give an optimal value for the set of objective functions that are being optimized. Thus, optimization techniques can be either a mathematical idea, an algorithm or a tool that find extreme values of an objective function. The broad classification of optimization techniques is shown in Figure 1.1. The oldest technique for solving optimization problem is the *Graphical* method. In this method, optimization is found by means of graphs. The major shortcomings of this technique are 1) limited to only two variables (2D), 2) less accurate, 3) time consuming and 4) difficult for non-linear problems.

The most commonly used technique is *Gradient Method*, this uses the gradient information of the objective function to find optimal solution. The major drawbacks of this method are; 1) it is best used on well behaved systems where there is one clear optima, 2) in case of multimodal problems global optima can not be found, 3) it takes many iterations to converge to local optima if curvature in different directions is very different and 4) this is applicable only for differentiable and continuous funtion. Another commonly used technique is *Direct search* method. This method does not need the gradient information of the problem.

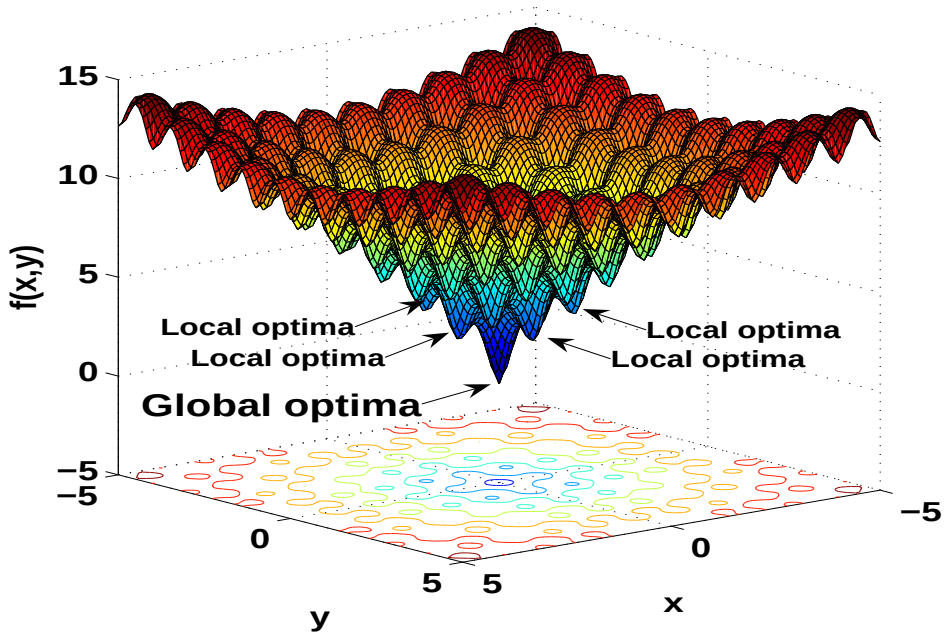

Figure 1.2: Diagram illustrating global and local optima

This include many iterative methods including Hooke-Jeeves, Nelder-Mead and Heuristic techniques. The Heuristic method for optimization has dominated the optimization domain very recently and is most sorted method for optimization. In Heuristic method optimum solution is found by means of Exploitation and Exploration. Exploitation: examines nearby space and moves towards the local optima. Exploration: causes deviation from local minima to other regions where improved solutions are possible. Among all the Heuristic methods Swarm Intelligence (SI) based methods have evolved as a popular method for optimization. Particle Swarm Optimization (PSO) is one of the SI based method, which is very simple to implement and produce equally good results. Since optimization plays a vital role in almost all scientific and engineering applications, hence there is a ever growing demand to develop efficient and robust optimization techniques. Several books, based on mathematical concepts of optimization are available in literature [1], [2], [3], [4] and [5].

1.1.1 Global Optimization

In case of multimodal, functions there are multiple optima and one global optima as shown in Figure 1.2. In general, there can be solutions that are locally optimal but not globally optimal. Consequently, global optimization problems are typically quite difficult to solve exactly. The primary aim of global optimization is to find the best solution of decision models, in presence of the multiple local solutions. Global optimization problems fall within the broader class of Non-Linear Programming (NLP).

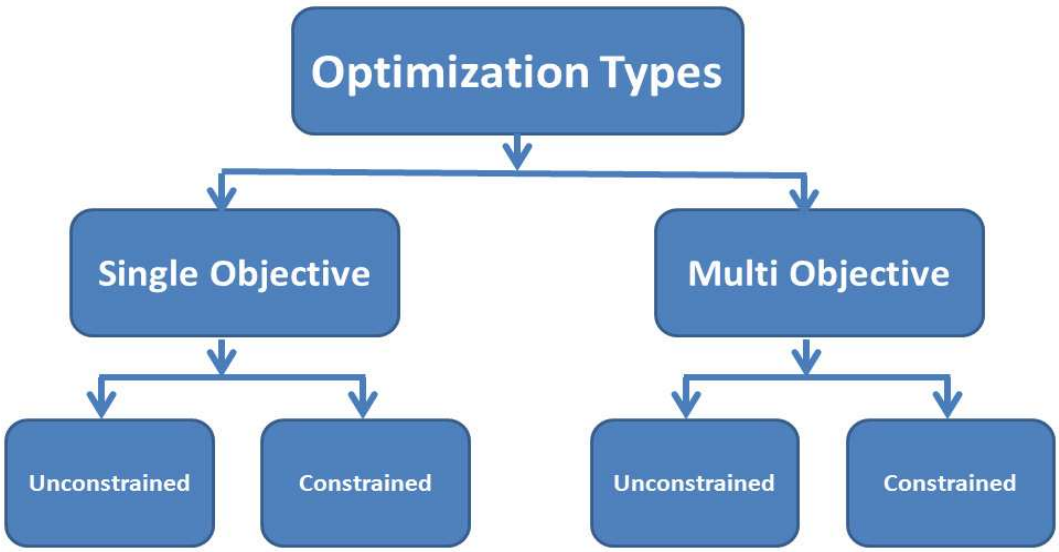

Figure 1.3: Diagram illustrating Optimization Types

1.1.2 Types of Optimization

The global optimization can broadly be categorized in to four major domain based on the number of objectives and or constraints involved. Figure 1.3 shows the different types of optimization problems that are discussed below.

1.1.2.1 Single Objective Unconstrained Optimization

The equation (1.1) states the generic optimization problem. In equation (1.1), if $q = 0$,$m = 0$ and $k = 1$, then this equation reduces to Single Objective Unconstrained Optimization (SOUO). This class of optimization problem involves only one objective function without any constraints. The simplified mathematical equation is given below.

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ &f(\overrightarrow{x}) \\ Where\ l_i \leq x_i \leq u_i, &i = 1, 2, \ldots, D \end{aligned} \tag{1.2}$$

1.1.2.2 Single Objective Constrained Optimization

In equation (1.1), if $q > 0$ and or $m > 0$ and $k = 1$, then this equation reduces to Single Objective Constrained Optimization (SOCO). This class of optimization problem involves only one objective function with atleast one constraint. The constraints may be either inequality ($q > 0$), or equality ($m > 0$), or both ($q > 0$, $m > 0$). The simplified mathematical equation is given below.

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ &f(\overrightarrow{x}) \\ subject\ to: \quad &g_i(\overrightarrow{x}) < 0, i = 1, 2, \ldots, q \\ &h_i(\overrightarrow{x}) = 0, i = q+1, q+2, \ldots, m \\ Where\ l_i \leq x_i \leq u_i, &i = 1, 2, \ldots, D \end{aligned} \tag{1.3}$$

1.1.2.3 Multi Objective Unconstrained Optimization

In equation (1.1), if $q = 0$,$m = 0$ and $k > 1$, then this equation reduces to Multi Objective Unconstrained Optimization (MOUO). This kind of optimization problem involves multiple objective functions without any constraints to be optimized

simultaneously. The simplified mathematical equation is given below.

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ & f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x}) \\ Where\ & l_i \leq x_i \leq u_i, i = 1, 2, \ldots, D \end{aligned} \tag{1.4}$$

1.1.2.4 Multi Objective Constrained Optimization

In equation (1.1), if $q \geq 1$,$m \geq 1$ and $k > 1$, then this equation reduces to Multi Objective Constrained Optimization (MOCO). This kind of optimization problem involve multiple objective functions with constraints. The constraints may be either inequality ($q > 0$), or equality ($m > 0$), or both ($q > 0$, $m > 0$). The simplified mathematical equation is given below.

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ & f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x}) \\ subject\ to: \quad & g_i(\overrightarrow{x}) < 0, i = 1, 2, \ldots, q \\ & h_i(\overrightarrow{x}) = 0, i = q+1, q+2, \ldots, m \\ Where\ & l_i \leq x_i \leq u_i, i = 1, 2, \ldots, D \end{aligned} \tag{1.5}$$

1.2 Swarm Intelligence

Swarm Intelligence (SI) is an innovative, distributed and intelligent system for solving optimization problems, these are developed from the inspiration of biological examples by swarming, flocking and herding phenomena. Conventional computing paradigms show difficulty in optimizing the real world problems. The real world problems often are characterized by noisy, incomplete data or multimodality due to their inflexible construction. Natural systems have evolved since decades to solve the optimization problems. These natural systems often contain many simple elements that, when working together, produces complex emergent behavior. The natural computing paradigms can be used, where conventional computing paradigm perform unsatisfactorily. Swarm Intelligence (SI) belongs to

one such natural computing paradigm.

Since more than five decades, the biologists have shown great interest in studying collective behavior of social animals such as insects, fish, birds and mammals. The first ever theoretical explanation of the collective behaviors in social insects was provided by French biologist Pierre-Paul Grasse. In 1984 Grasse [6] reported the collective behavior of African termites. The first flocking model was developed in 1987 by Craig Reynolds [7]. This was a bio-inspired computational model for simulating the animation of a flock of entities called boid [7]. Collective patterns and decision making was presented in 1989 by Deneubourg, Jean-Louis and Goss, Simon [8]. In 1991 the food foraging and the shortest path between the food sources and the nest in ants was studied by Deneubourg and Goss [9]. Beni and Wang [10] first introduced the term Swarm Intelligence in the context of cellular robotic systems in 1991. In 1992 Marco Dorigo introduced Ant Colony Optimization (ACO) algorithm. The ACO is a heuristic algorithm that is inspired by the food foraging behavior of ants. Particle Swarm Optimization (PSO) is developed by the inspiration of the social behavior of bird flocking or a school of fish, by Eberhart and Kennedy [11] in 1995. In 2005 Artificial Bee Colony (ABC) algorithm was introduced by Karaboga. Essentially ABC algorithm simulates the food foraging behavior of honey bees. The detailed algorithmic explanations of ABC and ACO algorithms are out of scope of the thesis.

1.2.1 Why Swarm Intelligence?

Swarm Intelligence (SI) is a growing research field in natural computing paradigm. It deals with natural and artificial systems composed of many simple individual. Each of them coordinate using decentralized control and self-organization. SI is the outcome of collective behaviors of simple individuals that interacts with each other and with their environment. It has the potential to solve complex, non-differentiable, discontinuous, multimodal and distributed problems. It offers an alternative way to design the systems that are either impossible or near impossible by conventional optimization algorithms. A typical SI algorithm has the following properties.

- It is composed of many simple individuals.
- The individuals are relatively homogeneous.

- The interactions among the individuals are based on simple behavioral rules.
- Individuals exchange information directly or via the environment.
- The overall behavior of the system results from the interactions of individuals with each other and with their environment
- Individuals have the division of labor and distributed task allocation system among them
- Individuals act in a coordinated way without the presence of a coordinator or of an external controller.
- Each individual has a stochastic behavior that depends on its local perception of the neighborhood.

Because of the above properties, SI has captured the prominent place in computational research domain.

1.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is one kind of the SI algorithm and is only a decade old in the optimization domain. PSO was introduced in 1995 by Kennedy and Eberhart [11],[12]. PSO being a stochastic algorithm exhibits many similarities with evolutionary algorithm for solving optimization problems. PSO essentially imitates the food foraging behavior of social life, such as a swarm of birds or school of fish [13]. The main source of swarm's search capability is the interaction among the members and the reaction to one another's findings to reach the goal. In PSO terminology each member of the swarm is called a particle. The word swarm comes from the irregular movement of the particles in the problem space. Every particle in the search space represents a potential solution. During search process every particle remembers its current position and self best position found so far called as personal best (*pbest*). All the particles explore the search space and the information collected by them is sorted to find the best particle in the swarm called the global best (*gbest*). The location of this member is communicated to all the particles and hence the flying trajectory of the particles is altered in the direction of the swarm's *gbest* and it's own *pbest*. Since the particle's position in a swarm represents the potential solution, hence each particle

is evaluated based on the fitness function to be optimized. The value of fitness function extrapolates the quality of solution. As the particle fly randomly in D-dimensional search space, the position and velocity of i^{th} particle are represented as $X_i = (x_{i,1}, x_{i,2}, x_{i,3}, \cdots, x_{i,D})$ and $V_i = (v_{i,1}, v_{i,2}, v_{i,3}, \cdots, v_{i,D})$ respectively. In real life applications D refers to the number of variables present in the objective function to be optimized. With increased iteration, particles of the swarm will move towards the global best position by keeping track of their ($pbest$) and with the influence of ($gbest$).

Figure 1.4 shows the details of cognitive and social component influence on the particle. In a D dimensional search space the $pbest$ of the i^{th} particle is represented as $pbest_i = (p_{i,1}, p_{i,2}, p_{i,3}, \cdots, p_{i,D})$ and the $gbest$ of the whole swarm is represented as $gbest = (g_1, g_2, g_3, \cdots, g_D)$. The PSO algorithm updates the velocity and position of each particle by the following equations (1.6) and (1.7) respectively.

$$\begin{aligned} V_{i,d}^{t+1} &= V_{i,d}^{t} + c_1 * rand_1 * (pbest_{i,d}^{t} - X_{i,d}^{t}) \\ &+ c_2 * rand_2 * (gbest_{d}^{t} - X_{i,d}^{t}) \qquad (1.6) \\ X_{i,d}^{t+1} &= X_{i,d}^{t} + V_{i,d}^{t+1} \qquad (1.7) \end{aligned}$$

where, c_1 and c_2 are the learning factors which determines the relative influence of cognitive and social component respectively. $rand_1$ and $rand_2$ are uniformly distributed random numbers in the range [0,1]. $V_{i,d}^{t}$, $X_{i,d}^{t}$ and $pbest_{i,d}^{t}$ are the velocity, position and the personal best of i^{th} particle in d^{th} dimension for the t^{th} iteration respectively. The $gbest_{d}^{t}$ is the global best of the swarm in d^{th} dimension for the t^{th} iteration.

1.3.1 PSO terminology

The terminologies used to describe PSO are discussed below

1) Particle:- An individual in the swarm (analogous to a bird in flock) is referred as a particle. All the particles in the swarm act independently and accelerate towards the solution.

2) Position:- Position refers to a particles's place in the search space. This is represented by coordinates on the x−y plane. In general, however, we can extend this idea to any N−dimensional space according the problem at hand. This N−

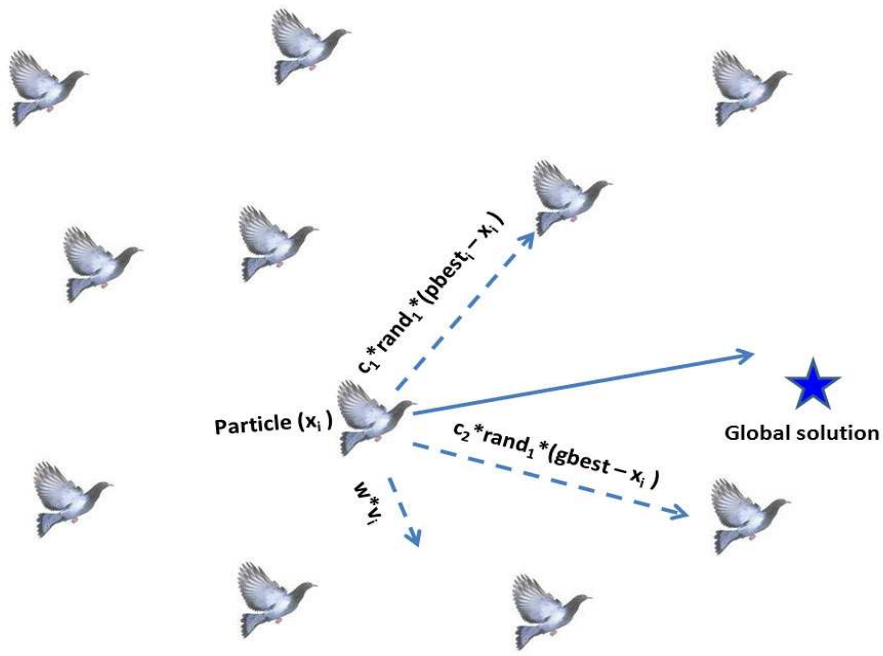

Figure 1.4: Diagram illustrating the influence of cognitive and social component

dimensional space is the solution space for the problem being optimized, where any set of coordinate represents a solution to the problem.

3) Fitness:- Fitness refers to the function or method to evaluate the goodness of a position in search space. It takes the coordinates in the solution space and returns a numerical value (goodness). The fitness function provides an interface between the physical problem and the optimization algorithm.

4) pbest:- This is the information of best location that the particle has found during search process. Each particle has its own *pbest* determined by the path that it has flown. At each point along its path, the particle compares the fitness value of its current location to that of *pbest*. If the current location has a better fitness, *pbest* is replaced with its current location.

5) gbest:- This is the information of best position in the whole swarm. For entire swarm there is one *gbest* to which each particle is attracted. At each point along their path every particle compares the fitness of their current location to that of *gbest*. If any particle is at a location of better fitness then *gbest* is replaced by that particle′s current position.

6) Cognitive factor:- It is a relative pull on the velocity of the particle by the best memory (*pbest*) of the same particle. This is the major factor that alters the

trajectory of the particle fly during due course of search.
7) Social factor:- It is a relative pull on the velocity of the particles by the best memory of the swarm (*gbest*). This is also the major factor to alters the trajectory of the particle fly.

1.3.2 PSO Foundation

PSO algorithm is based on the collective behaviors of the simple agents that interact locally with their environment to create global functional patterns and the intelligence in PSO results from social interaction. Evaluation, comparison and imitation of others, as well as learning from experience allow individuals to adapt to the environment and determine optimal pattern of behavior. Since the exact behavior of PSO can not be explained clearly and is not understood yet, hence an attempt is made with the following explanation.

1.3.2.1 PSO Model: A Newtonian Mechanical Model

The Newtonian mechanical model explains how iterative (velocity and position) equations of Particle Swarm Optimization are deduced from mechanics. Figure 1.4 depicts the relative impact of social and cognitive component on the flying trajectory of the particle. The position and velocity of each particle with mass m can be expressed according to the Newton's second law of motion as

$$\frac{d\overrightarrow{x}}{dt} = \overrightarrow{v}, \frac{d\overrightarrow{v}}{dt} = \overrightarrow{a} = \frac{\overrightarrow{F}}{m} \tag{1.8}$$

Where, $\overrightarrow{F}$ denotes the force on the particles, $\overrightarrow{a}$ and $\overrightarrow{v}$ are the acceleration and velocity. The forward differentiation of equation (1.8) converts it to iterative process as follows

$$\frac{\overrightarrow{x_t} - \overrightarrow{x_{t-1}}}{\Delta t} = \overrightarrow{v} \Longrightarrow \overrightarrow{x_t} = \overrightarrow{x_{t-1}} + \overrightarrow{v_t}\Delta t \tag{1.9}$$

$$\frac{\overrightarrow{v_t} - \overrightarrow{v_{t-1}}}{\Delta t} = \frac{\overrightarrow{F_{t-1}}}{m} \Longrightarrow \overrightarrow{v_t} = \overrightarrow{v_{t-1}} + \frac{\overrightarrow{F_{t-1}}}{m}\Delta t \tag{1.10}$$

Considering m and Δt to be unity in above equations (1.9) and (1.10) we get the following.

$$\overrightarrow{x_t} = \overrightarrow{x_{t-1}} + \overrightarrow{v_t} \tag{1.11}$$

$$\overrightarrow{v_t} = \overrightarrow{v_{t-1}} + \overrightarrow{F_{t-1}} \tag{1.12}$$

All the terms in equations (1.9) and (1.10) are known except force $\overrightarrow{F_{t-1}}$ at $(t-1)^{th}$ iteration. To model this $(\overrightarrow{F_{t-1}})$, the swarm characteristics are considered. The important feature of swarm is; any particle is influenced by the relative pull of *pbest* and *gbest*. This relative pull on swarm is equivalent to two spring like attraction and hence can be expressed as

$$\overrightarrow{F_{t-1}} = c_1(\overrightarrow{pbest_{t-1}} - \overrightarrow{x_{t-1}}) + c_2(\overrightarrow{gbest_{t-1}} - \overrightarrow{x_{t-1}}) \tag{1.13}$$

where c_1 and c_2 are Hook's constants of two springs. The c_1 and c_2 are considered as cognitive and social parts respectively. The behavior of the entire swarm can be expressed by writing the equations above in a matrix form. To inject the randomness into the swarm behavior, Hooke's constants c_1 and c_2 are multiplied by random number generators. Thus the final equation can be expressed as below

$$X_t = X_{t-1} + V_t \tag{1.14}$$

and

$$V_t = V_{t-1} + c_1 rand_1(P_{t-1} - X_{t-1}) + c_2 rand_2(G_{t-1} - X_{t-1}) \tag{1.15}$$

The equations (1.14) and (1.15) are the position and velocity update equations for PSO.

1.4 No Free Lunch Theorem

One of the most interesting development in optimization theory is No Free Lunch (NFL) theorem [14]. This theorem states that the performance of all optimization (search) algorithms, amortized over the set of all possible functions, is equivalent. The implications of this theorem are far reaching, since it implies that no algorithm can be designed so that it will be superior to a linear enumeration of the search space, or even a purely random search. The theorem is only defined over finite search spaces, however, and it is as yet not clear whether the result applies to infinite search spaces. All computer implementations of search algorithms will effectively operate on finite search spaces, though, so the theorem is directly applicable to all existing algorithms. Although the NFL [14] theorem states that all

algorithms perform equally well over the set of all functions, it does not necessarily hold for all subsets of this set. The set of all functions over a finite domain includes the set of all the permutations of this domain.

Chapter 2

Single Objective Unconstrained Optimization

This chapter introduces the concept of Single Objective Unconstrained Optimization (SOUO). The detailed literature survey for solving SOUO problems using Swarm Intelligence (SI), Particle Swarm Optimization (PSO) and other competitive algorithms are presented. It also addresses the major challenges of applying PSO in solving SOUO problems. This chapter is extended with research contributions made in the SOUO domain. The algorithmic and implementation details of developed PSO variants, i.e. i) Accelerated Effect Particle Swarm Optimizer (AEPSO), ii) Hyperspherical Acceleration Effect Particle Swarm Optimization (HAEPSO), iii) Integrated Learning Particle Swarm Optimizer (ILPSO) vi) Adaptive Accelerated Exploration Particle Swarm Optimizer (AAEPSO) are presented. The performance of developed algorithms are tested on complex numerical benchmark functions and evaluated on different statistical performance indicator such as mean, standard deviation and convergence graphs. The statistical significance of the algorithms are analyzed using Friedman′s test followed by Dunn′s multiple comparison test. These indicators are used for performance comparison of developed PSO variant with state of the art. The chapter is concluded with results and discussions.

2.1 Introduction

In Single Objective Unconstrained Optimization (SOUO) domain the objective is to minimize or maximize a single objective function without any constraints, except boundary conditions. The simplified mathematical representation of SOUO problems can be expressed as

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ &f(\overrightarrow{x}) \\ Where\ l_i \leq x_i \leq u_i, &i = 1, 2, \ldots, D \end{aligned} \tag{2.1}$$

Where $\overrightarrow{x}$ is the solution, f is the objective function to be optimized, l_i and u_i are the lower and upper bounds $\forall$ i $\in$ D of the search space.

2.2 Literature Survey

Particle Swarm Optimization (PSO) is a simple population based optimization algorithm introduced in 1995 by Kennedy and Eberhart [12, 11]. It is relatively recent (a decade old) addition to the field of natural computing (Swarm Intelligence). PSO has found widespread application in complex optimization domains and is currently a major research topic offering an alternative to the more established evolutionary computation techniques. There are many enhancements made to the original PSO, some resemble the original PSO and some deviate. The major and noticeable enhancements that have been made to the PSO in the last decade is arranged in chronological order as follows.

The original PSO [11, 12] was found to suffer from instability and premature convergence since particles move out of the solution space during search. In 1996 this was addressed by Eberhart et al. by restricting the velocity to a proportion of the initial search range [15]. The major problem prevailing with PSO is its premature convergence on early best solutions. Many strategies were developed to handle this issue. The first strategy to attempt this problem was introduced by Shi and Eberhart [16] in 1998. The algorithm [16] introduced the inertia weight with velocity factor to avoid premature convergence. The inertia weight influences the searching behavior of particles and hence it became another tuning parameter

in PSO. The larger inertia weight encourages global exploration while a smaller helps for local exploration by fine-tuning the current search space. Suitable selection of the inertia weight provides a balance between global and local exploration and thus requires less iteration to find the optimum solution. Initially the inertia weight was kept static during the entire search process for every particle in each dimension. However, with the due course of time dynamic inertia weights were introduced. Later in 2000 Shi and Eberhart [17] carried out research on the effect of inertia weight and they proposed linear decrease in inertia weight from 0.9 to 0.4 with iteration. This linear decrease of inertia weight allows initial exploration followed by acceleration towards global optima. In 2002 Clerc and Kennedy [18] introduced the constriction factor which restricts the velocity for improving convergence speed. The value of constriction factor was decided based on cognitive and social factors. The cognitive and social factors were given equal importance and they used the value to be 2.05 in their experiments. In 2002 Maurice and Kennedy [19] presented the detailed dynamics of the particles and proposed a generalized PSO model with a set of coefficients to control the convergence tendency. In 2003 Peram, Veeramachaneni and Mohan [20] developed Fitness-Distance Ratio Based Particle Swarm Optimization (FDRPSO) to avoid premature convergence. In FDRPSO [20] the particles move towards nearby particles of higher fitness, instead of attracting each particle towards global best position. They accomplished this objective by using the ratio of the relative fitness and the distance of other particles to determine the direction for changing the particle position. The premature convergence of PSO was further addressed in 2004 by Ratnaweera and Halgamuge [21]. In [21], a time-varying acceleration coefficients (TVAC) was introduced in addition to the time-varying inertia weight factor with PSO. In 2004 Bergh, Andries and Engelbrecht [22] developed Cooperative Approach to Particle Swarm Optimization (CPSO). In this approach [22], multiple swarms optimize different components of the solution vectors cooperatively and show remarkable enhancements to PSO performance in terms of quality of solution. In 2004 Mendes, Kennedy and Neves [23] proposed Fully Informed Particle Swarm (FIPSO). In FIPSO [23] each particle is influenced by the information of all particles rather than just with one best performer as in basic PSO. In 2005 Stefan Janson, and Martin Middendorf [24] developed a Hierarchical Particle Swarm Optimizer (HPSO). Where the particles are arranged in a dynamic hierarchical structure and the particles with best solutions move up or down in the

hierarchy. This hierarchical structure facilitates the larger influence of good particles on the swarm. The effect of weighting coefficients of particles on optimization process was investigated in Particle Swarm with Exponential Distribution (PSOE) by Renato, Krohling and Leandro in 2006. In PSOE [25], weighting coefficients were in exponential probability distribution instead of uniform distribution. Another modification to overcome the premature convergence was introduced in 2006 by Srinivas Pasupuleti and Roberto Batti is called Gregarious PSO (GPSO) [26]. In GPSO [26] algorithm, the re-initialization mechanism was used if the particles stuck close to the local minima. In every iteration either particles were accelerated towards the goal or be re-initialized. If the distance between the current particle and the global best is comparable then the particle velocity was re-initialized and clamped between (V_{min}, V_{max}) else the particle will move in the direction of global best. A novel PSO variant called as modified particle swarm optimizer with dynamic adaptation [27] was introduced by Yang et. al. in 2007. This variant adjusts the inertia weight dynamically, based on evolution speed factor and aggregation degree factor. Sabat et. al developed Combined Effect Comprehensive Learning Particle Swarm Optimizer (CECLPSO) [28] in 2007. The CECLPSO algorithm [28] makes use of two best elite particles (*gbest*) to alter the particles dynamics, instead of one elite particles (*gbest*) as in basic PSO [11]. The effect of second best particle has shown improvements in the results as compared to the only one best particle. PSO has proved its presence in discrete optimization problems also. In 2008 A hybrid particle swarm branch-and-bound (HPB) [29] optimizer for mixed discrete nonlinear programming was developed. The work has been done on tuning the social and cognitive factors of original PSO. During 2008 global best and local best model of PSO was combined and called GLBest-PSO [30]. GLBest-PSO [30] algorithm uses time varying inertia weight and acceleration coefficient to address premature convergence. The problem of premature convergence was further addressed with a novel idea called Accelerated Effect by Sabat et al. and proposed an Accelerated Exploration Particle Swarm Optimizer (AEPSO) algorithm [31]. The AEPSO [31] algorithm identifies the particle trapped in local optima, that are mainly responsible for premature convergence and accelerate them in the direction of best solution. This strategy helps the stagnated particles to come out the local optima and search the solution again. Further Sabat et al. integrated AEPSO algorithm with hyperspherical coordinate manipulation concept and proposed Hyperspherical Acceleration Effect Particle Swarm Optimizer (HAEPSO)

[32]. In HAEPSO algorithm [32], the particles position and velocity are updated in hyperspherical coordinate system instead of cartesian coordinate system and this was combined with AEPSO algorithm [31]. The advantage of hyperspherical coordinate system is, this coordinate system gives better control of particles's direction (as information is available in angle) and fine tunes the solution. This strategy [32] was very helpful when particles got stuck in local minima. In 2009 PSO with extrapolation (ePSO) [33] was introduced. In this algorithm, the current particle position was updated by extrapolating the global best particle position and the current particle positions in the search space. Apart from cognitive, social factor and inertia weight, research has been carried out on the size of the population also and thus an efficient population unitization technique called Efficient population utilization strategy for particle swarm optimizer was presented in [34]. Sabat et al. proposed Adaptive Accelerated Exploration Particle Swarm Optimizer [35]. Essentially AAEPSO [35] algorithm was an enhancement to AEPSO [31] with an adaptive acceleration and exploration. In AEPSO [31], the search patch of unit radius was maintained and hence restricted the scope of searching the solution around the global solution (*gbest*). Thus in AAEPSO, patch size was adjusted dynamically with iteration, and this in turn balances the exploration and exploitation. Further Sabat et. al. addressed the problem of premature convergence and quality of solution with higher dimensional problems and reported Integrated Learning Particle Swarm Optimizer (ILPSO) [36, 37]. The ILPSO [36, 37] algorithm essentially uses the concept of Hyperspherical Acceleration Effect Particle Swarm Optimizer (HAEPSO) [32] together with comprehensive learning approach [38].

2.3 Challenges in PSO for solving SOUO problems

Particle Swarm Optimization (PSO) follows very simple social model. Due to its simplicity and ease of implementation PSO has attracted many researchers across the globe since its inception in 1995. Though PSO has seen many advancements in its development, it still suffers with following major problems. The below mentioned inevitable problems persisting with PSO has given a scope for further research.

- Premature convergence: PSO ends up searching early best solutions. This is predominant in multimodal functions.
- Convergence speed: PSO explores the good solution in its early stage of search but get stagnated in the process of exploiting the global solution.
- Quality of solution. PSO produces low quality solutions due to inherent complexity, discontinuity and multimodality of the problems.
- Uncertainty in solutions. Due to stochastic nature, PSO produces different solutions in different runs.
- Instability: Instability caused by particles accelerating out of the solution space.
- Update strategy: PSO has simple solution update strategy and hence is not adequate to get the better solution in complex environment.

2.4 Research Contribution

As stated in the above section, lots of research work has been done since the inception of PSO. Due the problems prevailing with PSO; it has given further research direction for improving PSO. The following are the enhancements (algorithmic developments) that are carried out to address the above stated problems. The major enhancements are 1) Accelerated Effect Particle Swarm Optimizer (AEPSO) [31], 2) Hyperspherical Acceleration Effect Particle Swarm Optimization (HAEPSO) [32], 3) Integrated Learning Particle Swarm Optimizer (ILPSO) [36, 37] and 4) Adaptive Accelerated Exploration Particle Swarm Optimizer (AAEPSO) [35]. The detailed presentation of these algorithms is given in the following subsections.

2.4.1 Accelerated Effect Particle Swarm Optimizer

The major prevailing problem of basic PSO is premature convergence and poor quality of solutions. During due course of experiments it was found that some of the particles did not show any improvement in the solution with iterations. This is more pronounced in case of multimodal problems. Due to multiple local optima the particles may get stuck and stay in local optima for longer and may not come

out of it. This results to premature convergence of the swarm with low quality of solutions. The major motivational factor here is, to find out such problem causing particles and force them to come out of local optima.

The problem of premature convergence and poor quality of solution is addressed with a novel idea called Accelerated Effect. The Accelerated Effect is combined with Particle Swarm Optimization and is called as Accelerated Effect Particle Swarm Optimizer (AEPSO). The major part of AEPSO is to deal with the identification of the particles which are trapped in deep local optima, responsible for premature convergence (worst or diverged particles) and accelerate them in the direction of best solution. The detailed algorithm is explained in the following sections: 1) Identification of diverged particles and 2) Accelerating diverged particles in the direction of best solution.

2.4.1.1 Identification of diverged particles

During the search process, some of the particles get trapped in the local optima and may not be available for searching for good solutions there after. This is especially quite dominant in the case of higher dimensional objective functions having many local optima. These trapped particles are responsible for premature convergence and also for poor quality of solutions. AEPSO deals with the identification of the particles which are trapped in deep local optima, responsible for premature convergence. The diverged particles are selected based on the fitness of the particle and the Euclidean distance of the particles from the *gbest* particle. The farther the particle from the goal, the poorer result it gives and said to be diverged. These diverged particles will not participate in searching for a better solution there after. Similarly the particle which gives worst fitness are also treated to be diverged. Since the diverged particles are the major culprits and contribute to the premature convergence and hence special care need be taken for these particles. Only 20% of the population was selected as diverged particles. Experiments were conducted and found that the 20% of the population was a good choice.

2.4.1.2 Accelerating diverged particles

The diverged particles (identified in above subsection) will not participate in the search process. If these particles are not taken care of then they cause the solution to deteriorate and hence converge prematurely. There are two possibilities that can

be done with the selected diverged particles; they can be either be initialized to the *gbest* particle or to some other new position in the search range. Both the methods have their own limitations. If particles are initialized to *gbest* (i.e. making exact replica), the whole swarm may collapse and get completely stagnated. This is due to the fact that *gbest* of the current iteration is global solution or not is not known priori. There are chances that the *gbest* particle might also be stagnated and resulting into collapse of entire swarm. If the particles are regenerated randomly to new position in the search range, it takes lot of computational resources to converge the swarm to global solution. Thus in AEPSO, neither of the above is considered, but the diverged particles are accelerated near the particle with global solution. It is assumed that already the *gbest* particle has found the better solution and hence there could be some better solution nearby. The fixed sized search patch of unit radius was maintained around the *gbest* particle. The diverged particles are randomly placed in that patch. This helps the particles to come out of the deep local optima and take part in the search process.

The velocity and position update equation for AEPSO are

$$\begin{aligned} V_{i,d}^{t+1} &= w^t V_{i,d}^t + c_1^t * rand_1 * (pbest_{i,d}^t - X_{i,d}^t) \\ &+ c_2^t * rand_2 * (gbest_d^t - X_{i,d}^t) \qquad (2.2) \\ X_{i,d}^{t+1} &= X_{i,d}^t + V_{i,d}^{t+1} \qquad (2.3) \end{aligned}$$

Where w^t, c_1^t and c_2^t are linearly decreasing inertia & learning factors.

$$\begin{aligned} w^t &= W_{max} - \frac{W_{max} - W_{min}}{Maxiter} \times t \qquad (2.4) \\ c_1^t &= W_{max} - \frac{W_{max} - W_{min}}{Maxiter} \times t \qquad (2.5) \\ c_2^t &= W_{max} - \frac{W_{max} - W_{min}}{Maxiter} \times t \qquad (2.6) \end{aligned}$$

with $W_{max} = 0.9$ and $W_{min} = 0.2$. Figure 2.1 illustrates acceleration of the diverged particles towards global solution. In Figure 2.1 the particles are represented by small bold circles. The process of acceleration effect is explained by considering the population size of ten and number of diverged particles to be two (shown in red color). The normal particles are shown with blue circle and global best particles with green. The process is explained in four steps, initially in the first step diverged particles are far away from global solution and hence they will

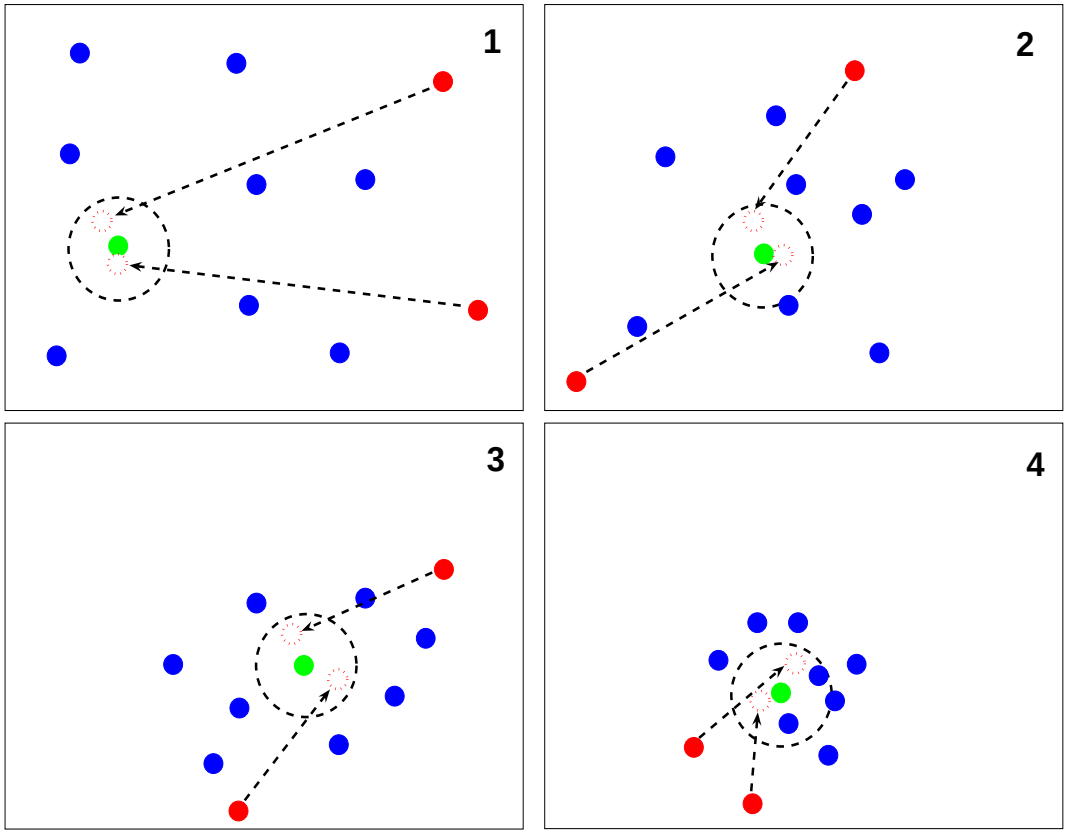

Figure 2.1: Diagram illustrating Exploration region around desired solution

be accelerated towards the best solution inside the black dotted bigger circle. The black dotted bigger circle is of unit radius and is the probable search region for diverged particles. The dotted small red circle shows the probable region that diverged particle may occupy. With increased iteration all the diverged particles will tend towards better solution as is seen from subsequent steps in Figure 2.1.

2.4.1.3 Implementation details of AEPSO Algorithm

Algorithm 2.4.1 show the detailed steps of AEPSO algorithm. In the initialization phase, the search range (X_{max}, X_{min}) is initialized and is problem dependant. The dimension (D) and population size (NP) are also initialized. The velocity is clamped to 20 percent of the search range i.e. $V_{max} \leftarrow 0.20 * (X_{max} - X_{min})$. The position X_i^0 and velocity V_i^0 of the population is randomly initialized in the search range. The initial personal best $pbest_i^0$ and global best $gbest^0$ is set after evaluating fitness f_i^0 of the all the particles. In the optimization phase t

and i are set as counters for iterations and population respectively. The velocity and position of the whole population are evaluated as per equations (2.3) and (2.2) respectively. The personal best $pbest_i^{t+1}$ and global best $gbest_i^{t+1}$ of all the particles are calculated after evaluating fitness f_i^{t+1} with new positions. This process continues till the stopping criteria is met or iteration counter reaches the

Algorithm 2.4.1 Pseudo code for AEPSO

Initialize
$Set \leftarrow X_{max}, X_{min}, D, NP$
$V_{max} \leftarrow 0.20 * (X_{max} - X_{min})$
$t \leftarrow 0, i \leftarrow 0$ ▷ t for iterations and i for particles
Randomly initialize particle's position $X_i^0 \in D$ ▷ D is dimension of the problem
Randomly initialize particle's velocity $V_i^0 \leq V_{max}$
Evaluate fitness function values f_i^0
$pbest_i^0 \leftarrow f_i^0, \;\; gbest^0 \leftarrow f_{best}^0$
Optimize
while $t \leq MaximumGeneration$ **do**
    while $i \leq NP$ **do**
Update velocity and position as per equation (2.2) and (2.3) respectively.
Evaluate fitness function values f_i^{t+1}
Find $pbest_i^{t+1}$
        if $f_i^{t+1} < pbest_i^t$ **then** ▷ for minimization problem
            $pbest_i^{t+1} \leftarrow f_i^{t+1}$
        end if
Find $gbest_d^{t+1}$
        if $gbest_d^{t+1} < gbest_d^t$ **then** ▷ for minimization problem
            $gbest_d^{t+1} \leftarrow gbest_d^t$
        end if
        $i \leftarrow i + 1$, *go to step* 9.
    end while
    if stop criteria not met **then**
        $t \leftarrow t + 1$, *go to step* 8.
    end if
    Identification and acceleration of diverged particles as per Algorithm 2.4.2
end while
Report results
Terminate

maximum limit. The identification and acceleration of diverged particles are done as per Algorithm 2.4.2. In Algorithm 2.4.2 the selection factor S_f for selecting diverged particles is set to 0.2 i.e 20% of the total population, m is the number of trapped particles to be selected ($m = S_f * NP$). The Acceleration and Exploration factor AE_f is initialized to 1.0, this maintains the search patch of unit radius around *gbest* particle. The diverged particles are categorized based on fitness and Euclidean distance from *gbest*. All the particles are arranged in ascending order based on the fitness measure and stored in $sorted_{fitness}$. The

identification of diverged particles are carried out by comparing the fitness of the said particle from the $sorted_{fitness}$, this essentially gives the index of the intended particle.

Algorithm 2.4.2 Pseudo code for identification & acceleration of diverged particles

```
Select → S_f                                   ▷ Selection factor for diverged particles
m ← S_f * NP                                   ▷ No. of diverged particles to be selected
Select → AE_f                                  ▷ Acceleration and Exploration factor
flag ← 0                                       ▷ to flip between two acceleration strategies
sorted_distance ← sort(distance)               ▷ Sort the distances
sorted_fitness ← sort(f^t)                     ▷ Sort the fitness
if mod(t, 10) = 0 then                         ▷ 10 Refresh interval, t Current Iteration
    a = gbest_d^t − AE_f, b = gbest_d^t + AE_f
    for k ← 1, m do
        for l ← 1, NP do
            if flag == 0 then
      Δ=sorted_fitness(NP − m) − f^t(l)
                if Δ == 0 then                 ▷ index of diverged particle based on fitness
                    X(l) =AE_f * rand * gbest
                end if
                flag ← 1
            else
  φ=sorted_distance(NP − m) − distance(l)
                if φ == 0 then                 ▷ index of diverged particle based on distance
                    X(l) =a + (b − a) * rand
                end if
                flag ← 0
            end if
        end for  l
    end for  k
end if
```

The current fitness of the particle is denoted as f and $sorted_{fitness}$ is the current fitness in ascending order, *sort* arranges the the fitness values of f in ascending order and stores in $sorted_{fitness}$. The $\triangle = 0$ condition finds out the exact particle which has poor fitness. This is achieved by subtracting f from $sorted_{fitness}$ and the particles index is captured for zero difference. The captured particles are termed as diverged particles. Similarly all the particles are arranged in ascending order based on distance of particles from *gbest* and stored in $sorted_{distance}$. The diverged particles based on Euclidean distance are similarly identified. For acceleration, the diverged particles are selected alternatively from $sorted_{fitness}$ and $sorted_{distance}$ to achieve this, a flag is maintained and updated as in Algorithm 2.4.2. To maintain the diversity in the population and not to monopolize the search process with only *gbest*, a refresh rate is maintained. The process of identification and acceleration

is carried out at refresh rate of 10 (at the end of every 10^{th} iteration; ex. in 1000 iterations and 10 refresh rate, acceleration will be only for 100 times). The experiments were conducted on different values of refresh rate, and we observed that refresh rate of 10 was optimum for getting good solution.

2.4.2 Hyperspherical Acceleration Effect Particle Swarm Optimization

Although previous work [22, 38, 19, 23, 20, 25, 24] on PSO has provided better solutions over different benchmark functions, the PSO world still aims to optimize the objective functions that have deep multiple local optima. Almost all the previous work on PSO used cartesian coordinates system for updating particle's position and velocity. The major difficulty with cartesian coordinate system is fine tuning of the solution. In the beginning of the search process PSO finds the solution by exploration where coarse tuning is sufficient, but as the search process proceeds and PSO enters into exploitation of solution, fine tuning is essential. In this stage premature convergence is seen due to multiple optima. For fine tuning the solution, a hyperspherical coordinate system is better than cartesian coordinates system, as in hyperspherical system a small change in particle coordinate results in large change in angles i.e. direction, and, that helps the particles to come out from deep local optima.

Thus to address the premature convergence, poor quality of solution and uncertainty in solution HAEPSO is proposed. Essentially HAEPSO [32] is an improvement over AEPSO [31] with major improvement of integrating hyperspherical coordinate systems update strategy. The selection of diverged particles and accelerating these towards global solution is similar to AEPSO algorithm explained in above section. The HAEPSO carries out hyperspherical manipulations with AEPSO as explained below.

2.4.2.1 Hyperspherical manipulations

The hyperspherical coordinate system is a generalization of spherical coordinates for higher dimensions. In the hyperspherical coordinate system of n-dimensional Euclidean space, the coordinates consist of a radial coordinate r, and $n-1$ angular coordinates $\phi_1, \phi_2, \cdots \phi_{n-1}$. The relation between n-dimensional Euclidean space

and hyperspherical space can be defined as

$$\begin{aligned}
x_1 &= rcos(\phi_1) \\
x_2 &= rsin(\phi_1)cos(\phi_2) \\
x_3 &= rsin(\phi_1)sin(\phi_2)cos(\phi_3) \\
&\vdots \\
x_{n-1} &= rsin(\phi_1)sin(\phi_2)\cdots \\
&\cdots\ sin(\phi_{n-2})cos(\phi_{n-1}) \\
x_n &= rsin(\phi_1)sin(\phi_2)\cdots \\
&\cdots\ sin(\phi_{n-2})sin(\phi_{n-1})
\end{aligned}$$

The premature convergence in PSO is mainly caused due to particles trapped in local optima. This problem can be resolved if the particles are sufficiently intelligent to make the variation in their coordinates minutely. Small changes can be made efficiently with the angle information. Since hyperspherical coordinates give information about the angle, HAEPSO uses this coordinate system. HAEPSO updates all the information of the particles by means of hyperspherical coordinates. Due to this the magnitude and the directions of the particles can be changed in a controlled manner and hence particles come out of the deep local minima to search for the better solution. Each particle will update its velocity and position according to the following equations (2.7) and (2.8) after converting into hyperspherical coordinates.

$$\begin{aligned}
V_{i,d}^{t+1} &= w^t * V_{i,d}^t + c_1^t * rand_1 * (pbest_{i,d}^t - X_{i,d}^t) \\
&+ c_2^t * rand_2 * (gbest_d^t - X_{i,d}^t) \qquad (2.7) \\
X_{i,d}^{t+1} &= X_{i,d}^t + V_{i,d}^{t+1} \qquad (2.8)
\end{aligned}$$

Where $V_{i,d}^t$, $X_{i,d}^t$, $gbest_d^t$ and $pbest_{i,d}^t$ are as stated above . The inertia weight (w^t) and acceleration coefficients (c_1^t and c_2^t) are decreased with iteration from 0.9 to 0.2 according to the equations (2.4), (2.5) and (2.6) respectively. The linear variation of inertia weight and acceleration coefficients used in this algorithm will further enhance the local search capability.

2.4.2.2 Implementation details of HAEPSO Algorithm

The practical implementation of HAEPSO involves the steps as shown in Algorithm 2.4.3. The implementation steps of HAEPSO are almost similar to AEPSO (Algorithm 2.4.1) with major difference of hyperspherical manipulation. The particles position and velocity is updated in hyperspherical coordinate system as per Algorithm 2.4.4. The Algorithm 2.4.4 shows the pseudo code for hyperspherical manipulation. In this algorithm x is the D-dimensional cartesian coordinate and (r, ϕ) is hyperspherical counter part of x. The origin of hyperspherical coordinate system $((r, \phi))$ is same as cartesian coordinate system x. The $rand$ is the uniformly distributed random number in the range of [0,1]. This algorithm takes the input in cartesian system, converts it into hyperspherical and implements the necessary manipulation i.e. updates and then converts it back into cartesian system.

Algorithm 2.4.3 Pseudo code for HAEPSO

Initialize
1: $Initialize \longrightarrow W_{max}, W_{min}, X_{max}, X_{min}, D, NP$
2: $V_{max} \leftarrow 0.20 * (X_{max} - X_{min})$
3: $Initialize \longrightarrow m$; No. of diverged particles to be selected.
4: $t \leftarrow 0, i \leftarrow 0$
5: Decrease w, c_1, c_2 as per (2.4), (2.5) and (2.6) with iterations respectively
6: Randomly initialize position $X_i^0 \in D$; $\forall$ i $\in$ NP.
7: Randomly initialize velocity $V_i^0 \leq V_{max}$; $\forall$ i $\in$ NP.
8: Evaluate fitness f_i^0 using particle positions.
9: $pbest_i^0 \leftarrow f_i^0$
10: $gbest^0 \leftarrow f_{best}^0$
Optimize
11: Update velocity and position as per (2.7) and (2.8).
12: Evaluate fitness f_i^{t+1} using particle positions.
13: Find $pbest_i^{t+1}$
14: **if** $f_i^{t+1} < pbest_i^t$ **then** ▷ for minimization problem
15: $pbest_i^{t+1} \leftarrow f_i^{t+1}$
16: **else**
17: $pbest_i^{t+1} \leftarrow pbest_i^t$
18: **end if**
19: Find $gbest_d^{t+1}$
20: **if** $gbest_d^{t+1} < gbest_d^t$ **then** ▷ for minimization problem
21: $gbest_d^{t+1} \leftarrow gbest_d^t$
22: **end if**
23: Identification and acceleration of diverged particles as per Algorithm 2.4.2
24: If stop criteria not met, increment t and go to step 11.
Report results
Terminate

Algorithm 2.4.4 Pseudo code for Hyperspherical manipulation

1: $r \leftarrow \sqrt{(\sum(x^2))}$
2: $\phi \leftarrow (180/pi) * acos(x/r)$
3: $r \leftarrow rand * r$
4: $\phi \leftarrow \phi + rand$
5: $x \leftarrow r * cos((\phi * pi)/180)$

2.4.3 Integrated Learning Particle Swarm Optimizer

The unsolved problems of PSO (premature convergence, poor quality of solution and instability in solution) are further addressed by Integrated Learning Particle Swarm Optimizer (ILPSO) [37, 36]. This algorithm is essentially a hybrid algorithm that has the components of AEPSO and HAEPSO integrated with well known comprehensive learning strategy [38]. Identification and acceleration of diverged particles in ILPSO are similar to AEPSO [31] discussed in the above section. The hyperspherical update is also similar to HAEPSO [32]. The concepts from these two algorithms are hybridized with well known comprehensive learning strategy [38]. The Comprehensive Learning strategy is discussed as below.

2.4.3.1 Comprehensive Learning

In Comprehensive Learning (CL) [38], the particles learn from different exemplar with different dimension and time. Comprehensive Learning (CL) has major differences compared to basic PSO [11, 12]. In basic PSO particles learn from *pbest* and *gbest* in every generation and dimension, where as in CL [38] a particle learns from either its own *pbest* or *gbest* or *pbest* of some other particles for a few iterations. And each dimension of a particle learns from a different exemplars (its own *pbest* or *gbest* or *pbest* of some other particles) for different dimensions for a few iterations. The comprehensive learning concept is integrated with stated proposal with the refreshing rate of 10 (explained in above section 2.4.1.3). The particle's velocity and position are updated by selecting either of the following velocity and

position update equation.

$$V_{i,d}^{t+1} = w^t * V_{i,d}^t + rand_1 * (gbest1_d^t - X_{i,d}^t) + rand_2 * (gbest2_d^t - X_{i,d}^t) \quad (2.9)$$

$$V_{i,d}^{t+1} = w^t * V_{i,d}^t + rand_1 * (pbest_{f,d}^t - X_{i,d}^t) + rand_2 * (gbest2_d^t - X_{i,d}^t) \quad (2.10)$$

$$V_{i,d}^{t+1} = w^t * V_{i,d}^t + rand_1 * (pbest_{i,d}^t - X_{i,d}^t) + rand_2 * (gbest2_d^t - X_{i,d}^t) \quad (2.11)$$

$$X_{i,d}^{t+1} = X_{i,d}^t + V_{i,d}^{t+1} \quad (2.12)$$

Algorithm 2.4.5 Pseudo code for ILPSO

Initialize
$Set \leftarrow X_{max}, X_{min}, NP, D$
$Select \leftarrow S_f \quad m \leftarrow S_f * NP$
$Select \leftarrow AE_f$
$t \leftarrow 0, i \leftarrow 0$ ▷ t for iterations and i for particles
Initialize position $X_i^0 \in D$ in R and velocity $V_i^0 \leq V_{max}$
Evaluate fitness function values f_i^0
$pbest_i^0 \leftarrow f_i^0, \quad gbest^0 \leftarrow f_{best}^0$
$flag \leftarrow 0$
Optimize
while $t \leq MaximumGeneration$ **do**
  while $i \leq NP$ **do**
Update velocity by select either of the velocity equation (2.9), (2.10), (2.11) as per Algorithm 2.4.4; ▷ Comprehensive Learning
  Update position (2.12) as per Algorithm 2.4.4
    Evaluate function fitness values f_i^{t+1}
    Find $pbest_i^{t+1}$
    if $f_i^{t+1} < pbest_i^t$ **then** ▷ for minimization problem
      $pbest_i^{t+1} \leftarrow f_i^{t+1}$
    else
      $pbest_i^{t+1} \leftarrow pbest_i^t$
    end if
    $i \leftarrow i + 1$, *go to step* 10.
    Find $gbest_d^{t+1}$ ▷ Minimum of $pbest_i^{t+1}$
    if $gbest_d^{t+1} < gbest_d^t$ **then** ▷ for minimization problem
      $gbest_d^{t+1} \leftarrow gbest_d^t$
    end if
  end while
  Identification and acceleration of diverged particles as per Algorithm 2.4.2
  if stop criteria not met **then**
    $t \leftarrow t + 1$, *go to step* 9.
  end if
end while
Report results
Terminate

2.4.3.2 Implementation details of ILPSO Algorithm

Algorithm 2.4.5 shows all the necessary steps for practical implementation of ILPSO. Identification and acceleration of diverged particles is almost similar to Algorithm 2.4.2 and the hyperspherical updates is similar to Algorithm 2.4.4. The velocity and position update equation for ILPSO are as in equations (2.9), (2.10), (2.11) and (2.12).

2.4.4 Adaptive Accelerated Exploration Particle Swarm Optimizer

The prevailing premature convergence problem of PSO is further addressed by Adaptive Accelerated Exploration Particle Swarm Optimizer (AAEPSO) [35]. Essentially AAEPSO [35] is an enhancement to AEPSO with an adaptive acceleration and exploration. The problem with AEPSO was, it maintained a constant search patch of unit radius. This restricts the scope of searching the solution around the global solution (*gbest*). Thus in AAEPSO [35] the patch size is adjusted dynamically with iteration, and this in turn balances the exploration and exploitation. The selection and acceleration process of particles in AAEPSO is almost similar to AEPSO explained in above subsection with following additions.

2.4.4.1 Adaptive Accelerated Exploration

To preserve the diversity among the swarm, discourage premature convergence and to avoid the swarm from collapsing, the following strategies are implemented. The diverged particles in the population are allowed to search the surrounding space of global solution, but not re-initialized to global solution. The acceleration and exploration power are imparted to the diverged particles is based on distance and fitness of the particles alternately for every 10 iterations (a refresh rate similar to AEPSO [31]). The farther the particle from the goal, the poorer solution it gives and said to be diverged. These diverged particles will not participate for searching the better solution there after. Similarly the particle which gives worst fitness are also treated to be diverged here. These particles are then accelerated in the direction of best particle in the swarm (i.e. the global solution) with more exploration power to search the space around the global solution. Here the number of diverged particles selected for further processing is 20% of the population.

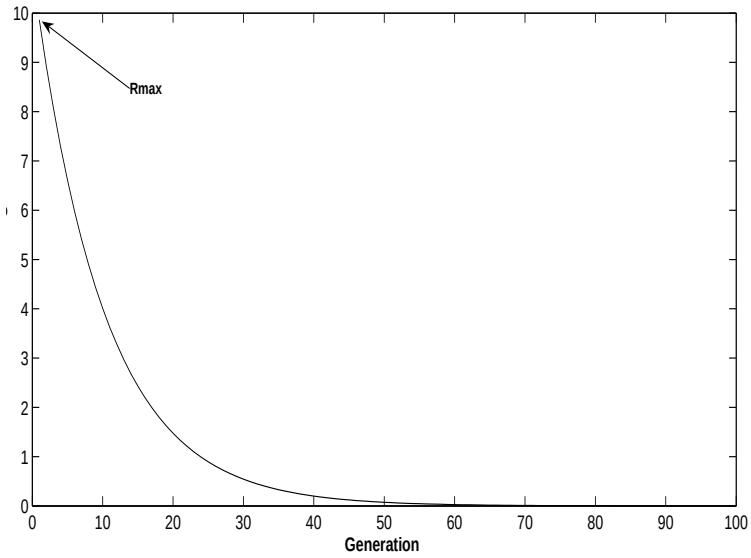

Figure 2.2: Diagram illustrating exponential decrease of exploration region

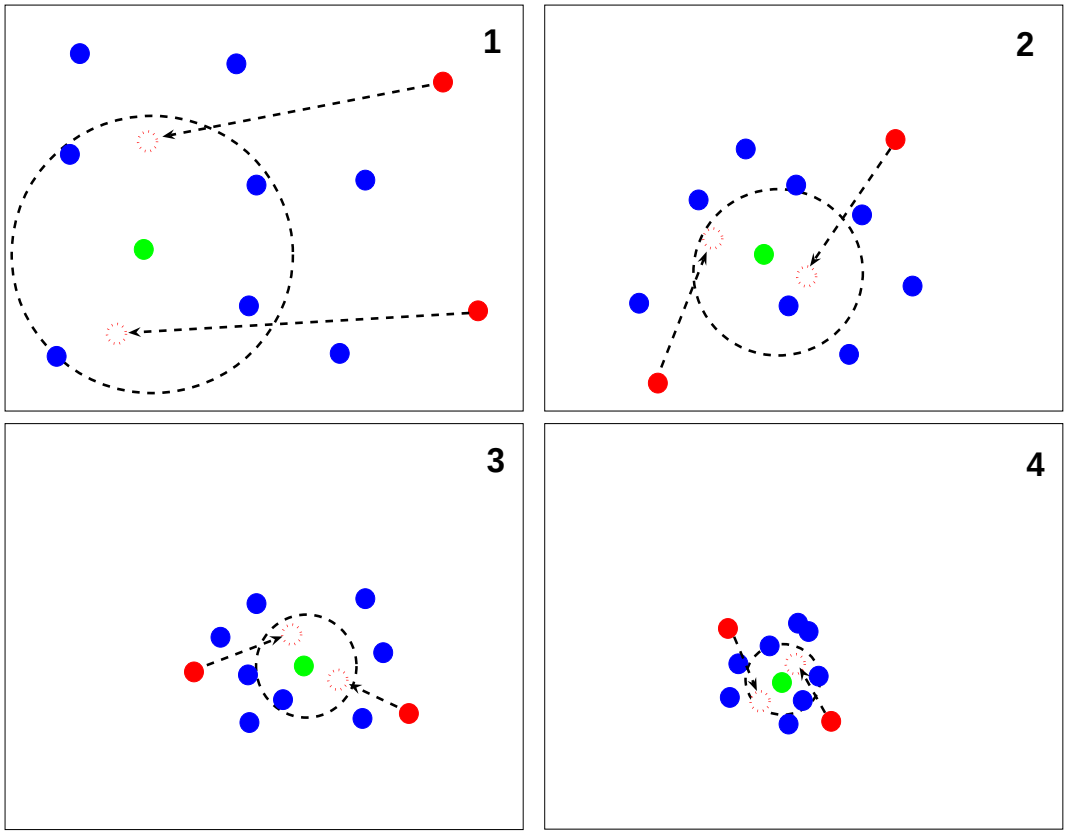

Figure 2.3: Diagram illustrating Exploration region around desired solution

$$AE_f^t = R_{max} * e^{\frac{-t}{u}} \tag{2.13}$$

Experiments were conducted and 20% was found to be the suitable percentage for an optimum result. Again from these diverged particles (20% of the population), we select randomly between 10% to 90% of the particles for acceleration. The exploration power imparted to the diverged particle follows exponential decay. In the beginning of search process the search range for the selected particles is same as problem search range around the global solution. With increase in iteration the search range decrease exponentially according to equation (2.13) and as shown in Figure 2.2. Where u is a arbitrary constant and is set to 10. We have conducted the experiments to estimate the value of u. The chosen value of u decreases exponentially from R_{max} to zero. R_{max} is the maximum search range of the problem. Thus, this strategy gives the complete freedom for the particles to explore the space around the global solution. The complete search process is depicted in Figure 2.3. The Figure 2.3 is almost similar to Figure 2.1, a notable difference is the size of the patch. The patch size decreases with iteration adaptively as per equation (2.13).

2.4.4.2 Implementation details of AAEPSO Algorithm

The practical implementation of AAEPSO involves the steps as shown in Algorithm 2.5.1. The implementation steps of AAEPSO are almost similar to AEPSO (Algorithm 2.4.1) with the major difference being adaptive acceleration and exploration. The selection factor S_f in AAEPSO is the random permutation in the range [10 - 90] % of the diverged population. The acceleration-exploration factor AE_f is decreased as per equation 2.13. The process of diverged particles selection and adaptive acceleration exploration is carried in Algorithm 2.5.2. The detailed implementation steps of diverged particles selection and adaptive acceleration exploration is shown in Algorithm 2.5.1.

2.5 Simulation

The simulations were carried out in Windows XP Operating System, on Pentium IV 2.6GHz with 512MB of RAM. The coding is done in Matlab 7.2 programming language. The population size and the maximum iterations considered for all

the algorithms are 25 and 10000 respectively. The population is initialized in the search range $[X_{min}, X_{max}]$. Where X_{max} and X_{min} are the maximum and minimum value in search range. All the PSO variants used here are initialized asymmetrically. The velocity is initialized with 20 percent of the search range i.e., $V_{max} = 0.20 * (X_{max} - X_{min})$. The stopping criteria is set to 10^{-250}. The results obtained are the average of 25 trials.

Algorithm 2.5.1 Pseudo code for AAEPSO

Initialize
$Set \leftarrow X_{max}, X_{min}, D, NP$
$V_{max} \leftarrow 0.20 * (X_{max} - X_{min})$
$t \leftarrow 0, i \leftarrow 0$ ▷ t for iterations and i for particles
Randomly initialize particle's position $X_i^0 \in D$ in R;
Randomly initialize particle's velocity $V_i^0 \leq V_{max}$
Evaluate fitness function values f_i^0
$pbest_i^0 \leftarrow f_i^0, \quad gbest^0 \leftarrow f_{best}^0$
Optimize
while $t \leq MaximumGeneration$ **do**
  Decrease AE_f Acceleration and Exploration factor exponentially as per equation 2.13.
  while $i \leq NP$ **do**
Update velocity (equation (2.2)) and position (equation (2.3))
Evaluate fitness function values f_i^{t+1}
    Find $pbest_i^{t+1}$
    if $f_i^{t+1} < pbest_i^t$ **then** ▷ for minimization problem
      $pbest_i^{t+1} \leftarrow f_i^{t+1}$
    end if
    $i \leftarrow i + 1$, *go to step* 10.
  end while
  Find $gbest_d^{t+1}$
  if $gbest_d^{t+1} < gbest_d^t$ **then** ▷ for minimization problem
    $gbest_d^{t+1} \leftarrow gbest_d^t$
  end if
  if stop criteria not met **then**
    $t \leftarrow t + 1$, *go to step* 8.
  end if
  Identification & acceleration of diverged particles as per Algorithm 2.5.2
end while
Report results
Terminate

The performance comparisons of proposed algorithms AEPSO [31], HAEPSO [32], ILPSO [37, 36] and AAEPSO [35] are carried out with the state of the art PSOw [16], EPSO [25], FDRPSO [20], GPSO [26], FIPSO [23] and CLPSO [38]. The algorithmic parameters of these algorithms are given below. We have used $c_1 = c_1 = 2$ for PSOw [16] and linearly decreasing inertia weight from 0.9 to 0.2 with iterations. For EPSO the value of a and b are chosen to be 0.3 and 1.0 as in

[25], and for FDRPSO [20] c_1, c_2, c_3 are set to 1. For GPSO [26] the parameters are $\gamma_{min} = 2$, $\gamma_{max} = 4$, $\delta = 0.5$, $\gamma_{initial} = 3.0$ and $\epsilon = 10^{-8}$ [26]. For FIPSO the parameters $C_1 = C_2 = 2.05$ $\phi = C_1 + C_2$ and the velocity update equation as in [23].

Algorithm 2.5.2 Pseudo code for identification & acceleration of diverged particles for AAEPSO

```
Select → S_f randomly between 10% to 90% of diverged particles   ▷ Selection factor for diverged particles
m ← S_f * NP                                                     ▷ No. of diverged particles to be selected
Select → AE_f at particular iteration (follows equation 2.13)    ▷ Acceleration and Exploration factor
flag ← 0                                                         ▷ to flip between two acceleration strategies
sorted_distance ← sort(distance)                                 ▷ Sort the distances
sorted_fitness ← sort(f^{t+1})                                   ▷ Sort the fitness
if mod(t, 10) = 0 then                                           ▷ 10 Refresh interval, t Current Iteration
    a = gbest_d^{t+1} − AE_f(t), b = gbest_d^{t+1} + AE_f(t)
    for k ← 1, m do
        for l ← 1, NP do
            if flag == 0 then
      Δ=sorted_fitness(NP − m) − f^{t+1}(l)
                if Δ == 0 then
                    X(l) =AE_f(t) * rand * gbest
                end if
                flag ← 1
            else
  φ=sorted_distance(NP − m) − distance(l)
                if φ == 0 then
                    X(l) =a + (b − a) * rand
                end if
                flag ← 0
            end if
        end for   l
    end for   k
end if
```

Since the PSO algorithm experience difficulty to optimize functions with multiple optima, therefore we focus on well-known standard benchmark functions [39] which have many local optima. The chosen functions have different degree of complexity including discontinuity, nondifferentiability, convexity, scalability and multimodality. The algorithms are validated and compared with a set of benchmark functions of dimensions of 10, 30, 50 and 100. The benchmark functions are further rotated and shifted for validation. The detailed definitions and characteristics of the functions are given in Appendix A. The performance comparisons are done based on conventional tests like average results, standard deviations and convergence graphs. Further to validate the test results, statistical significant test using Friedman′s test followed by Dunn′s multiple comparison test were performed.

The detailed definitions of performance metrices are given below.

2.5.1 Conventional metrices

The Conventional metrices include convergence graphs, average results and robustness (standard deviation).

2.5.1.1 Convergence

The convergence graphs are drawn in terms of mean fitness value achieved by each of the algorithms. This test is done to verify, how fast the algorithm converges. In other words this is carried out to determine the speed of convergence of the algorithms in terms of number of iterations and the quality of solution they have achieved.

2.5.1.2 Average Results

Swarm Intelligence and Evolutionary Algorithms are stochastic in nature and the solution produced by these algorithms vary form run to run, hence average results give better measure of the correctness. Average results give the complete statistical information ever achieved by the algorithms over 25 trials, where each algorithm is run for 10000 iterations in each trial. The best mean result achieved by the algorithms are tabulated in table and shown in bold.

2.5.1.3 Robustness

The robustness or stability of the algorithms are represented by the standard deviation. This verifies the deviation of algorithm from run to run. Due to the randomness involved in heuristic algorithms, the performance can not be concluded using the result of a single run. Therefore, different number of trials with different parameter initialization are required to conclude the effectiveness and efficiency of heuristic algorithm. An algorithm can be termed as a robust algorithm if it gives consistent results during different trials i.e. with low standard deviation. A 0.0 value of standard deviation indicate very stable nature of the algorithm, any other value indicates a variation.

2.5.2 Statistical metrices

The statistical metrices are widely used to confirm, whether a new proposed algorithm offers a significant improvement, or not, over the existing algorithm for a given problem [40]. These metrices are categorized into parametric and nonparametric classes. The nonparametric tests can perform pairwise and multiple comparisons. Pairwise statistical procedures perform individual comparisons between two algorithms. In order to carry out a comparison which involves more than two algorithms, multiple comparisons tests are used. For multiple comparisons between algorithms we used Friedman′s test followed by Dunn′s multiple comparison test for post hoc analysis [41].

2.5.2.1 Friedman′s test

The Friedman test is a nonparametric test that compares three or more paired groups. Since it is a multiple comparisons test, it is used to detect significant differences between the behaviors of two or more algorithms. The Friedman test converts original results to ranks. The Friedman test first ranks the values in each matched set (each row) from low to high. Each row is ranked separately. It then sums the ranks in each group (column). If the sums are very different, the P value will be small. The Friedman statistic is calculated from the sums of ranks and the sample sizes [41].

2.5.2.2 Dunn′s post hoc test

Dunn′s post hoc test compares the difference in the sum of ranks between two columns with the expected average difference. For each pair of columns, the P value is listed as > 0.05, < 0.05, < 0.01, or < 0.001. The calculation of the P value takes into account the number of comparisons that are made. If the null hypothesis is true (all data are sampled from populations with identical distributions, so all differences between groups are due to random sampling), then there is a 5% chance that at least one of the post tests will have $P < 0.05$. The 5% chance does not apply to each comparison but rather to the entire family of comparisons. Once the proper statistics have been computed using Friedman test, a p-value is computed. If the null hypothesis is rejected and there is a significant difference found between two algorithms then a post-hoc procedure to characterize these differences is used. We used Dunn′s test for post hoc analysis [41].

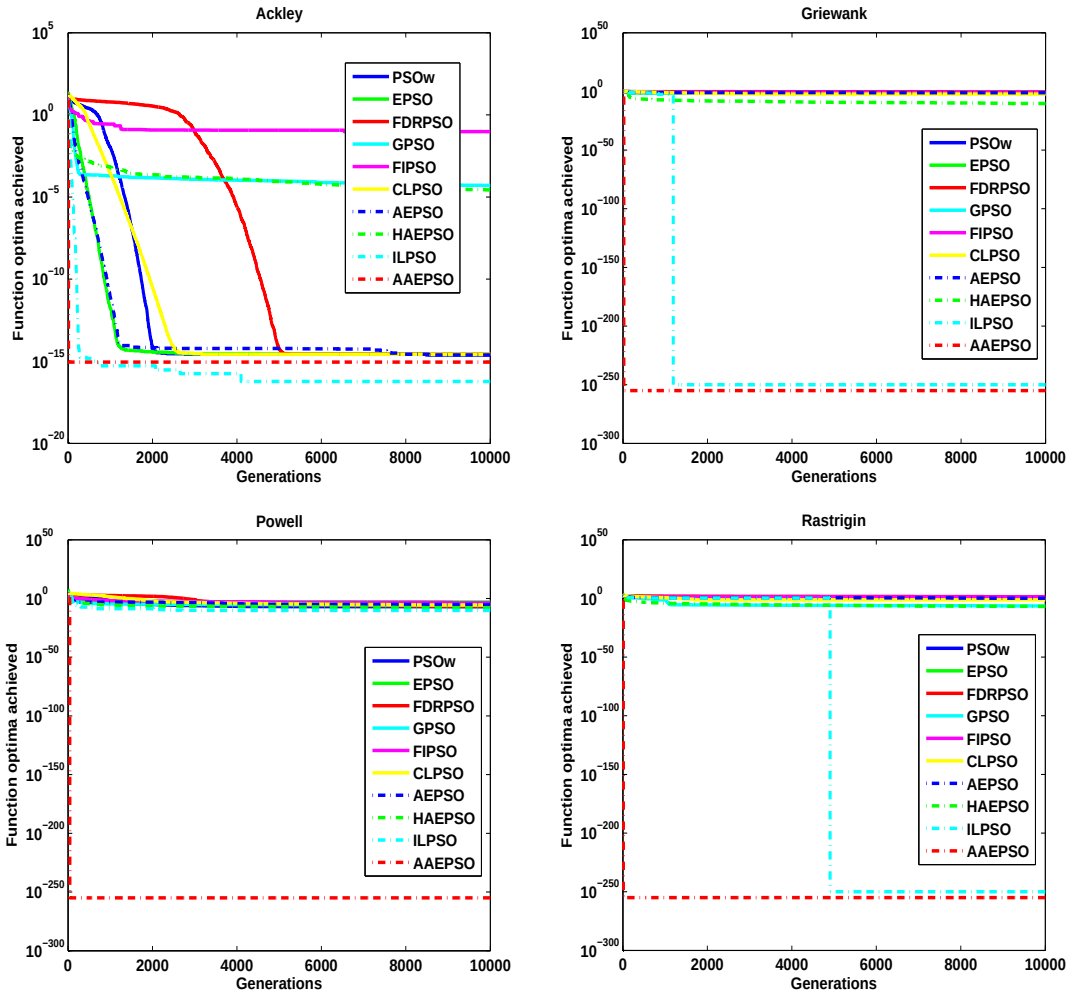

Figure 2.4: Convergence Graphs for Ackley, Griewank, Powell and Rastrigin of 10 dimension

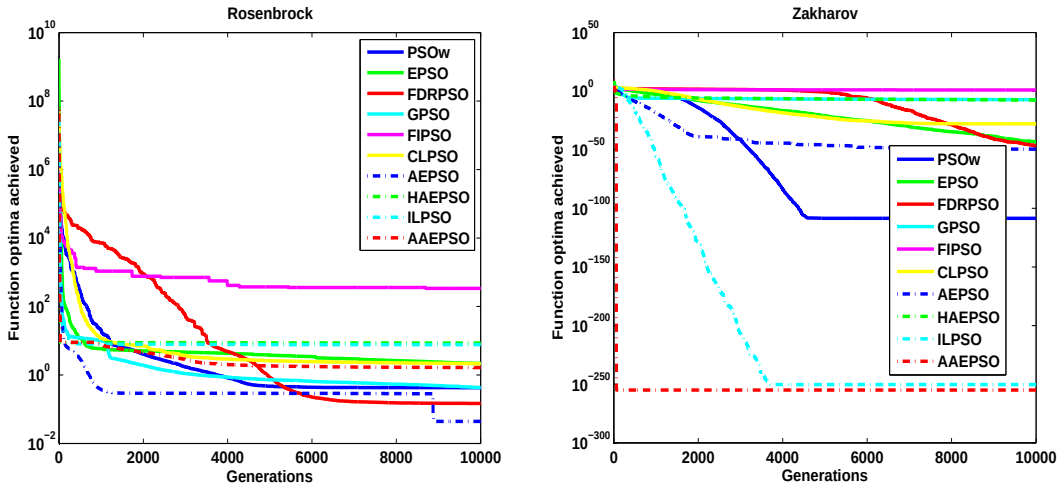

Figure 2.5: Convergence Graphs for Rosenbrock and Zakharov of 10 dimension

2.6 Results and Discussions

The results obtained by each of the algorithms are presented in the following categories.

2.6.1 Convergence Graphs

The Figure 2.4 and Figure 2.5 shows the convergence graphs obtained by the algorithms. The Figure 2.4 show the convergence characteristics on Ackley, Griewank, Powell and Rastrigin. And Figure 2.5 show the convergence on Rosenbrock and Zakharov functions. The convergence graphs presented are for basic functions of dimension 10. Since the convergence characteristics of respective rotated and shifted function does not show remarkable variation (hold the same degree of convergence) they are not presented here. From the convergence graphs it can be concluded that AAEPSO obtains good quality of solution and converges very quickly on functions Griewank, Powell, Rastrigin and Zakharov. ILPSO converges equally well on Ackley function. AEPSO shows good convergence on Rosenbrock function. The remarkable convergence of AAEPSO is observed on almost all the mentioned functions except Rosenbrock. AAEPSO achieves very good quality of solutions with very few iteration. The quick convergence of AAEPSO with good quality of solution shows that it can be an alternative choice where computational efforts are the major criteria.

Table 2.1: Average results obtained with 10 D problems

Func	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEPSO
Ackley	2.43e-15	2.66e-15	2.66e-15	5.06e-5	9.52e-2	2.66e-15	2.43e-15	2.69e-5	**5.92e-17**	8.88e-16
Griewank	6.74e-2	3.95e-2	3.97e-1	1.97e-2	1.36e-1	4.76e-3	7.67e-2	4.86e-11	1.00e-250	**1.00e-255**
Powell	1.19e-7	9.40e-4	1.55e-7	2.50e-6	1.82e-4	3.13e-6	7.15e-6	8.37e-9	6.70e-11	**1.00e-255**
Rastrigin	2.98	1.26	2.55e+1	4.06e-7	1.86e+1	6.63e-2	1.55	1.50e-7	1.00e-250	**1.00e-255**
Rosenbrock	4.25e-1	2.17	1.48e-1	4.26e-1	3.38e+2	2.14	**4.40e-2**	8.55	7.67	1.65
Zakharov	3.10e-109	4.82e-44	1.21e-47	5.18e-8	8.16	1.15e-28	3.08e-50	2.23e-8	5.03e-251	**1.00e-255**
rAckley	7.70e-2	3.21e-1	2.43e-15	5.40e-5	5.55e-1	2.66e-15	2.43e-15	3.19e-5	**1.78e-16**	8.88e-16
rGriewank	1.13e-1	1.12e-1	4.54e-1	9.86e-2	1.70e-1	2.05e-2	1.20e-1	7.00e-11	3.22e-2	**1.00e-255**
rPowell	4.25e-6	2.27e-3	2.43e-5	9.41e-5	2.88e+1	1.50e-3	3.68e-7	4.68e-9	1.20e-4	**1.00e-255**
rRastrigin	9.82	1.53e+1	3.32e+1	1.50e+1	1.75e-2	2.26	8.18	1.07	**00**	**1.00e-255**
rRosenbrock	4.22e+1	2.13e+1	1.10	5.72	1.57e+2	7.25	**2.32e-1**	8.86	8.24	1.32
rZakharov	8.77e-114	2.30e-45	2.20e-50	4.40e-8	2.22	2.30e-29	4.63e-53	3.14e-8	6.61e-251	**1.00e-255**
sAckley	**4.14e-16**	2.38e-1	8.88e-16	3.03e-6	1.96e+1	1.03e-11	7.17	1.94e+1	1.11e+1	8.88e-16
sGriewank	6.76e-2	1.17e-1	3.81e-1	2.87e-2	1.00	1.80e-2	2.25e-1	1.16	5.47e-1	**7.85e-4**
sPowell	**7.90e-8**	3.98e-1	1.26e-7	1.56e-6	1.24e+5	6.17e-4	8.84e-1	5.99e+4	1.28e+2	1.40e-7
sRastrigin	2.52	3.52	2.59e+1	**1.27e-9**	1.09e+3	1.33e-1	4.83e+1	8.84e+2	6.31e+1	1.86
sRosenbrock	6.24e-1	7.83	**1.72e-1**	3.28e-1	1.17e+7	4.26	2.68e+2	4.32e+7	3.09e+2	1.28
sZakharov	1.65e-30	1.54e+1	1.92e-29	1.38e-10	1.79e+3	5.06e-3	4.81e+1	1.91e+3	6.19	**1.00e-255**
srAckley	5.92e-17	2.23	**8.88e-16**	4.88e-1	1.88e+1	5.22e-14	9.43	1.89e+1	1.34e+1	1.33
srGriewank	1.10e-1	1.14e-1	4.19e-1	1.39e-1	6.91e-1	**3.91e-2**	1.53e-1	4.50e-1	4.13e-1	1.23e-1
srPowell	**4.25e-7**	1.10e-2	2.47e-5	1.47e-5	7.96e+4	2.34e-2	3.53e+1	1.06e+5	1.04e+2	1.23e-6
srRastrigin	1.01e+1	3.66e+1	3.24e+1	1.69e+1	1.68e+3	**7.37**	1.00e+2	2.33e+3	5.87e+1	9.22
srRosenbrock	3.92e+1	3.22e+1	**1.57**	2.80e+1	1.49e+6	1.99e+1	9.31	2.78e+6	1.44e+3	1.01e+1
srZakharov	**7.43e-30**	2.90	1.12e-28	8.35e-11	2.22e+3	4.56e-3	2.38e+1	2.98e+3	5.85	2.05e-28

2.6.2 Average Results

1) 10 Dimension: The average results archived by different algorithms on benchmark functions with dimension 10 are presented in Table 2.1. From Table 2.1 it can be observed that AAEPSO gives good quality of solutions on Griewank, Powell, Rastrigin, Zakharov, rotated Griewank, rotated Powell, rotated Rastrigin and rotated Zakharov. On shifted functions it shows good results on shifted Griewank, shifted Zakharov and shifted − rotated Griewank. ILPSO algorithm performs better compared to other PSO variants on the functions Ackley, rotated Ackley and rotated Rastrigin. PSOw achieves good results on shifted Ackley, shifted Powell, shifted − rotated Powel and shifted − rotated Zakharov. The FDRPSO also shows good result on shifted Rosenbrock, shifted − rotated Ackley and shifted − rotated Rosenbrock. AEPSO shows good result on only Rosenbrock and rotated Rosenbrock. GPSO achieves good result on shifted Rastrigin and CLPSO on shifted − rotated Rastrigin. Table 2.1 concludes that the AAEPSO algorithm is a good choice for normal objective functions. Similarly PSOw is the good choice for shifted functions. Table 2.2 presents the detailed Friedman test results on 10D problems. The results of Table 2.2 are calculated with Friedman statistic to be 45.88 and 84.67 for non shifted and shifted benchmark problems respectively. There is a statistically significant difference between the results obtained with FIPSO vs ILPSO, very significant difference between EPSO vs ILPSO on non shifted benchmark problems as seen from Table 2.2. The Table 2.1 concludes that ILPSO outperforms FIPSO and EPSO. Similarly significant, very significant and extremely significant statistical difference between the results obtained with GPSO, EPSO and FIPSO vs AAEPSO is found on non shifted benchmark problems. Again from Table 2.1 it can be concluded that AAEPSO perform well compared to GPSO, EPSO and FIPSO. AEPSO and HAEPSO has not shown any statistically significant difference on results compared to other state of the art. From Table 2.1, it can be verified that both AEPSO and HAEPSO algorithms perform equally well on non shifted benchmark problems compared to other state of art. On shifted problems following are the significant differences in the results. Statistically significant difference on the results between AEPSO and PSOw shows that PSOw perform well. ILPSO and PSOw shows significant results and PSOw perform well. There is again a statistically significant difference between AAEPSO vs HAEPSO and ILPSO, here AAEPSO perform well.

Table 2.2: Friedman test on 10D problems (P < 0.0001)

without shift	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEEPSO
PSOw	–	ns	ns	ns	ns	ns	ns	ns	ns	ns
EPSO	–	–	ns	ns	ns	ns	ns	ns	**s**	**vs**
FDRPSO	–	–	–	ns	ns	ns	ns	ns	ns	ns
GPSO	–	–	–	–	ns	ns	ns	ns	ns	**s**
FIPSO	–	–	–	–	–	ns	ns	ns	**es**	**es**
CLPSO	–	–	–	–	–	–	ns	ns	ns	ns
AEPSO	–	–	–	–	–	–	–	ns	ns	ns
HAEPSO	–	–	–	–	–	–	–	–	ns	ns
ILPSO	–	–	–	–	–	–	–	–	–	ns
AAEPSO	–	–	–	–	–	–	–	–	–	–

with shift	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEEPSO
PSOw	–	ns	ns	**es**	ns	ns	**s**	**es**	**vs**	ns
EPSO	–	–	ns	ns	ns	ns	ns	**s**	ns	ns
FDRPSO	–	–	–	ns	**es**	ns	ns	ns	ns	ns
GPSO	–	–	–	–	**es**	ns	ns	ns	ns	ns
FIPSO	–	–	–	–	–	ns	ns	ns	ns	ns
CLPSO	–	–	–	–	–	–	ns	ns	ns	ns
AEPSO	–	–	–	–	–	–	–	ns	ns	ns
HAEPSO	–	–	–	–	–	–	–	–	ns	**es**
ILPSO	–	–	–	–	–	–	–	–	–	**es**
AAEPSO	–	–	–	–	–	–	–	–	–	–

ns=not significant, s=significant, vs=very significant and es=extremely significant

Table 2.3: Average results obtained with 30 D problems

Func	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEPSO
Ackley	4.86e-1	2.11e+1	2.65e-8	1.38e-2	1.11	1.66e-1	3.14e-15	3.00e-5	**1.95e-15**	4.09e-15
Griewank	2.42e-2	1.29e+1	2.36e-1	6.26e-3	1.96e-1	2.14e-3	1.45e-6	1.22e-10	**00**	2.35e-2
Powell	1.27e-4	1.60e+9	3.67e-3	1.34e-2	2.63e+3	3.54e-1	2.56e-7	5.07e-8	**1.30e-16**	1.09e-4
Rastrigin	5.47e+1	3.85e+4	2.34e+2	8.21	1.15e+2	2.59e+1	3.49	7.58	**00**	5.97e-1
Rosenbrock	2.28e+1	2.46e+10	3.75e+1	6.36e+1	4.18e+3	5.47e+1	**2.16e+1**	2.86e+1	2.53e+1	2.74e+1
Zakharov	3.06e-3	1.25e+14	3.45e+2	8.64e-2	2.36e+1	2.23e+1	2.12e-27	1.12e-6	**6.13e-113**	1.76e-15
rAckley	2.11	2.13e+1	6.13e-9	2.60	5.48e-1	1.51e-1	2.90e-15	2.69e-5	**1.01e-15**	1.54e-1
rGriewank	9.52e-3	1.24e+1	5.74e-3	1.32e-2	3.90e-1	7.39e-3	**00**	1.24e-10	**00**	1.93e-2
rPowell	1.44e-1	3.01e+9	6.50e-1	7.48e-2	9.38e+2	3.87e+1	2.89e-6	**5.66e-8**	1.31e-3	1.29e-3
rRastrigin	1.05e+2	4.41e+4	2.74e+2	1.33e+2	4.21e+1	4.97e+1	1.99e+1	2.32e-7	**00**	4.25
rRosenbrock	6.10e+1	2.68e+10	1.10e+2	4.66e+1	1.27e+4	4.32e+1	2.45e+1	2.88e+1	2.84e+1	**1.16e+1**
rZakharov	2.23e-3	7.51e+10	3.15e+2	9.44e-2	3.38e+1	2.41e+1	3.04e-27	1.93e-7	**2.08e-137**	9.62e-9
sAckley	1.58	2.14e+1	**2.93e-8**	9.33e-4	2.06e+1	3.82e-1	1.92e+1	2.05e+1	2.06e+1	1.39e-1
sGriewank	6.73e-3	1.32e+1	1.55e-1	1.28e-2	2.36	8.87e-3	1.21	2.49	8.95e-1	**1.26e-3**
sPowell	**2.34e-4**	1.16e+9	5.20e-3	3.05e-2	4.98e+7	3.33e+3	7.01e+5	2.57e+7	1.36e+4	2.91e-4
sRastrigin	5.49e+1	4.78e+4	2.08e+2	**7.56**	8.13e+3	2.30e+1	1.72e+3	9.18e+3	3.53e+2	6.06e+1
sRosenbrock	3.00e+1	3.69e+10	3.25e+1	2.95e+1	1.93e+8	1.11e+2	1.61e+5	1.92e+8	4.13e+4	**1.02e+1**
sZakharov	**4.46e-3**	7.95e+13	3.45e+2	2.62e+1	1.03e+4	3.21e+2	4.79e+3	1.14e+4	3.96e+3	1.49e-1
srAckley	7.79	2.15e+1	**1.39**	1.95e+1	2.08e+1	2.62	1.98e+1	2.07e+1	2.09e+1	2.27
srGriewank	1.02e-2	1.44e+1	5.09e-3	1.38e-2	2.82	1.14e-2	1.35	3.13	8.99e-1	**1.16e-3**
srPowell	3.55e-1	2.82e+9	1.44e+1	1.60e-2	5.35e+7	3.06e+3	3.45e+5	5.76e+7	1.12e+4	**1.74e-3**
srRastrigin	9.10e+1	4.72e+4	2.62e+2	1.48e+2	8.10e+3	**8.37e+1**	1.73e+3	9.28e+3	3.69e+2	1.19e+2
srRosenbrock	4.22e+1	2.64e+10	2.80e+1	4.03e+1	1.42e+8	1.90e+3	1.89e+6	1.96e+8	7.01e+4	**2.50e+1**
srZakharov	2.49e-4	1.45e+12	2.98e+2	**2.03e-4**	8.60e+3	1.95e+2	4.32e+3	8.47e+3	3.10e+3	2.52e-2

2) 30 Dimension: The average results archived by different algorithms on benchmark functions with dimension 30 are presented in Table 2.3. From Table 2.3 it is observed that; the mean results of ILPSO over all other PSO variants surpasses on the functions Ackley, Griewank, Powel, Rastrigin, Zakharov, rotated Ackley, rotated Griewank, rotated Rastrigin and rotated Zakharov. AAEPSO shows good results on shifted environment. It shows comparatively good result on rotated Rosenbrock, shifted Griewank, shifted Rosenbrock, shifted − rotated Griewank, shifted − rotated Powel and shifted − rotated Rosenbrock. PSOw shows good result on shifted Powel, shifted Zakharov. FDRPSO shows good result on shifted Ackley and shifted − rotated Ackley. GPSO shows good result on shifted Rastrigin and shifted − rotated Zakharov. HAEPSO and CLPSO shows good results on rotated Powel and shifted − rotated Rastrigin respectively.

The results of Table 2.4 are calculated with Friedman statistic to be 83.89 and 94.79 for non shifted and shifted problems respectively. On non shifted problems a extremely statistical significant difference between the results of EPSO and FIPSO vs AEPSO is found, and significant difference found between GPSO and AEPSO. The Table 2.3 concludes that AEPSO outperforms well. HAEPSO shows extremely statistical significant and very significant difference with EPSO and FIPSO respectively, here HAEPSO is found to perform well. ILPSO shows extremely statistical significant difference with EPSO, FDRPSO and FIPSO, and very significant difference with GPSO and CLPSO, here ILPSO is found to perform well. ILPSO also shows statistical significant difference in results with PSOw. AAEPSO shows extremely statistical significant and significant difference with EPSO and FIPSO, here also AAEPSO is found to do well. On shifted problems, extremely significant difference is found between PSOw and EPSO and PSOw is found to perform well. FDRPSO and EPSO shows extremely significant difference and FDRPSO performs well. Extremely statistical significant difference between FDRPSO and GPSO, GPSO performs well compared to FDRPSO. Extremely statistical significant difference between PSOw, GPSO and FIPSO, here FIPSO shows the worst performance. FDRPSO performs well compared to FIPSO as the difference is statistically very significant. CLPSO shows extremely and only significant difference with EPSO and FIPSO. Here CLPSO performs well compared to FIPSO and worst compared to GPSO. The significant difference is found between AEPSO and PSOw, the PSOw performs well. Extremely and very significant statistical difference of HAEPSO shows the worst performance.

Table 2.4: Friedman test on 30D problems (P < 0.0001)

without shift	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEEPSO
PSOw	–	ns	ns	ns	ns	ns	ns	ns	**s**	ns
EPSO	–	–	ns	ns	ns	ns	**es**	**es**	**es**	**es**
FDRPSO	–	–	–	ns	ns	ns	ns	ns	**es**	ns
GPSO	–	–	–	–	ns	ns	**s**	ns	**vs**	ns
FIPSO	–	–	–	–	–	ns	**es**	**vs**	**es**	**s**
CLPSO	–	–	–	–	–	–	ns	ns	**vs**	ns
AEPSO	–	–	–	–	–	–	–	ns	ns	ns
HAEPSO	–	–	–	–	–	–	–	–	ns	ns
ILPSO	–	–	–	–	–	–	–	–	–	ns
AAEPSO	–	–	–	–	–	–	–	–	–	–

with shift	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEEPSO
PSOw	–	**es**	ns		**es**	ns	**s**	**es**	ns	ns
EPSO	–	–	**es**	**es**	ns	**es**	ns	ns	ns	ns
FDRPSO	–	–	–	ns	**vs**	ns	ns	**vs**	ns	ns
GPSO	–	–	–	–	**es**	ns	ns	**es**	ns	ns
FIPSO	–	–	–	–	–	**s**	ns	ns	ns	**s**
CLPSO	–	–	–	–	–	–	ns	**vs**	ns	ns
AEPSO	–	–	–	–	–	–	–	ns	ns	**vs**
HAEPSO	–	–	–	–	–	–	–	–	ns	**es**
ILPSO	–	–	–	–	–	–	–	–	–	**s**
AAEPSO	–	–	–	–	–	–	–	–	–	–

ns=not significant, s=significant, vs=very significant and es=extremely significant

Table 2.5: Average results obtained with 50 D problems

Func	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEPSO
Ackley	1.88	2.15e+1	1.47e-2	2.08e-1	3.16e-1	2.45	3.73e-15	3.45e-5	**2.31e-15**	1.16e-1
Griewank	1.16e-2	3.18e+1	7.93e-4	1.14e-2	1.03e-1	4.43e-3	1.09e-4	1.14e-10	**00**	2.62e-2
Powell	1.79	5.05e+9	4.92e+3	1.86e-1	1.10e-2	1.61e+1	5.25e-7	8.98e-8	**2.03e-12**	2.03e-3
Rastrigin	1.76e+2	1.20e+5	4.90e+2	8.34e+1	1.79e+2	1.22e+2	3.17	3.21e+1	**00**	1.45e+1
Rosenbrock	1.17e+2	1.39e+11	2.23e+2	2.00e+2	2.91e+3	1.54e+2	**4.19e+1**	4.85e+1	4.79e+1	6.66e+1
Zakharov	1.00e+2	1.89e+16	1.30e+3	9.37	2.25	4.87e+2	3.85e-24	1.66e-5	**2.92e-24**	3.82
rAckley	2.84	2.15e+1	2.61e-1	3.24	1.05	2.44	3.73e-15	2.67e-5	**1.95e-15**	1.11
rGriewank	8.12e-3	2.92e+1	6.46e-2	1.05e-2	2.19e-1	2.25e-5	**00**	1.23e-10	**00**	1.62e-2
rPowell	5.21e+2	8.29e+9	4.60e+3	1.19	5.44e+3	6.61e+2	4.04e-6	3.99e-8	**8.14e-13**	1.07e-1
rRastrigin	2.72e+2	1.25e+5	5.92e+2	2.72e+2	2.55e+2	1.67e+2	1.21e+1	8.18e-7	**00**	2.29e+1
rRosenbrock	1.38e+2	1.12e+11	3.21e+2	1.34e+2	1.34e+4	3.03e+2	**4.59e+1**	4.87e+1	4.88e+1	6.03e+1
rZakharov	9.95e+1	3.37e+13	1.25e+3	8.11	1.34e+2	5.05e+2	4.54e-24	2.51e-6	**2.10e-38**	5.88e-1
sAckley	1.80	2.15e+1	1.69e-2	**5.17e-3**	2.06e+1	1.43	1.89e+1	2.06e+1	2.06e+1	1.24
sGriewank	1.33e-2	3.81e+1	2.48e-3	6.13e-2	4.58	3.95e-1	2.11	5.34	1.01	**1.17e-3**
sPowell	4.78	6.85e+9	9.98e+3	6.63	8.47e+7	2.20e+4	3.67e+6	1.13e+8	5.21e+4	**7.26e-2**
sRastrigin	1.70e+2	1.50e+5	5.08e+2	4.75e+1	1.61e+4	**1.16e+2**	5.78e+3	1.81e+4	7.64e+2	1.60e+2
sRosenbrock	9.30e+1	1.30e+11	8.88e+2	1.11e+2	6.52e+8	1.59e+3	6.38e+7	8.29e+8	1.14e+5	**7.02e+1**
sZakharov	3.65e+2	3.85e+16	1.33e+3	2.58e+1	2.00e+4	4.58e+3	1.31e+4	2.17e+4	1.50e+4	**8.20e-1**
srAckley	4.74	2.15e+1	**6.04e-1**	1.92e+1	2.06e+1	2.99	1.92e+1	2.07e+1	2.08e+1	3.13
srGriewank	4.19e-3	3.49e+1	4.11e-3	8.02e-2	4.45	1.01e-1	2.12	4.75	1.01	**2.65e-3**
srPowell	2.95e+3	8.98e+9	1.44e+4	1.04e+4	4.16e+7	2.27e+4	2.23e+6	6.68e+7	5.67e+4	**3.52e+1**
srRastrigin	2.51e+2	1.35e+5	5.17e+2	3.42e+2	1.31e+4	2.40e+2	4.47e+3	1.48e+4	7.55e+2	**1.91e+2**
srRosenbrock	7.11e+1	1.50e+11	1.18e+2	8.25e+1	1.05e+9	8.60e+3	1.37e+8	1.23e+9	2.31e+5	**6.22e+1**
srZakharov	2.99e+2	9.71e+15	1.21e+3	2.56e+1	2.36e+12	5.02e+3	2.13e+4	3.51e+12	1.88e+4	**1.33e-1**

3) 50 Dimension: The average results archived by different algorithms on benchmark functions with dimension 50 are presented in Table 2.5. The Table 2.5 also indicate that once again ILPSO and AAEPSO surpasses all other PSO variants in achieving good results. ILPSO surpasses all other PSO variants on the functions Ackley, Griewank, Powel, Rastrigin, Zakharov, rotated Ackley, rotated Griewank, rotated Powel, rotated Rastrigin and rotated Zakharov. AAEPSO shows good results on shifted environment. It shows comparatively good result on shifted Griewank, shifted Powel, shifted Rosenbrock, shifted Zakharov, shifted − rotated Griewank, shifted − rotated Powel, shifted − rotated Rastrigin, shifted − rotated Rosenbrock and shifted − rotated Zakharov. AEPSO shows good result on Rosenbrock, rotated Rosenbrock and rotated Griewank. GPSO shows good result only on shifted Ackley.

The statistically significant results analysis on 50 dimension problems are presented in Table 2.6. The results of Table 2.6 are calculated with Friedman statistic to be 88.55 and 97.76 for non shifted and shifted problems respectively. On non shifted problems PSOw shows statistically significant difference with EPSO and here PSOw does well. AEPSO shows extremely statistical significant difference with EPSO and FDRPSO, similarly it shows very significant difference with FIPSO. From Table 2.5 it can be concluded that AEPSO performs extremely well compared to EPSO, FDRPSO and FIPSO. Extremely statistical significant difference of HAEPSO is found over EPSO and very significant difference FDRPSO and FIPSO. This confirms that HAEPSO performs very well compared to these algorithms (Table 2.5). ILPSO show extremely significant difference and very significant difference compared PSOw, EPSO, FDRPSO, GPSO, FIPSO and CLPSO. From Table 2.5 it can be concluded that ILPSO performs extremely well compared to these algorithms. AAEPSO shows statistically very significant difference only with EPSO and from the Table 2.5 it can be concluded that AAEPSO perform very well compared EPSO and equally well with other algorithms.

On shifted set of problems PSOw, CLPSO and FDRPSO perform well compared to EPSO and HAEPSO. From Table 2.6 and Table 2.5 it can be concluded that FIPSO gives worst performance compared to its counterparts. The extremely statistical significant difference of AAEPSO with EPSO, FIPSO, HAEPSO and ILPSO shows that AAEPSO perform very well on shifted problems.

Table 2.6: Friedman test on 50D problems ($P < 0.0001$)

without shift	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEEPSO
PSOw	–	**s**	ns	ns	ns	ns	ns	ns	**es**	ns
EPSO	–	–	ns	ns	ns	ns	**es**	**es**	**es**	**vs**
FDRPSO	–	–	–	ns	ns	ns	**es**	**vs**	**es**	ns
GPSO	–	–	–	–	ns	ns	ns	ns	**vs**	ns
FIPSO	–	–	–	–	–	ns	**vs**	**vs**	**es**	ns
CLPSO	–	–	–	–	–	–	**s**	ns	**es**	ns
AEPSO	–	–	–	–	–	–	–	ns	ns	ns
HAEPSO	–	–	–	–	–	–	–	–	ns	ns
ILPSO	–	–	–	–	–	–	–	–	–	ns
AAEPSO	–	–	–	–	–	–	–	–	–	–

with shift	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEEPSO
PSOw	–	**es**	–	ns	**vs**	ns	ns	**es**	ns	ns
EPSO	–	–	**es**	**es**	ns	**es**	ns	ns	ns	**es**
FDRPSO	–	–	–	ns	**s**	ns	ns	**vs**	ns	ns
GPSO	–	–	–	–	**vs**	ns	ns	**es**	ns	ns
FIPSO	–	–	–	–	–	ns	ns	ns	ns	**es**
CLPSO	–	–	–	–	–	–	ns	**vs**	ns	ns
AEPSO	–	–	–	–	–	–	–	ns	ns	ns
HAEPSO	–	–	–	–	–	–	–	–	ns	**es**
ILPSO	–	–	–	–	–	–	–	–	–	**vs**
AAEPSO	–	–	–	–	–	–	–	–	–	–

ns=not significant, s=significant, vs=very significant and es=extremely significant

Table 2.7: Average results obtained with 100 D problems

Func	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEPSO
Ackley	4.16	2.15e+1	4.43	2.78	1.71	7.70	5.51e-15	2.76e-5	**3.02e-15**	5.60e-1
Griewank	3.17e-2	7.86e+1	4.93e-1	1.86e-2	1.99e-1	3.36e-1	**00**	2.96e-10	**00**	6.65e-3
Powell	1.11e+4	3.73e+10	1.37e+5	1.05e+2	6.80	1.49e+4	1.08e-7	1.61e-7	**3.60e-10**	1.72e-1
Rastrigin	6.79e+2	2.76e+5	1.22e+3	5.01e+2	2.17e+2	9.05e+2	1.02	1.85e-6	**00**	6.46e+1
Rosenbrock	1.42e+3	5.63e+11	3.84e+5	8.55e+2	2.49e+5	6.33e+4	**9.29e+1**	9.81e+1	9.85e+1	3.37e+2
Zakharov	8.25e+3	1.23e+19	8.04e+3	1.04e+3	7.72e+2	5.63e+3	1.80e-20	1.52e-4	**1.47e-20**	8.52e+1
rAckley	5.72	2.16e+1	5.09	4.56	7.81e-1	7.91	4.44e-15	2.80e-5	**3.38e-15**	1.52
rGriewank	5.86e-2	7.65e+1	6.54e-1	3.32e-2	1.37e-5	4.93e-1	**00**	1.58e-10	**00**	3.47e-3
rPowell	1.90e+4	3.59e+10	2.05e+5	3.73e+2	9.93	2.19e+4	8.48e-8	2.01e-7	**1.32e-8**	2.81e+1
rRastrigin	8.03e+2	3.06e+5	1.50e+3	1.14e+3	1.11e-1	1.01e+3	**00**	1.02e-6	**00**	2.31e+2
rRosenbrock	1.31e+3	4.09e+11	5.02e+5	8.32e+2	4.47e+4	7.28e+4	**9.50e+1**	9.88e+1	9.87e+1	4.42e+2
rZakharov	8.80e+3	1.04e+17	8.42e+3	8.94e+2	1.13e+3	5.72e+3	3.19e-20	2.95e-5	**2.73e-20**	6.11e+2
sAckley	5.88	2.16e+1	4.53	3.81	2.09e+1	8.31	1.95e+1	2.08e+1	2.12e+1	**2.68**
sGriewank	3.35e-2	9.21e+1	5.87e-1	5.70e-1	8.84	1.07	5.32	9.44	1.17	**2.14e-2**
sPowell	3.02e+4	2.88e+10	3.85e+5	5.33e+5	3.67e+8	6.85e+5	7.48e+7	3.91e+8	9.32e+5	**4.19e-1**
sRastrigin	**7.00e+2**	3.51e+5	1.34e+3	4.41e+2	3.15e+4	1.10e+3	1.71e+4	3.41e+4	2.43e+3	8.22e+2
sRosenbrock	1.42e+3	5.88e+11	4.91e+5	4.91e+4	1.54e+9	4.90e+5	3.43e+8	1.70e+9	2.17e+6	**1.14e+3**
sZakharov	2.64e+4	7.11e+18	1.50e+4	3.48e+3	3.86e+4	3.19e+4	2.78e+4	3.72e+4	5.72e+4	**1.50e+1**
srAckley	1.23e+1	2.16e+1	**5.70**	1.96e+1	2.10e+1	9.17	1.97e+1	2.09e+1	2.12e+1	6.39
srGriewank	3.76e-2	7.81e+1	5.80e-1	1.09	9.18	1.17	5.40	9.69	1.17	**9.86e-3**
srPowell	4.47e+4	3.66e+10	3.84e+5	1.74e+5	2.16e+8	3.79e+5	3.66e+7	2.05e+8	4.08e+4	**3.07e+1**
srRastrigin	4.98e+2	8.85e+3	8.90e+2	5.38e+2	1.53e+3	4.19e+2	8.47e+2	1.49e+3	1.42e+3	**4.09e+2**
srRosenbrock	4.20e+4	3.78e+13	1.92e+7	2.12e+5	1.58e+011	4.04e+7	4.25e+10	1.78e+11	1.63e+7	**8.22e+2**
srZakharov	3.77e+2	7.22e+14	3.25e+2	4.23e+1	1.61e+3	8.99e+2	2.16e+6	4.10e+012	1.53e+4	**4.22e+2**

4) 100 Dimension: The average results archived by different algorithms on benchmark functions with dimension 100 are presented in Table 2.7. The Table 2.7 also indicate that once again ILPSO and AAEPSO surpasses all other PSO variants in achieving good results. ILPSO surpass all other PSO variants on the functions Ackley, Griewank, Powel, Rastrigin, Zakharov, rotated Ackley, rotated Griewank, rotated Powel, rotated Rastrigin and rotated Zakharov. AAEPSO shows good results on shifted environment. It shows comparatively good result on shifted Ackley, shifted Griewank, shifted Powel, shifted Rosenbrock, shifted Zakharov, shifted − rotated Griewank, shifted − rotated Powel, shifted − rotated Rastrigin, shifted − rotated Rosenbrock and shifted − rotated Zakharov. AEPSO shows good result on Griewank, Rosenbrock, rotated Griewank, rotated Rastrigin and rotated Rosenbrock. FDRPSO and PSOw show good result only on shifted − rotated Ackley and shifted Rastrigin.

The statistically significant results analysis on 100 dimension problems are presented in Table 2.8. The results of Table 2.8 are calculated with Friedman statistic to be 101.1 and 91.20 for non shifted and shifted problems respectively. On non shifted problems GPSO and FIPSO show significant difference with FDRPSO and here GPSO does well. AEPSO shows extremely significant difference with almost all the algorithms as seen from Table 2.8 and after referring to Table 2.7 it can be concluded that AEPSO does extremely well compared to other algorithms. Similarly ILPSO also shows extremely significant difference with all other algorithms and this also confirms the extremely good performance of ILPSO over other algorithms as is evident from Table 2.7 also. HAEPSO also shows better performance over other algorithms. AAEPSO perform extremely well compared to EPSO, FDRPSO, and FIPSO evident from Table 2.7 and Table 2.8.

On shifted set of problems PSOw, CLPSO and FDRPSO perform well compared to EPSO and HAEPSO. From Table 2.8 and Table 2.7 it can be concluded that FIPSO gives the worst performance among the whole set. The extremely significant difference of AAEPSO with EPSO, FIPSO, AEPSO, HAEPSO and ILPSO shows that AAEPSO performs very well on shifted problems.

Table 2.8: Friedman test on 100D problems ($P < 0.0001$)

without shift	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEEPSO
PSOw	–	ns	ns	ns	ns	ns	**es**	**s**	**es**	ns
EPSO	–	–	ns	**s**	**vs**	ns	**es**	**es**	**es**	**es**
FDRPSO	–	–	–	ns	ns	ns	**es**	**es**	**es**	**s**
GPSO	–	–	–	–	ns	ns	**s**	ns	**s**	ns
FIPSO	–	–	–	–	–	ns	ns	ns	**es**	**es**
CLPSO	–	–	–	–	–	–	**es**	**vs**	**es**	ns
AEPSO	–	–	–	–	–	–	–	ns	ns	ns
HAEPSO	–	–	–	–	–	–	–	–	ns	ns
ILPSO	–	–	–	–	–	–	–	–	–	ns
AAEPSO	–	–	–	–	–	–	–	–	–	–

with shift	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEEPSO
PSOw	–	**es**	ns	ns	**es**	ns	ns	**es**	ns	ns
EPSO	–	–	**es**	**es**	ns	**es**	ns	ns	ns	**es**
FDRPSO	–	–	–	ns	**s**	ns	ns	**s**	ns	ns
GPSO	–	–	–	–	**es**	ns	ns	ns	ns	ns
FIPSO	–	–	–	–	–	ns	ns	ns	ns	**es**
CLPSO	–	–	–	–	–	–	ns	ns	ns	ns
AEPSO	–	–	–	–	–	–	–	ns	ns	**vs**
HAEPSO	–	–	–	–	–	–	–	–	ns	**es**
ILPSO	–	–	–	–	–	–	–	–	–	**vs**
AAEPSO	–	–	–	–	–	–	–	–	–	–

ns=not significant, s=significant, vs=very significant and es=extremely significant

2.6.3 Robustness

The results related to robustness are presented from Table 2.9 to 2.12 as standard deviation. The best standard deviation is shown with bold.

1) 10 Dimension: The standard deviation achieved by different algorithms on benchmark function with dimension 10 are presented in Table 2.9. From Table 2.9 it can be observed that AAEPSO shows excellent robustness on Ackley, Griewank, Powell, Rastrigin, Zakharov, rotated Ackley, rotated Griewank, rotated Powell, rotated Rastrigin, rotated Zakharov, shifted Zakharov and shifted − rotated Rastrigin. ILPSO algorithm performs better compared to other PSO variants on the functions Griewank, Rastrigin, Zakharov, rotated Rastrigin and rotated Zakharov. The robustness of PSOw is also seen on shifted Powell, shifted − rotated Powel and shifted − rotated Zakharov. FDRPSO is robust on Ackley, shifted Ackley, shifted − rotated Ackley and shifted − rotated Rosenbrock. The robustness of CLPSO is seen on functions Ackley, rotated Ackley, shifted Griewank and shifted − rotated Griewank. GPSO shows good robustness on Rastrigin and rotated Rosenbrock. EPSO, AEPSO and HAEPSO shows good robustness on Ackley, rotated Rosenbrock and Rosenbrock respectively.

Table 2.9: standard deviation obtained with 10 D problems

Func	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEPSO
Ackley	9.17e-16	**00**	**00**	1.49e-5	3.59e-1	**00**	9.17e-16	1.48e-5	1.47e-15	**00**
Griewank	2.94e-2	2.28e-2	6.93e-2	1.39e-2	2.38e-1	4.88e-3	9.79e-2	4.04e-11	**00**	**00**
Powell	1.50e-7	1.53e-3	1.21e-7	1.02e-6	5.23e-4	4.38e-6	1.80e-5	9.62e-9	2.49e-10	**00**
Rastrigin	1.19	1.09	1.41e+1	2.84e-7	3.62e+1	2.57e-1	2.68	1.66e-7	**00**	**00**
Rosenbrock	3.93e-1	2.62	1.23e-1	1.42	8.55e+2	1.81	9.87e-2	**9.25e-2**	5.21e-1	2.58e-1
Zakharov	1.20e-108	1.64e-43	4.65e-47	3.53e-8	1.31e+1	4.30e-28	1.19e-49	4.01e-8	**00**	**00**
rAckley	2.98e-1	6.84e-1	9.17e-16	1.48e-5	1.37	**00**	9.17e-16	1.94e-5	1.47e-15	**00**
rGriewank	6.09e-2	8.51e-2	5.02e-2	4.33e-2	2.28e-1	1.48e-2	6.64e-2	6.94e-11	1.12e-1	**00**
rPowell	2.45e-6	8.10e-3	3.26e-5	3.42e-5	6.82e+1	1.68e-3	4.89e-7	2.95e-9	4.56e-4	**00**
rRastrigin	3.46	7.11	8.33	8.62	6.55e-2	1.22	7.87	4.13	**00**	**00**
rRosenbrock	1.09e+2	4.11e+1	1.39	2.06e+1	3.30e+2	6.20	**2.94e-2**	3.74e-2	3.88e-1	3.69e-1
rZakharov	3.39e-113	5.70e-45	8.49e-50	4.27e-8	8.58	3.37e-29	1.79e-52	5.47e-8	**00**	**00**
sAckley	1.25e-15	6.71e-1	00	7.29e-7	3.83e-1	3.48e-11	6.39	5.29e-1	8.37	**00**
sGriewank	3.26e-2	8.06e-2	1.16e-1	1.05e-2	8.20e-2	**8.75e-3**	6.15e-2	1.01e-1	8.70e-2	3.68e-2
sPowell	**5.83e-8**	1.38	1.41e-7	5.19e-7	4.51e+4	7.42e-4	3.27	4.28e+4	1.29e+2	1.10e-7
sRastrigin	1.63	1.88	1.37e+1	**6.22e-10**	1.68e+2	3.50e-1	2.74e+1	2.23e+2	1.29e+1	1.12
sRosenbrock	1.05	8.44	1.49e-1	**8.51e-3**	6.82e+6	2.89	4.39e+2	1.86e+7	1.28e+2	5.96e-1
sZakharov	2.51e-30	9.21	1.65e-29	1.70e-10	2.79e+2	1.19e-2	6.58e+1	6.22e+2	1.99	**00**
srAckley	1.63e-15	4.95	**00**	8.43e-1	7.98e-1	1.03e-13	7.84	7.00e-1	8.21	5.14
srGriewank	6.06e-2	7.68e-2	7.66e-2	9.77e-2	7.99e-2	**2.23e-2**	1.07e-1	1.06e-1	6.40e-2	5.61e-2
srPowell	**2.12e-7**	4.24e-2	3.47e-5	5.49e-6	3.36e+4	4.50e-2	1.20e+2	2.11e+5	8.67e+1	7.35e-7
srRastrigin	4.44	1.17e+1	1.16e+1	9.07	4.61e+2	2.22	3.16e+1	4.41e+2	7.91	**4.24**
srRosenbrock	6.54e+1	5.74e+1	**2.81**	5.23e+1	5.31e+5	2.72e+1	2.80e+1	2.44e+6	1.08e+3	6.72e+1
srZakharov	**1.20e-29**	4.12	2.61e-28	5.71e-11	8.19e+2	8.39e-3	3.55e+1	7.76e+2	1.24	2.91e-28

2) 30 Dimension: The standard deviation archived by different algorithms on function with dimension 30 are presented in Table 2.10. From Table 2.10 it can be observed that ILPSO shows excellent robustness on Ackley, Griewank, Rastrigin, Zakharov, rotated Ackley, rotated Griewank, rotated Rastrigin, rotated Zakharov. AAEPSO algorithm performs better compared to other PSO variants on the functions shifted Powell, shifted Rosenbrock, shifted − rotated Powell. HAEPSO shows its good stability on Powell, Rosenbrock, rotated Powell and rotated Rosenbrock. The robustness of PSOw is also seen on shifted Griewank and shifted Zakharov. FDRPSO is robust on shifted Ackley and shifted − rotated Rosenbrock. The robustness of CLPSO is seen on functions shifted − rotated Rastrigin and shifted − rotated Griewank. GPSO shows superior stability on shifted Powel and shifted − rotated Zakharov. Ackley, rotated Ackley, shifted Griewank and shifted − rotated Griewank. GPSO shows robustness on Rastrigin and rotated Rosenbrock. EPSO, AEPSO and HAEPSO shows robustness on Ackley, rotated Rosenbrock and Rosenbrock respectively. AEPSO and FIPSO show good results on functions rotated Griewank and shifted − rotated Ackley respectively.

Table 2.10: standard deviation obtained with 30 D problems

Func	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEPSO
Ackley	7.20e-1	8.71e-1	5.97e-8	4.34e-3	2.81	4.40e-1	1.25e-15	1.63e-5	**1.47e-15**	2.14e-15
Griewank	2.46e-2	6.35	3.91e-1	1.22e-2	4.06e-1	4.55e-3	5.61e-6	1.25e-10	**00**	3.33e-2
Powell	4.79e-5	1.86e+9	3.43e-3	1.52e-2	8.13e+3	6.74e-1	3.76e-7	**6.25e-8**	5.0222e-3	8.16e-5
Rastrigin	1.24e+1	2.58e+4	5.35e+1	3.64	1.71e+2	7.16	9.59	2.94e+1	**00**	1.59
Rosenbrock	2.54e+1	1.91e+10	3.03e+1	4.03e+1	1.18e+4	2.96e+1	4.66e-1	**3.54e-2**	6.86	1.98e+1
Zakharov	1.10e-2	1.96e+14	6.67e+1	4.93e-2	6.24e+1	1.68e+1	1.94e-27	1.68e-6	**2.37e-112**	5.47e-15
rAckley	5.25e-1	2.83e-1	4.97e-9	8.23e-1	1.93	4.07e-1	9.17e-16	1.39e-5	**1.83e-15**	5.98e-1
rGriewank	1.17e-2	6.18	8.25e-3	9.41e-3	4.95e-1	9.81e-3	**00**	1.37e-10	**00**	4.16e-2
rPowell	3.27e-1	2.31e+9	1.54	5.34e-2	1.95e+3	4.79e+1	3.97e-6	**5.07e-8**	2.72e-3	5.97e-4
rRastrigin	2.90e+1	2.72e+4	2.60e+1	2.78e+1	1.11e+2	8.86	1.85e+1	2.67e-7	**00**	6.05
rRosenbrock	6.17e+1	1.91e+10	1.32e+2	3.88e+1	4.50e+4	3.88e+1	3.22e-1	**1.87e-2**	2.00e-1	1.06e+1
rZakharov	6.97e-3	2.85e+11	4.79e+1	4.81e-2	8.96e+1	2.11e+1	4.79e-27	2.29e-7	**8.05e-137**	3.02e-8
sAckley	5.12	1.13e-1	8.97e-8	2.30e-4	**8.70e-2**	7.11e-1	6.37e-1	9.41e-2	2.83e-1	3.70e-1
sGriewank	**7.59e-3**	4.82	3.16e-1	1.17e-2	1.17e-1	9.56e-3	1.04e-1	1.84e-1	4.78e-2	1.72e-2
sPowell	2.16e-4	1.13e+9	7.28e-3	2.51e-2	9.07e+6	2.09e+3	5.05e+5	8.69e+6	4.97e+3	**1.36e-4**
sRastrigin	1.06e+1	2.88e+4	5.27e+1	**5.48**	3.57e+2	1.06e+1	5.98e+2	7.02e+2	5.59e+1	1.95e+1
sRosenbrock	3.12e+1	1.86e+10	2.02e+1	3.03e+1	5.41e+7	4.85e+1	2.92e+5	2.69e+7	3.14e+4	**1.57e+1**
sZakharov	1.66e-2	9.71e+13	6.63e+1	2.48e+1	6.51e+2	2.08e+2	9.15e+2	8.00e+2	8.35e+2	1.86e-1
srAckley	8.97	7.46e-2	5.39	4.74e-1	**5.94e-2**	6.63e-1	3.12e-1	8.35e-2	1.40e-1	4.61
srGriewank	1.28e-2	5.41	6.35e-3	1.66e-2	9.44e-2	**8.78e-3**	1.19e-1	2.30e-1	4.42e-2	1.44e-2
srPowell	1.05	2.04e+9	5.32e+1	3.32e-3	1.04e+7	1.85e+3	1.76e+5	2.23e+7	3.82e+3	**3.07e-2**
srRastrigin	2.42e+1	1.98e+4	5.98e+1	6.28e+1	4.53e+2	**2.12e+1**	5.73e+2	7.59e+2	4.63e+1	3.69e+1
srRosenbrock	2.72e+1	2.39e+10	**1.03e+1**	2.47e+1	1.50e+7	6.93e+3	1.55e+6	4.30e+7	2.82e+4	2.18e+1
srZakharov	2.98e-4	3.31e+12	7.17e+1	**1.09e-4**	9.51e+2	1.11e+2	6.45e+2	8.34e+2	7.38e+2	5.54e-2

3) 50 Dimension: The standard deviations achieved by different algorithms on benchmark function with dimension 50 are presented in Table 2.11. From Table 2.11 it can be observed that ILPSO shows excellent robustness on Ackley, Griewank, Rastrigin, rotated Ackley, rotated Griewank, rotated Rastrigin and rotated Zakharov. AAEPSO algorithm performs better compared to other PSO variants on the functions shifted Powell, shifted Rastrigin, shifted Rosenbrock, shifted − rotated Powell. HAEPSO shows its good stability on Powell, Rosenbrock, rotated Powell and rotated Rosenbrock, shifted Zakharov, shifted − rotated Powell and shifted − rotated Griewank. The robustness of HAEPSO is seen on Powell, Rosenbrock, rotated Powell, rotated Rosenbrock and hifted − rotated Ackley. GPSO shows good robustness on shifted Ackley, shifted − rotated Rosenbrock and shifted − rotated Zakharov. AEPSO good robustness on Zakharov and rotated Griewank. CLPSO and FDRPSO show their stability on functions shifted − rotated Rastrigin and rotated Griewank respectively.

Table 2.11: standard deviation obtained with 50 D problems

Func	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEPSO
Ackley	7.44e-1	5.58e-2	1.07e-2	2.83e-1	6.11e-1	6.12e-1	1.72e-15	1.58e-5	**1.12e-15**	3.65e-1
Griewank	1.16e-2	3.21	2.32e-3	1.32e-2	3.27e-1	6.23e-3	3.46e-4	1.33e-10	**00**	3.04e-2
Powell	3.79	2.83e+9	6.07e+3	1.84e-1	3.46e-2	1.88e+1	9.74e-7	**8.85e-8**	6.00e-4	1.16e-3
Rastrigin	4.04e+1	2.62e+4	1.31e+2	3.22e+1	2.94e+2	4.39e+1	7.56	1.01e+2	**00**	1.00e+1
Rosenbrock	6.93e+1	5.32e+10	1.29e+2	7.53e+1	9.21e+3	3.70e+1	4.98e-1	**3.49e-2**	6.77e-1	3.68e+1
Zakharov	4.85e+1	1.43e+16	1.44e+2	2.95	6.83	1.39e+2	**3.08e-24**	1.76e-5	9.23e-2	6.23
rAckley	7.68e-1	6.16e-2	3.76e-1	5.03e-1	2.20	8.88e-1	1.72e-15	1.33e-5	**1.50e-15**	8.79e-1
rGriewank	9.84e-3	3.56	1.77e-1	1.27e-2	4.02e-1	3.72e-5	**00**	1.43e-10	**00**	2.97e-2
rPowell	6.15e+2	5.16e+9	3.93e+3	1.14	1.72e+4	2.78e+2	9.00e-6	**3.21e-8**	1.87e-4	1.04e-1
rRastrigin	6.18e+1	2.81e+4	6.10e+1	8.03e+1	3.24e+2	3.76e+1	2.51e+1	6.91e-7	**00**	2.09e+1
rRosenbrock	1.15e+2	7.10e+10	4.18e+2	5.65e+1	4.20e+4	3.36e+2	4.69e-1	**1.57e-2**	9.59e-2	2.62e+1
rZakharov	9.67e+1	5.33e+13	1.86e+2	3.44	2.22e+2	1.13e+2	4.68e-24	3.52e-6	**5.93e-38**	1.12
sAckley	5.32e-1	7.73e-2	9.34e-3	**5.34e-4**	4.53e-2	6.90e-1	4.44e-1	6.10e-2	3.95e-1	5.51e-1
sGriewank	1.63e-2	8.06	**4.02e-3**	5.20e-2	1.89e-1	2.43e-1	2.54e-1	1.97e-1	1.68e-2	8.87e-3
sPowell	8.68	2.23e+9	8.35e+3	1.84e+1	7.42e+6	1.20e+4	1.87e+6	2.98e+7	4.10e+4	**5.67e-2**
sRastrigin	3.35e+1	2.13e+4	8.79e+1	4.01e+1	4.74e+2	4.43e+1	1.11e+3	8.46e+2	7.85e+1	**3.33e+1**
sRosenbrock	3.69e+1	4.75e+10	1.39e+3	3.85e+1	5.54e+7	1.03e+3	2.35e+7	8.85e+7	3.38e+4	**3.60e+1**
sZakharov	1.98e+2	1.97e+16	2.56e+2	2.84e+1	8.62e+2	6.92e+2	3.21e+3	1.92e+3	1.93e+3	**7.37e-1**
srAckley	5.29	6.04e-2	6.42e-1	3.71e-1	9.78e-2	5.48e-1	2.83e-1	**4.17e-2**	2.36e-1	1.05
srGriewank	4.51e-3	7.26	6.61e-3	1.35e-1	1.55e-1	7.02e-2	3.42e-1	2.21e-1	1.57e-2	**1.28e-3**
srPowell	2.66e+3	5.11e+9	6.72e+3	7.21e+3	1.24e+7	1.38e+4	8.95e+5	1.79e+7	1.48e+4	**3.29e+1**
srRastrigin	6.95e+1	2.36e+4	7.33e+1	8.12e+1	7.71e+2	**3.99e+1**	9.07e+2	8.35e+2	6.74e+1	1.16e+2
srRosenbrock	3.66e+1	7.87e+10	5.35e+1	**3.60e+1**	1.41e+8	1.26e+4	6.66e+7	1.29e+8	1.04e+5	4.58e+1
srZakharov	2.00e+2	1.62e+16	2.59e+2	**4.39e+1**	5.99e+12	4.76e+2	6.33e+3	6.97e+12	5.62e+3	9.21e+1

4) 100 Dimension: The standard deviation achieved by different algorithms on function with dimension 100 are presented in Table 2.12. From Table 2.12 it can be observed that ILPSO shows excellent robustness on Ackley, Griewank, Powell, Rastrigin, Zakharov, rotated Ackley, rotated Griewank, rotated Powell, rotated Rastrigin, rotated Zakharov, shifted Griewank, shifted Rastrigin and shifted − rotated Griewank. AAEPSO algorithm performs better compared to other PSO variants on the functions namely shifted Powell, shifted Rosenbrock, shifted Zakharov, shifted − rotated Powell, shifted − rotated Rastrigin and shifted − rotated Rosenbrock. HAEPSO shows good stability on Rosenbrock, rotated Rosenbrock, shifted Ackley, shifted − rotated Ackley and shifted − rotated Rastrigin. AEPSO shows good robustness on Griewank, rotated Griewank and rotated Rastrigin. GPSO is stable on shifted − rotated Zakharov.

Table 2.12: standard deviation obtained with 100 D problems

Func	PSOw	EPSO	FDRPSO	GPSO	FIPSO	CLPSO	AEPSO	HAEPSO	ILPSO	AAEPSO
Ackley	7.87e-1	5.82e-2	1.04	4.49e-1	2.78	1.01	1.50e-15	1.10e-5	**1.12e-15**	7.51
Griewank	3.10e-2	8.05	1.53e-1	7.71e-3	4.20e-1	2.97e-1	**00**	2.84e-10	**00**	1.60e-2
Powell	4.38e+3	7.33e+9	4.72e+4	1.51e+2	2.15e+1	5.84e+3	1.58e-7	1.31e-7	**3.41e-10**	1.62e-1
Rastrigin	7.99e+1	4.54e+4	1.46e+2	1.03e+2	4.55e+2	1.14e+2	2.15	1.63e-6	**00**	5.31e+1
Rosenbrock	7.79e+2	2.17e+11	3.08e+5	5.81e+2	4.94e+5	7.65e+4	1.07	**1.53e-2**	5.52e-1	1.07e+2
Zakharov	2.58e+3	4.48e+18	1.35e+3	1.78e+2	1.29e+3	5.41e+2	2.12e-20	1.39e-4	**1.78e-20**	1.13e+2
rAckley	6.87e-1	4.09e-2	8.65e-1	7.50e-1	2.46	8.10e-1	1.87e-15	1.02e-5	**1.50e-15**	1.35
rGriewank	5.16e-2	1.30e+1	2.93e-1	2.06e-2	3.40e-5	1.76e-1	**00**	2.40e-10	**00**	4.43e-3
rPowell	9.31e+3	1.57e+10	1.10e+5	2.74e+2	1.93e+1	8.51e+3	1.20e-7	1.70e-7	**2.21e-8**	2.64e+1
rRastrigin	5.82e+1	5.83e+4	2.63e+2	2.00e+2	3.04e-1	1.72e+2	**00**	6.84e-7	**00**	1.00e+2
rRosenbrock	8.36e+2	1.20e+11	6.52e+5	4.51e+2	8.71e+4	7.70e+4	1.20	**1.33e-2**	7.81e-2	2.09e+2
rZakharov	3.92e+3	1.22e+17	1.64e+3	1.54e+2	3.15e+3	1.10e+3	6.13e-20	2.78e-5	**1.91e-20**	5.95e+2
sAckley	4.91	5.73e-2	9.25e-1	7.73e-1	3.67e-2	2.34	1.80e-1	**2.66e-2**	2.55e-1	2.15
sGriewank	5.58e-2	1.06e+1	2.80e-1	2.16e-1	2.08e-1	5.30e-2	2.50e-1	2.00e-1	**1.76e-2**	2.07e-1
sPowell	9.65e+3	1.16e+10	1.10e+5	7.41e+5	1.70e+7	1.45e+5	2.38e+7	5.67e+7	4.15e+5	**5.45e-1**
sRastrigin	8.58e+1	3.61e+4	1.74e+2	2.51e+2	8.63e+2	3.10e+2	2.54e+3	1.17e+3	**1.19e+2**	1.89e+2
sRosenbrock	1.70e+3	2.57e+11	2.97e+5	7.81e+4	4.25e+7	6.04e+5	9.13e+7	5.49e+7	4.66e+5	**9.50e+2**
sZakharov	7.20e+3	4.25e+18	3.52e+3	1.51e+3	1.45e+3	3.15e+3	3.85e+3	9.63e+2	1.47e+4	**2.89e+1**
srAckley	7.99	4.55e-2	1.32	1.11e-1	4.22e-2	8.97e-1	3.74e-1	**3.55e-2**	2.08e-1	1.70
srGriewank	2.75e-2	1.19e+1	3.25e-1	8.31e-2	2.10e-1	6.94e-2	4.23e-1	2.69e-1	**1.22e-2**	1.41e-1
srPowell	1.54e+4	1.63e+10	1.41e+5	4.84e+5	3.15e+7	1.66e+5	1.28e+7	1.85e+7	6.29e+3	**1.52e+1**
srRastrigin	1.25e+2	1.35e+3	3.11e+1	4.55e+1	3.90e+1	7.83e+1	1.94e+2	**2.23e+1**	5.20e+1	4.16e+2
srRosenbrock	7.62e+4	1.42e+13	1.46e+7	3.79e+5	1.60e+10	6.22e+7	1.19e+10	8.83e+9	7.13e+6	**4.57e+2**
srZakharov	1.36e+2	7.01e+14	4.60e+1	1.97e+1	3.65e+1	9.60e+1	8.12e+6	2.68e+012	4.34e+3	**4.21e+1**

2.7 Conclusions

This chapter introduced Single Objective Unconstrained Optimization concept (SOUO). This chapter also presented the detailed challenges that PSO suffers while solving SOUO problems. It was found that the major challenges of PSO are premature convergence, poor quality of solution and uncertainty in solution, this is especially dominant on complex multimodal functions. To address the problem of premature convergence and poor quality of solution, the AEPSO algorithm was proposed. AEPSO first separates the particles responsible for premature convergence (diverged) that give poor quality of solution. This occurs due to the fact that the particles are stuck in local minima. AEPSO then accelerates these particles under the guidance of global best solution, essentially these particles were forced to leave their current position and search the solution space around global solution. The stagnated particles were further processed by the introduction of hyperspherical coordinate updates along with acceleration and was called as HAEPSO. In HAEPSO the information of the particles were first converted from cartesian coordinate system into hyperspherical coordinate system, then all the updates were carried out. This strategy helps in gaining the greater control of the particles flying trajectory. Further all the strategies developed were integrated together with well known comprehensive learning and the new algorithm was called as ILPSO. AEPSO has solved the prevailing problem of PSO to some extent and again this also gives scopes for further research. AEPSO accelerated the diverged particles with constant size search patch of unit radius around global solution. This strategy helped more in exploitation of the solution than exploration. This is addressed in AAEPSO where the search patch of dynamic size was maintained. In AAEPSO the acceleration of diverged particles towards global solution was carried out adaptively. The size of the search patch was more in the beginning of search process and it decreases exponentially with the iteration. This strategy in particular helped for more exploration of search space around global solution in the beginning and more exploitation as the search process proceeds. The comprehensive analysis of these developed algorithms were carried out on set of complex multimodal functions. These functions were further rotated and shifted for thorough validation of proposed algorithms. The dimensions of the problems was chosen to be 10, 30, 50 and 100. The results were analyzed using well known performance metrices; Conventional and Statistical. The conventional metrices

include convergence graphs, average of the results and robustness (standard deviation). The statistical metrices include Friedman′s test followed by Dunn′s multiple comparison test. The statistical results in terms of convergence, average results and robustness clearly proves that the developed algorithms shows remarkable improvements in the quality of solution and hence addresses the prevailing problems of PSO, this was further supported by statistical metrices (Friedman′s and Dunn′s test). The ILPSO also has shown the superior performance over other algorithms including HAEPSO. The performance of ILPSO was not affected with the increase in the dimensions of the problems (evident from statistical results), specially on normal and rotated problems. The AAEPSO has shown excellent convergence and quality of solutions on lower dimension of normal and rotated problems, but as dimensions of the problem increased the AAEPSO failed to achieve good results. The major feature of AAEPSO has also been proved on shifted and shifted rotated problems for higher dimension. Among the four developed PSO variants ILPSO and AAEPSO have shown competitive performance among themselves and outperform other two (AEPSO and HAEPSO).

Chapter 3

Single Objective Constrained Optimization

This chapter introduces the concept of Single Objective Constrained Optimization (SOCO) problems along with the detailed literature survey for solving SOCO problems using PSO and other competitive algorithms. It presents the algorithmic and implementation details of proposed Stochastic Ranking PSO (SRPSO) for solving SOCO numerical benchmark and engineering design problems. The performance of developed algorithm is validated using different performance indicators such as average results (mean) and robustness (standard deviation) are presented. These indicators are further used for performance comparison of developed algorithm with the current state of art. The chapter is concluded with results and discussions.

3.1 Introduction

Most of the real world science and engineering optimization problems are complex and nonlinear. They have a number of linear and or nonlinear constraints associated with them. Thus it became necessary for an optimization algorithm to handle constraints while optimizing objective function. The Single Objective Constrained Optimization (SOCO) problems are the problems to either to minimize or maximize a single objective function under given constraints such as inequality, equality, upper bound and lower bound. The SOCO usually involves multiple constraints, these could be nonlinear, discontinuous, nondifferentiable and this

increases the difficulty while optimization. In addition to these difficulties the objective and constraints may be multimodal and have multiple optima. The major challenging issues of the SOCO is to obtain the optimum solution for an objective function while satisfying the constraints simultaneously. The simplified mathematical representation of SOCO problems can be expressed as

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ &That\ optimizes\ f(\overrightarrow{x}) \\ subject\ to: \quad g_i(\overrightarrow{x}) &< 0, i = 1, 2, \ldots, q \\ h_i(\overrightarrow{x}) &= 0, i = q+1, q+2, \ldots, m \\ Where\ l_i \leq x_i &\leq u_i, i = 1, 2, \ldots, D \end{aligned} \tag{3.1}$$

Where f is the objective function, g_i and h_i are the inequality and equality constraints respectively. The values l_i and u_i $\forall$ i $\in$ D are the lower and upper bounds of the solution defining the search space.

The common method to handle the constraints is to convert constrained optimization problem into an unconstrained problem and apply unconstrained optimization algorithms. Mathematically this conversion is implemented with the introduction of penalty function for constraints as

$$\psi(x) = f(x) + r_k \phi(g_i(x); i = 1, 2, \ldots, m)$$

where $\phi \geq 0$ is a real valued function that imposes a penalty. The penalty on each constraint ($g_i \forall i \in m$) is imposed by the penalty factor r_k. The aim is to decrease the fitness of infeasible solutions in order to favor the selection of feasible solutions. The feasible solution has less or no constraints violation and satisfy all the constraints. Similarly infeasible solution has constraints violation and does not satisfy all the constraints.

3.2 Challenges in Single Objective Constrained Optimization

Most of the real world science and engineering design problems are associated with constraints. The presence of complex nonlinear constraints poses challenges to the algorithms. The major challenges in solving these are as follows.

1. The SOCO problems are associated with multiple constraints; these constraints will diverge the solution.

2. The associated constraints may be linear and or non-linear; this makes the problem more complex.

3. Often the search space available for these kind of problems may be complex, and the the feasible (physically realizable) region in the search space can be very small.

4. Generally the objectives and constraints are nondifferentiable and or non-continuous. Hence it will be impossible for point based algorithms to find the solution.

5. Due to the complex solution space the problems could be multimodal, and the algorithms based on initial solution may get stuck local optima leading to premature convergence.

3.3 Literature Survey

Due to the multimodal nature of found in objective function and constraints, the conventional optimization algorithms tend to get trapped local optima while solving complex non-linear functions. Hence conventional methods will not solve such problems to the desired accuracy level. During the last few decades, Swarm Intelligence (SI) and Evolutionary Algorithms (EA) have emerged as an alternative for such complex science and engineering design problems.

Constraint-handling algorithm are mainly based on three different methods i.e.,repair method, tournament selection method and penalty method [42, 43]. In repair method an infeasible solution maps into a feasible one. The repair is made for evaluation purpose and the repaired individual replaces the original one in

the population. The tournament selection method gives simple strategy to select feasible solution. If the two solutions are feasible then one with the best objective function value will be selected. If one is feasible and the other is infeasible, then the feasible one is chosen. If both are infeasible, then the one that less violates the constraints is chosen. The penalty method [44] converts constrained problems into unconstrained problem with the introduction of penalty factor. This method is used to penalize the infeasible solutions by adding a penalty term to the objective function value. The major problem with this approach is the selection of the penalty factor itself. Improper selection of penalty factor leads to the premature convergence.

In 1995 Michalewicz et al. [45] proposed an approach to handle the constraints. In this work both infeasible and feasible solutions were generated and constraints were satisfied while evaluating solutions. The violation of constraints for each solution was considered separately, and the relationship between infeasible and feasible solutions were exploited to guide the search. The disadvantage with this work was, the lack of proper metrices to measure the relationship between feasible and infeasible solutions. The most common method for constraints handling is the penalty method, but the major problem is the selection of penalty factor. The improper selection of penalty factor leads to either overpenalization or underpenalization. This over and underpenalization problem was addressed by Runarsson and Yao in 2000 and proposed Stochastic Ranking (SR) [46]. In this approach the individuals were assigned the ranks based on the feasibility and constraints violations. This approach does not introduce any new factors for adjustments but relies on a simple bubble sort mechanism to rank the individuals. This approach has proved to be very effective in solving under constraints [46]. In 2002 Parsopoulos and Vrahatis [47] introduced a multi-stage non-stationary penalty function with PSO to handle constraints. The non-stationary penalty function adjusts penalty values dynamically. This approach gives good results only to some kind of problems. Hu and Eberhart [48] proposed a feasibility preservation strategy for determining the best particle. In this approach the particles fly through the problem space during optimization and explore new solutions. This new solution was used to check if the new explored solutions satisfy all the constraints to be preserved for future utilization. This algorithm has proved to be efficient for solving nonlinear optimization problems with nonlinear inequality constraints. The strategies proposed in [48, 47] of non-stationary multistage penalty and feasibility preservation for handling con-

straints were analyzed in details by Coath and Halgamuge [49] in 2003. In [49] the comparative performance assessment of each method was carried out in terms of accuracy and rate of convergence. Hu et al. in 2003 [50], proposed an algorithm based on Particle Swarm Paradigm for nonlinear constrained optimization. The algorithm finds initial feasible population and ensures feasibility during the entire optimization process. The algorithm [50] proved to be efficient for solving nonlinear optimization problems with inequity constraints. Ray and Liew proposed an algorithm [51] based on social behavior of human and insect societies in 2003. Algorithm [51] show that the social interactions are better and faster than biological evolution. In this algorithm every society had a leader and society gets influenced by the leader, leaders improve through inter society information exchange and migrations. In 2004, He et al. introduced fly-back mechanism [52], this method returns an infeasible particle to its previous feasible position. The major drawback of this algorithm was the requirement of an all feasible initial population. In 2005 Montes et al., presented a simple multi-membered evolution strategy (SMES) [53] to solve constrained nonlinear optimization problems. In [53], a simple diversity mechanism is used, this improved the algorithm′s exploitation capabilities and computational cost. In 2006 Amirjanov et al., proposed a Changing Range Genetic Algorithm (CRGA) [54] in which the size of the search space of feasible region adaptively shifts and shrinks by employing feasible and infeasible solutions in the population to reach the global optimum [54]. This algorithm has many control parameters and efficiency of this method mainly depends on proper selection of these parameters. Krohling et al. proposed co-evolutionary particle swarm optimization with Gaussian distribution (CPSO-GD) [55] to solve constrained optimization. In [55], the Gaussian probability distribution was used to generate the acceleration coefficients of PSO instead of uniform probability distribution. This approach [55] has proved to be effective and robust. Tessema et al. proposed a Self Adaptive Penalty Function (SAPF) [56] technique with Genetic Algorithm (GA) for solving constrained optimization problems. It uses self adaptive penalty function which encourages infeasible individuals with low objective function values and low constraint violations [56]. The feasible individuals in [56] were used to guide the population towards finding feasible solutions. This technique was simple to implement and did not introduce any new parameter for tuning. During 2007 Huang et al., proposed a Co-evolutionary Differential Evolution (CDE) [57] algorithm to solve the constrained optimization problems. In this algorithm [57]

a special penalty function was designed to handle the constraints and then CDE was employed to perform the search in spaces for both, solutions and penalty factors [57]. This facilitates the solutions and penalty factors to evolve interactively and self-adaptively resulting to satisfactory solutions and suitable penalty factor simultaneously. In order to tackle the problem of fine-tuning of penalty functions and to balance objective function against the degree of constraint violation, Shimizu et al. proposed an algorithm [58] in 2007. They [58] used a ranking technique and a percentage-based tolerance adjustment scheme with evolutionary strategy. This algorithm does not require a user-specified penalty coefficient or any other tuning parameters. He and Wang proposed co-evolutionary particle swarm optimization approach (CPSO) [59] in 2007. In CPSO [59] algorithm co-evolution was used to adapt penalty factors and was integrated with particle swarm optimization for handling constraint problems. In 2008 Jiao and Tang [60] proposed Constrained Engineering Optimization via Particle Swarm Optimization (CEOPSO). The constraint handling mechanism in CEOPSO [60] preserves some of the infeasible individuals in the population during search process. Zhang et al. in 2008 proposed dynamic stochastic selection multimember differential evolution (DSS-MDE) [61]. In 2009 Wang et al. [62], introduced multiobjective way of handling constraints and proposed Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. In 2009 Mezura et al. [63], proposed a variant of PSO for constraints handling and was called Improved Particle Swarm Optimization (IPSO). Zahara et al. in 2009 proposed hybrid Nelder-Mead simplex method and Particle Swarm Optimization (NM-PSO) [64]. The NM-PSO [64] algorithm includes gradient repair method and constraint fitness priority-based ranking method. Majid et al. in 2010 proposed two new harmony search (HS) [65] algorithms for engineering optimization design problems. The first algorithm is called proposed harmony search (PHS) and this introduces a new definition of bandwidth. The second algorithm is called improving proposed harmony search (IPHS) used non-uniform mutation to enhance accuracy and convergence rate of PHS algorithm. Leandro Coelho proposed quantum-behaved particle swarm optimization (QPSO) [66]. In QPSO [66] algorithm harnesses the quantum mechanics theories and used with PSO, essentially mutation operator with Gaussian probability distribution was incorporated. Sabat et al. proposed Stochastic Ranking Particle Swarm Optimization (SRPSO) [67, 68] for handling the constraints in 2010. The SRPSO algorithm integrates the well known stochas-

tic ranking [46] along with AEPSO [31] algorithm for handling the constraints while solving SOCO problems. In this algorithm particles were ranked as per stochastic ranking [46] and the lower ranked particles were accelerated towards the feasible solution (global solution) using AEPSO algorithm [31].

3.4 Research Contribution

Particle Swarm Optimization (PSO) is a population-based simple SI algorithm and is very popular in single objective unconstrained optimization [69, 35, 32]. However the basic PSO, like other evolutionary algorithms, lacks an explicit mechanism to handle constraints which are often found in science and engineering optimization problems. These prevailing problems and necessity of SOCO has been the motivation for development of an efficient and robust optimization algorithm for SOCO. One of the famous and common method to handle the constraints is by converting constraint optimization problem in to an unconstrained problem with the introduction of penalty factor [44]. The penalty factor is mainly an exterior and the main aim is to decrease the fitness of infeasible solutions so that the selection of feasible solution is favored. Mathematically the constrained problem (equation (3.1)) can equivalently be converted in to an unconstrained problem with the introductions of penalty factor as

$$\psi(x) = f(x) + r_k \phi(g_i(x); i = 1, 2, \ldots, m)$$

where $\phi \geq 0$ is a real valued function that imposes a penalty. The penalty on each constraint is imposed by the penalty factor r_k. Although the above penalty method works well for certain constrained optimization problems, selecting the penalty factor r_k remains to a challenge. If the penalty factor is chosen to be too small, an infeasible solution may not be penalized enough (underpenalization), resulting a final infeasible solution. If the penalty factor is too large, a feasible solution is very likely to be found (overpenalization), but could be of poor quality [46]. Thus underpenalization and overpenalization are not good for handling constraints.

Despite its simplicity, a penalty function requires the definition of penalty factors to determine the severity of the penalization, and these values depend on the problem being solved [46]. Due to this major disadvantage, several alternative constraint-handling algorithms have been proposed. Stochastic ranking technique

has been proposed [46] to maintain the required balance between objective function and penalty function. This technique uses stochastic bubble-sort algorithm to rank the individuals for generating offsprings for the next generation. In order to solve complex constrained problems, PSO algorithm is hybridized with stochastic ranking [46] along with accelerated exploration technique. The proposed algorithm hence is named as Stochastic Ranking Particle Swarm Optimization (SRPSO) [67, 68].

3.4.1 Stochastic Ranking Particle Swarm Optimization

Many studies have been done for solving constrained optimization problems using EA [43, 44, 46, 45, 70, 71] and PSO [72, 73]. However, it still remains a challenge to devise more efficient and effective techniques for handling constraints. PSO shows better performance on unconstrained optimization problems and is popular for its fast convergence [35, 32]. However, PSO neither explicitly nor implicitly has the mechanism to handle constraints. Due to its faster convergence and effectiveness of stochastic ranking [46] for constrained handling, SRPSO is proposed by integrating both stochastic ranking and AEPSO for solving constrained optimization problems. The stochastic ranking [46] technique uses a simple bubble-sort algorithm to rank the individuals producing offsprings for the next generation. In stochastic ranking, a probability P_f is introduced to rank the individuals. Generally two adjacent individuals are used for comparison and ranking. If both are in feasible space, an individual with smaller objective values will get higher rank. If both adjacent individuals are in infeasible space, P_f is used to compare and rank. The individual with smaller objective value will occupy the rank with the probability P_f. If one particle is in feasible space and the other one is in infeasible space, then the particle in feasible space gets the higher rank. Similarly all individuals are ranked.

3.4.2 Implementation details of SRPSO Algorithm

Algorithm 3.4.1 shows all the necessary steps for practical implementation of SRPSO algorithm. In the initialization phase, the search range (X_{max}, X_{min}) is initialized depending on the problem. The population size (NP) is initialized randomly in the search range and velocity is restricted to 20% of the search range i.e. $V_{max} \leftarrow 0.20 * (X_{max} - X_{min})$. The cognitive and social factors initialized

to 1.479 i.e. $c_1 = 1.479, c_2 = 1.479$. The objective function $f(x)$ and penalty function $\phi(g(x))$ of all particles are evaluated. The initial personal best *pbest* and global best *gbest* is set after evaluating fitness f and constraints of the all the particles. In the optimization phase t is set as counter for iterations and i for particles. The velocity and position of the whole population is evaluated as per equations (2.3) and (2.2) respectively. Algorithm 3.4.2 shows simple bubble-sort algorithm for ranking the particles. In Algorithm 3.4.2, s_w represents number of swaps, NP is population size, $\phi(X_j)$ is constraint violation for X_j particle, $f(X_j)$ represents fitness calculation for X_j particle, $swap(X_j, X_{j+1})$ exchanges particle X_j with X_{j+1} and *rand* gives a uniformly distributed random number in the range $[0, 1]$. The particles are compared based on constrained violation and fitness of the particles, detailed steps are shown in Algorithm 3.4.2. All the particles are categorized as either feasible or infeasible, and are ranked using Algorithm 3.4.2. The balance between underpenalization and overpenalization is achieved by setting the probability P_f to be less than $\frac{1}{2}$ [46]. The highest ranked particle is treated as global best (*gbest*) for the current iteration and is compared with previous *gbest*, and the one with minimum objective is considered as *gbest* for next generation. The personal best *pbest* of the particles are also decided based on the acquired rank. The highest ranked μ individuals out of NP are selected for the next generation (μ is set as $\frac{NP}{7}$). In the next generation rest of the particles ($NP - \mu$) are regenerated from the highest-ranked μ particles as per AEPSO (Algorithm 2.4.1). Regeneration in SRPSO is done at a refresh rate of 10 similar to AEPSO. This process continues till the stopping criteria is met or the iteration counter reaches the maximum count.

Algorithm 3.4.1 Pseudo code for SRPSO

Initialize
$Set \leftarrow X_{max}, X_{min}, D, NP$
$V_{max} \leftarrow 0.20 * (X_{max} - X_{min})$
$t \leftarrow 0, i \leftarrow 0$ ▷ t for iterations and i for particles
Randomly initialize particle's position $X_i^0 \in D$ in R;
Randomly initialize particle's velocity $V_i^0 \leq V_{max}$
Evaluate fitness function values f_i^0
$pbest_i^0 \leftarrow f_i^0, \quad gbest^0 \leftarrow f_{best}^0$
Optimize
while $t \leq MaximumGeneration$ **do**
  while $i \leq NP$ **do**
Update velocity and position as per equation (2.2) and (2.3) respectively.
Evaluate fitness function values f_i^{t+1}
Find $pbest_i^{t+1}$
    if $f_i^{t+1} < pbest_i^t$ **then**
      $pbest_i^{t+1} \leftarrow f_i^{t+1}$
    end if
Find $gbest_d^{t+1}$
    if $gbest_d^{t+1} < gbest_d^t$ **then**
      $gbest_d^{t+1} \leftarrow gbest_d^t$
    end if
    $i \leftarrow i + 1$, *go to step* 9.
  end while
  if stop criteria not met **then**
    $t \leftarrow t + 1$, *go to step* 8.
  end if
  Rank the individuals according to Stochastic Ranking as per Algorithm 3.4.2
  Select the highest ranked μ particles as per Algorithm 2.4.2.
  Generate NP particles from μ individuals as per Algorithm 2.4.2.
end while
Report results
Terminate

Algorithm 3.4.2 Pseudo code for Stochastic Ranking using bubble sort procedure

for $i \leftarrow 1$, S_w **do**
  for $j \leftarrow 1$, $NP - 1$ **do**
    if $(\phi(X_j) = \phi(X_{j+1}) = 0)$ **or** $(rand < P_f)$ **then**
      if $(f(X_j) > f(X_{j+1}))$ **then**
        $swap(X_j, X_{j+1})$
      end if
    else
      if $(\phi(X_j) > \phi(X_{j+1}))$ **then**
        $swap(X_j, X_{j+1})$
      end if
    end if
  end for j
    if no swaps done break
end for i

3.5 Simulation

The simulations were carried out in Windows XP Operating System, on Pentium IV 2.6GHz with 512MB of RAM. The coding is done in Matlab 7.2 programming language. The population size and the maximum iterations considered for all the algorithms are 25 and 1000 respectively. The population is initialized in the search range $[X_{min}, X_{max}]$. Where X_{max} and X_{min} are the maximum value and minimum value of search range. The value of control parameters for SRPSO algorithm are $c_1 = c_2 = 1.479$ and $P_f = 0.45$.

The performance comparisons of proposed algorithm SRPSO is carried out with the state of the art CRGA [54], SAPF [56], CDE [57], CPSO-GD [55], SMES [53], Ray and Liew [51], NM-PSO [64], CPSO [59], IHS [65], DSS-MDE [61] and QPSO [66]. The simulations were conducted separately on fifteen constrained numerical functions and five engineering design problems. The best results obtained by SRPSO algorithm on fifteen constrained numerical functions are compared with the CRGA, SAPF, CDE, CPSO-GD and SMES algorithms [54, 56, 57, 55, 53, 51]. For engineering design problems, performance of SRPSO is compared with CDE, Ray and Liew, NMPSO, CPSO, HS and DSS-MDE algorithms.

Since PSO algorithms experience difficulty in optimizing functions associated with constraints, the focus is on well-known standard benchmark functions CEC 2006 [74]. Furthermore, the efficiency of SPRSO is validated on five engineering design problems viz., 1) Welded beam design problem, 2) Pressure vessel design problem, 3) Three-bar truss design problem 4) Speed reducer design problem and 5) Spring design [43, 51]. The chosen test problems are characterized by different difficulties in linear, nonlinear, quadratic, different-dimensionality and with different difficult constraints. The detailed definition of benchmark problems and engineering design problems can be found in Appendix B.

3.6 Results and Discussions

This section present the discussions on the results obtained by SRPSO on fifteen benchmark functions. The discussion is extended with the comparison of SRPSO to the current state-of-the-art techniques [54, 56, 57, 55, 53]. The performance of proposed SRPSO algorithm is also validated on five engineering design problems 1) Welded beam design problem, 2) Pressure vessel design problem, 3) Three-

bar truss design problem 4) Speed reducer design problem and 5) Himmelblau Nonlinear problem [43, 51].

3.6.1 Numerical benchmark functions

The complete statistical results obtained by proposed SRPSO on 15 standard benchmark function are tabulated in Table 3.1. The first two columns of Table 3.1 show the constrained functions and their corresponding optima. This table summarizes the best, worst, median and mean results over 25 independent runs. The robustness of the algorithm is tested with standard deviation which is also tabulated in Table 3.1 along with Feasible Rate (FR). The FR denotes the percentage of the solutions which are in the feasible space. The number of violated constraints at the median solution is represented as c in Table 3.1 where the sequence of three numbers indicate the number of violations (including inequality and equalities) by more than 1.0, more than 0.01 and more than 0.0001 respectively. The mean value of violations of all constraints at the median solution is represented as v in Table 3.1. It can be concluded from Table 3.1 that the proposed SRPSO algorithm consistently found global optima for all benchmark functions except g07 and g10 functions. SRPSO also seems to be a robust technique for constrained optimization problems as the standard deviation of solutions obtained in multiple runs is very low except g05, g06 and g10.

Table 3.1: Optimum results, standard deviation, number and mean value of the constrained violations at the median solution obtained by SRPSO

Func	optimum	best	median	worst	c	v	mean	std	FR
g01	-15	-15.00	-15.00	-15.00	0,0,0	0	-15.00	5.27e-12	100
g02	-0.8036191	-0.8034681	-0.7933138	-0.7572932	0,1,1	0.16	-0.7886154	1.31e-3	100
g03	-1.0005001	-0.9996365	-0.9986063	-0.9965328	0,0,0	0	-0.9984904	8.18e-5	100
g04	-30665.539	-30665.539	-30665.539	-30665.364	0,0,0	0	-30665.526	4.05e-3	100
g05	5126.4967	5126.4985	5127.6362	5145.9263	0,0,0	0	5129.9011	5.11	100
g06	-6961.8139	-6961.81397	-6961.8139	-6323.3140	0,0,0	0	-6916.1371	138.331	100
g07	24.306209	24.312803	24.360653	24.885038	0,1,1	0	24.38	1.13e-2	100
g08	-0.0958250	-0.0958250	-0.0958250	-0.0958250	0,0,0	0	-0.0958250	2.80e-11	100
g09	680.63006	680.63043	680.65104	680.76645	0,0,0	0	680.66052	3.33e-3	100
g10	7049.248	7076.397	7262.878	8075.923	1,2,2	522	7340.69640	255.37	100
g11	0.75	0.75	0.75	0.75	0,1,1	0	0.75	9.44e-5	100
g12	-1	-1	-1	-1	0,0,0	0	-1	2.62e-11	100
g15	961.71502	961.71517	961.71710	961.77126	0,0,0	0	961.72076	1.12e-2	100
g16	-1.9051553	-1.9051553	-1.9051553	-1.9051553	0,0,0	0	-1.9051553	1.12e-11	100
g24	-5.5080133	-5.5080133	-5.5080133	-5.5080133	0,0,0	0	-5.5080133	2.69e-11	100

SRPSO is compared with five recent approaches; CRGA[54], SAPF[56], CDE[57], CPSO-GD[55] and SMES[53].

Table 3.2: Comparison of best results of SRPSO with state of art

Func	SRPSO	CRGA[54]	SAPF[56]	CDE[57]	CPSO-GD[55]	SMES[53]
g01	-15.00	-14.9977	-15.00	-15.00	-15.00	-15.00
g02	-0.80346805	-0.802959	-0.803202	-0.794669	NA	-0.803601
g03	-0.9997	-0.9997	-1.000	NA	NA	-1.000
g04	-30665.539	-30665.520	-30665.401	-30665.539	-30665.539	-30665.539
g05	5126.4985	NA	NA	NA	NA	NA
g06	-6961.8139	-6956.251	-6961.046	-6961.814	NA	-6961.814
g07	24.312803	24.882	24.838	NA	24.711	24.327
g08	-0.095825041	-0.095825	-0.095825	NA	NA	-0.095825
g09	680.63004	680.726	680.773	680.771	680.678	680.632
g10	7076.397	7114.743	7069.981	NA	7055.6	7051.903
g11	0.75	0.750	0.749	NA	NA	0.75
g12	-1	-1.000000	-1.000000	-1.000000	NA	-1.000
g15	961.7151	NA	NA	NA	NA	NA
g16	-1.9051553	NA	NA	NA	NA	NA
g24	-5.5080133	NA	NA	NA	NA	NA

Best result comparison: Table 3.2 shows the best result obtained by SRPSO for all the considered benchmark functions along with other technique. SRPSO gives comparatively good results on almost all the benchmark functions except g03.

Worst result comparison: The worst results obtained by SRPSO and other techniques are tabulated in Table 3.3. SRPSO shows comparatively good results for g01, g05, g07, g08, g12, g15, g16, g24 benchmark functions on worst results.

Mean result comparison: Table 3.4 shows the mean result obtained by SRPSO along with other techniques. SRPSO shows comparatively good results on most of the benchmark functions except g03, g04, g06, g09, and g10. As shown in Table 3.2 - Table 3.4, the proposed SRPSO method gives better result as compared to CRGA, SAPF, CDE, CPSO-GD, SMES in terms of the quality of solution and robustness as a measure of best and standard deviation values respectively.

Table 3.3: Comparison of worst results of SRPSO with state of art

Func	SRPSO	CRGA[54]	SAPF[56]	CDE[57]	CPSO-GD[55]	SMES[53]
g01	-15.000000	-14.9467	-13.097	-15.0000	-14.994	-15.000
g02	-0.757293	-0.722109	-0.745712	-0.779837	NA	-0.751322
g03	-0.9965328	-0.9931	-0.887	NA	NA	-1.000
g04	-30665.3644038	-30660.313	-30656.471	-30665.509	NA	-30665.539
g05	5145.9262874086544	NA	NA	NA	NA	NA
g06	-6323.3140341	-6077.123	-6943.304	-6901.285	NA	-6952.482
g07	24.885038	27.381	33.095	NA	27.166	24.843
g08	-0.095825041	-0.095808	-0.092697	NA	NA	-0.095825
g09	680.76645813	682.965	682.081	685.144	681.371	680.719
g10	8075.923755	10826.09	7489.406	NA	11458	7638.366
g11	0.750364	0.757	0.757	NA	NA	0.75
g12	-1.000000	-1.000000	-0.999548	-1.000000	NA	-1.000
g15	961.77126304593492	NA	NA	NA	NA	NA
g16	-1.9051552584358129	NA	NA	NA	NA	
g24	-5.508013271495793	NA	NA	NA	NA	NA

Table 3.4: Comparison of mean results of SRPSO with state of art

Func	SRPSO	CRGA[54]	SAPF[56]	CDE[57]	CPSO-GD[55]	SMES[53]
g01	-15.000000	-14.9850	-14.552	-15.0000	-14.997	-15.000
g02	-0.788615	-0.764494	-0.755798	-0.785480	NA	-0.785238
g03	-0.9985	-0.9972	-0.964	NA	NA	-1.000
g04	-30665.526026	-30664.398	-30665.922	-30665.536	NA	-30665.539
g05	5129.9010819813429	NA	NA	NA	NA	NA
g06	-6916.1370274885467	-6740.288	-6953.061	-6960.603	NA	-6961.284
g07	24.38	25.746	27.328	NA	25.709	24.475
g08	-0.095825041366072852	-0.095819	-0.095635	NA	NA	-0.095825
g09	680.66052250760072	681.347	681.246	681.503	680.7810	680.643
g10	7340.6964029884484	8785.149	7238.964	NA	8464.2	7253.047
g11	0.75	0.752	0.751	NA	NA	0.75
g12	-1.000000	-1.000000	-0.99994	-1.000000	NA	-1.000
g15	961.72076523564249	NA	NA	NA	NA	NA
g16	-1.9051552584503268	NA	NA	NA	NA	NA
g24	-5.508013271537056	NA	NA	NA	NA	NA

3.6.2 Engineering design problems

The complete design solutions obtained by proposed SRPSO on five engineering design problems are tabulated in Table 3.5. The first column of Table 3.5 show the algorithms. This table also summarizes the statistical results i.e., best, mean,

worst, function evaluations (FEs) and standard deviation obtained for each design problem.

Welded beam design problem

This problem aims to minimize the cost of beam subject to constraints on shear stress, bending stress, bucking load, and the end deflection. Four continuous design variables are the thickness of the beam, width of the bean, length of the weld, and the weld thickness. The best feasible solution found by SRPSO is $f(0.22104332, 5.8282757, 6.6883208, 0.22104332) = 1.7248665802025513$. This problem has also been solved in the recent past [57, 51, 66, 59, 64, 65, 61]. A comparison of simulation result statistics is presented in Table 3.5. The proposed SRPSO gives better performance with regard to quality of solution (mean of all runs), speed (number of function evaluations), and robustness in terms of standard deviation as compared to other standard and recently reported approaches.

Pressure vessel design problem

This problem aims to minimize the fabrication cost of cylindrical pressure vessel subjects to constraints on thickness of pressure vessel, thickness of the head, inner radius of the vessel and length of the vessel. The best solution obtained by proposed SRPSO is $f(0.80866029, 0.3997212, 41.899497, 180.39996) = 5886.19835373$. This solution is far better than all the existing schemes in terms of all considered statistical parameters. It also demonstrates the superiority of proposed SRPSO algorithm for finding the global minima as compared to other techniques with regard to number of function evaluations.

Speed reducer design problem

In this constrained optimization problem, the weight of speed reducer is to be minimized subject to constraints on bending stress of the gear teeth, surface stress, transverse deflections of the shafts, and stresses in the shafts. As shown in Table 3.5, the best solution obtained by SRPSO is $f(2.7540737, 0.72184566, 17.571564, 7.8446617, 7.7938804, 3.2070532, 5.1558205) =2506.9424074072431$ in comparision to the available methods for this problem. The total number of evaluations is 20,000 in SRPSO, while 54,456 number of function evaluations in Ray and Liew [51]. Although the proposed method gives a better solution, the standard deviation is quite high.

Three-bar truss design problem

This problem deals with the design of a three-bar truss structure where the volume is to minimized subject to stress constraints. The best feasible solution found by

Table 3.5: Comparison of engineering design problem results of SRPSO with state of art

	best	mean	worst	std	FEs
		WELDED BEAM [51]			
SRPSO	1.72486658	1.72489934	1.72542212	1.12e-6	20,000
CDE [57]	1.733461	1.768158	1.824105	2.2e-02	240,000
Ray and Liew [51]	2.3854347	3.0025883	6.3996785	9.6e-01	33,095
NM-PSO [64]	1.724717	1.726373	1.733339	0.3e-02	200,000
CPSO [59]	1.728024	1.748831	1.782143	0.129e-01	80.000
HS [65]	1.7248	NA	NA	NA	65,300
DSS-MDE[61]	2.3809	2.38095	2.38095	2.1e-10	24,000
		PRESSURE VESSEL [43]			
SRPSO	5886.1984	5942.8353	6315.0147	80.4459	20,000
CDE [57]	6059.7340	6085.2303	6371.0455	4.3e+01	240,000
GQPSO[66]	6059.7208	6440.3786	7544.4925	448.4711	NA
CPSO [59]	6061.0777	6147.1332	6363.8041	86.4545	200,000
NM-PSO [64]	5930.3137	5946.7901	5960.0557	9.16	80,000
HS [65]	7197.730	NA	NA	NA	100,000
		THREE-BAR STRUSS [51]			
SRPSO	263.895844	263.897780	263.907955	3.02e-03	20,000
Ray and Liew [51]	263.89584654	263.90335672	263.96975638	1.3e-02	17,610
DSS-MDE [61]	263.89584	263.89584	263.89584	9.2e-7	15000
		SPEED REDUCER [51]			
SRPSO	2506.942407	2756.525088	2947.825502	91.766319	20,000
Ray and Liew [51]	2994.744241	3001.758264	3009.964736	4.0	54,456
HS [65]	2994.4	NA	NA	NA	20,000
DSS-MDE[61]	2994.47106	2994.47106	2994.47106	3.5e-12	30000
		HIMMELBLAU [43]			
SRPSO	-31025.554042	-31019.990288	-31005.091619	5.990052	20,000
CPSO [59]	-31025.600	NA	NA	NA	NA
HS [65]	-31025.565	NA	NA	NA	20,100

SRPSO is f(0.7887701, 0.40799905)=263.8958, matches with the reported best-known result for this problem. A comparison of results presented in Table 3.5 shows that SRPSO perform equally well to Ray Liew [51] and DSS-MDE [61], in terms of quality of solution and robustness.

Himmelblau Nonlinear problem

The best feasible solution found for this problem by SRPSO is f(78, 33.049666, 27.133319,45, 44.776691)=-31025.554041580137. The obtained objective function value is comparable and even better than then existing methods. To summarize the result statistics reported in Table 3.5, the proposed SRPSO algorithm has substantial capability to handle various constrained engineering optimization problems efficiently with regard to quality of solution, robustness and speed (number of function evaluations).

3.7 Conclusions

This chapter introduced the concept of Single Objective Constrained Optimization (SOCO) along with the detailed literature survey on constraints handling techniques. This chapter also presented the detailed challenges that PSO faces in handling constraints. The most common and popular way of handling constraints is the penalty method. This method essentially converts constrained optimization problems to unconstrained optimization problems with the introduction of a penalty factor to penalize the constraints. The major problem with this approach was the value of penalty factor; greater value results into overpenalization and lesser leads to underpenalization. It was found from the literature that the Stochastic Ranking (SR) was the simple method to find a balance between over and underpenalization. The SR mechanism has proved its strength in handling the constraints. This has motivated for further research on PSO with SR integration. To address the problem of premature convergence and poor quality of solution, and incorporating the constrained handling method in PSO; Stochastic Ranking Particle Swarm Optimization Ranking (SRPSO) was proposed. SRPSO essentially a hybrid algorithm of AEPSO and stochastic ranking. This algorithm rank the particles based on stochastic ranking and accelerates the infeasible particles in the direction of feasible particles by maintaining the diversity in the population. The comprehensive experimental analysis of SRPSO was carried out with present state of the art on varied standard constrained benchmark problems with different complexities. Further SRPSO was applied on constrained engineering design problems and was compared with state of art algorithms. The comprehensive analysis was carried out using metrices like average of the results, robustness (standard deviation) and function evaluations (FEs). The SRPSO has shown competitive performance on par with state of art on numerical benchmark problems and superior performance on engineering design problems.

Chapter 4

Multi Objective Unconstrained Optimization

This chapter introduces the concept of Multiobjective Unconstrained Optimization (MOUO). It also presents a detailed literature survey for solving MOUO problems using Particle Swarm Optimization (PSO) and other competitive techniques including Swarm Intelligence (SI) techniques. The major challenges faced by PSO while solving MOUO problems are also addressed in this chapter. The chapter is extended with research contribution made in MOUO by developing a novel PSO variant namely Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (MOAAEPSO). The algorithmic and implementation details of MOAAEPSO is also presented. The performance of developed algorithm is illustrated in terms of Non Pareto compliant performance indicators such as pareto front graphs, inverted generational distance, spacing, diversity and convergence metrices while solving benchmark functions. The statistical significance of the algorithms are also analyzed using Pareto compliant hypervolume indicator and nonparametric ANOVA comparison test. These indicators are used for performance comparison of MOAAEPSO with state of the art. The chapter is concluded with results and discussions.

4.1 Introduction

Multiobjective Unconstrained Optimization (MOUO) is an important and challenging topic in science and engineering domain. The main objective of MOUO

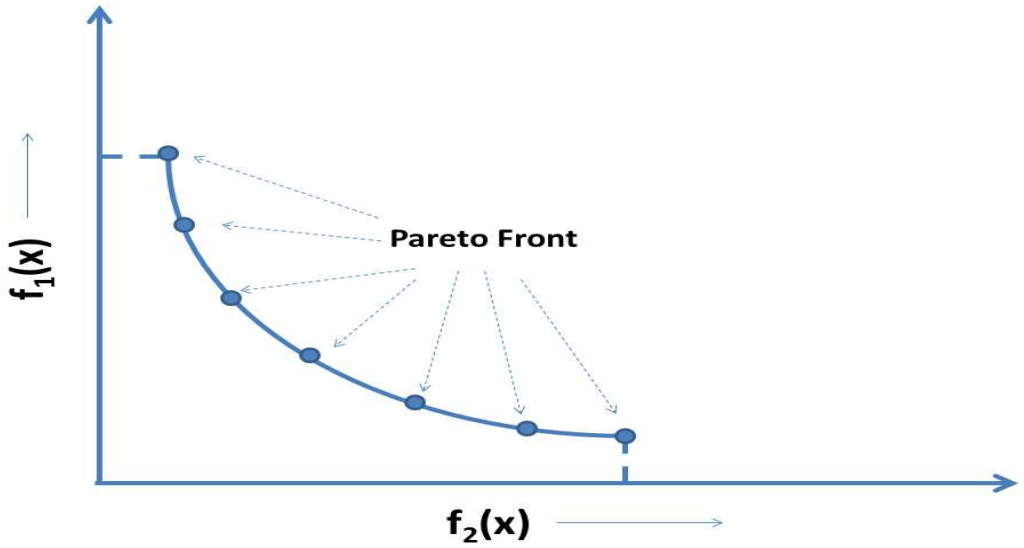

Figure 4.1: Pareto Front

is to minimize or maximize a set of objective functions (some times conflicting) simultaneously without any constraints. The multiple objectives may be non-linear, nondifferentiable, discontinuous and multimodal with many local optima. Mathematically the MOUO problem is defined as

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ & f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x}) \\ Where\ & l_i \leq x_i \leq u_i, i = 1, 2, \ldots, D \end{aligned} \tag{4.1}$$

In the above equation k is the number of objectives to be optimized simultaneously. The values l_i and u_i $\forall$ i $\in$ D are the lower and upper bounds defining the search space. Since multi-objective problem consists of multiple objectives. Hence there can be a set of optimal solutions to the given problem. The tradeoff amongst them is defined in terms of Pareto optimality (shown in Figure 4.1).

4.1.1 Pareto optimality

The vector $\mathbf{u} \in R$ is said to dominate the vector $\mathbf{v} \in R$, denoted by $\mathbf{u} \prec \mathbf{v}$, iff $\forall\ i\ \in 1, 2 \ldots k\ :\ f_i(\mathbf{u}) \leq f_i(\mathbf{v})$ and $\exists\ j\ \in\ 1, 2 \ldots k\ :\ f_j(\mathbf{u}) < f_j(\mathbf{v})$. A decision

vector $\mathbf{x} \in R$ is said to be Pareto optimal with respect to $\mathbf{R}$ iff there is no other decision vector that dominates $\mathbf{x}$ in $\mathbf{R}$. The set of all Pareto optimal solutions in the decision space is termed as Pareto optimal set and the corresponding set of objective vector is termed as Pareto optimal front as shown in the Figure 4.1. The aim of mutiobjective optimization algorithm is to obtain Pareto optimal front accurately.

4.2 Literature Survey

Multiobjective Unconstrained Optimization (MOUO) has emerged as a key research area in the field of science and engineering in the last few decades. Real world optimization problem demands efficient algorithms for solving multiple and often conflicting objectives simultaneously [75, 76, 77, 78, 79]. Evolutionary and Swarm Intelligence based algorithms have contributed a lot in this filed, comprehensive surveys of which can be found in [80, 81, 82]. A lot more literature of Particle Swarm Optimization on MOUO problems can be found in [83, 84, 85, 86].

The work on MOUO proposed by Fonseca and Fleming [87] and Tanino et al. [88] in 1993, involves evaluating rank of the members of a population by using both the Pareto dominance and preference information. In 1994 Horn et al. presented a Pareto based algorithm called as niched pareto genetic algorithm [89] for solving multiobjective optimization problems. In [89] the Genetic Algorithm (GA) was modified to deal with multiple objectives with the integration of Pareto dominance and niching pressure. This helps to spread its population along the pareto optimal surface. The strength of this algorithm was shown only on few and simple problems. In 1999 Zitzler and Thiele [90] presented a detail study on Multiobjective evolutionary algorithms and strength of the pareto approach. Furthermore they introduced an algorithm named as Strength Pareto Evolutionary Algorithm (SPEA) for multiobjective problems. This algorithm [90] maintains an external memory to store nondominated solutions and incorporates clustering approach to maintain the size of external memory. In [90], the concepts of non-domination and a secondary population of non-dominated points were used. With every iteration the secondary population was updated with the non-dominated offspring, and all dominated elements were discarded. If the archive size exceeds its maximum limit then the clustering mechanism groups all currently non-dominated solutions into a pre-defined number of clusters and picks a representative solution from each

cluster, thereby ensuring population diversity. This algorithm has shown superior performance only on 0/1 knapsack problem. In 2000, Knowles and Corne proposed a memetic Pareto archived evolution strategy to solve MOUO problems [91]. This algorithm [91] introduces a Pareto ranking-based selection method and couples it with a partition scheme in objective space. It used two different archives to save non-dominated solutions. Knowles and Corne proposed Pareto archived evolution strategy (PAES) [92]. This algorithm [92] concentrates mainly on local search and uses external archive to store the Pareto optimal solutions. The archive was updated with the following strategy; if a new solution is non dominated to any member of the archive then it enters into the archive by discarding the members of archive it dominates. If the archive reaches maximum size, then the new nondominated solution acceptance will be decided by a histogram-like density measure over a hyper-grid division of the objective space. This has shown very good spread of non-dominated solutions in a fixed-size archive. The PAES proved to be efficient but on limited test problems. Leung and Wang [93] proposed uniform design and GA for solving multiobjective problems. This algorithm [93] used multiple fitness functions for guiding the search process. In [93], each fitness function was the weighted sum of the normalized objective function. This strategy helped in spreading the solution on the Pareto frontier in the objective space. In this strategy uniform design was used to select weights and generate good initial population for searching efficiently. The nondominated sorting genetic algorithm (NSGA) [94] was proposed by Deb et al. in 2001. In NSGA [94], a simple sorting algorithm was used for sorting nondominated solutions to obtain Pareto fronts of MOUO problems. To maintain the population diversity, a distance metric was chosen to calculate the proximity measure between two solutions. Though this was the fist of its kind and popular; it has problems like large computational complexity. This was because nondominated sorting procedure was invoked in every generation. Further the mechanism lacks the ability to retain the good solutions found during search process and lacks even the diversity-preservation mechanism. In the year 2001, Jin et al. integrated dynamic weights with evolutionary algorithm and proposed Evolutionary Dynamic Weighted Aggregation (EDWA) [95] for solving MOUO problems. The weighted sum approach was efficient in multiobjective optimization, but it generated one Pareto solution in one run and could not find the concave part of the Pareto front. Thus EDWA [95] introduced dynamic weighted aggregation and was able to obtain the Pareto set in one run irrespective

of whether Pareto was convex or concave. In 2002, Zitzler et al. developed SPEA2 [96, 97] using non dominated sorting to optimize multiple objectives simultaneously. Essentially SPEA2 [96, 97] was an improved version of SPEA [90] with major differences of a well defined fitness assignment technique, a density estimation technique and an improved external archive truncation method. In SPEA2 [96, 97] fitness assignment was modified with integration of density information, and the size of the external archive was fixed and did not vary as in SPEA [90]. When the number of nondominated solutions exceed maximum and the archive is full, then a clustering technique was used to eliminate the less dominated members from the archive to maintain a fixed size. In SPEA2 [96, 97] only the members from the archive participate in the mating process. These techniques incorporated with SPEA have shown greater improvements in solving multiobjective problems. Laumanns et al. [98] combined the convergence and diversity in evolutionary algorithms to solve multiobjective optimization problems. Laumanns et al. in [98] pointed out the major problems that occur with multi-objective evolutionary algorithms (MOEAs) when trying to obtain true Pareto-optimal solutions. To overcome the drawback persisting with major MOEAs, Laumanns et al. in [98] proposed strategies for ϵ-dominance archiving. The ϵ is a positive integer and the value is application specific, and it further determines the maximal size of the archive. The update strategy in ϵ-dominance archive is as follows; the new solutions are accepted only if they are not ϵ-dominated by any other solution of the current archive. If a solution is accepted, all dominated solutions are removed. Parsopoulos and Vrahatis applied Particle Swarm Optimization (PSO) method on Multiobjective Optimization [99] in 2002. In [99] the weighted aggregation technique, fixed and adaptive weights were considered for solving MOUO problems. Furthermore PSO was hybridized with Genetic Algorithms to develop a multi-swarm PSO that can cope effectively with MO problems [99]. A Fast and Elitist Multiobjective Genetic Algorithm (NSGA-II) [100] was developed in 2002 by Deb et al. This algorithm [100] has become very popular and is widely used for solving MOUO problems. Essentially this [100] algorithm overcame the major drawback in NSGA [94], with the introduction of fast sorting method. The sorting method had a simple bubble sort mechanism and this was not called or invoked in every iteration as in NSGA. In NSGA-II, two entities were calculated viz. domination count (number of solutions which dominate the new solution) and a set of solutions that the solution dominates. This requires lesser comparisons

and the computational complexity is reduced by a great extent. NSGA-II [100] has used crowded-distance to maintain the diversity instead of a simple distance metric [94].

Yen and Lu developed dynamic multiobjective evolutionary algorithm (DMOEA) [101] in 2003. In DMOEA [101] a cell-based rank and density estimation strategy was incorporated for computing dominance and diversity of the dynamically varying population. Due to the dynamic strategy, the population could grow or shrink and this was decided by whether an individual would survive or be eliminated. In 2004, Coello et al. applied PSO for multiobjective optimization [102]. In this algorithm [102] an archive was used to maintain the non-dominated solutions found, and a mutation operator was used to maintain the population diversity. The members of the external archive were also used to guide the swarm further to find the solution and therefore increase the convergence speed. In [102] the faster convergence of PSO proved to be fatal in some cases while solving MOUO problems and hence a mutation operator was introduced. The mutation operator tries to explore all the particles in the beginning of the search process and it decreases with iteration. A Pareto-frontier differential evolution (PDE) algorithm was proposed by Sarker and Abbass [103] in 2004. In [103] Gaussian distribution was used for initializing population. In [103] reproduction in each generation was only among non-dominated solutions and only the dominating individuals were kept in the population. Though this approach seems to be simple and efficient, it has been tested only on simple MOUO problems.

Tan et al. proposed a distributed cooperative co-evolutionary algorithm (CCEA) [104] for multiobjective optimization in 2006. In CCEA a divide-and-conquer approach was used to decompose the solution into smaller components and this evolves multiple solutions in the form of cooperative sub populations. The CCEA maintains an external archive for Pareto fronts. Further it was extended with a distributed concept and proposed a distributed cooperative coevolutionary algorithm (DCCEA). The DCCEA [104] allows inter-communication of subpopulation that increases the computational speed. It was proved that DCCEA reduces the execution time without sacrificing the performance of CCEA [104]. Emmerich et al. proposed Gaussian Random Field Metamodels (GRFM) [105] an efficient search method for evolutionary algorithms (EA). The GRFM [105] predicts new solution by exploiting the previously recorded information and estimates the confidence of this prediction. This algorithm selects only the promising members

in every iteration and evaluations are carried out only for the selected members. This reduces the computational cost to a greater extent. In classical comprehensive learning PSO (CLPSO) [106] the velocity of a particle is updated on the basis of *gbest* of the swarm, *pbest* of the concerned particle, and the *pbests* of another particle chosen at random. Suganthan et al. in 2006 extended CLPSO to multiobjective optimization and called it the Multiobjective Comprehensive Learning PSO (MOCLPSO) [107]. MOCLPSO [107] incorporates non-dominated sorting, bounded external archive and updates particle positions and velocities as in CLPSO. The MOCLPSO uses a learning strategy whereby the historical best information of all other particles is used to update the particles velocity. This particular strategy enables the diversity of the swarm to be preserved and discourage premature convergence. This algorithm performs better than NSGA II for some testbench functions [107]. A remarkable work in EA with decomposition named as MultiObjective Evolutionary Algorithm based on Decomposition(MOEA/D) [108] was proposed by Zhang and Li in 2007. It was based on conventional aggregation approaches in which a MOUO problem was decomposed into a number of scalar objective optimization problems (SOPs). The objective of each SOP called a subproblem, was a weighted aggregation of the individual objectives. Neighborhood relations among these subproblems were defined based on the distances between their aggregation weight vectors. This has produced remarkable results in MOUO domain. A Time Variant Multi-Objective Particle Swarm Optimization (TV-MOPSO) [84] to solve MOUO problems was developed by Tripathi et al. in 2007. In TV-MOPSO [84] important parameters of PSO (inertia weight and acceleration coefficients) were made adaptive i.e. it changes with iteration. This adaptive nature of the parameters help the algorithm to explore the search space more efficiently. Further to improve the diversity in the Pareto-optimal solutions, two well known approaches; the hyper-grid (TV-MOPSO-H) and clustering (TV-MOPSO-C) were integrated with PSO. Sanchis et al. in 2008, proposed an integrated multiobjective optimization with priori preferences using genetic algorithms [109]. In this algorithm [109] user preferences (about the solution) are also taken into consideration in the search process. The preference functions were built to reflect the decision-makers (DM) interest with meaningful parameters for each objective. The preference functions are further solved by GA. A multiple swam algorithm was proposed in 2008 by Leong and Yen [110]. In this algorithm cell-based rank density estimation, population growing and declining strategies,

and adaptive local search, were incorporated. The dynamic population strategy within the multiple-swarm MOPSO was proposed instead of fixed population. Additionally this algorithm [110] includes an adaptive local archive to improve the diversity within each swarm. This algorithm has shown competitive results with improved diversity and convergence at less computational cost. Gong et al. in 2008 proposed a non-dominated neighbor immune algorithm (NNIA) for multiobjective optimization [111]. The selection strategy emphasizes on less crowded solutions. The NNIA [111] uses an external archive to store nondominated individuals. Some of the individuals from the archive that was less-crowded were used for further cloning, recombination, and mutation. This algorithm [111] paid more attention to the less-crowded regions and used proportional cloning, recombination and hyper-mutation operators it generate new solutions. An efficient nondominated sorting method was incorporated with evolutionary algorithms for multiobjective optimization by Fang et al. in 2008 [112]. Essentially this [112] improves the sorting method found in NSGA-II [100]. In order to reduce the time complexity to generate non-dominated fronts, [112] uses a new data structure called the dominance tree and a divide-and-conquer mechanism. The non-dominated sorting method presented in [112] claims to be faster than NSGA-II [100]. In 2009 Rachmawati and Srinivasan proposed a preference based MOEA to find the knee region in the Pareto Front. In this algorithm a preference based focus was achieved by optimizing a set of linear weighted sums of the original objectives and the weights were carefully selected based on a user-specified parameter. Differential Evolution (DE) is one of the popular evolutionary algorithms for the optimization oof a problem. In DE, a trial vector is generated using a predefined strategy; the parameters of the vector are tuned by a trial and error method [113]. Self adaptive DE avoids tuning of parameters by trial and error method [114] and thus MultiObjective Self adaptive Differential Evolution (MOSaDE) was proposed as an extension of self adaptive DE for multiobjective optimization problems [115] in 2009 by Huang et al. Zhao and Suganthan [116] in 2010 proposed Multi-objective evolutionary algorithm with an ensemble of external archives. This algorithm [116] essentially removes the tuning difficultly of ϵ parameter in ϵ - dominance sorting. The well proven sorting method in the literature of MOUO is ϵ - dominance sorting, but the major drawback lies in the selection of the ϵ parameter. Choosing the optimum value of ϵ to maintain diversity in a population is difficult, as it is problem dependant. This difficulty was removed in [116] with the integration of the en-

semble of external archives and the ensemble of ϵ parameter with PSO. Wang et al. proposed a self-adaptive differential evolution with elitist archive and crowding entropy based diversity measure for MOUO problems [117]. In this approach, a crowding entropy based diversity measure is applied to maintain an elitist archive. There is plenty of literature available and interested readers can refer the works proposed in [118, 119, 120, 121, 122, 77, 123, 124, 125, 126, 127, 128, 129, 130, 75, 131, 83, 132, 133]. Though several algorithms have been reported, there are persisting issues with MOUO problems, as discussed in following section.

4.3 Challenges in Multi Objective Unconstrained Optimization

The presence of multiple (often conflicting) objectives pose challenges to the optimization algorithms. The major challenges in solving Multi Objective Unconstrained Optimization problems are as follows:

1. MOUO problems have multiple objectives; the objectives could be conflicting with each other and hence difficult to optimize.

2. The multiple objectives may be linear or non-linear; this makes the problem more complex.

3. Often the search space available for these kind of problems may be complex, and the the feasible (physically realizable) region in the search space may be small.

4. Since in most of the cases the objectives are non differentiable and non-continuous, it is impossible to solve by point based algorithms.

5. Due to complex solution space the problems could be multimodal, and the algorithms based on initial solution may be stuck up local minima and get stagnated.

6. PSO neither internally nor externally have the mechanism to optimize multiple objectives of MOUO.

7. PSO has the perennial problem of premature convergence and is more pronounced in complex, multimodal and conflicting problems.

4.4 Research Contribution

MOUO problems usually generate multiple solutions, the conventional optimizing algorithms fail to optimize MOUO problems but population based stochastic algorithms (evolutionary algorithms and swarm intelligence) are well suited for solving MOUO problems. The population based algorithms have their own limitations, as these algorithms don't have built in mechanisms to handle multiple objectives simultaneously. Many of these algorithms focus on the tuning and ordering of the Pareto optimal fronts. The PSO is no exception, as it also does not have internal or external mechanism to handle MOUO problems. Basic PSO suffers with the perennial problem of premature convergence while solving complex objectives and thus is a more serious concern for MOUO problems. Not much attention has been paid towards applying improved variants of PSO for MOUO problems [131]. Sabat et al. proposed an Accelerated Exploration Particle Swarm Optimizer (AAEPSO) [35] for complex single objective problems. The key feature of this algorithm is to identify the particles which are responsible for premature convergence and accelerate them in the direction of the best particle in the swarm (*i.e.* the global solution). This helps the particles to come out of the deep local optima and become available to search for better solution again. The successful and promising results of AAEPSO [35] motivated to apply this on MOUO problems. Hence Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (MOAAEPSO) is proposed.

4.4.1 Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer

The Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (MOAAEPSO) is an improved variant of AAEPSO [35] for solving MOUO problems. MOAAEPSO applies the adaptive acceleration and exploration found in [35] along with non dominated sorting and external archive for solving MOUO problems. The implementation concept of the proposed algorithm i.e. *Selection*, *Acceleration*, *Exploration* and *Adaptive Acceleration and Exploration* are almost similar to the AAEPSO [35] as explained in the Chapter 2 with the major addition of Pareto optimality and external archive. The MOUO problems can not be explained with a single solution as these involve multiple objectives and hence

need multiple solutions. The multiple solutions are the tradeoffs and can best be explained with Pareto optimality. The multiple solutions (Pareto optimal) produced during search process are stored in the external archive. The detailed MOAAEPSO steps are explained in Algorithm 4.4.1.

Algorithm 4.4.1 Pseudo code of MOAAEPSO

Initialization
Initialize the swarm of size NP:
Initialize position $'X'$ and velocity $'V'$ of the particles randomly in D -dimensional search range (X_{max}, X_{min}).
Initialize Selection factor S_f for dominated particles with random permutation in [10-90].
Evaluate the fitness values of all particles.
Set the current position as *pbest* and the particle with most nondominated as *gbest*.
Initialize the bounded Archive to size of population
Set $count_{A_f} = 0$
Optimize
for $t \leftarrow 1$, $Maxgen$ **do**
$w^t, c_1^t, c_2^t = w_{max} - ((w_{max} - w_{min})/Maxgen) * t \;\; Where \;\; w_{max} = 0.9, w_{min} = 0.2$
Decrease AE_f exponentially as per equation (2.13)
Update velocity (equation 2.2) and position (equation 2.3)
Evaluate fitness function for each particle.
Update *pbest*: If current fitness dominates the previous then set current position as *pbest* else retain *pbest*.
Update *gbest*: If best of all the current *pbest* dominates the previous *gbest* then set best *pbest* as *gbest* else retain *gbest*.
Find the Pareto-optimal solutions.
Check the domination status of new solution with the solutions in the archive.
  if X dominates any member of Archive **then**
    Delete the dominated member and insert X in the Archive
  else if X is neither dominated by the members of Archive or the members by X **then**
    Insert X in the Archive
  else if X is dominated by the members of Archive **then**
    reject X
  end if
  if Size of Archive exceeds maximum size **then**
    use crowding distance to eliminate less dominated particles.
  end if
  if Mod(t,10)=1 **then**
    Find distance of each particle from *gbest*.
    Sort the particles based on their dominance and distance from *gbest*.
    Select S_f number of diverged particles from sorted particles.
    Accelerate these selected particles around nondominated using AE_f as
    $count_{A_f} = count_{A_f} + 1$
    a=gbest - $A_f(count_{A_f})$ and b=gbest + $A_f(count_{A_f})$
      for $i \leftarrow 1$, S_f **do**
        for $d \leftarrow 1$, D **do**
          $X_{i,d} = a_d + (b_d - a_d) * rand$
        end for
      end for
  end if
end for t
continue optimizing until stopping criteria or exceeding maximum iteration
Report results
Terminate

4.4.2 Implementation details of MOAAEPSO Algorithm

The Algorithm 4.4.1 shows all necessary steps for the practical implementation of MOAAEPSO algorithm. The steps of MOAAEPSO are almost similar to

AAEPSO algorithm [35], with major additions of Pareto optimality and bounded archive for multiobjective optimization. The process of diverged particle selection and adaptive acceleration exploration is similar to AAEPSO except with minor difference relating to MOUO concept. The difference is, while solving MOUO problem, there will be multiple points produced, all may be equally good and hence can be explained well with help of Pareto optimality (a trade off) as is evident from the Figure 4.1. Thus selection and updating of *pbest* and *gbest* in MOAAEPSO is not the same as in AAEPSO. In MOAAEPSO, *pbest* and *gbest* are selected and updated based on dominance nature. The most nondominated solution at a particular iteration is treated as *gbest* and similarly *pbest*. Here after whenever *pbest* and *gbest* are referred in MOUO, they refer to nondominated solutions. In Algorithm 4.4.1 the population of size NP is initialized in the search space. The fitness of whole population is evaluated and the current position is set as *pbest* and best nondominated particle in swarm is set as *gbest*. The selection factor S_f is initialized, this factor is responsible for selecting the dominated (diverged) particles. The acceleration factor AE_f is decreased exponentially with iterations linearly. The velocity and position of all the particles are evaluated and the fitness of the new solution is evaluated. Based on the dominance of the solution, the *pbest* and *gbest* are updated. If current fitness dominates the previous then the current position is set as *pbest* else the previous *pbest* is retained. Similarly if the best of all the current *pbest* dominates the previous *gbest* then best of *pbest* is set as *gbest* else previous *gbest* is retained. MOAAEPSO implements bounded external archive for storing nondominated solutions [107]. The update strategy of the archive is as follows. If new solution dominates the member of the archive then dominated member from the archive is deleted and the new solution is inserted in the archive. The new solution will enter the archive if neither this dominates members of the archive nor members of the archive dominates the new solution. The new solution will be rejected if it is dominated by all the members of the archive. If the archive size exceeds the predefined limit (here size of the archive is same as the population size) then crowding distance is used to eliminate the less dominated members from the archive. The adaptive acceleration and exploration of diverged (dominated) particles are carried out around the nondominated solution. This process is carried out for every 10 iterations (refresh rate). This refresh rate of 10 prevents particles from collapse and stagnation. The process of acceleration can best be explained with the help of Figure 4.2. This figure shows

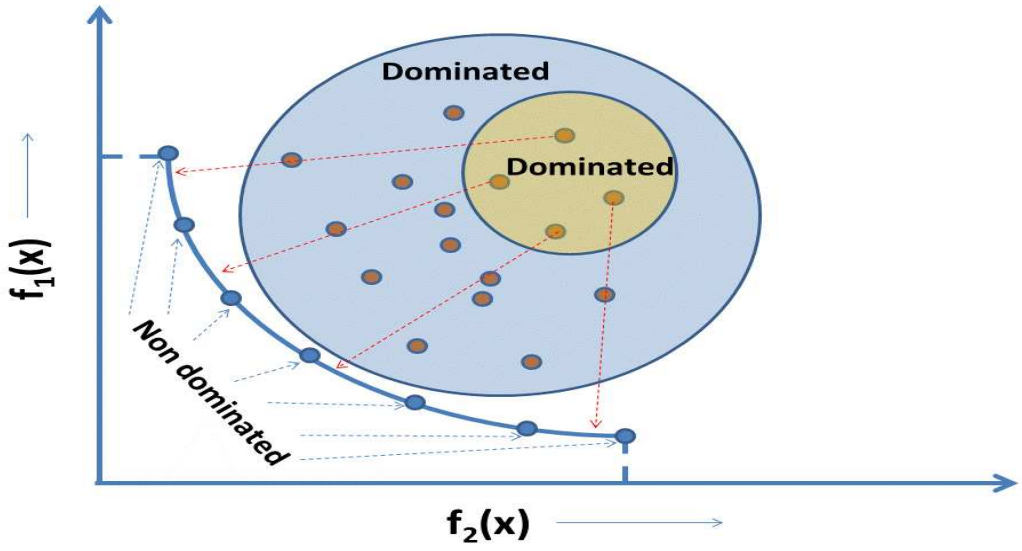

Figure 4.2: MOUO Acceleration process

the Nondominated and Dominated points (particles); the light blue circle shows all the dominated points and light yellow circle show only selected (dominated) points for further acceleration. The few selected (dominated) points are forced to come closer and search for a better solution near the nondominated solutions.

4.5 Simulation

The simulations were carried out in Windows XP Operating System, on Pentium IV 2.6GHz with 512MB of RAM. The algorithm is implemented using Matlab 7.2 programming language. The population size and the maximum iterations considered for all the algorithms are 100 and 1000 respectively. The population is initialized in the search range $[X_{min}, X_{max}]$, where X_{max} and X_{min} are the maximum value and minimum value in the search range. The velocity is initialized with 20 percent of the search range i.e. $V_{max} = 0.20 * (X_{max} - X_{min})$. Simulation results are for an average of 25 runs. The performance comparisons of the proposed algorithm MOAAEPSO is carried out with MOSaDE [115] and MOCLPSO [107] over seventeen different benchmark functions such as KUR, FON, SCH, ZDT1, ZDT2, ZDT3, ZDT4 and UF1 to UF10 [134]. These functions have different characteristics in terms of convexity, discontinuity and nonuniformity [134]. The

detailed definition of these functions are available in Appendix C. The algorithmic parameters of the MOSaDE considered are [115]. The algorithmic parameters for MOCLPSO are: Population size (NP) = 100, archive size (A) = 100, learning probability (P_c) = 0.1, and elitism probability (P_m) = 0.4 [107]. The parameters used in the proposed algorithm are selected carefully after several trials to obtain satisfactory results. Further the algorithms are validated using different performance metrices namely pareto front graphs, inverted generational distance, spacing, diversity and convergence metrics while solving benchmark functions. The statistical significance of the algorithms are also analyzed using hypervolume indicator and ANOVA comparison test.

4.5.1 Performance metrices

For measuring the performance of multiobjective optimization algorithms, visual comparisons were common in the infancy of evolutionary algorithms. In case of multiobjective optimization, we aim at approximating a set of Pareto-optima solutions which are a set of trade-offs instead of a single solution. There are mainly two techniques being used for performance measurement of multiobjective optimization algorithms; 1) attainment function approach which models the set of trade off solutions as a probability density functions in the objective space, and 2) other is indicator approach which summarizes the outcome of the algorithm as a quantitative performance and performs statistical analysis on the corresponding measures. There are two sets of performance assessment metrices: one is Non Pareto compliant and other is Pareto compliant. The most commonly used Non Pareto compliant metrices include inverted generational distance, spacing, diversity and convergence metrics [107]. The Pareto compliant indicator includes Hypervolume indicator. Further the comparisons are done with nonparametric test namely ANOVA test. The detailed descriptions of these metrices are presented in below subsections. Non Pareto compliant metrices are described under common name as conventional metrices and Pareto compliant metrices as statistical metrices..

4.5.1.1 Conventional metrices

The most commonly used conventional performance metrices often called Non Pareto compliant metrices are discussed here.

Inverted Generational Distance Metric (IGD:) IGD [107] is used to estimate the closeness of elements in an approximated Pareto front with respect to the elements of true Pareto front. It is defined as

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{|n^*|} \tag{4.2}$$

where $|n^*|$ is set of uniformly distributed points along the true Pareto front and d_i is the Euclidean distance between the elements in Pareto front found so far by the algorithm (approximated Pareto front) and its nearest neighbor in the known true Pareto front. A smaller IGD value indicates better performance.

Spacing (S): Spacing [107] measures the range variance of the neighboring solutions in the Pareto front obtained from the algorithm. It is defined as:

$$\mathbf{S} = \sqrt{\frac{1}{(n-1)} \sum_{i=1}^{n} (\overline{d} - d_i)^2}$$

where n is the number of non dominated solutions generated by the algorithm and

$$d_i = \min_j \left(\sum_{k=1}^{m} |f_m^i(\overrightarrow{x}) - f_m^j(\overrightarrow{x})| \right) \;\; i,\; j \;=\; 1,\; 2, \cdots, n$$

and $\overline{d}$ is the mean distance of all d_i. m is the number of objectives. A zero value for this metric indicates that all elements of Pareto fronts are equi-spaced.

Convergence Metric (CM): CM [107] measures the extent of convergence of Pareto solutions to the known optimal Pareto front. It is defined as

$$\mathbf{CM} = \frac{1}{n} \sum_{i}^{n} d_i \tag{4.3}$$

where n is the number of non dominated solutions obtained by the algorithm and d_i is the Euclidean distance (in objective space) between the i^{th} non dominated solution and the nearest member of the known Pareto optimal front. A smaller value of CM denotes a better convergence performance.

Spread or Diversity Metric (DM): DM [107] measures the extent of

spread among the solutions. It is defined as

$$\mathbf{DM} = \frac{\sum_{m=1}^{M} d_m^e + \sum_{i=1}^{n-1} |d_i - \overline{d}|}{\sum_{m=1}^{M} d_m^e + (n-1)\overline{d}} \tag{4.4}$$

where d_m^e is the Euclidean distance between the two extreme solutions of Pareto optimal front and the boundary solutions of the nondominated solutions corresponding to m^{th} objective function. d_i is the Euclidean distance between i^{th} nondominated solution and corresponding true Pareto front. $\overline{d}$ is the mean value of all d_i. Zero value indicates an ideal distribution. Smaller DM indicates better diversity of the non dominated set.

Pareto front graphs: Pareto front graphs are pictorial way of analyzing how well the algorithms obtain Pareto optimal solutions. Since MOUO problems usually consists of multiple solutions and hence trade off solutions are obtained. These trade off solutions can visually be seen by means of Pareto front graphs.

4.5.1.2 Statistical metrices

Following are the two performance metrices used for statistical comparisons. Hypervolume indicator is a performance measure under the category of Pareto compliance and ANOVA test under the category of nonparametric statistical test.

Hypervolume Indicator (I_H): Hypervolume indicator is one of the indicator based approaches for performance assessment of multiobjective optimization algorithms [135]. This indicator measures the hypervolume of that portion of objective space which is weakly dominated by an approximation set A and is to be maximized. In order to measure this quantity, the objective space must be bounded if it is not, then a bounding reference point that is dominated by all point should be used. The hypervolume indicator has a desirable property : whenever A dominates B the $I_H(A) > I_H(B)$ provided that the bounding point is strictly dominated by all the points in A and B. In this work hypervolume indicator is evaluated using the tool [136].

ANOVA test: Parametric tests have been commonly used in the analysis of experiments in computational intelligence. For example, a common way to test whether the difference between the results of two algorithms is non-random is to compute a paired t-test, which checks whether the average difference in their performance over the problems is significantly different from zero. When comparing

a set of multiple algorithms, the common statistical method for testing the differences between more than two related sample average is the repeated-measures ANalysis Of VAriance (ANOVA). The ANOVA is a collection of statistical models, and their associated procedures, that provides a statistical test of whether or not the average results of several groups are all equal, and therefore generalizes t−test to more than two groups [41].

4.6 Results and Discussions

Comprehensive analysis of proposed algorithm with the state of the art is carried out with different metrices separately for each metric as follows.

4.6.1 Pareto front graphs

Figure 4.3 - Figure 4.19 presents the approximated Pareto fronts obtained using the algorithm for seventeen benchmark functions which includes ten benchmark functions UF1-UF10 of CEC 2009 [134]. Each figure has three parts demonstrating the results of MOSaDE, MOCLPSO and the proposed MOAAEPSO algorithm respectively. Symbol $'o'$ represents the true Pareto front obtained analytically; $'*'$ denotes approximated Pareto front obtained by the algorithm. Figure 4.3 presents the Pareto fronts obtained from three algorithms for KUR function. KUR function is characterized by a tradeoff between convexity and discontinuity. The values of decision variables are difficult to find in the solution space as the optimal Pareto front is not continuous. Figure 4.3 clearly shows that proposed algorithm is able to find Pareto solutions accurately. Figure 4.4 compares the Pareto fronts obtained from the three algorithms on FON function. FON function is characterized by a tradeoff between nonconvexity and nonlinearity. From the plot, it is clearly evident that MOAAEPSO performance is comparable to MOCLPSO and MOSaDE. Figure 4.5 presents the comparative Pareto fronts obtained for SCH function. SCH function is a single variable bi-objective function having convex Pareto front. For this function, all three algorithms give equally good and similar to analytical Pareto front. Figure 4.6 presents the Pareto fronts on ZDT1 function which is a convex problem. It can be observed that both MOCLPSO and MOAAEPSO are able to find solutions same as analytical Pareto front. Figure 4.7 and Figure 4.8 present the Pareto fronts of ZDT2 and ZDT3 functions respectively. ZDT2

is a non convex continuous function whereas ZDT3 is a discontinuous function. For both these functions, MOAAEPSO gives better front compared to other algorithms. Figure 4.9 presents the Pareto front obtained from MOSaDE, MOCLPSO and MOAAEPSO algorithm on ZDT4 function. This function deal with multimodal landscape and it has 219 local fronts. MOSaDE fails to find global Pareto front while MOAAEPSO and MOCLPSO are able to provide global Pareto front. The approximated Pareto fronts on UF1 - UF9 CEC 2009 benchmark functions using MOSaDE, MOCLPSO and MOAAEPSO are shown in figures from Figure 4.10 to Figure 4.19. From these figures it is evident that MOAAEPSO finds approximated front close to true Pareto front as compared to MOCLPSO and MOSaDE.

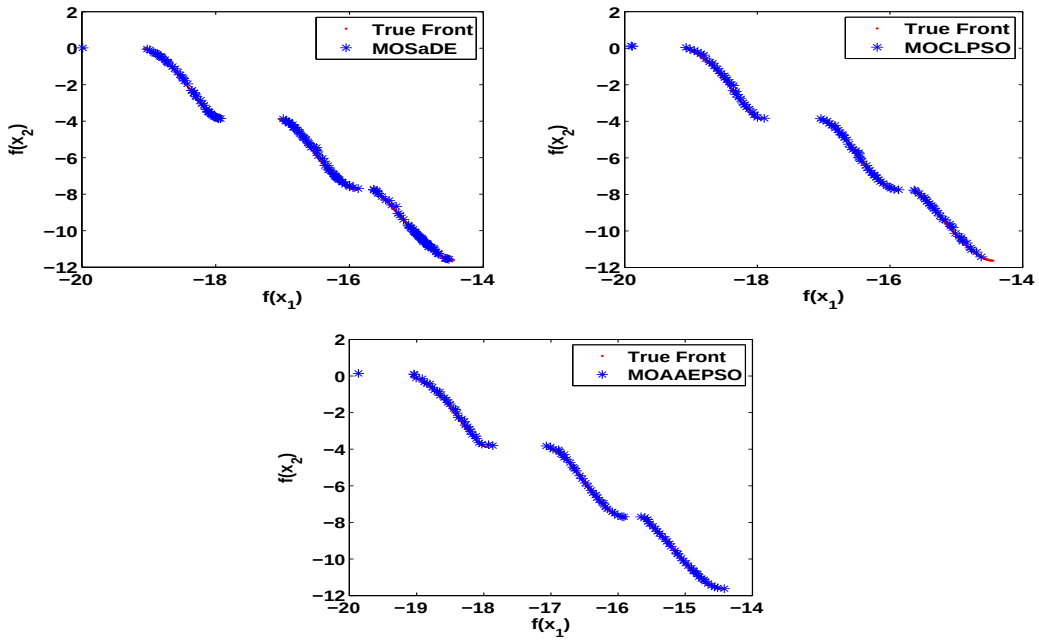

Figure 4.3: Pareto fronts on KUR

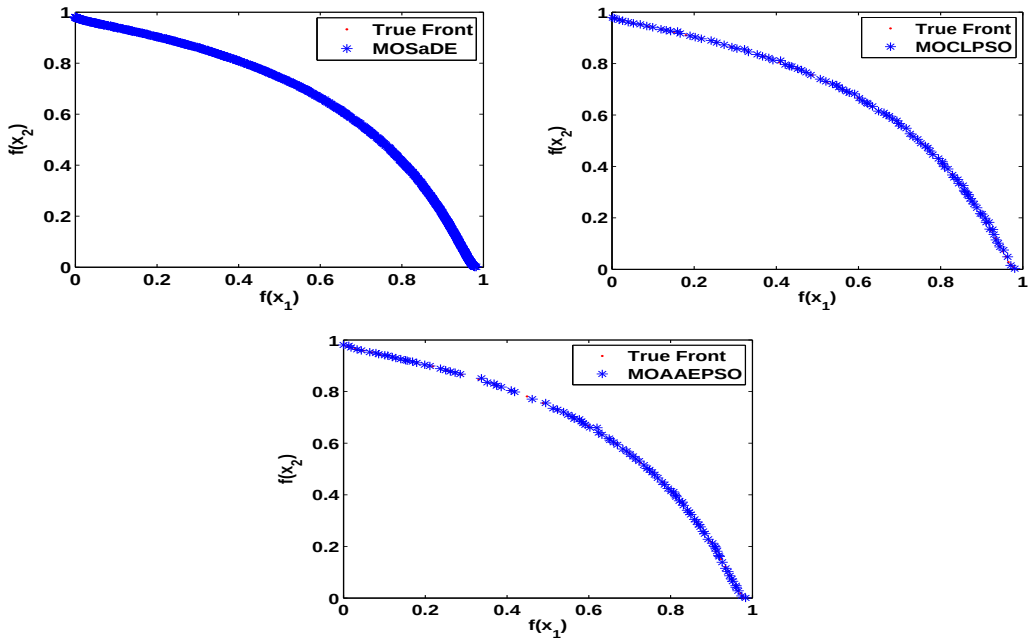

Figure 4.4: Pareto fronts on FON

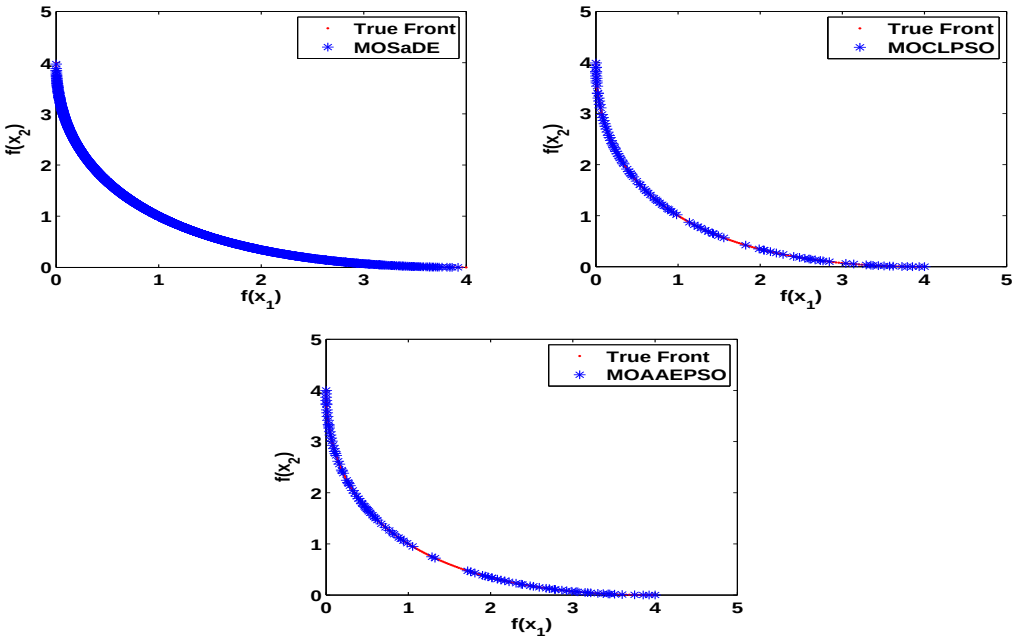

Figure 4.5: Pareto fronts on SCH

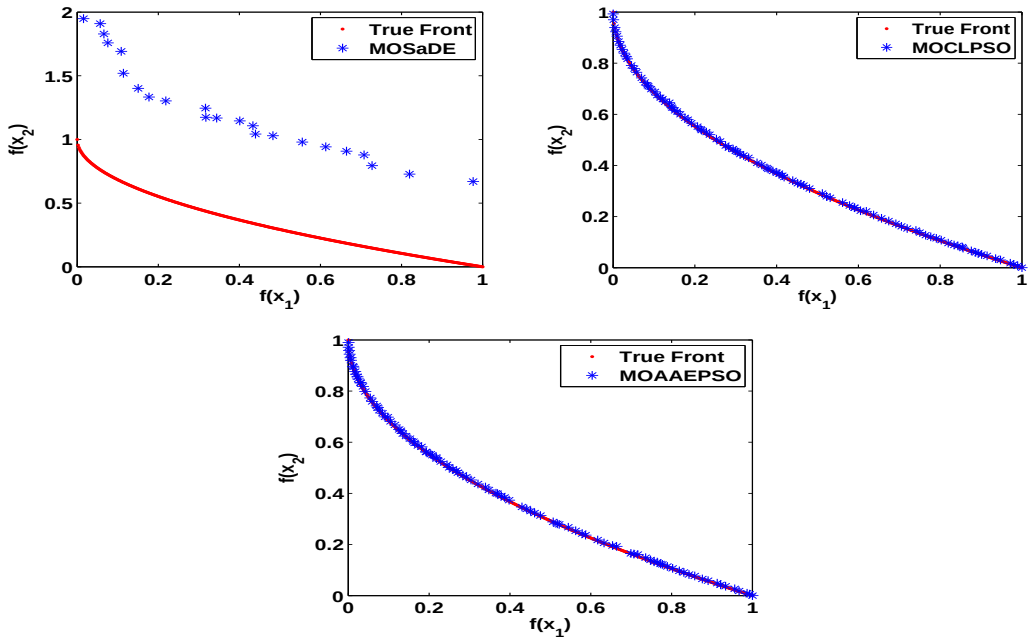

Figure 4.6: Pareto fronts on ZDT1

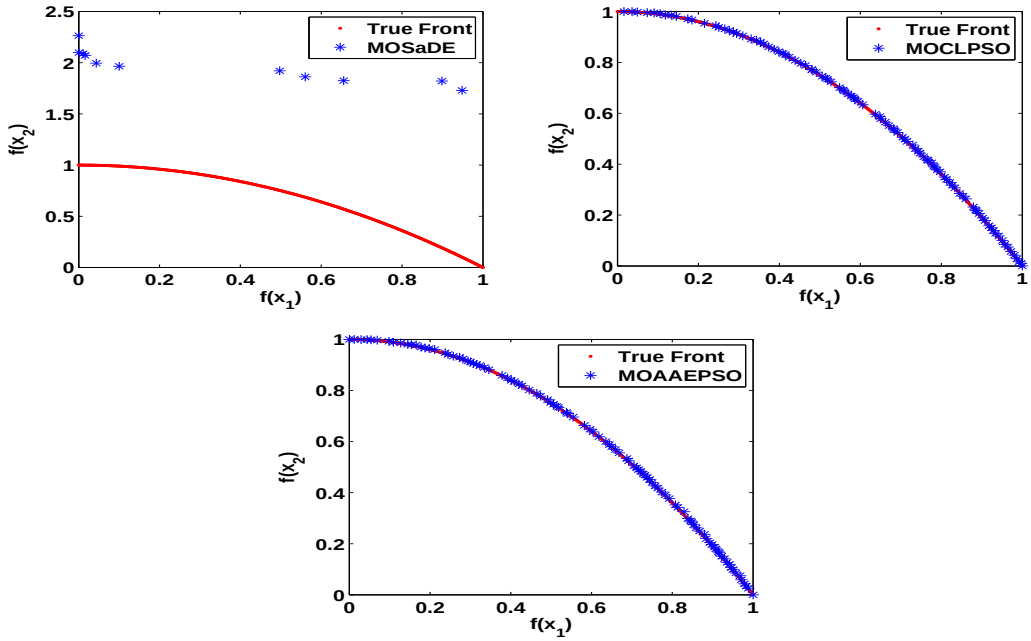

Figure 4.7: Pareto fronts on ZDT2

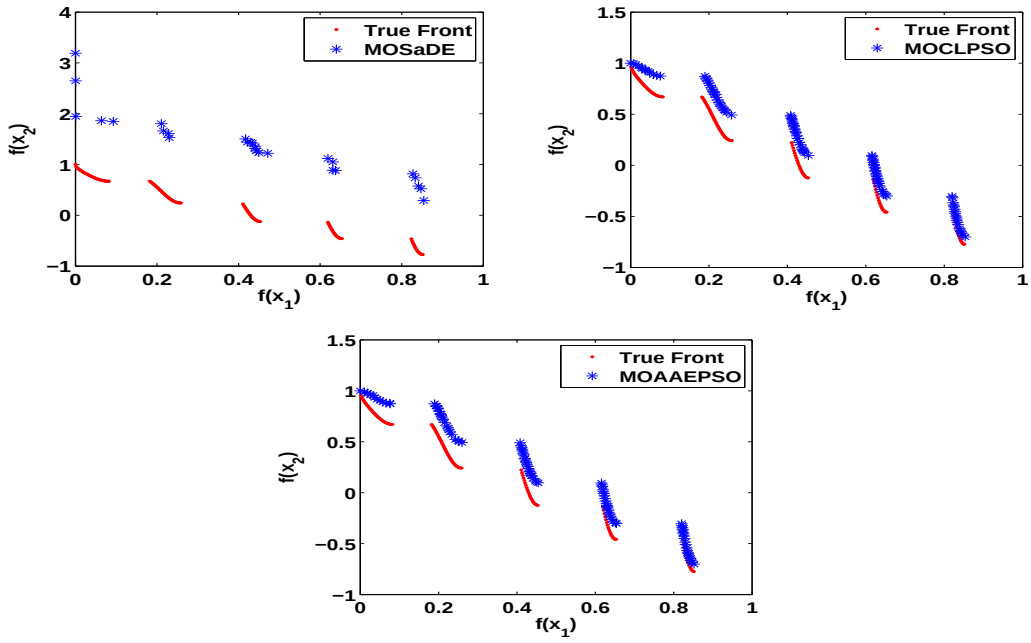

Figure 4.8: Pareto fronts on ZDT3

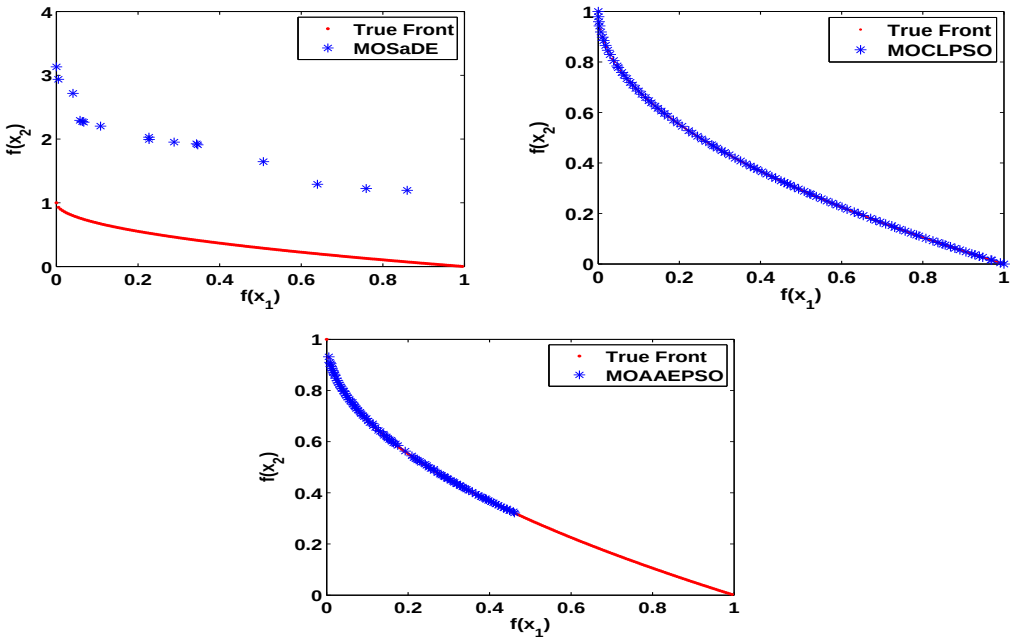

Figure 4.9: Pareto fronts on ZDT4

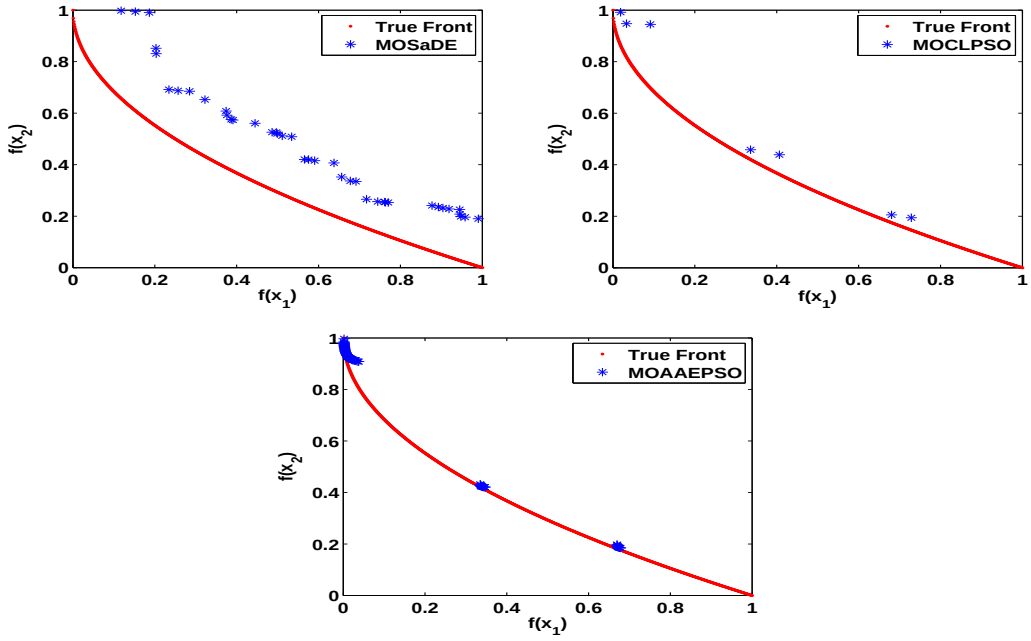

Figure 4.10: Pareto fronts on UF1

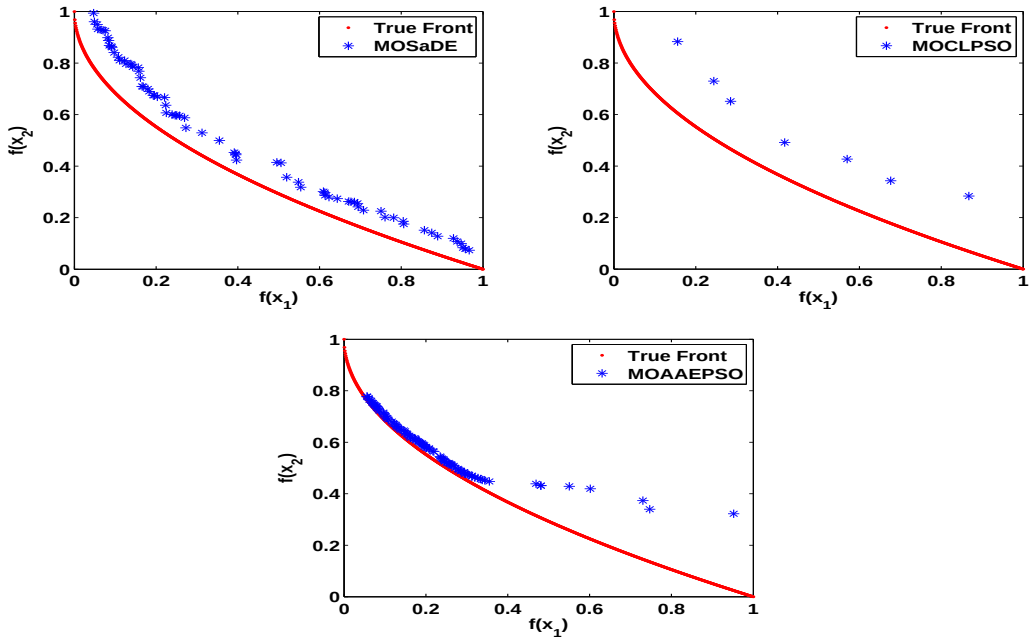

Figure 4.11: Pareto fronts on UF2

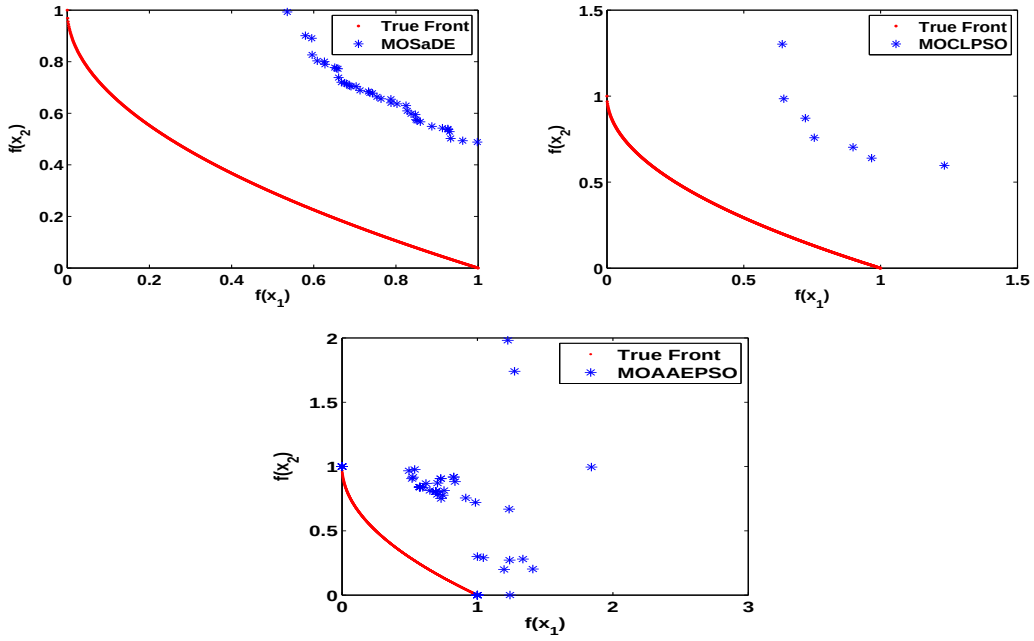

Figure 4.12: Pareto fronts on UF3

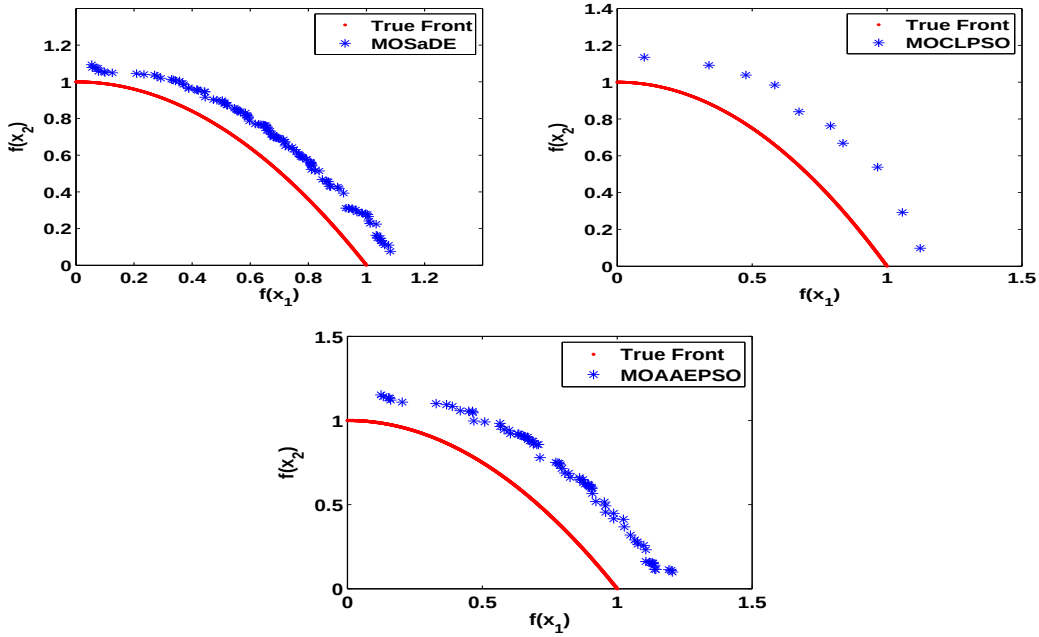

Figure 4.13: Pareto fronts on UF4

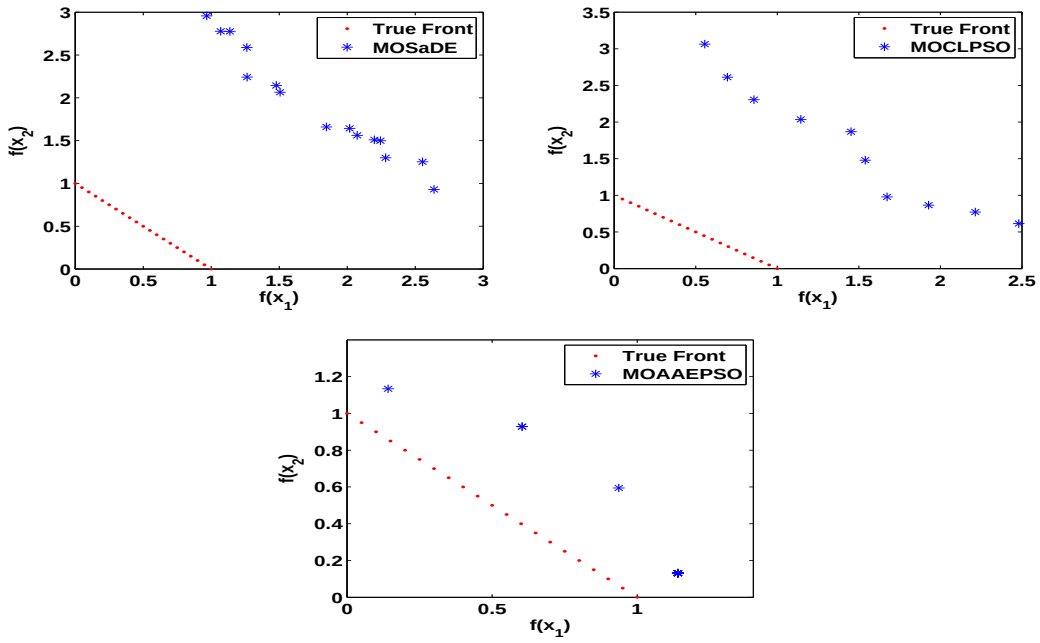

Figure 4.14: Pareto fronts on UF5

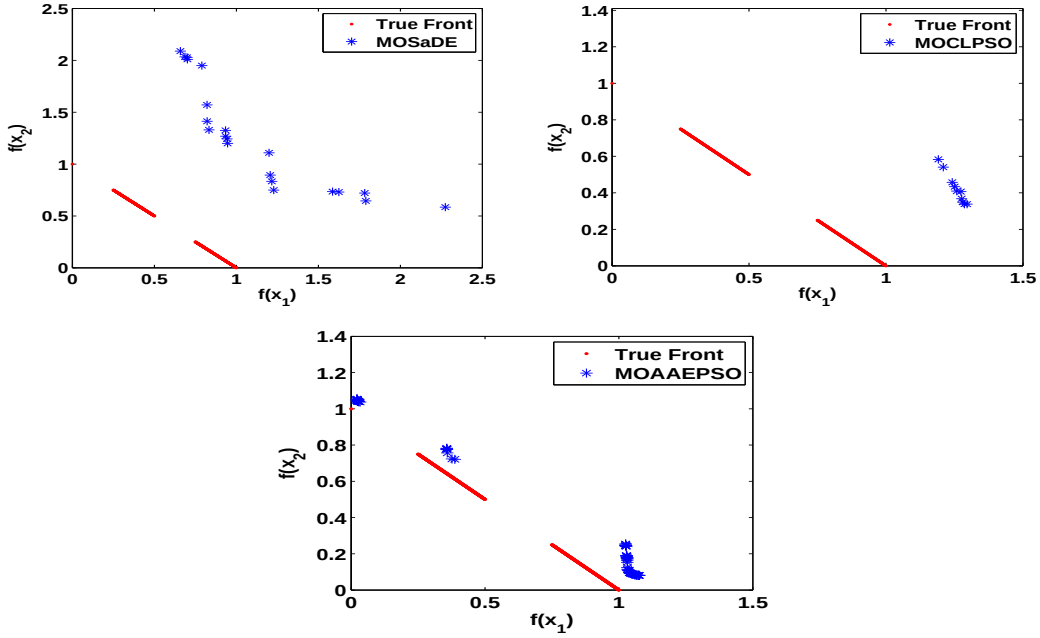

Figure 4.15: Pareto fronts on UF6

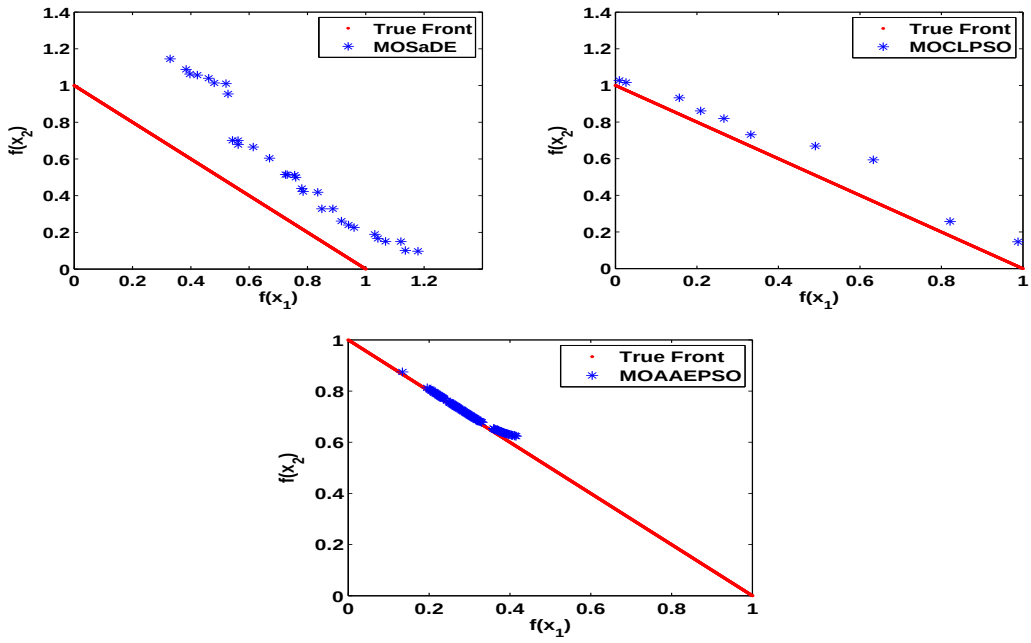

Figure 4.16: Pareto fronts on UF7

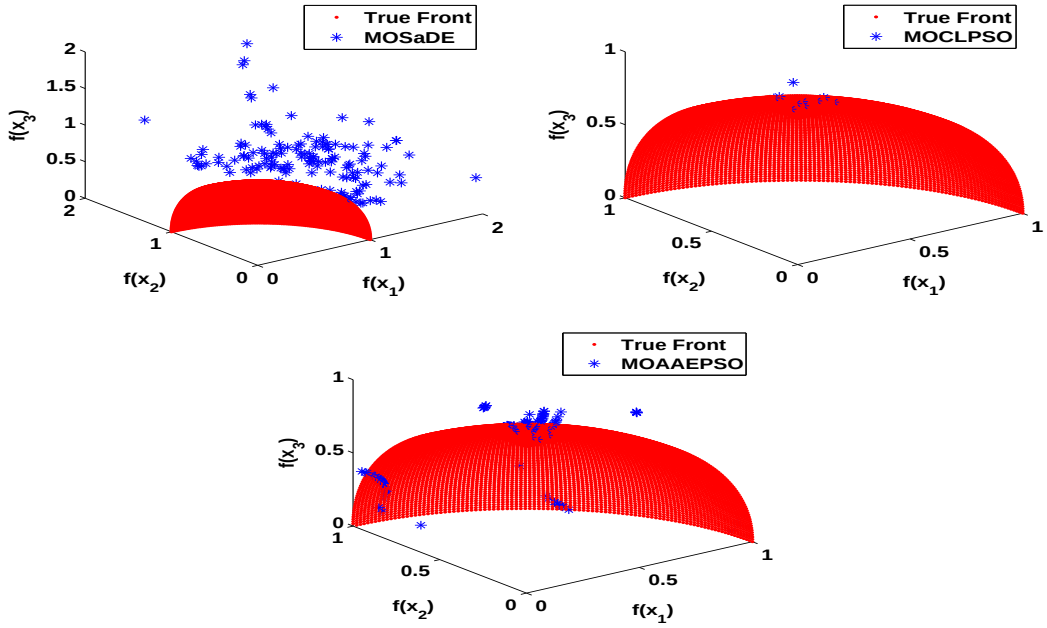

Figure 4.17: Pareto fronts on UF8

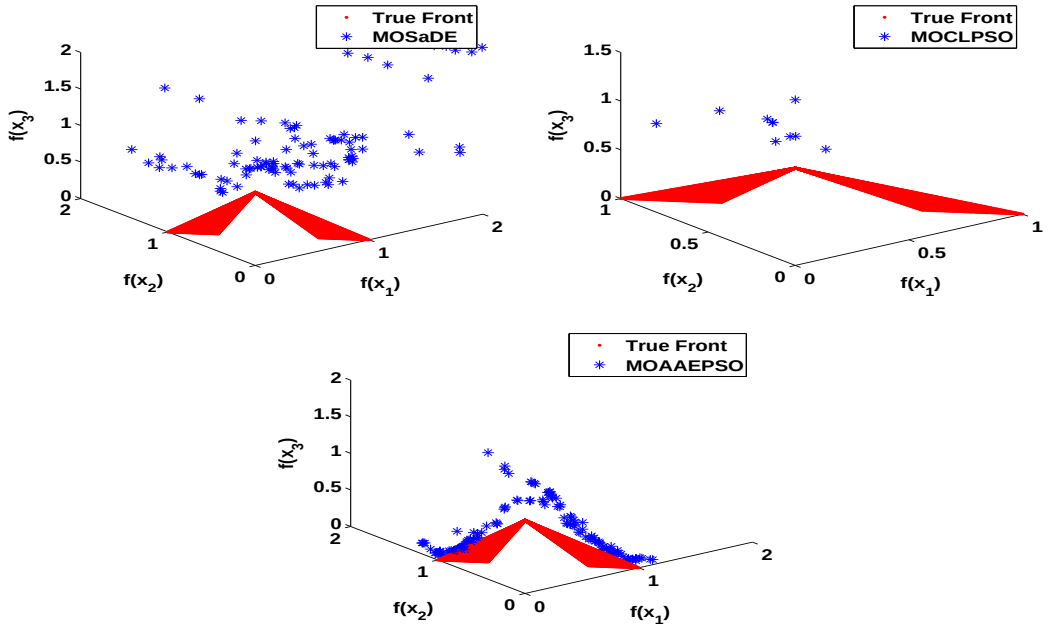

Figure 4.18: Pareto fronts on UF9

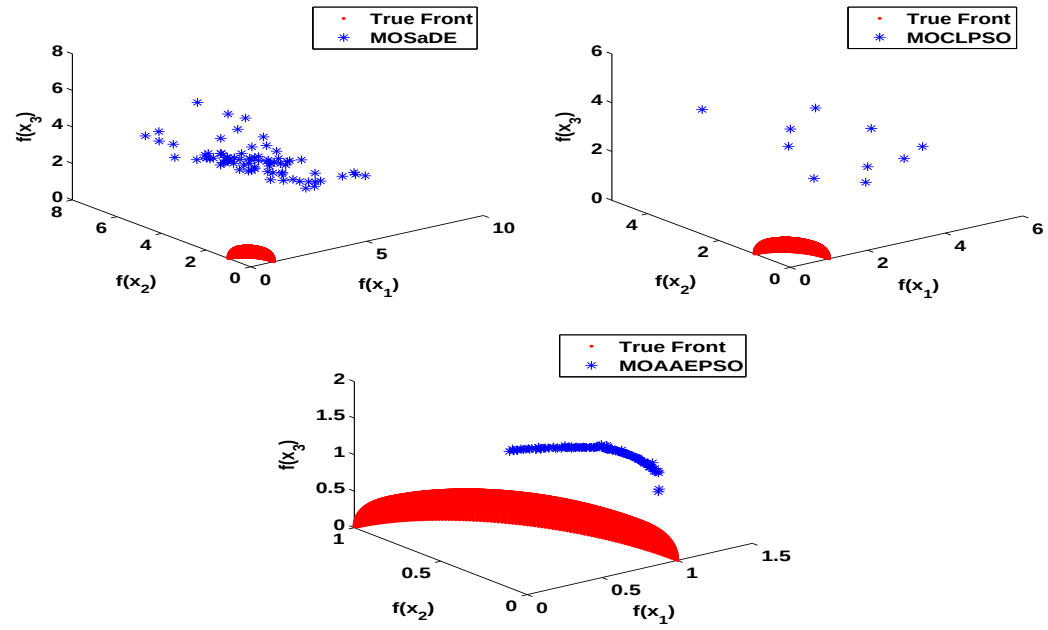

Figure 4.19: Pareto fronts on UF10

4.6.2 Average numerical results

The average of numerical results in terms of IGD, Spacing (S), Diversity (DM) and Convergence metrics (CM) are presented in Table 4.1 and Table 4.2. The mean and standard deviations (inside parenthesis) of IGD, Spacing, CM and DM on KUR, FON, SCH and ZDT1 - ZDT4 functions are tabulated in Table 4.1. Similarly Table 4.2 presents the mean and standard deviations (inside parenthesis) of IGD, Spacing, CM and DM on UF1 to UF10 functions. The proposed algorithm MOAAEPSO is better in terms of IGD, CM, Spacing, DM and robustness as compared to MOSaDE and MOCLPSO algorithms on KUR function, as evident from Table 4.1.

Table 4.1: Results for MOUO functions: mean(std)

KUR	IGD	Convergence Metric	Spacing	Diversity Metric
MOAAEPSO	**1.09e-3**(**2.19e-4**)	**1.63e-2**(**1.33e-3**)	**3.75e-2 4.87e-3**)	**1.32**(3.73e-4)
MOCLPSO	2.89e-3(1.81e-3)	4.08e-2(1.95e-2)	4.14e-2(9.74e-3)	1.33(1.83e-4)
MOSaDE	2.09e-3(2.17e-3)	4.98e-2(5.79e-3)	8.85e-2(1.31e-2)	1.32(**2.30e-5**)
FON				
MOAAEPSO	**1.96e-3**(1.63e-4)	**1.10e-2**(8.70e-4)	**1.28e-2**(1.09e-3)	**7.94e-1**(**2.36e-4**)
MOCLPSO	2.08e-3(1.97e-4)	1.13e-2(1.24e-3)	1.36e-2(1.75e-3)	7.94e-1(6.94e-5)
MOSaDE	2.09e-3(**1.09e-005**)	1.24e-2(**8.49e-005**)	9.77e-2(**5.84e-005**)	2.61(1.48e-002)
SCH				
MOAAEPSO	**8.67e-4**(**4.96e-5**)	**9.71e-4**(**3.35e-4**)	**1.28e-3**(1.10e-3)	**3.70e-1**(2.26e-4)
MOCLPSO	9.05e-4(5.07e-5)	1.01e-2(5.99e-4)	1.31e-2(**5.77e-4**)	3.73e-1(**1.44e-4**)
MOSaDE	6.35e-3(1.39e-4)	2.84e-3(3.06e-4)	3.71e-3(5.90e-4)	2.46(4.33e-2)
ZDT1				
MOAAEPSO	**4.69e-5**(**8.83e-7**)	**8.62e-4**(2.10e-5)	1.24e-3(**3.68e-5**)	**2.74e-1**(**2.67e-4**)
MOCLPSO	5.37e-5(9.02e-7)	1.07e-3(**1.99e-5**)	**1.14e-3**(6.56e-5)	2.78e-1(4.22e-4)
MOSaDE	2.30e-2(2.19e-3)	5.12e-1(4.85e-2)	1.27e-1(9.86e-2)	7.51e-1(4.21e-2)
ZDT2				
MOAAEPSO	7.71e-2(1.08e-1)	1.72(2.41)	8.21e-3(4.71e-4)	8.80e-1(5.41e-2)
MOCLPSO	**5.20e-5**(**1.79e-6**)	**1.00e-3**(**3.63e-5**)	**1.21e-3**(**4.69e-5**)	**2.26e-1**(**8.20e-6**)
MOSaDE	4.77e-2(3.33e-3)	1.05(7.38e-2)	1.28e-1(9.91e-2)	7.53e-1(2.80e-2)
ZDT3				
MOAAEPSO	**3.92e-4**(2.71e-5)	**3.64e-3**(2.97e-4)	5.17e-3(8.06e-4)	6.89e-1(9.89e-5)
MOCLPSO	4.00e-4(**1.88e-5**)	3.88e-3(**1.51e-4**)	**4.84e-3**(**4.05e-4**)	**6.88e-1**(**4.86e-5**)
MOSaDE	5.77e-2(3.30e-3)	6.51e-1(3.39e-2)	9.45e-2(3.52e-2)	8.62e-1(1.06e-2)
ZDT4				
MOAAEPSO	**6.63e-3**(**2.03e-2**)	**8.59e-2**(**2.64e-2**)	**2.73e-1**(**8.51e-1**)	1.22(5.26e-2)
MOCLPSO	4.15e-2(5.24e-2)	5.54e-1(7.26e-1)	8.93e-1(8.54e-1)	1.22(**1.65e-2**)
MOSaDE	6.58e-2(9.10e-3)	9.23e-1(1.26e-1)	2.21(2.47)	**1.15**(4.38e-2)

Only MOSaDE shows robustness for DM. Table 4.1 also reveals that the MOAAEPSO algorithm is better than other two algorithm in terms of IGD, CM, Spacing, DM

metrices and robustenss on FON and SCH functions. MOSaDE shows robustness for achieving CM and Spacing on FON functions, where as MOCLPSO shows robustness for achieving Spacing and DM on SCH function.

Table 4.2: Results for MOUO functions: mean(std)

UF1	IGD	Convergence Metric	Spacing	Diversity Metric
MOAAEPSO	**9.73e-3**(9.22e-3)	2.69e-1(2.66e-1)	**1.46e-2**(2.86e-2)	**4.11e-1**(1.38e-1)
MOCLPSO	4.82e-3(2.20e-3)	**1.35e-1**(**5.63e-2**)	3.42e-2(**1.58e-2**)	4.40e-1(8.91e-2)
MOSaDE	1.12e-2(**3.02e-4**)	8.41e-1(9.23e-2)	7.05e-2(1.69e-2)	5.01e-1(**6.73e-2**)
UF2				
MOAAEPSO	**2.10e-3**(**7.30e-5**)	**6.54e-2**(**2.41e-3**)	**1.52e-2**(**8.62e-3**)	**3.98e-1**(4.87e-2)
MOCLPSO	5.10e-3(3.09e-4)	1.54e-1(1.01e-2)	9.46e-2(3.62e-2)	4.49e-1(**2.39e-2**)
MOSaDE	1.21e-2(4.23e-3)	3.02e-1(1.08e-1)	1.19e-2(1.25e-2)	5.59e-1(5.98e-2)
UF3				
MOAAEPSO	1.62 e-2(5.1e-3)	4.67e-1(1.6e-1)	1.0(0.23)	9.07ee-1(1.02e-1)
MOCLPSO	1.81e-2(4.23e-4)	5.70e-1(1.33e-2)	1.25e-1(8.08e-2)	6.64e-1(4.13e-2)
MOSaDE	**1.42e-2**(**1.70e-4**)	**4.47e-1**(**5.0e-3**)	**3.63e-2**(**1.75e-2**)	**5.96e-1**(**3.43e-2**)
UF4				
MOAAEPSO	9.14e-1(5.87e-1)	2.89e+1(1.86e+1)	9.27e-1(7.03e-2)	9.88e-1(7.45e-3)
MOCLPSO	6.63e-3(4.29e-4)	2.06e-1(1.35e-2)	9.26e-2(1.90e-2)	3.79e-1(1.77e-2)
MOSaDE	**3.08e-3**(**5.52e-5**)	**9.67e-2**(**1.88e-3**)	**6.95e-3**(**1.19e-3**)	**3.44e-1**(**9.47e-3**)
UF5				
MOAAEPSO	**1.43e-1**(5.15e-2)	**6.37e-1**(2.27e-1)	**2.60e-2**(**5.48e-2**)	**1.37e-1**(2.69e-1)
MOCLPSO	3.64e-1(1.28e-1)	1.66(5.89e-1)	2.05e-1(8.70e-2)	7.37e-1(6.84e-2)
MOSaDE	3.84e-1(**1.70e-2**)	1.76(**7.88e-2**)	1.91e-1(1.26e-1)	7.69e-1(**3.26e-2**)
UF6				
MOAAEPSO	**4.56e-3**(**5.42e-4**)	**1.18e-1**(**2.72e-2**)	**1.02e-1**(1.63e-1)	9.96e-1(2.33e-4)
MOCLPSO	2.25e-2(4.81e-3)	6.82e-1(1.26e-1)	5.33e-2(**9.37e-2**)	9.98e-1(**1.67e-4**)
MOSaDE	2.73e-2(1.47e-3)	6.59e-1(4.48e-2)	1.84e-1(1.44e-1)	9.99e-1(2.52e-4)
UF7				
MOAAEPSO	**1.10e-2**(**2.34e-3**)	**2.67e-1**(5.41e-2)	2.85e-2(3.25e-2)	**2.15e-1**(**5.75e-2**)
MOCLPSO	9.15e-2(7.43e-2)	3.40e-1(1.87e-1)	5.10e-2(3.70e-2)	2.71e-1(1.70e-1)
MOSaDE	7.21e-2(9.89e-3)	7.07e-1(**2.57e-2**)	5.18e-2(**2.45e-2**)	3.55e-1(6.09e-2)
UF8				
MOAAEPSO	**3.85e-3**(8.93e-4)	**3.23e-1**(7.50e-2)	**9.10e-2**(6.47e-2)	**9.16e-1**(5.79e-4)
MOCLPSO	5.07e-3(1.19e-3)	4.35e-1(9.16e-2)	1.14e-1(3.94e-2)	9.25e-1(**1.74e-4**)
MOSaDE	4.39e-3(**3.96e-4**)	4.17e-1(**4.23e-2**)	9.63e-2(**3.21e-2**)	9.26e-1(4.47e-4)
UF9				
MOAAEPSO	**2.33e-3**(4.15e-4)	**2.13e-1**(3.78e-2)	**1.26e-1**(2.65e-1)	**9.68e-1**(2.84e-3)
MOCLPSO	6.08e-2(5.63e-4)	5.63e-1(5.69e-2)	2.04e-1(1.09e-1)	9.75e-1(**7.83e-5**)
MOSaDE	4.83e-2(**3.17e-4**)	4.65e-1(**2.72e-2**)	1.69e-1(**5.29e-2**)	9.75e-1(1.45e-4)
UF10				
MOAAEPSO	**9.02e-3**(**1.47e-3**)	**8.49e-1**(1.50e-1)	**1.81e-3**(**2.07e-3**)	**9.25e-1**(**1.14e-4**)
MOCLPSO	3.23e-1(3.03e-3)	3.22(3.03e-1)	4.65e-1(1.e-1)	9.28e-1(3.25e-4)
MOSaDE	3.92e-1(2.60e-3)	3.92e+0(**2.61e-1**)	5.37e-1(1.91e-1)	9.31e-1(8.87e-4)

Table 4.3: Conclusion of mean results from Table 4.1 and 4.2

Metric	KUR	FON	SCH	ZDT1	ZDT2	ZDT3	ZDT4
IGD	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOCLPSO	MOAAEPSO	MOAAEPSO
CM	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOCLPSO	MOAAEPSO	MOAAEPSO
S	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOCLPSO	MOCLPSO	MOCLPSO	MOAAEPSO
DM	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOCLPSO	MOCLPSO	MOSaDE

Metric	UF1	UF2	UF3	UF4	UF5	UF6
IGD	MOAAEPSO	MOAAEPSO	MOSaDE	MOSaDE	MOAAEPSO	MOAAEPSO
CM	MOCLPSO	MOAAEPSO	MOSaDE	MOSaDE	MOCLPSO	MOAAEPSO
S	MOAAEPSO	MOAAEPSO	MOSaDE	MOSaDE	MOAAEPSO	MOAAEPSO
DM	MOAAEPSO	MOAAEPSO	MOSaDE	MOSaDE	MOAAEPSO	MOAAEPSO

Metric	UF7	UF8	UF9	UF10
IGD	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOAAEPSO
CM	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOAAEPSO
S	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOAAEPSO
DM	MOAAEPSO	MOAAEPSO	MOAAEPSO	MOAAEPSO

For ZDT1 function, MOAAEPSO is better in terms of IGD and convergence metric compared to other two algorithms, where as in terms of spacing, diversity metric and robustness, it is competitive with others as in Table 4.1. Table 4.1 also reveals that on ZDT2 function, MOCLPSO is better in terms all four indicative metrices and robustness while MOAAEPSO is competitive on ZDT3. For the ZDT4 function, MOAAEPSO is better in terms of IGD, Convergence metric and spacing. For the diversity metric, it is competitive with MOCLPSO. In summary we can conclude that the quality of solution of the proposed MOAAEPSO algorithm is better than other two algorithms for all the functions except ZDT2. The diversity for the proposed algorithm is as good and sometimes better than the other algorithms except for ZDT2 function. Table 4.2 presents the mean and standard deviations (inside parenthesis) of the performance metrices on CEC 2009 benchmark functions. MOAAEPSO performs better than the other two algorithms in terms of IGD, spacing, CM, DM and robustness on all the CEC 2009 benchmark functions except UF3 and UF4. On UF3 and UF4, MOSaDE dominates all the metrices including robustness.

The summary of performances of all the three algorithms on all benchmark functions are consolidated in Table 4.3. This table indicates the best algorithm for a function in terms of all four performance evaluation parameters. From this table we find that MOAAEPSO algorithm performs almost better than other two algorithms on all the benchmark functions except ZDT2, UF3 and UF4. In case of the UF3 and UF4 functions, MOSaDE algorithm is found to perform the best.

4.6.3 Statistical metrices

The statistical significance of the algorithms is analyzed using the hypervolume indicator followed by ANOVA comparison test. Table 4.4 presents the hypervolume indicator for all the numerical benchmark functions considered above and Table 4.5 presents the ANOVA test results on the hypervolume indicator. Table 4.4 presents the Hypervolume data on the approximation pareto fronts obtained by MOAAEPSO, MOCLPSO and MOSaDE algorithms. The Table 4.5 shows the results after ANOVA test on the hypervolume data in Table 4.4 with α (confidence level) to be 0.05. In Table 4.5 $'SS'$ is Sums of Squares, $'DF'$ is Degrees of Freedom, $'MS'$ is Mean Squares and $'F'$ is Test Statistic. The test statistic value F is 4.249. Using an α of 0.05, we have $F_{0.05;2,18} = 3.555$ (from F distribution

table). Since the test statistic is much larger than the critical value, we conclude that there is a (statistically) significant difference among the average values. The $p-$value for F =4.249 is 0.0898, so the test statistic is significant at that level. This statistically significant difference in the average results show that, MOAAEPSO fails on Pareto compliance metrices.

Table 4.4: Hypervolume indicator

Funtion	MOAAEPSO	MOCLPSO	MOSaDE
FON	0.30917	0.30676	0.31584
KUR	0	0	0
SCH	0.82546	0.82769	0.82838
UF1	0.32723	0.33166	0.6189
UF2	0.21067	0.53614	0.63385
UF3	0.5	0.32752	0.33511
UF4	0.30152	0.20317	0.32088
UF5	0.087749	0.21283	0.2594
UF6	0.21772	0.025388	0.46604
UF7	0.094189	0.2251	0.4064
UF8	0.58721	0.19626	0.70702
UF9	0.9751	0.27155	0.71322
UF10	0.002768	0.16959	0.48569
ZDT1	0.59174	0.6578	0.64471
ZDT2	0.32558	-1	0.27237
ZDT3	0.62696	0.6277	0.50842
ZDT4	0.3016	0.33584	0.72769
ZDT6	0	0.38037	0.029946

Table 4.5: ANOVA test

Source	SS	DF	MS	F
Treatment (between columns)	0.1512	2	0.07560	4.249
Individual (between rows)	2.242	18	0.1246	
Residual (random)	1.055	36	0.02932	
Total	3.449	56		

4.7 Conclusions

This chapter introduced the concept of Multi Objective Constrained Optimization (MOCO) along with a detailed literature survey on multiple objective handling algorithms. This chapter also presents in detail the challenges that PSO faces in handling multiple objectives in optimization. It was found that, the major challenges of PSO are premature convergence, uncertainty and poor quality solution, this is especially dominant while solving complex multimodal conflicting objective problems. To address the problem of premature convergence and poor quality solution in PSO while solving multiple (often conflicting) objectives, the Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (MOAAEPSO) was proposed. While solving multiple objectives PSO produces nondominated and dominated particles, and MOAAEPSO first separates the dominated particles (diverged) that gives less nondominated solution and are responsible for premature convergence. MOAAEPSO then accelerates these particles under the guidance of nondominated solutions. Essentially these particles were forced to leave their current position and search the solution space around nondominated solution. In MOAAEPSO the acceleration of diverged particles towards nondominated solution was done adaptively. A search patch was maintained around the nondominated solution of the current iteration. The size of the search patch was decreased exponentially in a controlled manner. The size of the search patch was kept to be more in the beginning of search process and was decreased with the iteration. This strategy, in particular, helped more exploration of search space around nondominated solutions in the beginning and more exploitation as search process proceeds. The comprehensive analysis of the proposed algorithms was carried out on set of complex MOUO problems with the present state of art. The results in terms of pareto front graphs and various parametric measures clearly shows that the proposed algorithm shows remarkable improvements in the solution and hence addresses the prevailing problems of PSO. Further the analysis was carried out using Pareto compliant indicator (hypervolume), and it confirms that proposed algorithm on this.

Chapter 5

Multi Objective Constrained Optimization

This chapter presents the concept of Multi Objective Constrained Optimization (MOCO). It also presents a detailed literature survey for solving MOCO problems using Particle Swarm Optimization (PSO) and other competitive algorithms including Swarm Intelligence (SI). The major challenges faced by PSO while solving MOCO problems are also addressed in this chapter. The chapter is extended with research contribution made in MOCO with a novel PSO variant. The novel PSO variant namely Constrained Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (CMOAAEPSO) is developed for solving MOCO problems. The algorithmic and implementation details of CMOAAEPSO is presented in details. The performance of the developed algorithm is illustrated in terms of Non Pareto compliant performance indicators such as pareto front graphs, inverted generational distance and convergence metrices while solving benchmark functions and engineering design problems. The statistical significance of the algorithms are also analyzed using Pareto compliant hypervolume indicator and nonparametric Wilcoxon signed rank test test. These indicators are used for a performance comparison of CMOAAEPSO with state of the art. The chapter is concluded with results and discussions.

5.1 Introduction

Multiobjective Constrained Optimization (MOCO) is an important, difficult and challenging topic in engineering and computation domain [137]. The MOCO usually involves multiple (some times conflicting) objectives and associated constraints (often complex). The complex constraints together with conflicting objective increase the difficulty in optimization. The major challenging issues of the MOCO problems is to handle multiple objectives and their associated constraints. Mathematically the MOCO problems are the problems involving minimization or maximization of multiple objective functions under given constraints such as inequality, equality, upper and lower bound. The simplified mathematical representation of MOCO problem is as follows

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ & f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x}) \\ subject\ to: \quad & g_i(\overrightarrow{x}) < 0, i = 1, 2, \ldots, q \\ & h_i(\overrightarrow{x}) = 0, i = q+1, q+2, \ldots, m \\ Where\ & l_i \leq x_i \leq u_i, i = 1, 2, \ldots, D \end{aligned} \tag{5.1}$$

Where k is the number of objectives to be optimized simultaneously, g_i, and h_i are the inequality and equality constraints respectively. The values of l_i and u_i $\forall$ i $\in$ D are the lower and upper bounds defining the search space respectively, and D is the dimension of the problem. The multiple objectives in MOCO usually consist of conflicting objectives. In this case, a single solution does not describe the optimal decision. Hence there can be a set of optimal solutions to the given problem. The tradeoffs amongst them is defined in terms of Pareto optimality as explained in Figure 4.1 of Chapter 4.

5.2 Literature Survey

Most of the real world optimization applications usually consist of multiple objectives and associated constraints. Hence Multiobjective Constraint Optimization (MOCO) deals with solving a set of conflicting objectives function while satisfy-

ing a set of constraints. Although different algorithms are available in literature for solving multiobjective constraint optimization problems [138], there is still demand for an efficient algorithm for solving MOCO problems. A detailed review of different algorithms for solving MOCO problems can be found in [138]. There are lots of algorithm available to solve the problems involving multiple objectives and problems with constraints separately.

In 1997, Binh and Korn [139] proposed an algorithm called MultiObjective Evolution Strategy for MOCO problems (MOES). This algorithm considers a degree of violation of constraints for infeasible solutions and objective function vector to select the potential parents. The infeasible individuals were ranked based on their degree of constraints violation. Jimenez et al. in 1998 proposed Constrained multiobjective optimization by an evolutionary algorithm [140]. In this algorithm [140], nondominated sorting method was used to find Pareto-optimal solution, the ranking selection and crowding factor model was used to maintain diversity in the population. Kurpati et al. in 2002 suggested four constraint handling improvements for Multi-Objective Genetic Algorithms (MOGA) [141]. These improvements were made in the fitness assignment stage of MOGA and were all based upon a $Constraint - First - Objective - Next$ model. In the first strategy only objective function for whole population was evaluated and inferior individuals were identified, the constraint function was evaluated only for inferior individuals in the population. This reduces the computational cost involved in evaluating constraints. In the second strategy, the amount of infeasibility was taken into account while handling constraints and penalizes the infeasible individuals depending on average constraint violation. In the third strategy instead of considering the amount of constraint violation into account, it considered the number of violated constraints, and penalized the infeasible individuals depending on the number of constraints that the individual violates. The fourth strategy was essentially the combination of second and third. In 2003 Chafekar et al. proposed two methods for solving MOCO problems using steady state GAs [142]. First method was called objective exchange genetic algorithm for design optimization (OEGADO). The OEGADO runs several GAs concurrently where each GA optimizes one objective and exchanges information about its objective with the others. The second method was called objective switching genetic algorithm for design optimization (OSGADO). The OSGADO optimize each objective sequentially with a common population for all objectives. In 2004, Ji et al. presented a divided range

multi-objective PSO for distributed computing [143]. Here symbiosis mechanism was used for constrained handling. In symbiosis mechanism, the feasible particles evolve towards the Pareto front and infeasible particles toward feasibility guided by a non-feasibility function. Ji et al. [143], claimed that the Pareto fronts were achieved but it was not supported with experimental results. The non-feasibility function used to guide the infeasible particles was also not clear. Osman et al. in 2006 proposed an An iterative co-evolutionary algorithm (IT-CEMOP) for a multiobjective optimization problem with nonlinear constraints [144]. This algorithm was based on the concept of co-evolution and repair algorithm for handling nonlinear constraints and maintains a finite-sized archive for nondominated solutions. This algorithm also makes use of ϵ-dominance and this has guaranteed convergence and diversity. In 2007, Reddy et al. presented Elitist-Mutated Multi-Objective Particle Swarm Optimization (EM-MOPSO) [145] based on Pareto dominance and constraint dominance concept. The *gbest* particle was selected randomly from an external repository, where all the non-dominated solutions were stored. The *pbest* particle was determined by the Pareto-dominance concept. This approach however was well tested only on four MOCO problems and may well fail on other. Since this approach uses the external repository concept, computation cost of the algorithm increases while implementation [146]. Zhuhong Zhang [147] in 2007 developed Immune optimization algorithm for constrained nonlinear multiobjective optimization problems. This algorithm was based on Pareto optimality and simple interactive metaphors between antibody population and multiple antigens. The algorithm in [147] devised two mechanisms; one for constraints and the other for multiple objectives. This algorithm proved to do well on high dimensional complex optimization problems with multiple constraints. In 2008, Singh et al. applied Simulated Annealing (SA) on constrained multi-objective optimization problems [148]. The algorithm [148] minimizes constraint violation by moving along approximate descent direction while operating on infeasible solutions. It maintains an external archive for non-dominated solutions and the acceptance probability of a new solution was determined by its feasibility status, and its domination status as compared to the current solution and the solutions in the archive. Yonas et al. in 2009 proposed constraint handling in multiobjective evolutionary optimization [149]. In this algorithm adaptive penalty function and a distance measure was used, here objective space was modified for better performance and less constraint violation of an individual. The modified objective functions were used in nondom-

inated sorting to facilitate the search of optimal solutions not only in the feasible space but also in the infeasible regions. The search in the infeasible space was designed to exploit those individuals with better objective values and lower constraint violations. The number of feasible individuals in the population was used to guide the search process either towards finding more feasible solutions or in search for Pareto optimal solutions. This method was simple to implement and does not need any parameter tuning. In 2009 many algorithms were developed for solving MOCO problems specially CEC 2009 problems found in [134]. Minzhong Liu at. al. proposed Multi-Objective Evolutionary Algorithm with Domain Decomposition (DMOEA-DD) algorithm [150]. In this algorithm the domain decomposition technique was used to divide the feasible domain of decision variables into several sub domains. In each sub domain, the DMOEA was used to search the Pareto optimal solutions and each sub domain exchanges the information by genetic operators. Hai lin Liu and Xueqiang Li proposed multiobjective evolutionary algorithm based on determined weight and sub-regional search [151]. This algorithm divides the multiobjective optimization space into small regions and individuals in same region operated with evolutionary operator. The information between the individuals of other regions was exchanged through their offsprings and again re-divided into regions. Kukkonen et al. proposed Generalized Differential Evolution 3 (GDE3) for constrained multi-objective problems [152]. This algorithm implements three different methods for diversity maintenance. The first methods was based on selection; weakly constraint-dominated previous solution solution was replaced with the current solution. Here no nondominated sorting and external repository was used. In the second methods the selection was based on crowdedness; current solution replace the previous solution if it is feasible and non-dominating. With the third method, the size of the population is reduced using non-dominated sorting and crowding distance for diversity preservation. Zamuda et al. proposed Differential Evolution with Self-adaptation and Local Search for Constrained Multiobjective Optimization (DECMOSA-SQP)[153] which uses the self-adaptation mechanism for solving MOCO problems. Tseng et al. proposed Multiple Trajectory Search for Unconstrained/Constrained Multi-Objective (MTS) [154] Optimization, with local search methods for solving MOCO problems. In the beginning of search process, this algorithm [154] uses very large sized neighborhood, and size of neighborhood decreases with iteration. Karthik Sindhya et al. proposed Local Search Based Evolutionary Algorithm [155] for solving MOCO problems. This algorithm

has used a clustering technique for searching better Pareto optimal solutions. Chih-Ming et al. has enhanced MOEA/D[108] with Guided Mutation and Priority Update for solving MOCO problems [156]. This algorithm [156], uses a guided mutation operator as a replacement for the differential evolution operator found in MOEA/D[108], additionally it as used priority update for maintaining uniformly distributed Pareto font. In 2010, Layak Ali et al. proposed Constrained Multi Objective Adaptive and Accelerated Exploration Particle Swarm Optimizer (CMOAAEPSO) [157] for handling MOCO problems. The CMOAAEPSO algorithm [157] was an extension of AAEPSO [35] with the integration of a bounded archive, crowding distance and pareto optimality for solving MOCO problems. In this algorithm nondominated particles were separated from dominated and feasible particles from nonfeasible. The nondominated-feasible particles were desirable and other set of particles were undesirable, Hence these particles were accelerated in the direction of desired particles, the acceleration was similar to [35]. Hemant Kumar et al. in 2010 developed Constrained Pareto Simulated Annealing (C-PSA) [138] for constrained multi-objective optimization. Essentially C-PSA was an extended version of [148], specifically considers constraints handling effectively. However the standard versions of PSO, like other evolutionary algorithms, lack an explicit mechanism to handle constraints that are often found in science and engineering optimization problems. This necessitates the development of efficient and robust optimization algorithms for solving MOCO problems.

5.3 Challenges in Multi Objective Constrained Optimization

Most of the real world science and engineering design problems can be framed as Multi Objective Constrained Optimization (MOCO). The presence of conflicting objectives and complex nonlinear constraints poses challenges to the optimization algorithms. The major challenges in solving MOCO problems are as follows.

1. MOCO problems have multiple objectives; some time these objectives could be conflicting each other and hence make optimization difficult.

2. The multiple objectives are again associated with multiple constraints; these constraints may cause the solution to diverge.

3. The associated constraints may be linear and or nonlinear; this makes the problem more complex.

4. Often the search space available for these kind of problems may be very complex, and the the feasible (physically realizable) region in the search space can be very small.

5. Due to the complex solution space the problems could be multimodal, and the algorithms based on initial solution startup may fail and get stuck in local optima.

6. PSO neither internally nor externally has the ability to handle either multiple objectives or constraints.

7. PSO itself has the perennial problem of premature convergence especially on multimodal problems.

5.4 Research Contribution

In MOCO, more than one conflicting objectives needs to be optimized simultaneously while satisfying a set of constraints. In single objective optimization using PSO, a single *gbest* of the swarm directs all the particles to obtain global solution. In the other way single *gbest* guides the particles to find global solution. In case of multiobjective optimization (MO), the number of non dominated solutions are more than one in the Pareto front. So each solution in the pareto front is a potential *gbest* solution of the problem. In the original PSO, *gbest* guides the swarm and in some cases it fails to find global solution. In some situations, the solutions are stuck in local optima in search of global minima. In subsequent generation, the stuck particle will search around the local optima only. To solve this issue Sabat et al. proposed an improved PSO variant, namely Adaptive Accelerated Exploration PSO (AAEPSO) [35] for solving single objective optimization problems.

The successful and promising results of AAEPSO [35] is the major motivation to apply it on MOCO problems. To solve MOCO problems, AAEPSO is extended with the concept of Pareto optimality, bounded archive and particle probing to solve MOCO problems. Thus Constrained Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (CMOAAEPSO) for solving MOCO problems is developed. The diverged particles selection and acceleration is almost

similar to AAEPSO [35] with MOCO requirements. Additionally particle probing is incorporated and explained in section 5.4.2.

5.4.1 Constrained Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer

The CMOAAEPSO separates the problem causing particles and forces them towards the best possible solution. Since it deals with MOCO problems, the selection of problem causing particles is a tricky issue. Based on constraint violations the particles are categorized as either feasible (that satisfies all the given constraints) or infeasible (that violates all or at least one constraints). Similarly based on the principle of Pareto optimality, the particles are categorized as dominated and nondominated. By combining above two principles, a particle may belong to either of four categories a) Nondominated-Feasible, b) Dominnated-Feasible, c) Nondominated-Infeasible and d) Dominated-Infeasible as shown in Figure 5.1. As a fact, the Nondominated-Feasible particles possess the desired solution and the particles that belong to either of the other three categories are treated as undesired or diverged particles. Proposed algorithm presents a novel strategy to identify the diverged particles and accelerate them towards the nondominated feasible solution [35]. The detailed implementation steps for the proposed algorithm is presented in Algorithm 5.4.1. The proposed algorithm is divided and explained under the heading; selection, acceleration and exploration, particle probing, external archive. The selection, adaptive acceleration and exploration steps are almost similar to the MOAAEPSO algorithm as discussed in Chapter 4 with the difference of diverged particle selection and acceleration. Here the diverged particles are selected based on the feasibility and dominance of the particles. The nonfeasible dominant, feasible dominant and nonfeasible nondominant particles are termed as diverged particles. These diverged particles are then accelerated towards feasible nondominated solution. This process is best be explained with help of Figure 5.2.

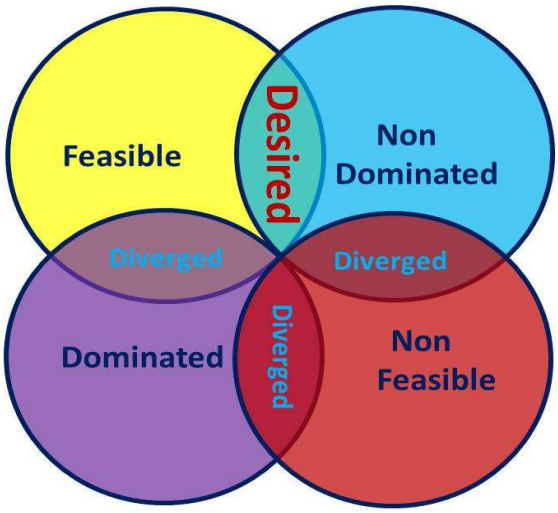

Figure 5.1: Diagram illustrating MOCO Solution Space

Figure 5.2 shows the Nondominated − Feasible, Nondominated − Infeasible, Dominated − Feasible and Dominated − Infeasible solutions (particles); the light blue circle shows Nondominated − Infeasible, Dominated − Feasible and Dominated − Infeasible solutions and they are called as diverged. Few among them are selected for further acceleration, one solution from Nondominated − Infeasible and Dominated − Feasible are selected for acceleration. Two solutions are selected from Dominated − Infeasible group for acceleration. The few selected solutions are forced to come closer and search for the better solution nearby nondominated − feasible.

5.4.2 Particle probing

Particle probing concept is an extension of acceleration and exploration of search range around elite particle found in [35] to all *pbest* particles. In accelerated and exploration [35]; the search range is restricted to only around the elite particle.

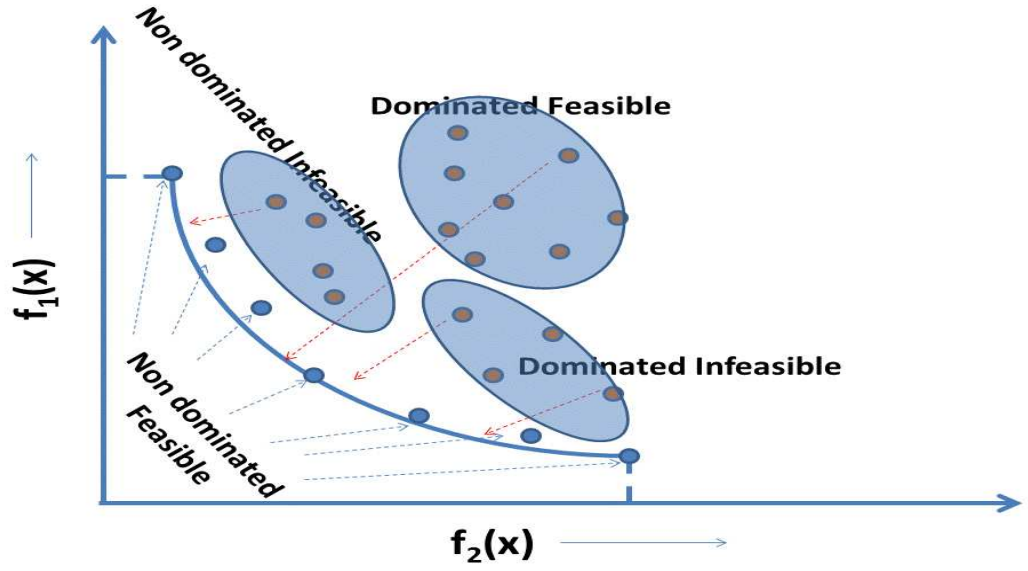

Figure 5.2: CMOAAEPSO Acceleration process

As far as the unconstrained single objective is concerned the single best solution is expected and hence a single elite particle exists, but when solving constrained multiobjective problems, there will be multiple solutions and all will be equally good. The acceleration and exploration of the region around single elite particle found in [35] is extended to the multiple solutions. The best solutions in the history of PSO iterations will be in the form of *pbest* particles. Thus to solve constrained multiobjective problems the diverged particles are accelerated towards the multiple nondominated-feasible solution spaces (the *pbest*). The random number of selected diverged particles are accelerated towards respective *pbest* particles. This gives more freedom to search the solution space in a controlled manner. The process of particle probing is best explained with the help of Figure 5.3. In Figure 5.3, the red circles represents all the *pbest* particles; the three big circles denote the search range around the *pbest*. The black circle inside the big circle denotes the probable solution that are generated. The yellow circle at the bottom of the Figure 5.3 denote the best of the *pbest* solution after contention period. The *pbest* particles (red circles) are selected and the random solutions are generated (black circle) as per patch size (big circle) around the *pbest*. These new solutions are evaluated for their fitness and the best among them are selected (yellow circles). To preserve the population the number of newly generated particles (yellow circles) will be same as initially selected diverged particles (red circles).

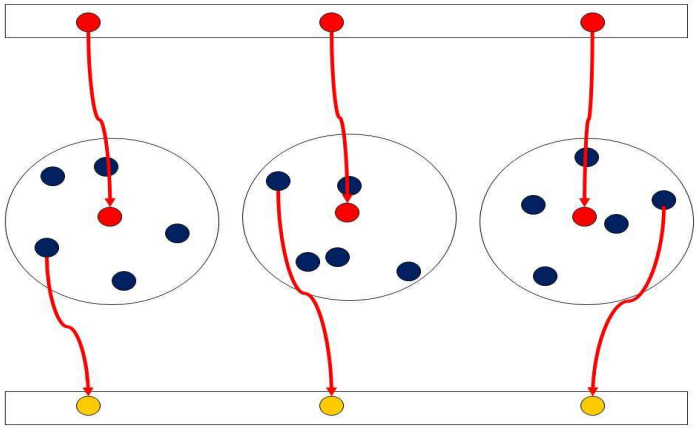

Figure 5.3: Particle probing for AAEPSO

5.4.3 External Archive

While solving constrained multiobjective problems multiple solutions are generated and hence the size of internal archive (in the form of *pbest*) is not sufficient to store all these solutions and hence an external archive is maintained. The external archive preserves the pareto optimal solutions that are produced during due course of the search. We have used bounded archive to preserve the good solutions and the size of archive is same as that of population. Initially, the archive is empty and as the generation progresses, good (nondominated-feasible) solutions enter the archive. The good solutions obtained in each generation are compared one by one with the solutions in the current archive. The archive is updated with following four strategies. (1) If the current solution does not dominate the members of archive, the new solution is rejected. (2) If the current solution dominates the members of the archive, then new solution enters the archive. (3) If the archive exceeds maximum predefined size then, the dominated members are reject. (4) If the archive size exceeds the predefined limit (here size of the archive is same as the population size) then crowding distance is used to eliminate the less dominated members from the archive.

5.4.4 Implementation details of CMOAAEPSO Algorithm

Algorithm 5.4.1 shows all the necessary steps for practical implementation of CMOAAEPSO algorithm. The implementation steps of CMOAAEPSO are almost similar to MOAAEPSO with particle probing and feasibility of solution as an added component. The process of diverged particles selection and adaptive acceleration exploration is similar to MOAAEPSO. In Algorithm 5.4.1 the population of size NP is initialized in the search space. The fitness value of whole population is evaluated and the current position is set as the *pbest* and best

Algorithm 5.4.1 Pseudo code of CMOAAEPSO

Initialization
Initialize the swarm of size NP:
Initialize position $'X'$ and velocity $'V'$ randomly in the search range (X_{max}, X_{min}).
Initialize Selection (S_f) and Acceleration Exploration (AE_f) factor as in [35] .
Evaluate the fitness values of all particles.
Set the current position as *pbest* and the particle with the best position as *gbest*.
Initialize the bounded Archive to the size of population
Optimize
for $t \leftarrow 1$, $Maxgen$ **do**
$w^t, c_1^t, c_2^t = w_{max} - ((w_{max} - w_{min})/Maxgen) * t \;\; Where \;\; w_{max} = 0.9, w_{min} = 0.2$
Decrease AE_f exponentially as per equation (2.13)
Update velocity (equation 2.2) and position (equation 2.3)
Evaluate fitness and constraints violations for each particle.
Find the Feasible solutions.
Find the Pareto-optimal of feasible solutions.
Update *pbest*: If current fitness dominates the previous then set current position as *pbest* else retain *pbest*.
Update *gbest*: If best of all the current *pbest* dominates the previous *gbest* then set best *pbest* as *gbest* else retain *gbest*.
Check the dominating status of new solution (X) with the solutions in the archive.
  if X dominates any member of Archive **then**
    Delete the dominated member and insert X in the Archive
  else if X is neither dominated by the members of Archive or the members by X **then**
    Insert X in the Archive
  else if X is dominated by the members of Archive **then**
    reject X
  end if
  if Size of Archive exceeds maximum size **then**
    use crowding distance to eliminate less dominated particles.
  end if
Apply Algorithm 2.5.2 to diverged particles.
end for t
continue optimizing until stopping criteria or exceeding maximum iteration
Report results
Terminate

feasible nondominated particle is set as *gbest*. The selection factor S_f is initialized, this factor is responsible for selecting the diverged particles. The acceleration

factor AE_f is decreased exponentially with iterations. Velocity and position of all the particles are evaluated as per equations (2.2) and (2.3) respectively. The fitness of the new solution is evaluated based on the updated position. Based on the dominance and feasibility of solution the *pbest* and *gbest* are updated. If current fitness dominates the previous then the current position is set as *pbest* else the previous *pbest* is retained. Similarly if the best of all the current *pbest* dominates the previous *gbest* then best *pbest* is set as *gbest* else previous *gbest* is retained. CMOAAEPSO also implements bounded external archive for storing feasible nondominated solutions. The archive is updated as as explained in section 5.4.3.

5.5 Simulation

The simulations were carried out in Windows XP Operating System, on Pentium IV 2.6GHz with 512MB of RAM. The coding is done in Matlab 7.2 programming language. The population size and the maximum iterations considered for all the algorithms are 100 and 1000 respectively. The population is initialized in the search range $[X_{min}, X_{max}]$. X_{max} and X_{min} are the maximum and minimum value in search range. The velocity is initialized with 20 percent of the search range i.e., $V_{max} = 0.20 * (X_{max} - X_{min})$. The results obtained are the average of 25 trials. The performance of the proposed algorithm is compared on standard benchmark functions CF1 to CF10 from CEC 2009 [158] and numerical functions chosen from [159, 144]. These functions have different characteristics in terms of convexity, discontinuity and non uniformity in objective and constrained functions. The detailed definition and characteristics of these functions are given in Appendix D.

Further the effectiveness of algorithm is validated on four engineering design problems [159, 144]; a) Welded beam design problem [159], b) Two-bar truss design problem [159], c) Speed reducer design problem [144] and d) Gear train design problem [159]. The discussion is extended with the comparison of proposed CMOAAEPSO algorithm with the current state of art [153, 154, 155, 156, 138, 151, 150, 152].

Two different set of performance metrics such as I) Conventional metrices; that includes a) Inverted Generational Distance (IGD) [158], b) Convergence Metric (CM) [107], c) Robustness and d) Pareto fronts and II) Statistical metrices; that include Hypervolume indicator followed by Wilcoxon signed rank test. Conven-

tional metrices are under the category of non Pareto compliant where as statistical metrices are under the category of Pareto compliant.

5.5.1 Performance metrices

For measuring the performance of constrained multiobjective optimization algorithms, two sets of performance assessment metrices are used: one is Non Pareto compliant and other is Pareto compliant. The most commonly used Non Pareto compliant metrices include Pareto front graphs, inverted generational distance and convergence metrics [107]. The Pareto compliant indicator such as Hypervolume indicator is used. Further the comparisons are done with nonparametric test namely Wilcoxon signed rank test. The details of these metrices are defined in Chapter 4 and the brief descriptions of these metrices are presented in below subsections. Non Pareto compliant metrices are described under common name as conventional metrices and Pareto compliant metrices as statistical metrices..

5.5.1.1 Conventional metrices

The most commonly used conventional performance metrices often called Non Pareto compliant metrices are discussed here.

Inverted Generational Distance Metric (IGD:) IGD [107] is used to estimate the closeness of elements in approximated Pareto front with respect to the elements of true Pareto front. A smaller IGD value indicates better performance. The detailed definition is given in Chapter 4.

Convergence Metric (CM): CM [107] measures the extent of convergence of Pareto solutions to the known optimal Pareto front. A smaller value of *CM* denotes a better convergence performance. The detailed definition is given in Chapter 4.

Pareto front graphs: Pareto front graphs are pictorial way of analyzing how well the algorithms obtain Pareto optimal solutions. Explained in Chapter 4.

5.5.1.2 Statistical metrices

Following are the two performance metrices used for statistical comparisons. Hypervolume indicator is a performance measure under the category of Pareto compliance and Wilcoxon signed rank test falls under the category of nonparametric statistical test.

Hypervolume Indicator (I_H): Hypervolume indicator is one of the indicator based approaches for performance assessment of multiobjective optimization algorithms [135]. In this work hypervolume indicator is evaluated using the tool [136]. Detailed definition of Hypervolume Indicator is given in Chapter 4.

Wilcoxon signed rank test: Pairwise comparisons are the simplest kind of statistical tests that a researcher can apply within the framework of an experimental study. Such tests are directed to compare the performance of two algorithms when applied to a common set of problems. Wilcoxon's test is one test for this kind of a pair test. Wilcoxon's signed rank test is a nonparametric test that compares two paired groups. This test compares two matched groups, without assuming that the distribution of the before-after differences follows a Gaussian distribution. Essentially it compares the median of a column of numbers against a hypothetical median [41].

5.6 Results and Discussions

The performance of proposed CMOAAEPSO algorithm is compared with present state of art algorithms on functions CF1 to CF10 [134] and functions chosen from [159, 144]. Further algorithms are validated on engineering design problems [159, 144]. The results are discussed separately for numerical benchmark functions and engineering design problems.

5.6.1 Numerical benchmark functions

Comprehensive analysis of proposed algorithm with the state of the art is carried out on numerical benchmark functions with different metrices separately as follows.

5.6.1.1 Pareto front graphs

The Pareto fronts obtained by CMOAAEPSO are presented from Figure 5.4 to Figure 5.5. Figure 5.4 show the Pareto fronts obtained by proposed CMOAAEPSO algorithm on 5 benchmark functions CF1 to CF5. And Figure 5.4 show the Pareto fronts obtained by proposed CMOAAEPSO algorithm on 5 benchmark functions CF6 to CF10. From Figure 5.4 and Figure 5.5 it is evident that the proposed algorithm gives good front on functions CF1, CF2, CF6 and CF8, where as it

gives competitive fronts with respect to true front on functions except CF3, CF5 and CF7.

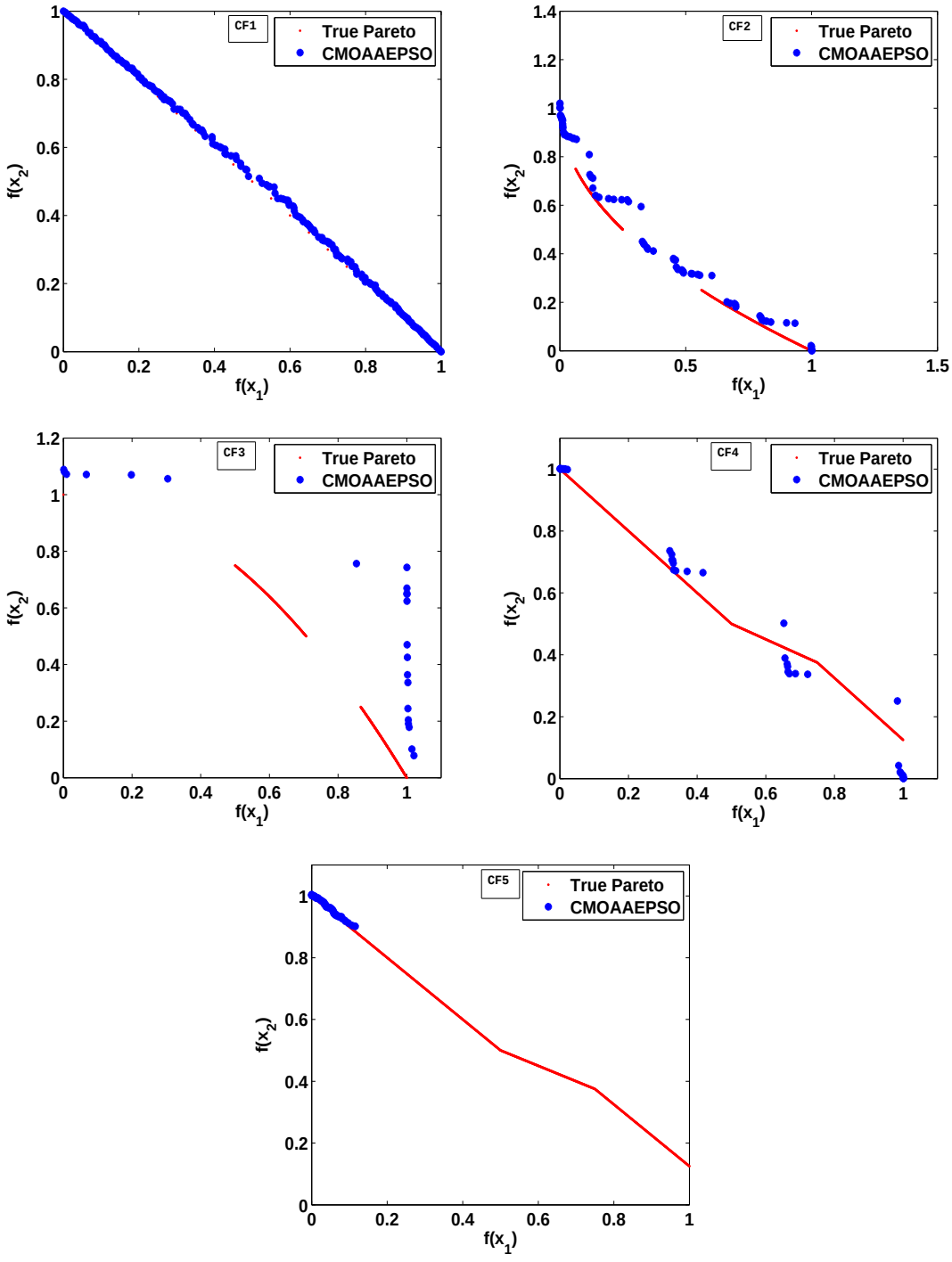

Figure 5.4: Pareto fronts on CF1 to CF5 by CMOAAEPSO

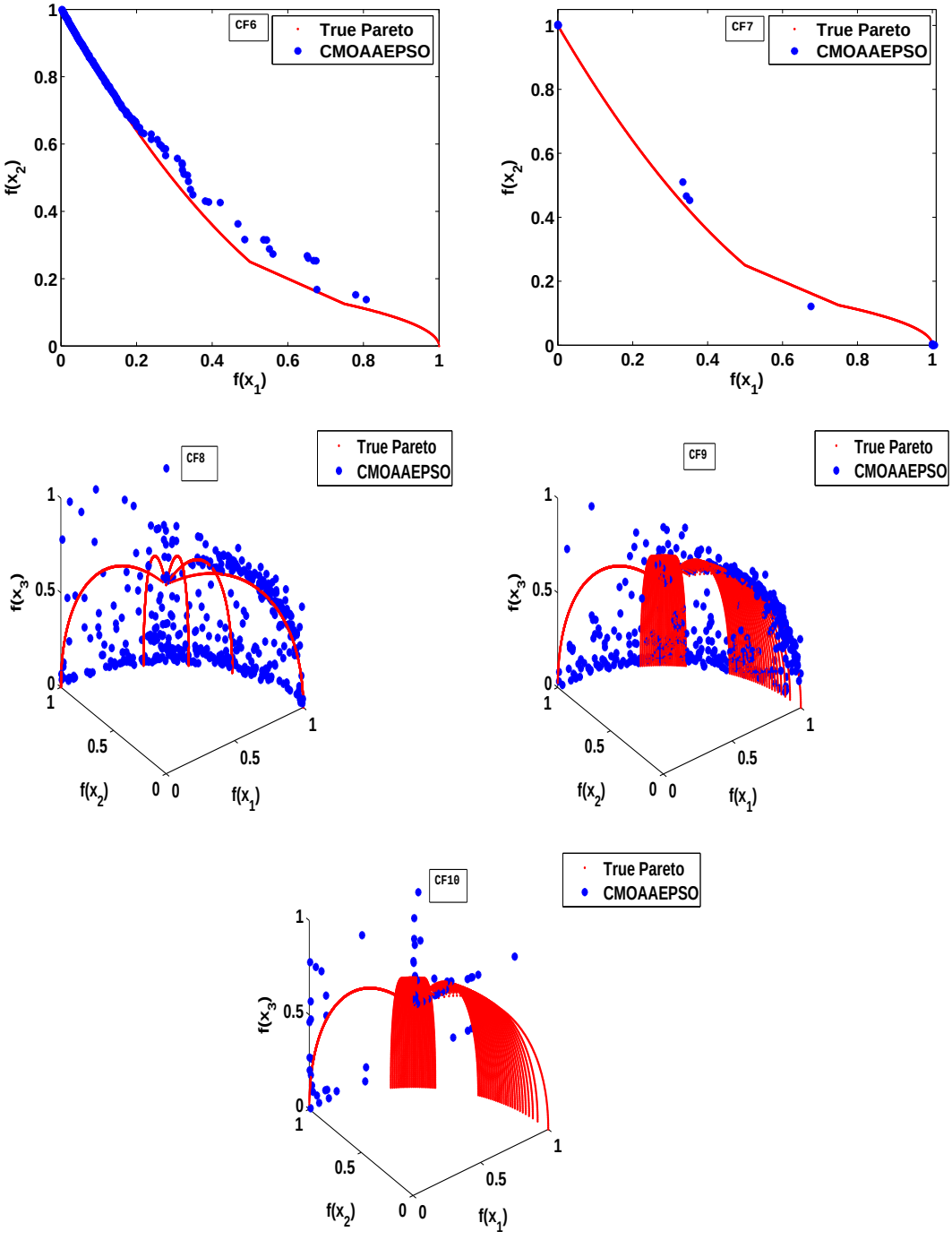

Figure 5.5: Pareto fronts on CF6 to CF10 by CMOAAEPSO

5.6.1.2 Average numerical results

The detailed results obtained by CMOAAEPSO on benchmark function chosen from CEC 2009 are presented in Table 5.1 and Table 5.2. The Table 5.1 presents complete statistical results (mean, best, worst and standard deviation) of IGD values obtained by CMOAAEPSO and similarly Table 5.2 presents convergence metrics. Both of these tables reveal the efficiency and effectiveness of the proposed algorithm. The proposed algorithm also shows its appreciable stability as is evident from the Tables 5.1 and 5.2. The comprehensive analysis of CMOAAEPSO on numerical benchmark functions with present state-of-the art are presented in Table 5.3. The results of algorithms used for comparison are taken from their respective literature [153, 154, 155, 156, 138, 151, 150, 152]. From the results presented in Table 5.3 it is evident that the CMOAAEPSO algorithm meets or beats most of the other competitor algorithms. The CMOAAEPSO ranks third on CF1 beating all presented algorithm except [151] and [138]. On CF2 CMOAAEPSO achieves 6^{th} rank and shows better result compared to [154], [138] and [153], similarly on CF3 it shows better result compared to [155], [156] and [153]. CMOAAEPSO ranks 4^{th} on CF4 and is better than all except [154], [152] and [150]. On CF5 CMOAAEPSO shows a poorer rank (8^{th}) and shows better results only with [156].

Table 5.1: Average IGD values obtained on (CF1 to CF10)

Function	mean	std	best	worst
CF1	5.0817e-3	1.3204e-3	1.0559e-3	1.2887e-2
CF2	2.5834e-2	2.6217e-2	1.0105e-2	1.2015e-1
CF3	2.0869e-1	3.0248e-1	9.0680e-3	7.8984e-1
CF4	1.3049e-2	6.3370e-3	8.1638e-3	1.0379e-1
CF5	5.1238e-1	1.1501e-1	2.6996e-1	7.4508e-1
CF6	3.7513e-2	3.9970e-3	1.3111e-2	4.6228e-2
CF7	2.3216e-1	2.0389e-1	1.1003e-1	6.1970e-1
CF8	1.0521e-1	1.5672e-2	9.6538e-2	1.4982e-1
CF9	1.0201e-1	1.4273e-2	6.1348e-2	1.6336e-1
CF10	3.8588e-1	6.5988e-2	1.0458e-1	8.1108e-1

The 5^{th} rank on CF6 shows CMOAAEPSO is better than [138], [152], [153] and [156]. The 6^{th} rank on CF7 shows CMOAAEPSO is better than [155], [153] and [156]. CMOAAEPSO shows comparatively good performance on CF8 and ranks third along with [150], [151] and next to [151], [154] on CF9. Finally CMOAAEPSO ranks 6^{th} on CF10 and is better than [152], [153] and [138].

Table 5.2: Average Convergence Metric values on (CF1 to CF10)

Function	mean	std	best	worst
CF1	1.0817e-2	1.3204e-3	8.8559e-3	1.2887e-2
CF2	5.5834e-2	2.6217e-2	3.1095e-2	1.2015e-1
CF3	2.9869e-1	3.0248e-1	7.9680e-2	7.8984e-1
CF4	9.0049e-2	6.3370e-3	8.0638e-2	1.0379e-1
CF5	6.1238e-1	1.1501e-1	3.6996e-1	7.4508e-1
CF6	3.7513e-2	3.9970e-3	3.3111e-2	4.6228e-2
CF7	4.3416e-1	2.0389e-1	1.3003e-1	6.1970e-1
CF8	1.1521e-1	1.5672e-2	9.6538e-2	1.4982e-1
CF9	1.3201e-1	1.4273e-2	1.1348e-1	1.6336e-1
CF10	6.8588e-1	6.5988e-2	6.5458e-1	8.1108e-1

5.6.1.3 Statistical metrices

The statistical significance of the algorithms are analyzed using hypervolume indicator followed by Wilcoxon comparison test. Table 5.4 presents the hypervolume indicator for all the numerical benchmark functions considered above and Table 5.5 presents the Wilcoxon matched-pairs signed rank test on hypervolume indicator. Table 5.4 presents the Hypervolume data on approximation Pareto fronts obtained by CMOAAEPSO and True pareto fronts. The Table 5.5 shows the results after Wilcoxon matched-pairs signed rank test on hypervolume data in Table 5.4. Since the difference between the medians of approximated pareto front obtained by CMOAAEPSO and True pareto front is not significant, hence it can be concluded that the results obtained by CMOAAEPSO are valid and the explanation made using conventional metrices holds good.

Table 5.3: Comparative results of average IGD obtained on CF1 to CF10

Rank	CF1		CF2		CF3		CF4	
1	LiuLiAlgo [151]	8.5000e-4	DMOEADD	2.1000e-3	DMOEADD	5.6305e-2	DMOEADD	6.9900e-3
2	C-PSA [160]	3.4200e-3	LiuLiAlgo	4.2000e-3	MTS	1.0446e-1	GDE3	7.9900e-3
3	**AAEPSO** [35]	**5.0817e-3**	MOEADGM	8.0000e-3	GDE3	1.2750e-1	MTS	1.1009e-2
4	NSGAIILS [155]	6.9200e-3	NSGAIILS	1.1830e-2	C-PSA	1.4430e-1	**AAEPSO**	**1.3049e-2**
5	MOEADGM [156]	1.0800e-2	GDE3	1.5970e-2	LiuLiAlgo	1.8290e-1	LiuLiAlgo	1.4230e-2
6	DMOEADD [150]	1.1310e-2	**AAEPSO**	**2.5834e-2**	**AAEPSO**	**2.0869e-1**	NSGAIILS	1.5760e-2
7	MTS [154]	1.9180e-2	MTS	2.6770e-2	NSGAIILS	2.3994e-1	MOEADGM	7.0700e-2
8	GDE3 [152]	2.9400e-2	C-PSA	3.4030e-2	MOEADGM	5.1340e-1	DECMOSA	1.5265e-1
9	DECMOSA [153]	1.0773e-1	DECMOSA	9.4600e-2	DECMOSA	1000000	C-PSA	5.88743

Rank	CF5		CF6		CF7		CF8	
1	DMOEADD	1.5770e-2	LiuLiAlgo	1.3948e-2	DMOEADD	1.9050e-2	DMOEADD	4.7501e-2
2	MTS	2.0770e-2	DMOEADD	1.5020e-2	MTS	2.4690e-2	LiuLiAlgo	6.0746e-2
3	GDE3	6.7990e-2	MTS	1.6160e-2	GDE3	4.1690e-2	**AAEPSO**	**1.0521e-1**
4	LiuLiAlgo	1.0973e-1	NSGAIILS	2.0130e-2	LiuLiAlgo	1.0446e-1	NSGAIILS	1.1093e-1
5	NSGAIILS	1.8420e-1	**AAEPSO**	**3.7513e-2**	C-PSA	1.8571e-1	GDE3	1.3872e-1
6	DECMOSA	4.1275e-1	C-PSA	5.9197e-2	**AAEPSO**	**2.3216e-1**	DECMOSA	1.7634e-1
7	C-PSA	4.4729e-1	GDE3	6.1990e-2	NSGAIILS	2.3345e-1	MOEADGM	4.0610e-1
8	**AAEPSO**	**5.1238e-1**	DECMOSA	1.4782e-1	DECMOSA	2.6049e-1	MTS	1.0854
9	MOEADGM	5.4460e-1	MOEADGM	2.0710e-1	MOEADGM	5.356e-1	C-PSA	NA

Rank	CF9		CF10	
1	LiuLiAlgo	5.0549e-2	MTS	1.3765e-1
2	MTS	8.5139e-2	DMOEADD	1.6213e-1
3	**AAEPSO**	**1.0201e-1**	LiuLiAlgo	1.9741e-1
4	NSGAIILS	1.0560e-1	MOEADGM	2.8540e-1
5	GDE3	1.1450e-1	NSGAIILS	3.5920e-1
6	DECMOSA	1.2713e-1	**AAEPSO**	**3.8588e-1**
7	MOEADGM	1.4080e-1	GDE3	4.9233e-1
8	DMOEADD	1.4343e-1	DECMOSA	5.0705e-1
9	C-PSA	NA	C-PSA	NA

Table 5.4: Hypervolume indicator

Funtion	TRUE	CMOAAEPSO
CF1	0.47317	0.48087
CF2	0.61297	0.57419
CF3	0.27198	0.26765
CF4	0.45436	0.35911
CF5	0.45226	0.1378
CF6	0.64354	0.69081
CF7	0.6405	0.10553
CF8	0.42678	0.95701
CF9	0.45002	0.86734
CF10	0.448	0

Table 5.5: Wilcoxon matched-pairs signed rank test

Source	Results
P value	0.6250
P value summary	ns
Are medians signif. different? ($P < 0.05$)	No
Sum of positive, negative ranks	33.00 , -22.00
Sum of signed ranks (W)	11.00
Was the pairing significantly effective?	No

5.6.2 Engineering design problems

The effectiveness of proposed CMOAAEPSO [157] algorithm is further validated on four engineering design problems chosen from [159, 144]. The engineering design problems are; a) Welded beam design problem [159], b) Two-bar truss design problem [159], c) Speed reducer design problem [144] and d) Gear train design [159]. The Pareto front graphs obtained by CMOAAEPSO on numerical functions BHN, OSY and SRK chosen from [159, 144] are presented in Figure 5.6. Similarly Pareto fronts on the engineering design problems chosen from [159, 144]

are presented from Figure 5.7 to Figure 5.10. The numerical results obtained by the CMOAAEPSO on the engineering design problems are presented from Table 5.6 to Table 5.9. Since these engineering design problems are MOCO problems, they are characterized by multiple solutions and all the solutions are equally important. Presenting all the possible numerical values of these problems is difficult and hence only extreme values are reported. The extreme values are, where one functions attains the maximum and other the minimum and vice versa. The solutions pertaining to these extreme objective values are also reported.

5.6.2.1 Pareto front

Pareto fronts obtained by CMOAAEPSO on BHN, OSY, SRK Figure 5.6. From Figure 5.6 it can be clearly concluded that proposed algorithm obtains excellent Pareto fronts.

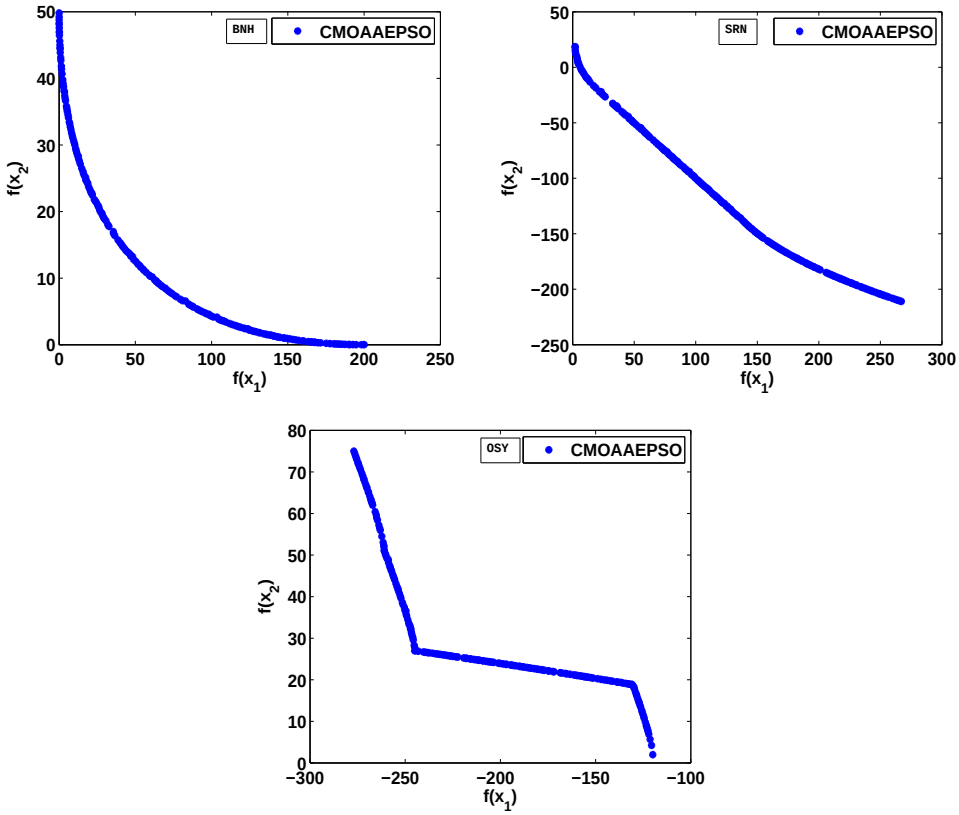

Figure 5.6: Pareto fronts on BNH, SRN and OSY

5.6.2.2 Welded beam design problem:

The objective of the welded beam design is to minimize the cost of fabrication and the end deflection [159]. The desired parameters are thickness of the beam (b), width of the beam (t), length of weld (l), and weld thickness (h) for which the cost of the beam and the deflection at the open end are minimum. The values are tabulated in Table 5.6. For minimum cost of (3.80711749689925), the set of parameter obtained are (h,l,t,b)=(0.801013, 7.009815, 0.09031,0.09100). For this minimum cost, the deflection is found to be 0.00795636796864. Similarly the minimum deflection of 4.447337842544898e-4 is obtained by the set of parameter (h,l,t,b)=(4.93598660079272, 10.0000, 0.1250, 0.1000). For this minimum deflection the cost is found to be 33.54406293268443. This is evident from Figure 5.7.

Table 5.6: Parameter values obtained on Welded beam design

Parameters	Values	cost	deflection
(h,l,t,b)	(0.801013, 7.009815, 0.09031,0.09100)	(3.80711749689925)	0.00795636796864
(h,l,t,b)	(4.93598660079272, 10.0000, 0.1250, 0.1000)	33.54406293268443	4.447337842544898e-4

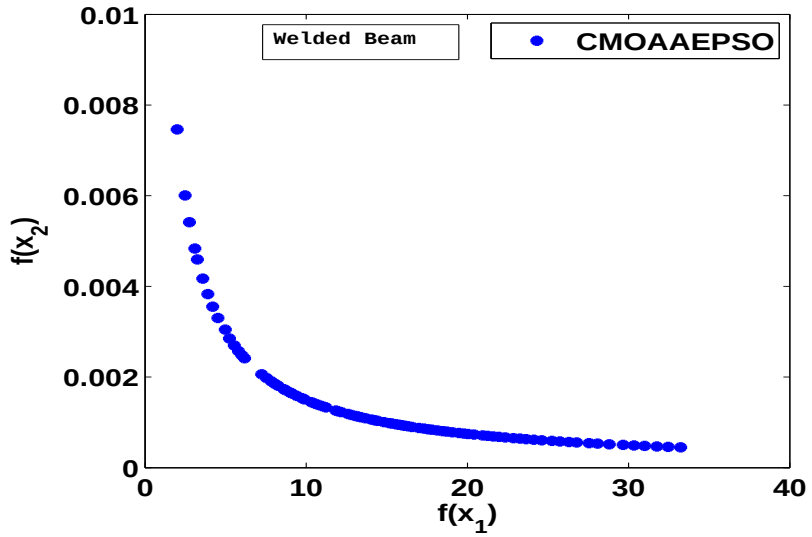

Figure 5.7: Pareto front on Welded beam design

5.6.2.3 Two-bar truss design problem:

This problem is aimed at designing the truss for minimum volume (minimum cost of fabrication) and minimizing stresses [159]. The values are tabulated in Table 5.7.

The minimum volume of (0.00312823966649) is obtained with inter bar distances (d_1, d_2, d_3) being (0.00065884461377, 0.000100, 1.88908672578715). With this set of parameters the stress in the bars is 9.051737919401531e+5. Similarly the minimum stress obtained with inter bar distances (0.00395438056030, 0.0100000, 3.000000) is (8.432740427115679e+3). With this stress the volume obtained is 0.05139467940320. This is evident from Figure 5.8.

Table 5.7: Parameter values obtained on Two-bar truss design

Parameters	Values	stress	volume
(d_1, d_2, d_3)	(0.00065884461377, 0.000100, 1.88908672578715)	9.051737919401531e+5	0.00312823966649
(d_1, d_2, d_3)	(0.00395438056030, 0.0100000, 3.000000)	8.432740427115679e+3	0.05139467940320

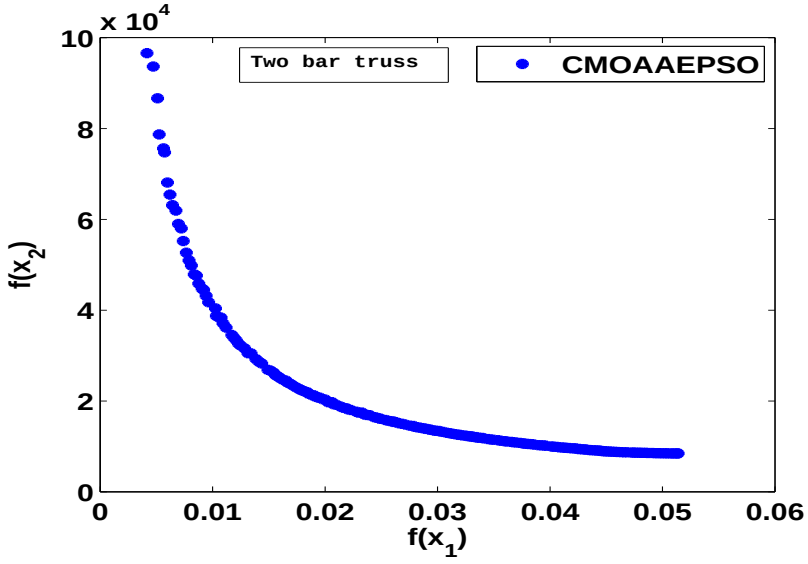

Figure 5.8: Pareto front on Two-bar truss design

5.6.2.4 Speed reducer design problem:

The objective of speed reducer design problem is to find the minimum volume of a gear box (the weight) and minimize the stress in gear shafts [144]. There are seven design variables x_1=Width of the gear face (cm), x_2=Teeth module (cm), x_3=Number of pinion teeth (Integer), x_4=Shaft 1 length between bearings (cm), x_5=Shaft 2 length between bearings (cm), x_6=Diameter of shaft 1 (cm) and x_7= Diameter of shaft 2 (cm). The values are tabulated in Table 5.8. The minimum gear box volume of (2.3524e+3) is obtained with design parameters $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$=(2.6000, 0.7000, 17.0000, 7.3000, 7.3000, 2.9000, 5.0000). With this set of parameters the stress in gear shafts is obtained 1.6960e+3. Similarly the minimum stress in gear shafts obtained (694.2336) with the design parameters to be $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$= (2.6000, 0.8000, 28.0000, 7.3000, 7.3000, 3.9000, 5.0000). With these parameters the gear box volume obtained is 5.3567e+3. This is evident from Figure 5.9.

Table 5.8: Parameter values obtained for Speed reducer design

Parameters	Values	stress	volume (the weight)
$(x_1, \ldots, x_7)$	(2.6000, 0.7000, 17.0000, 7.3000, 7.3000, 2.9000, 5.0000)	1.6960e+3	2.3524e+3
$(x_1, \ldots, x_7)$	(2.6000, 0.8000, 28.0000, 7.3000, 7.3000, 3.9000, 5.0000)	694.2336	5.3567e+3

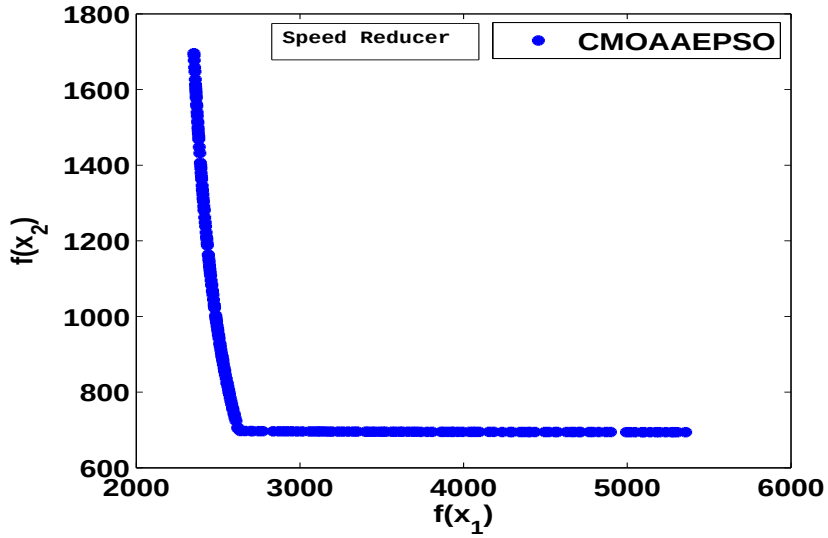

Figure 5.9: Pareto front on Speed reducer design

5.6.2.5 Gear train design:

The objective of the gear train design is to find the number of teeth in each of the four gears so as to minimize (i) the error between the obtained gear ratio and a required gear ratio of 1/6.931 and (ii) the maximum size of any of the four gears [159]. The values are tabulated in Table 5.9. This problem has four variables and they are number of teeth (t_1, t_2, t_3, t_4). Since the number of teeth must be integral the four variables are considered as integers. The minimum error obtained with design parameters (12, 12, 31, 31) is 3.096463758747167e-5. With these parameters the maximum size of four gears is 31. Similarly the maximum size of four gears of 12 is obtained with design parameters of (12, 12, 12, 12) and the error is 0.73225787401136. The solution of this may be in range as depicted from the Figure 5.10.

Table 5.9: Parameter values obtained on Gear train design

Parameters	Values	error	maximum size
$(t_1, \ldots, t_4)$	(12, 12, 31, 31)	3.096463758747167e-5	31
$(t_1, \ldots, t_4)$	(12, 12, 12, 12)	0.73225787401136	12

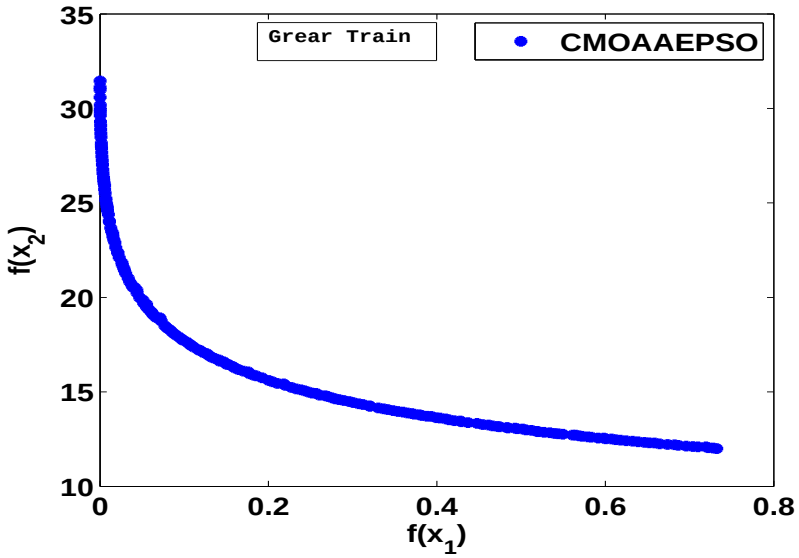

Figure 5.10: Pareto front on Gear train design

5.7 Conclusions

This chapter introduced the concept of Multi Objective Constrained Optimization (MOCO) along with a detailed literature survey. This chapter presented the challenges that PSO faces in handling multiple objectives and their associated constraints in detail. To address the problem of premature convergence and poor quality solution in PSO while solving multiple (often conflicting) objectives and constraints, the Constrained Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (CMOAAEPSO) was proposed. CMOAAEPSO divides the solution space in to four regions; a) Nondominated-Feasible, b) Dominnated-Feasible, c) Nondominated-Infeasible and d) Dominated-Infeasible. Among these four, only Nondominated-Feasible solution is the desired set and the others are undesired. The particles that belong to the undesired solution space were termed as diverged particles and special care was taken. CMOAAEPSO then accelerated these particles under the guidance of the desired solution (feasible-nondominated solution). Essentially these particles were forced to leave their current position and search the solution space around desired solution. In CMOAAEPSO the acceleration of diverged particles towards global solution was also done adaptively. The size of the search patch was more in the beginning of search process and it decreases exponentially with the iteration. This strategy in particular helps more exploration of search space around global solution in the beginning and more exploitation as search process proceeds. Further CMOAAEPSO has introduced particles probing concept. The particle probing concept was an extension of AAEPSO with single elite particles to all the feasible-nondominated particles. The comprehensive analysis of the developed algorithm was carried out on set of complex numerical benchmark and engineering design problems with the present state of the art. The analysis was carried out using conventional metrices (igd, convergence metrics and robustness), statistical (hypervolume and Wilcoxon tests) and graphical measures with the help of pareto fronts. The CMOAAEPSO has shown competitive performance comparable to the state of art on numerical benchmark problems and superior performance on engineering design problems.

Chapter 6

Particle Swarm Optimization Universal Solver

This chapter presents a user friendly Graphical User Interface (GUI) for solving optimization problems called Particle Swarm Optimization Universal Solver (PSOUS). The detailed literature survey on different GUI tools are presented. It also presents different optimization domains in brief. This chapter further discusses the developed algorithms in brief that are integrated together in PSOUS. The detailed architecture of PSOUS and different options available in PSOUS are discussed. The chapter is concluded with remarks on PSOUS.

6.1 Introduction

Particles Swarm Optimization (PSO) is a population based stochastic optimization algorithm, that simulates food foraging behavior of a swarm. PSO is highly effective for solving optimization problems, writing a computer program for implementing the PSO algorithms as per users need requires certain programming expertise along with considerable amount of time and effort. Since this task can be tedious and time consuming, the use of a PSO toolbox is more advantageous. There are toolboxes that are often developed for a specific problem [161] or for a specific optimization domain, but there is no single generic tool that can solve all the four types of optimization problem. Here the universal PSO solver tool is presented that encompasses all the domains of the optimization. There are mainly four types of optimization domains; they are unconstrained single objec-

tive, unconstrained multi objective, constrained single objective and constrained multi objective. Since the work has been carried out on all the possible domains of the optimization and has reported good algorithms in every domain, this has motivated us to develop an universal PSO tool, namely Particle Swarm Optimizer Universal Solver (PSOUS). All the reported algorithms are integrated in this solver to solve any kind of optimization problem. PSOUS is an interactive, graphically oriented and has all the attributes to be user friendly, developed in MATLAB environment.

6.2 Literature Survey

There are different optimization tools presented in literature. In 1999, Ulungu et al. [162] presented a tool called MOSA (MultiObjective Simulated Annealing) for solving multiobjective combinatorial optimization problems. This tool [162] is not modular and contains limited functionalities. In 2000 Erin Maneri and Wodek Gawronski developed a Graphical User Interface (GUI) to design Linear Quadratic Gaussian (LQG) [161] controllers applicable to antennas and radio telescopes. LQG algorithm has been used to control beam wave-guide and small antennas. The GUI in [161], was developed to simplify the design process and has user-friendly interface. In 2001, Tan et al. [163] developed an interactive graphical user interface (GUI) based on multiobjective evolutionary algorithm for computer-aided multiobjective optimization. This tool has used the concept of Pareto optimality for nondominated solutions distributing along the tradeoffs uniformly. It has features like goal and priority settings for better decision-making in multiobjective optimization, dynamic population size, settings for constraint handling, multiple goals specification for logical operation, adaptive niching scheme for uniform population distribution. In 2002, Luisa et al. proposed a generative and goal-oriented design tool [164]. The tool focuses on the aspects related to the environmental performance of buildings. It used Genetic Algorithms (GAs) as search procedure to look for optimized design solutions in terms of thermal and lighting performance in a building. It also helps in design problems like the choice of construction materials, design of shading elements, sizing of lighting and mechanical systems for buildings. In 2003, Johannes et al. proposed a Software Pipeline Optimization Tool (SPOT) [165]. The SPOT was based on visualization of the scheduled assembly code by a two-dimensional interactive schedule

editor. This has been equipped with feedback mechanisms of data dependencies and resource allocation conflicts. In 2004, Bartz et al. [166] developed a software package for development, analysis and application of multiobjective evolutionary algorithms and called as kit for evolutionary algorithms (KEA). The KEA tool [166] was written in Java and offers an object-oriented design to evaluate multi-objective fitness functions and display the progress of optimization in a dynamic display or results of optimization in a static visualization mode. In 2005, Deb and Chaudhuri [167] suggested an interactive procedure called an Interactive Evolutionary Multi-objective Optimization tool (I-EMO) which provides selecting one solution from the front and it helps to focus to on a single preferred solution. The I-EMO incorporated a decision-maker in the evolutionary optimization process and it provides a single solution at the end. In 2006, Hakanen et al. developed a tool [168], for chemical process that combines the rigorous process calculations of the BALAS process simulator and the interactive multiobjective optimization method NIMBUS. With this design tool, the designer can consider several conflicting performance criteria simultaneously. The interactive nature of this tool allows the designer to learn about the behavior of the problem. This tool [168] was validated on two applications viz. paper making and power plants. In 2007, Kollat and Reed [169] developed Visually Interactive Decision-making and design using Evolutionary multiobjective Optimization (VIDEO). In VIDEO users can navigate visually over large multiobjective solution sets and identify one or more optimal designs. This was demonstrated for a long-term groundwater monitoring (LTM) application. In 2008, the Genetic and Evolutionary Algorithm Toolbox (GEATbx) [170] was developed by Pohlheim. This tool was developed in Matlab and provided global optimization capabilities. It solves problems that are multimodal, discontinuous, stochastic, nonlinear and nondifferentiable. In 2009, Tahir et al. proposed a tool for multiobjective evolutionary algorithms [171]. The software tool in [171] was written in $C\#$ and gives the facility for the development, analysis and application of multiobjective evolutionary algorithms. It has features like visualizing of the progress and the results of optimization in a dynamic or static mode. This has been developed and well tested for multiobjective benchmark problems and two engineering design problems. In 2010, Boschetti et al. developed a tool for management decisions on ecological problems [172]. This method allows the goal of the management strategies as a result of the interaction between the user and the model, rather than being defined a priori.

6.3 Optimization domains

There are mainly four types of optimization domains; they are unconstrained single objective, unconstrained multi objective, constrained single objective and constrained multi objective. Most of the real world optimization problems can be converted into any of these domains. The simplified and generic mathematical representation is

$$\begin{aligned} Find\ \overrightarrow{x} &= (x_1, x_2, \ldots, x_D) \\ That\ optimizes\ & f_1(\overrightarrow{x}), f_2(\overrightarrow{x}), \ldots, f_k(\overrightarrow{x}) \\ subject\ to: \quad & g_i(\overrightarrow{x}) < 0, i = 1, 2, \ldots, q \\ & h_i(\overrightarrow{x}) = 0, i = q+1, q+2, \ldots, m \\ Where\ & l_i \leq x_i \leq u_i, i = 1, 2, \ldots, D \end{aligned} \tag{6.1}$$

Where $\overrightarrow{x}$ is the solution, k is the number of objectives that are to be simultaneously optimized, g_i, and h_i are the inequality and equality constraints respectively. D is the dimension of the problem, l_i and u_i are the lower and upper bounds $\forall$ $i \in D$ of the search space. In unconstrained single objective optimization there will be only one objective ($k = 1$) and no constraints ($q = m = 0$) and in case of constrained single objective ($k = 1$) optimization there will be constraints ($q > 0$ or $m > 0$) along with single objective. Similarly in unconstrained multi objective there will more than one objective ($k > 1$) for optimization without any constraints ($q = m = 0$) and in case of constrained multi objective there will be constraints ($q > 0$ or $m > 0$) along with multiple objectives ($k > 1$) for optimization.

6.4 Particle Swarm Optimizer Universal Solver

The Particle Swarm Optimizer Universal Solver (PSOUS) is a software tool that possesses the essential attributes of being interactive and graphically oriented, developed in MATLAB. Particle Swarm Optimizer (PSO) is highly effective at solving optimization problems, but writing a computer program for implementing the PSO algorithms as per the user's need requires certain programming expertise along with considerable amount of time and effort. Since this task of implemen-

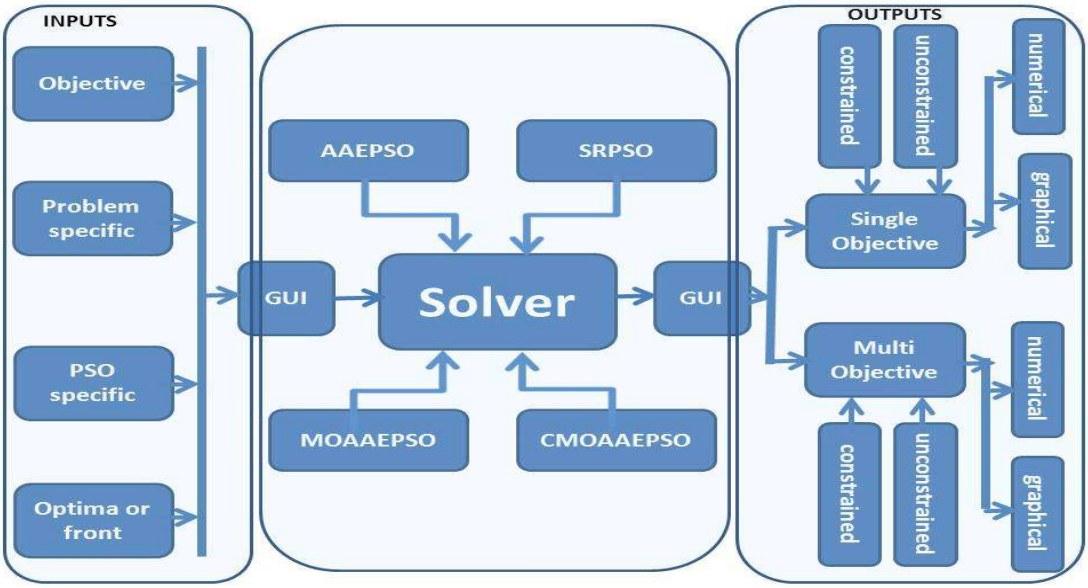

Figure 6.1: PSOUS Architecture

tation can be tedious and time consuming, the use of a PSO toolbox is more advantageous. There are toolboxes that are often developed for specific problem [161] or for specific type of optimization domain, but there is no single generic tool that can solve any of the four type of optimization problem. Here the universal PSO solver tool namely Particle Swarm Optimizer Universal Solver (PSOUS) is presented. It encompasses all the four domains of the optimization. The research work was carried out on all the said four optimization domains and reported the best performing algorithms in each of the domain. In unconstrained single objective optimization an Adaptive Accelerated Exploration Particle Swarm Optimizer (AAEPSO) was developed. The AAEPSO essentially identifies the particles responsible for premature convergence and accelerate them towards global solution. In constrained single objective optimization domain the Stochastic Ranking Particle Swarm Optimization (SRPSO) was developed. SRPSO is an integration of well known Stochastic Ranking along with AEPSO for handling constraints. Similarly Multi objective Adaptive Accelerated Exploration Particle Swarm Optimizer (MOAAEPSO) was developed for unconstrained multi objective optimization. MOAAEPSO also an extended version of AAEPSO was developed for optimizing multiple objective to generate Pareto optimal solution. Further AAEPSO was extended to handle constrained multi objective optimization problems and hence

reported Constrained Multi objective Adaptive Accelerated Exploration Particle Swarm Optimizer (CMOAAEPSO).

6.4.1 Architecture of PSOUS

The detailed architecture PSOUS is shown in Figure 6.1. PSOUS architecture shows the interconnections of inputs, algorithms and outputs. The input subsection of PSOUS architecture consists of 1) Objective, 2) Problems specific, 3) PSO specific and 4) Optima or front. The *Objective* input subsection of PSOUS gives the facility of specifying a type of problem. The type of problem may belong to either of four above mentioned optimization domain. The *Problem specific* subsection of POSUS accepts the problem related details like fitness function written in matlab, search range of the problem and number of objective. The *PSO specific* subsection accepts the PSO algorithmic related parameters like population size and maximum number of generations. Similarly *Optima or front* subsection of PSOUS accepts the known optima (single objective optimization) or true Pareto front (multi objective optimization) incase of standard benchmark problems. While solving problem of unknown optima or front, this should be left blank. The output subsection of PSOUS provides the output depending upon the type of the problem. The visual outputs are provided in two different format; 1) numerical and 2) graphical. The numerical results are problem dependant, for unconstrained and constrained single objective optimization the outputs are, optima achieved by the algorithm and solution to the problem. In case of known benchmark functions an error (absolute difference of known and achieved optima) will also be displayed. Similarly for unconstrained and constrained multi objective optimization the numerical outputs are, inverted generational distance, spacing, diversity and convergence metrics. The graphical output is also problem dependant, for unconstrained and constrained single objective optimization, it is an error graph (in case of standard benchmark functions) or fitness convergence graph (in case of normal function). Similarly for unconstrained and constrained multi objective optimization the graph is pareto front. The heart of the PSOUS is the algorithm integration with input and output subsection through GUI. This subsection consists of algorithms for specific type of optimization. Depending upon the type of the problem, a specific algorithm is made active; for unconstrained single objective optimization an AAEPSO is invoked, for unconstrained multi objective optimiza-

tion MOAAEPSO is invoked, for constrained single objective optimization SRPSO is invoked and for constrained multi objective optimization CMOAAEPSO is invoked.

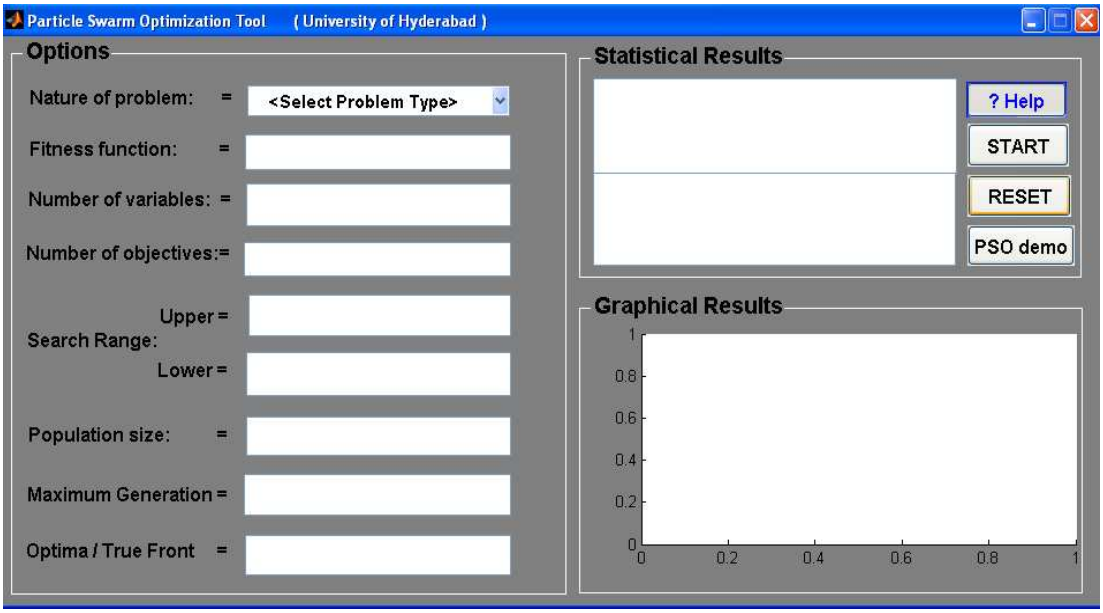

Figure 6.2: PSOUS Tool options

6.4.2 PSOUS options

PSOUS is a graphical tool that provides many user interactive options shown in Figure 6.2. The PSOUS has the facility of selecting types of optimization problem and accepts objective functions as matlab ($'.m'$) file. It also gives the facility to enter problem specific parameters like number of objectives, variables, search range (upper and lower limit) and stopping criteria. The algorithms terminates when it reaches the maximum number of iterations or stopping criteria value. For single objective unconstrained and constrained benchmark problems stopping criteria is the known optima value. For multi objective unconstrained and constrained benchmark problems true pareto front is the stopping criteria. It means the algorithm will run till the approximated pareto front is same as the true pareto front or it reaches maximum iterations, whichever is earlier. The option available in PSOUS to accepts optima or front as single value or set of values in ($'.dat'$) file format. The PSOUS accepts PSO specific parameters like population size and maximum number of iterations. It calls the appropriate algorithm (AAEPSO,

SRPSO, MOAAEPSO or CMOAAEPSO) and produces the output. The output from PSOUS can be seen in two different format, statistical and graphical. The statistical results are problem dependant; for single objective unconstrained and constrained problems, the output are; achieved optima, difference of true and achieved optima as an error, and the solution vector. In case of multi objective unconstrained and constrained problems, the statistical results are in the form of inverted generational distance, spacing, diversity and convergence metrics. The PSOUS also has an option for providing graphical output. For single objective unconstrained and constrained, it is an error convergence graph over iterations. For multi objective unconstrained and constrained it is a true pareto front and an approximated pareto front obtained by the algorithm. PSOUS has the following options:

1. Nature of problem: Here user can select the type of optimization problem.

2. Fitness function It accepts the objective function as ($'.m'$) matlab file. For example in case of
   a) Single Objective Unconstrained Optimization:
   $function f = Griewank(x)$
   $[ps, n] = size(x)$
   $fr = 4000; s = 0; p = 1;$
   $for j = 1 : n; s = s + x(j)^2; end$
   $for j = 1 : n; p = p * cos(x(j)/sqrt(j)); end$
   $f = s/fr - p + 1;$

   b) Single Objective Constrained Optimization:
   $function[f, g] = g08(x)$
   $f = ((sin(2 * pi * x(:, 1)).^3). * sin(2 * pi * x(:, 2)))./((x(:, 1).^3). * (x(:, 1) + x(:, 2)));$
   $g(:, 1) = x(:, 1).^2 - x(:, 2) + 1;$
   $g(:, 2) = 1 - x(:, 1) + (x(:, 2) - 4).^2;$

   c) Multi Objective Unconstrained Optimization:
   $function fit = UF1(x)$
   $x = x'; [dim, num] = size(x); tmp = zeros(dim, num);$
   $tmp(2 : dim, :) = (x(2 : dim, :) - sin(6.0 * pi * repmat(x(1, :), [dim - 1, 1]) + pi/dim * repmat((2 : dim)', [1, num]))).^2;$
   $tmp1 = sum(tmp(3 : 2 : dim, :)); tmp2 = sum(tmp(2 : 2 : dim, :));$
   $fit(1, :) = x(1, :) + 2.0 * tmp1/size(3 : 2 : dim, 2);$
   $fit(2, :) = 1.0 - sqrt(x(1, :)) + 2.0 * tmp2/size(2 : 2 : dim, 2);$
   $fit(3, :) = 0;$
   $fit = fit';$
   $cleartmp;$

   d) Multi Objective Constrained Optimization:
   $function[y, phi] = CF1(x)$
   $x = x'; a = 1.0; N = 10.0;$
   $[dim, num] = size(x); Y = zeros(dim, num);$
   $Y(2 : dim, :) = (x(2 : dim, :) - repmat(x(1, :), [dim - 1, 1]).^(0.5 + 1.5 * (repmat((2 : dim)', [1, num]) - 2.0)/(dim - 2.0))).^2;$
   $tmp1 = sum(Y(3 : 2 : dim, :));$
   $tmp2 = sum(Y(2 : 2 : dim, :));$
   $y(1, :) = x(1, :) + 2.0 * tmp1/size(3 : 2 : dim, 2);$
   $y(2, :) = 1.0 - x(1, :) + 2.0 * tmp2/size(2 : 2 : dim, 2);$
   $c(1, :) = y(1, :) + y(2, :) - a * abs(sin(N * pi * (y(1, :) - y(2, :) + 1.0))) - 1.0;$
   $y(3, :) = 0.0; clear Y; y = y'; phi = c';$

3. Number of variables: It accepts the number of variables of problem.
4. Number of objectives: It accepts the number of objectives of the problem.
5. Search Range: It accepts the upper and lower limit of the search space. User need to enter the ranges in the form of array. For example in case of 3 dimensional problem, the upper limit (treated $'1'$) search range is given as [1 1 1], and lower limit (treated $'0'$) is given as [0 0 0].
6. Population Size: It accepts size of the population.
7. Maximum generation: It accepts maximum number of iterations.
8. Optima / True Front: For known problems like benchmark functions the known optima can be given here. For unknown optima the algorithm will run for maximum number of iterations. For single objective, only one value is given and for multiobjective, true pareto front should given as ($'.dat'$) file. The ($'.dat'$) file should be a in column format.
9. Help: This gives the appropriate guidelines to user about the tool.
10. START: PSOUS will start the process after clicking this.
11. RESET: This will reset the PSOUS options selected earlier.
12. PSO demo: This option shows the working of basic PSO graphically.

The below figures from Figure 6.3 to Figure 6.6 show typical snapshots of PSOUS while solving different kind of problems.

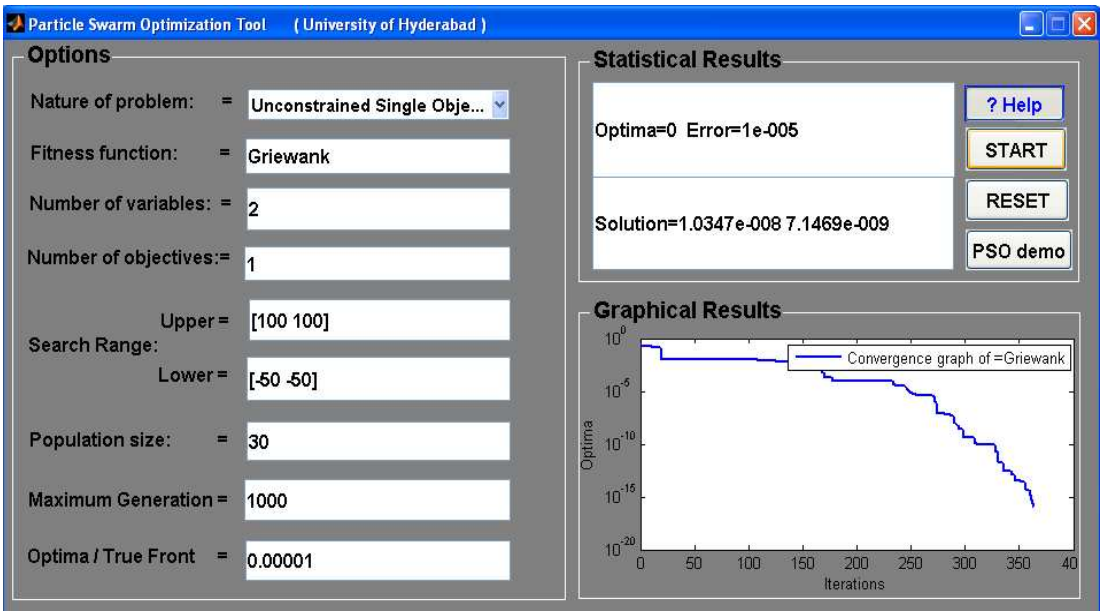

Figure 6.3: Snapshot for single objective unconstrained optimization problem

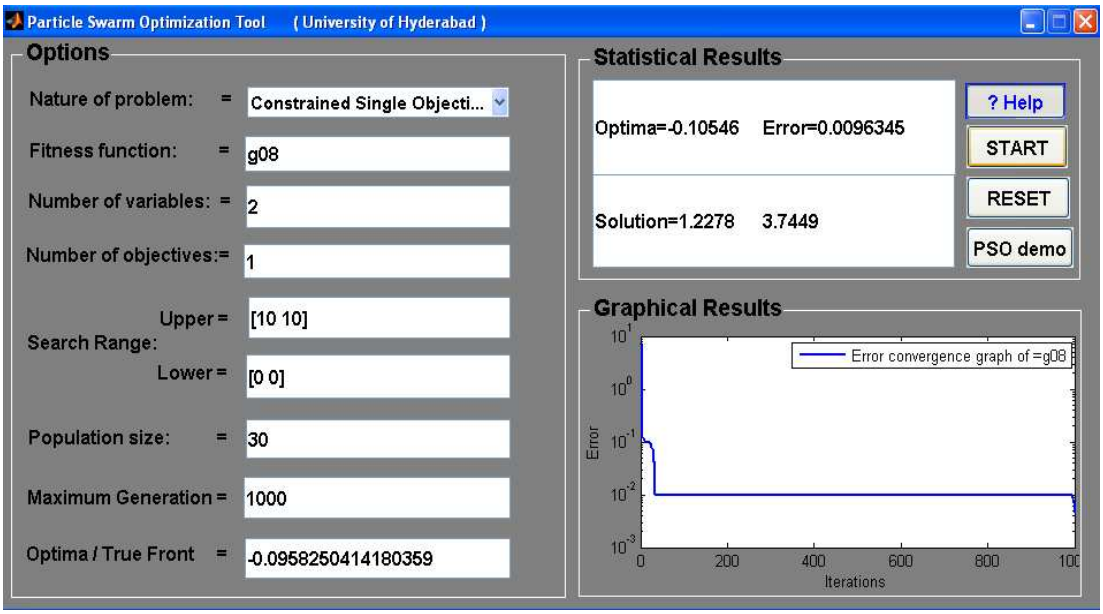

Figure 6.4: Snapshot for single objective constrained optimization problem

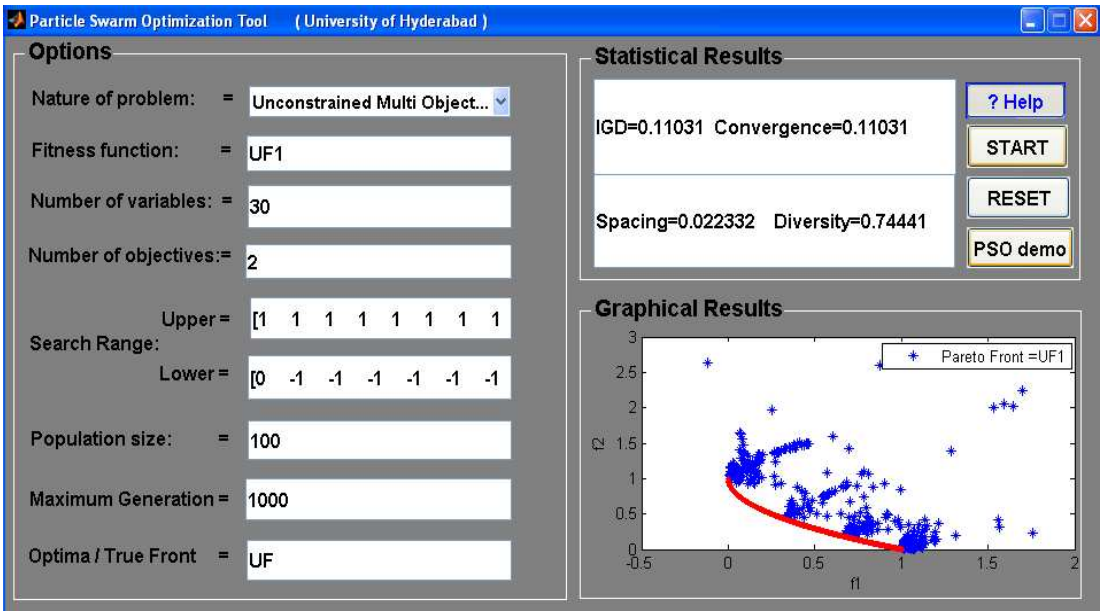

Figure 6.5: Snapshot for multi objective unconstrained optimization

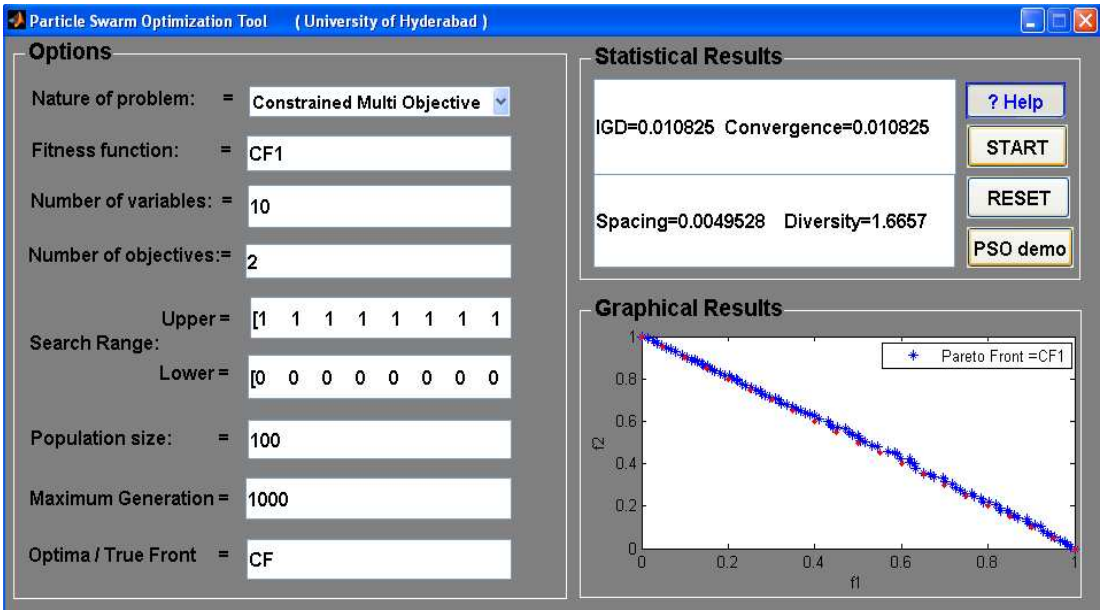

Figure 6.6: Snapshot for multi objective constrained optimization problem

6.5 Conclusions

Particles Swarm Optimization (PSO) is a population based stochastic optimization algorithm. Due to the simplicity of PSO, it has attracted many researchers across the globe. This chapter presented a detailed architecture of Particles Swarm Optimization Universal Solver (PSOUS) that encompasses all the domains of the optimization. The PSOUS is a software tool developed in MATALB with user friendly interfaces and possesses the essential attributes of being interactive and graphically oriented. PSOUS provides the necessary facilities to solve any kind of optimization problem. Rigorous experimental testing was carried out and found that the PSOUS tool works well on any kind of optimization problem. To the best of the author's knowledge this tool is first generic tool of its kind in the optimization domain. This tool will give the researchers across the globe to carry out further research and validate the results. This tool will be helpful even to industrialists and design engineers.

Chapter 7

Conclusions and Future scopes

This chapter presents the over all conclusions of research carried out on all the possible domains of optimization. The chapter is concluded with future scopes of research.

7.1 Conclusions

This thesis presented the basics of optimization, details of Swarm Intelligence and Particles Swarm Optimization (PSO). The detailed literature survey and research contribution on each of the four possible domains of optimization such as Single Objective Unconstrained Optimization (SOUO), Single Objective Constrained Optimization (SOCO), Multi Objective Unconstrained Optimization (MOUO) and Multi Objective Constrained Optimization (MOUO) are presented.

The major problems encountered by Particles Swarm Optimization (PSO) while solving four optimization domains were discussed. It was found that PSO shows premature convergence, poor quality of solution and uncertainty in solution while soving SOUO problems. To address the problem of premature convergence and poor quality of solution, the AEPSO algorithm was proposed. AEPSO first finds the particles responsible for premature convergence (diverged) that gives poor quality of solution. This occurs because the particles get stuck in deep local minima. AEPSO then accelerates these particles under the guidance of global best solution. Essentially these particles were forced to leave their current position and search the solution space around global solution. The stagnated particles were further manipulated by the introduction of hyperspherical coordinate updates

along with acceleration. This strategy is incorporated in proposed HAEPSO. In HAEPSO the information of the particles were first converted from cartesian coordinate system into hyperspherical coordinate system, then all the updates were carried out. This strategy helps in gaining the greater control of the particles flying trajectory. Further all the strategies developed were integrated together with well known comprehensive learning strategy and the new hybrid algorithm was called as ILPSO. AEPSO has elevated the prevailing problem of PSO to some extent. AEPSO accelerated the diverged particles with constant size search patch of unit radius (constant size) around global solution. This strategy helped more in exploitation of the solution than exploration. This is addressed in AAEPSO where the search patch size was adjusted dynamically. In AAEPSO the acceleration of diverged particles towards global solution was carried out adaptively. The size of the search patch was more at the beginning of search process and it decreases exponentially with the iteration. This strategy in particular helped for more exploration of search space around global solution in the beginning and more exploitation as search process proceeds. The comprehensive analysis of these developed algorithms were carried out on a set of complex multimodal functions. These functions were further rotated and shifted for thorough validation of proposed algorithms. The dimensions of the problems were chosen to be 10, 30, 50 and 100. The results were analyzed using well known performance metrices; Conventional and Statistical. The conventional metrices include convergence graphs, average of the results and robustness (standard deviation). The statistical metrices include Friedman′s test followed by Dunn′s multiple comparison test. The statistical results in terms of convergence, average results and robustness clearly proved that the developed algorithms shows remarkable improvements in the quality of solution and hence addresses the prevailing problems of PSO, this was further supported by statistical significant tests (Friedman′s and Dunn′s test). The ILPSO has shown superior performance over other algorithms including HAEPSO. The performance of ILPSO was not effected with the increase in the dimensions of the problems (evident from statistical results), specially on normal and rotated problems. ILPSO has shown excellent performance on normal and rotated problems of higher dimension. The AAEPSO has shown excellent convergence and quality of solutions on lower dimension of normal and rotated problems, but as dimensions of the problem increases, the AAEPSO failed to achieve good results. The major feature of AAEPSO has also been proved on shifted and shifted rotated prob-

lems for higher dimension. Among the four developed PSO variants ILPSO and AAEPSO has shown competitive performance among themselves and outperform other two (AEPSO and HAEPSO) algorithms.

PSO has neither internally nor externally has the mechanisms to handle constraints while solving SOCO problems. To incorporate the constrained handling method in PSO, Stochastic Ranking (SR) was integrated and hence proposed Stochastic Ranking Particle Swarm Optimization Ranking (SRPSO). Stochastic Ranking (SR) was the simple method to balance between over and underpenalization, additionally it is not dependant on nature of problem. SRPSO essentially a hybrid algorithm of AEPSO and stochastic ranking. This algorithm rank the particles based on stochastic ranking and accelerates the infeasible particles in the direction of feasible particles by maintaining the diversity in the population. The comprehensive experimental analysis of SRPSO was carried out with present state of the art on varied standard constrained benchmark problems with different complexities and on constrained engineering design problems and was compared with state of the art. The comprehensive analysis was carried out using metrices like average of the results, robustness (standard deviation) and function evaluations (FEs). The SRPSO has shown competitive performance with the state of art algorithms on numerical benchmark problems. The remarkable feature of SRPSO was seen in robustness test, i.e., SRPSO produced consistent results from run to run. It shows superior performance on engineering design problems compared to the current state of art algorithms. The SRPSO achieved good quality of solution with less number of functions evaluations with respect to current state of art. This algorithm serves to be an alternative option for solving engineering design problems where computational cost is the major constraint.

PSO finds difficulty while solving MOUO problems as it lack the mechanism to handle multiple objectives simultaneously. To address the problems of PSO and incorporate the multiple objective optimization Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (MOAAEPSO) was proposed. While solving multiple objectives PSO produces nondominated and dominated particles, and MOAAEPSO first separates the dominated particles (diverged) that gives less nondominated solution and are responsible for premature convergence. MOAAEPSO then accelerates these particles under the guidance of nondominated solutions. Essentially these particles were forced to leave their current position and search the solution space around nondominated solution. In MOAAEPSO

the acceleration of diverged particles towards nondominated solution was done adaptively. A search patch was maintained around the nondominated solution of the current iteration. The size of the search patch was decreased exponentially in a controlled manner and was kept to be more in the beginning of search process and decreased exponentially with the iteration. This strategy, in particular, helped more exploration of search space around nondominated solutions in the beginning and more exploitation as search process proceeds. The comprehensive analysis of the proposed algorithm was carried out on a set of complex CEC 2009 MOUO problems with the present state of the art techniques. The results in terms of pareto front graphs and various parametric measures clearly shows that the proposed algorithm shows remarkable improvements in the solution and hence addresses the prevailing problems of PSO. Further the analysis was carried out using Pareto compliant indicator (hypervolume). However the proposed algorithm is not superior to other algorithms in terms of hypervolume indicator.

Similarly PSO, neither has the mechanisms to handle constraints nor multiple objectives, while solving MOCO problems. To address the problem of premature convergence and poor quality solution while solving multiple (often conflicting) objectives and constraints, the Constrained Multiobjective Adaptive Accelerated Exploration Particle Swarm Optimizer (CMOAAEPSO) was proposed. CMOAAEPSO divides the solution space in to four regions; a) Nondominated-Feasible, b) Dominnated-Feasible, c) Nondominated-Infeasible and d) Dominated-Infeasible. Among these four, only Nondominated-Feasible solution is the desired and others are undesired solutions. The particles that belong to the undesired solution space were termed as diverged particles. CMOAAEPSO selects and accelerate these particles under the guidance of desired solution (feasible-nondominated solution). Essentially these particles were forced to leave their current position and search the solution space around the desired solution. In the proposed algorithm CMOAAEPSO, the acceleration of diverged particles towards global solution was carried out adaptively. The size of the search patch was more in the beginning of search process and it decreases exponentially with the iteration. This strategy in particular helps more exploration of search space around global solution in the beginning and more exploitation as search process proceeds. Further CMOAAEPSO has introduced particles probing concept. The particle probing concept was an extension of AAEPSO with single elite particles to all the feasible-nondominated particles. The comprehensive analysis of the developed algorithm was carried out

on set of complex numerical benchmark and engineering design problems with the present state of the art techniques. The analysis was carried out using conventional metrices (IGD, convergence metrics and robustness), statistical (hypervolume and Wilcoxon tests) and graphical measures with the help of pareto fronts. The CMOAAEPSO has shown competitive performance with state of the art on numerical benchmark problems chosen from CEC 2009. It has shown the superior performance on engineering deign problems such as; welded beam design, two-bar truss design, speed reducer design and gear train design problems.

Finally Particles Swarm Optimization Universal Solver (PSOUS) tool was developed. The PSOUS is a software tool developed in MATALB with user friendly interfaces and possesses the essential attributes of being interactive and graphically oriented. The PSOUS provides the necessary facilities to solve any class of optimization problems. Rigorous experimental testing was carried out and found that the PSOUS tool works well on any kind of optimization problem. To the best of authors knowledge this tool is first generic tool of its kind in the optimization domain. This tool will give the researchers across the globe to carry out further research and validate the results. This tool will be helpful even to industrialists and design engineers.

7.2 Future scopes

Research is a continuous and iterative process. The work presented in the thesis is not an exception. There are several interesting directions for further research and development based on the work in this research thesis.

1. On SOUO
    - PSO has major problem of premature convergence and produce poor quality of solutions. ILPSO and AAEPSO algorithms can be extended by hybridizing with other evolutionary or swarm techniques.
    - Can develop an intelligent algorithm that can sense and escape local optima and for this HAEPSO and AAEPSO can be hybridized together and tested.
    - ILPSO can be extended and tested for higher dimensions (say more than 100).
    - The search patch of AAEPSO can be made adaptive to nature of problem instead of decreasing exponentially with iterations.
    - PSO is very good algorithm for exploration (finding solutions though of poor quality in early stages), but has poor exploitation capability. PSO and its variants like AAEPSO can be hybridized with algorithms with good exploitation capability.
    - AAEPSO can be extended further for solving real-world dynamic problems.
2. On SOCO
    - PSO neither internally nor externally has the mechanism to handle constraints, thus more efficient constraint handling techniques can be integrated.
    - PSO update equations can be modified to accommodate the constraints.
    - Search patch of AAEPSO can be made adaptive based on constraint violations instead of decreasing with iterations.
3. On MOUO

- PSO neither internally nor externally has the mechanism to handle multiple objectives, thus more efficient techniques can be integrated.
- Multiple sub swarms can be maintained in MOAAEPSO, where each sub swarm optimize one objective and inter sub swarm information can be exchanged.
- The major problem with PSO is selecting gbest particle, more efficient method of selection can be used.
- MOAAEPSO can be run parallel with other bio-inspired algorithms for solving each objective, with certain information exchange.

4. On MOCO

- PSO neither internally nor externally has the mechanism to handle multiple objectives and constraints, thus more efficient techniques can be integrated.
- Two different swarms can be maintained in CMOAAEPSO, one for multiple objectives and other for constraints.
- CMOAAEPSO can be hybridized with other bio-inspired algorithms.

5. On tool box

- Comprehensive GUI tool can be developed for researchers to test state of the art with different options for tuning parameters of the algorithms.
- A common platform for all bio-inspired algorithms can be developed.
- User interactions can be made as part of search process.

Appendix A

Single Objective Unconstrained Benchmark Functions

Particle Swarm Optimization (PSO) like other evolutionary algorithms experience difficulty in optimizing functions with multiple optima, therefore the focus is on well-known standard benchmark functions [18] which have many local optima

A.1 Normal Benchmark functions

Ackley

The Ackley function can be unimodal or multimodal. At a low resolution the landscape of this is unimodal; however, the second exponential term covers the landscape with many small peaks and valleys.

$$\textbf{Definition: } f_1 = 20 + e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_i (x_i)^2}} - e^{-\frac{1}{n}} \sum_i^n cos(2\pi x_i)$$

$$\textbf{Search Space: } -30 \leq x_i \leq 30, i = 1, 2, \ldots, n$$

$$\textbf{Global optima: } x^* = (0, \ldots, 0); f_1(x^*) = 0$$

Griewank

Griewank function is a multimodal with multiple optima. It has a product term, introducing interdependency between the variables. This is intended to disrupt optimization techniques that work on one variable at a time. The term of the

summation produces a parabola, while the local optima are above the parabola.

$$\textbf{Definition: } f_2 = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{N} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$
$$\textbf{Search Space: } -600 \leq x_i \leq 600, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } x^* = (0, \ldots, 0); f_2(x^*) = 0$$

Powel

The Powel function is also highly multimodal and has several optima which are non-symmetrical and are randomly distributed.

$$\textbf{Definition: } f_3 = \sum_{i=1}^{n/4} (x_{4i-3} + 10x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^4$$
$$\textbf{Search Space: } -5 \leq x_i \leq 5, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } x^* = (0, \ldots, 0); f_3(x^*) = 0$$

Rastrigin

It is also multimodal function, where the value of multiple peaks increase as the distance from the global optimum point increases. The function is constructed from sphere, by adding a modulator term $'\alpha \cos(2\Pi x_i)'$. It is a nonlinear multimodal, however, the location of the optima are regularly distributed. It is fairly difficult problem due large search space and large number of local optima.

$$\textbf{Definition: } f_4 = \sum_{i=1}^{D} \left(x_i^2 - 10cos(2\pi x_i) + 10\right)$$
$$\textbf{Search Space: } -5.12 \leq x_i \leq 5.12, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } x^* = (0, \ldots, 0); f_4(x^*) = 0$$

Rosenbrock

The Rosenbrock function is characterized by an extremely deep valley along the parabola that leads to the global optima. Due to the non-linearity of the valley, many algorithms converge slowly because they change the direction of the search repeatedly. The function has a long gully with very steep walls and almost flat

bottom.

$$\textbf{Definition: } f_5 = \sum_{i=1}^{D-1} \left[100 \left(x_i^2 - x_{i+1} \right)^2 + (x_i - 1)^2 \right]$$
$$\textbf{Search Space: } -100 \leq x_i \leq 100, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } x^* = (1, \ldots, 1); f_5(x^*) = 0$$

Zakharov

Zakharov function is simple, strongly convex and can be considered as unimodal or multimodal.

$$\textbf{Definition: } f_6 = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=1}^{n} 0.5 i x_i \right)^2 + \left(\sum_{i=1}^{n} 0.5 i x_i \right)^4$$
$$\textbf{Search Space: } -10 \leq x_i \leq 10, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } x^* = (0, \ldots, 0); f_6(x^*) = 0$$

A.2 Rotated Benchmark functions

The functions are rotated by left multiplying the orthogonal matrix $'M'$ to the original value x to obtain the new rotated variable $y = M * x$. This variable y is used to calculate the fitness value. If $'M'$ is a D dimensional rotation matrix

$$M = \begin{bmatrix} m_{11} & m_{12} & \cdots & m_{1D} \\ m_{21} & m_{22} & \cdots & m_{2D} \\ \cdots & \cdots & \cdots & \cdots \\ m_{D1} & m_{D2} & \cdots & m_{DD} \end{bmatrix}, \quad then$$

$$y_k = m_{k1}x_1 + m_{k2}x_2 + m_{k3}x_3 + \cdots + m_{kD}x_D$$
$$Where \ k = 1, 2, 3, \cdots, D.$$

Rotated Ackley

$$\textbf{Definition: } f_7 = 20 + e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_i (y_i)^2}} - e^{-\frac{1}{n}} \sum_i^n cos(2\pi y_i),$$
$$Where\ y_i = M * x_i$$
$$\textbf{Search Space: } -30 \leq y_i \leq 30, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } y^* = (0, \ldots, 0); f_7(y^*) = 0$$

Rotated Griewank

$$\textbf{Definition: } f_8 = \sum_{i=1}^{D} \frac{y_i^2}{4000} - \prod_{i=1}^{N} cos\left(\frac{y_i}{\sqrt{i}}\right) + 1$$
$$Where\ y_i = M * x_i$$
$$\textbf{Search Space: } -600 \leq y_i \leq 600, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } y^* = (0, \ldots, 0); f_8(y^*) = 0$$

Rotated Powel

$$\textbf{Definition: } f_9 = \sum_{i=1}^{n/4} (y_{4i-3} + 10y_{4i-2})^2 + 5(y_{4i-1} - y_{4i})^2 +$$
$$(y_{4i-2} - y_{4i-1})^4 + 10(y_{4i-3} - y_{4i})^4$$
$$Where\ y_i = M * x_i$$
$$\textbf{Search Space: } -5 \leq y_i \leq 5, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } y^* = (0, \ldots, 0); f_9(y^*) = 0$$

Rotated Rastrigin

$$\textbf{Definition: } f_{10} = \sum_{i=1}^{D} \left(y_i^2 - 10cos(2\pi y_i) + 10\right)$$
$$Where\ y_i = M * x_i$$
$$\textbf{Search Space: } -5.12 \leq y_i \leq 5.12, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } y^* = (0, \ldots, 0); f_{10}(y^*) = 0$$

Rotated Rosenbrock

$$\textbf{Definition: } f_{11} = \sum_{i=1}^{D-1} \left[100 \left(y_i^2 - y_{i+1} \right)^2 + (y_i - 1)^2 \right]$$
$$Where \ y_i = M * x_i$$
$$\textbf{Search Space: } -100 \leq y_i \leq 100, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } y^* = (1, \ldots, 1); f_{11}(y^*) = 0$$

Rotated Zakharov

$$\textbf{Definition: } f_{12} = \sum_{i=1}^{n} y_i^2 + \left(\sum_{i=1}^{n} 0.5 i y_i \right)^2 + \left(\sum_{i=1}^{n} 0.5 i y_i \right)^4$$
$$Where \ y_i = M * x_i$$
$$\textbf{Search Space: } -10 \leq y_i \leq 10, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } y^* = (0, \ldots, 0); f_{12}(y^*) = 0$$

A.3 Shifted Benchmark functions

The functions are shifted by adding dynamic optima in the search range to the original value x to obtain the new shifted variable $y = x + O$. This variable y is used to calculate the fitness value. The dynamic shift is calculated as $O = R_{min} + 2 * R_{max} * rand(1, D)$, Where R_{max} and R_{min} is maximum and minimum of search range. The rand is the uniform random number in [0,1], and D is the dimension of the problem.

Shifted Ackley

$$\textbf{Definition: } f_{13} = 20 + e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_i (z_i)^2}} - e^{-\frac{1}{n}} \sum_{i}^{n} cos(2\pi z_i),$$
$$Where \ z_i = x - O \ and \ O = R_{min} + 2 * R_{max} * rand(1, D)$$
$$\textbf{Search Space: } -30 \leq z_i \leq 30, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } z^* = Dynamic; f_{13}(z^*) = 0$$

Shifted Griewank

$$\textbf{Definition: } f_{14} = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{N} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$$
$$Where\ z_i = x - O\ and\ O = R_{min} + 2 * R_{max} * rand(1, D)$$
$$\textbf{Search Space: } -600 \leq z_i \leq 600, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } z^* = Dynamic; f_{14}(z^*) = 0$$

Shifted Powel

$$\textbf{Definition: } f_{15} = \sum_{i=1}^{n/4} (z_{4i-3} + 10z_{4i-2})^2 + 5(z_{4i-1} - z_{4i})^2 + (z_{4i-2} - z_{4i-1})^4 + 10(z_{4i-3} - z_{4i})^4$$
$$Where\ z_i = x - O\ and\ O = R_{min} + 2 * R_{max} * rand(1, D)$$
$$\textbf{Search Space: } -5 \leq z_i \leq 5, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } z^* = Dynamic; f_{15}(z^*) = 0$$

Shifted Rastrigin

$$\textbf{Definition: } f_{16} = \sum_{i=1}^{D} \left(z_i^2 - 10cos(2\pi z_i) + 10\right)$$
$$Where\ z_i = x - O\ and\ O = R_{min} + 2 * R_{max} * rand(1, D)$$
$$\textbf{Search Space: } -5.12 \leq z_i \leq 5.12, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } z^* = Dynamic; f_{16}(z^*) = 0$$

Shifted Rosenbrock

$$\textbf{Definition: } f_{17} = \sum_{i=1}^{D-1} \left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$$
$$Where\ z_i = x - O\ and\ O = R_{min} + 2 * R_{max} * rand(1, D)$$
$$\textbf{Search Space: } -100 \leq z_i \leq 100, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } z^* = Dynamic; f_{17}(z^*) = 0$$

Shifted Zakharov

$$\textbf{Definition: } f_{18} = \sum_{i=1}^{n} z_i^2 + \left(\sum_{i=1}^{n} 0.5 i z_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i z_i\right)^4$$
$$Where\ z_i = x - O\ and\ O = R_{min} + 2 * R_{max} * rand(1, D)$$
$$\textbf{Search Space: } -10 \leq z_i \leq 10, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } z^* = Dynamic; f_{18}(z^*) = 0$$

A.4 Shifted and Rotated Benchmark functions

The function is Shifted by above (as in A.3) procedure and then rotated (as in A.2).

Shifted and Rotated Ackley

$$\textbf{Definition: } f_{19} = 20 + e^{-\frac{1}{5}\sqrt{\frac{1}{n}\sum_i (z_i)^2}} - e^{-\frac{1}{n}} \sum_{i}^{n} cos(2\pi z_i),$$
$$Where\ y_i = x - O,\ O = R_{min} + 2 * R_{max} * rand(1, D)$$
$$z_i = M * y_i$$
$$\textbf{Search Space: } -30 \leq z_i \leq 30, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } z^* = Dynamic; f_{19}(z^*) = 0$$

Shifted and Rotated Griewank

$$\textbf{Definition: } f_{20} = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{N} cos\left(\frac{z_i}{\sqrt{i}}\right) + 1$$
$$Where\ y_i = x - O,\ O = R_{min} + 2 * R_{max} * rand(1, D)$$
$$z_i = M * y_i$$
$$\textbf{Search Space: } -600 \leq z_i \leq 600, i = 1, 2, \ldots, n$$
$$\textbf{Global optima: } z^* = Dynamic; f_{20}(z^*) = 0$$

Shifted and Rotated Powel

$$\textbf{Definition: } f_{21} = \sum_{i=1}^{n/4} (z_{4i-3} + 10z_{4i-2})^2 + 5(z_{4i-1} - z_{4i})^2 + (z_{4i-2} - z_{4i-1})^4 + 10(z_{4i-3} - z_{4i})^4$$

$$Where\ \ y_i = x - O,\ \ O = R_{min} + 2 * R_{max} * rand(1, D)$$

$$z_i = M * y_i$$

$$\textbf{Search Space: } -5 \leq z_i \leq 5, i = 1, 2, \ldots, n$$

$$\textbf{Global optima: } z^* = Dynamic; f_{21}(z^*) = 0$$

Shifted and Rotated Rastrigin

$$\textbf{Definition: } f_{22} = \sum_{i=1}^{D} \left(z_i^2 - 10cos(2\pi z_i) + 10\right)$$

$$Where\ \ y_i = x - O,\ \ O = R_{min} + 2 * R_{max} * rand(1, D)$$

$$z_i = M * y_i$$

$$\textbf{Search Space: } -5.12 \leq z_i \leq 5.12, i = 1, 2, \ldots, n$$

$$\textbf{Global optima: } z^* = Dynamic; f_{22}(z^*) = 0$$

Shifted and Rotated Rosenbrock

$$\textbf{Definition: } f_{23} = \sum_{i=1}^{D-1} \left[100\left(z_i^2 - z_{i+1}\right)^2 + (z_i - 1)^2\right]$$

$$Where\ \ y_i = x - O,\ \ O = R_{min} + 2 * R_{max} * rand(1, D)$$

$$z_i = M * y_i$$

$$\textbf{Search Space: } -100 \leq z_i \leq 100, i = 1, 2, \ldots, n$$

$$\textbf{Global optima: } z^* = Dynamic; f_{23}(z^*) = 0$$

Shifted and Rotated Zakharov

$$\textbf{Definition: } f_{24} = \sum_{i=1}^{n} z_i^2 + \left(\sum_{i=1}^{n} 0.5 i z_i\right)^2 + \left(\sum_{i=1}^{n} 0.5 i z_i\right)^4$$

$$Where\ y_i = x - O,\ O = R_{min} + 2 * R_{max} * rand(1, D)$$

$$z_i = M * y_i$$

$$\textbf{Search Space: } -10 \leq z_i \leq 10, i = 1, 2, \ldots, n$$

$$\textbf{Global optima: } z^* = Dynamic; f_{24}(z^*) = 0$$

Appendix B

Single Objective Constrained Benchmark Functions

Particle Swarm Optimization (PSO) like any other population based algorithm, does not have internal or external mechanism to handle constraints. PSO finds itself difficult to optimize constraint problems. We have considered here well known numerical benchmark [74] and engineering design [43, 51] problems with varied complexity.

B.1 Numerical Benchmark functions

Table B.1: Characteristics of single objective constrained functions

Problem	n	Type of Function	ρ	LI	NI	LE	NE	a
g01	13	quadratic	0.0111 %	9	0	0	0	6
g02	20	nonlinear	99.9971 %	0	2	0	0	1
g03	10	polynomial	0.0000%	0	0	0	1	1
g04	5	quadratic	52.1230 %	0	6	0	0	2
g05	4	cubic	0.0000 %	2	0	0	3	3
g06	2	cubic	0.0066 %	0	2	0	0	2
g07	10	quadratic	0.0003 %	3	5	0	0	6
g08	2	nonlinear	0.8560%	0	2	0	0	0
g09	7	polynomial	0.5121%	0	4	0	0	2
g10	8	linear	0.0010%	3	3	0	0	6
g11	2	quadratic	0.0000 %	0	0	0	1	1
g12	3	quadratic	4.7713%	0	1	0	0	0
g15	3	quadratic	0.0000 %	0	0	1	1	2
g16	5	nonlinear	0.0204 %	4	34	0	0	4
g24	2	linear	79.6556%	0	2	0	0	2

Table B.2: Summary of single objective constrained benchmark functions

Fun	n	$f(\vec{x})$	Bounds
g01	13	-15.0000000000	$0 \leq x_i \leq 1$ $(i = 1, \ldots, 9)$, $0 \leq x_i \leq 100$ $(i = 10, 11, 12)$ and $0 \leq x_{13} \leq 1$
g02	20	-0.8036191042	$0 \leq x_i \leq 10$ $(i = 1, \ldots, n)$
g03	10	-1.0005001000	$0 \leq x_i \leq 1$ $(i = 1, \ldots, n)$
g04	5	-30665.5386717834	$78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45$ and $27 \leq x_i \leq 45$ $(i = 3, 4, 5)$
g05	4	5126.4967140071	$0 \leq x_1 \leq 1200, 0 \leq x_2 \leq 1200$, $-0.55 \leq x_3 \leq 0.55$ and $-0.55 \leq x_4 \leq 0.55$
g06	2	-6961.8138755802	$13 \leq x_1 \leq 100$ and $0 \leq x_2 \leq 100$
g07	10	24.3062090681	$-10 \leq x_i \leq 10$ $(i = 1, \ldots, 10)$
g08	2	-0.0958250415	$0 \leq x_1 \leq 10$ and $0 \leq x_2 \leq 10$
g09	7	680.6300573745	$-10 \leq x_i \leq 10$ $(i = 1, \ldots, 7)$
g10	8	7049.2480205286	$100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000$ $(i = 2, 3)$ *and* $10 \leq x_i \leq 1000$ $(i = 4, \ldots, 8)$
g11	2	0.7499000000	$-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$
g12	3	-1.0000000000	$0 \leq x_i \leq 10 (i = 1, 2, 3)$ *and* $p, q, r = 1, 2, \ldots, 9$
g15	3	961.7150222899	$0 \leq x_i \leq 10 (i = 1, 2, 3)$
g16	5	-1.9051552586	$704.4148 \leq x_1 \leq 906.3855, 68.6 \leq x_2 \leq 288.88$, $0 \leq x_3 \leq 134.75, 193 \leq x_4 \leq 287.0966$, $25 \leq x_5 \leq 84.1988$
g24	2	-5.5080132716	$0 \leq x_1 \leq 3$ and $0 \leq x_2 \leq 4$

g01

$$\textbf{Definition: } f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 + \sum_{i=5}^{1} 3x_i$$

$$\textbf{Subject to: } g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \leq 0$$

$$g_3(\vec{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \leq 0$$

$$g_4(\vec{x}) = -8x_1 + x_{10} \leq 0$$

$$g_5(\vec{x}) = -8x_2 + x_{11} \leq 0$$

$$g_6(\vec{x}) = -8x_3 + x_{12} \leq 0$$

$$g_7(\vec{x}) = -2x_4 - x_5 + x_{10} \leq 0$$

$$g_8(\vec{x}) = -2x_6 - x_7 + x_{11} \leq 0$$

$$g_9(\vec{x}) = -2x_8 - x_9 + x_{12} \leq 0$$

$$\textbf{Search Space: } 0 \leq x_i \leq 1 \ \ (i = 1, \ldots, 9), 0 \leq x_i \leq 100 (i = 10, 11, 12), 0 \leq x_{13} \leq 1$$

$$\textbf{Global optima: } x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1); f(x^*) = -15$$

g02

$$\textbf{Deinition: } f(x) = -\mid \frac{\sum_{i=1}^{n} cos^4(x_i) - 2\prod_{i=1}^{n} cos^2(x_i)}{\sqrt{\sum_{i=1}^{n} ix_i^2}} \mid$$

$$\textbf{Subject to: } g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0$$

$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0$$

$$\textbf{Search Space: } 0 \leq x_i \leq 10 \ \ (i = 1, \ldots, n), \ \ where \ \ n = 20$$

$$\textbf{Global optima: } x^* = (3.16246061572185, 3.12833142812967, 3.09479212988791,$$
$$3.06145059523469, 3.02792915885555, 2.99382606701730, 2.95866871765285,$$
$$2.92184227312450, 0.49482511456933, 0.48835711005490, 0.48231642711865,$$
$$0.47664475092742, 0.47129550835493, 0.46623099264167, 0.46142004984199,$$
$$0.45683664767217, 0.45245876903267, 0.44826762241853, 0.44424700958760,$$
$$0.44038285956317); f(x^*) = -0.80361910412559$$

g03

$$\textbf{Definition: } f(x) = -(\sqrt{n})^n \prod_{i=1}^{n} x_i$$

$$\textbf{Subject to: } h_1(x) = \sum_{i=1}^{n} x_i^2 - 1 \leq 0$$

$$\textbf{Search Space: } 0 \leq x_i \leq 1 \ \ (i = 1, \ldots, n), \ \ where \ \ n = 10$$

$$\textbf{Global optima: } x^* = (0.31624357647283069, 0.31624357414338339,$$
$$0.31624357801234592, 0.31624357566401789, 0.31624357820552606,$$
$$0.31624357738855069, 0.31624357547294951, 0.31624357716488393,$$
$$0.31624357815592030, 0.31624357614737491);$$
$$f(x^*) = -1.00050010001000$$

g04

Definition: $f(x) = 5.3578547x_2^3 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$

Subject to:

$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$

$$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$

$$g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_2^3 - 110 \leq 0$$

$$g_4(x) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_2^3 + 90 \leq 0$$

$$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

Search Space: $78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45$ *and* $27 \leq x_i \leq 45$ $(i = 3, 4, 5)$

Global optima: $x^* = (78, 33, 29.9952560256815985, 45, 36.7758129057882073)$;

$f(x^*) = -3.066553867178332e + 4$

g05

Definition: $f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$

Subject to: $g_1(x) = -x_4 + x_3 - 0.55 \leq 0$

$$g_2(x) = -x_3 + x_4 - 0.55 \leq 0$$

$$h_3(x) = 1000sin(-x_3 - 0.25) + 1000sin(-x_4 - 0.25) + 894.8 - x_1 = 0$$

$$h_4(x) = 1000sin(x_3 - 0.25) + 1000sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0$$

$$h_5(x) = 1000sin(x_4 - 0.25) + 1000sin(x_4 - x_3 - 0.25) + 1294.8 = 0$$

Search Space: $0 \leq x_1 \leq 1200, 0 \leq x_2 \leq 1200, -0.55 \leq x_3 \leq 0.55$ *and* $-0.55 \leq x_4 \leq 0.55$

Global optima: $x^* = (679.945148297028709, 1026.06697600004691,$

$0.118876369094410433, -0.39623348521517826)$;

$f(x^*) = 5126.4967140071$

g06

$$\textbf{Definition: } f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$
$$\textbf{Subject to: } g_1(x) = -(x_1 - 5)^2 - (x_2 - 5)^2 + 100 \leq 0$$
$$g_2(x) = (x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$$
$$\textbf{Search Space: } 13 \leq x_1 \leq 100 \ \textit{and} \ 0 \leq x_2 \leq 100$$
$$\textbf{Global optima: } x^* = (14.09500000000000064, 0.8429607892154795668);$$
$$f(x^*) = -6961.81387558015$$

g07

$$\textbf{Definition: } f(x) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3 - 10)^2 + 4(x_4 - 5)^2$$
$$+(x_5 - 3)^2 + 2(x_6 - 1)^2 + 5x_7^2 + 7(x_8 - 11)^2 + 2(x_9 - 10)^2 + (x_{10} - 7)^2 + 45$$
$$\textbf{Subject to: } g_1(x) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \leq 0$$
$$g_2(x) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \leq 0$$
$$g_3(x) = -8x_1 + 2x_2 + 5x_9 - 2x_10 - 12 \leq 0$$
$$g_4(x) = 3(x_1 - 2)^2 + 4(x_2 - 3)^2 + 2x_3^2 - 7x_4 - 120 \leq 0$$
$$g_5(x) = 5x_21 + 8x_2 + (x_3 - 6)^2 - 2x_4 - 40 \leq 0$$
$$g_6(x) = x_21 + 2(x_2 - 2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \leq 0$$
$$g_7(x) = 0.5(x_1 - 8)^2 + 2(x_2 - 4)^2 + 3x_5^2 - x_6 - 30 \leq 0$$
$$g_8(x) = -3x_1 + 6x_2 + 12(x_9 - 8)^2 - 7x_{10} \leq 0$$
$$\textbf{Search Space: } -10 \leq x_i \leq 10 (i = 1, \ldots, 10)$$
$$\textbf{Global optima: } x^* = (2.17199634142692, 2.3636830416034,$$
$$8.77392573913157, 5.09598443745173, 0.990654756560493,$$
$$1.43057392853463, 1.32164415364306, 9.82872576524495,$$
$$8.2800915887356, 8.3759266477347);$$
$$f(x^*) = 24.30620906818$$

g08

$$\textbf{Definition: } f(x) = \frac{\sin^3(2\Pi x_1)\sin(2\Pi x_2)}{x_1^3(x_1+x_2)}$$

$$\textbf{Subject to: } g_1(x) = x_1^2 - x_2 + 1 \leq 0$$

$$g_2(x) = 1 - x_1 + (x_2 - 4)^2 \leq 0$$

$$\textbf{Search Space: } 0 \leq x_i \leq 10 (i = 1, 2)$$

$$\textbf{Global optima: } x^* = (1.22797135260752599, 4.24537336612274885);$$

$$f(x^*) = -0.0958250414180359$$

g09

$$\textbf{Definition: } f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2$$

$$+10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$$

$$\textbf{Subject to: } g_1(x) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$$

$$g_2(x) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$$

$$g_3(x) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$$

$$g_4(x) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$$

$$\textbf{Search Space: } -10 \leq x_i \leq 10 (i = 1, \ldots, 7)$$

$$\textbf{Global optima: } x^* = (2.33049935147405174, 1.95137236847114592,$$

$$-0.477541399510615805, 4.36572624923625874, -0.624486959100388983,$$

$$1.03813099410962173, 1.5942266780671519);$$

$$f(x^*) = 680.630057374402$$

g10

$$\textbf{Definition: } f(x) = x_1 + x_2 + x_3$$
$$\textbf{Subject to: } g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0$$
$$g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0$$
$$g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0$$
$$g_4(x) = -x_1x_6 + 833.33252x_4 + 100x_1 - 83333.333 \leq 0$$
$$g_5(x) = -x_2x_7 + 1250x_5 + x_2x_4 - 1250x_4 \leq 0$$
$$g_6(x) = -x_3x_8 + 1250000 + x_3x_5 - 2500x_5 \leq 0$$
$$\textbf{Search Space: } 100 \leq x_1 \leq 10000, 1000 \leq x_i \leq 10000 (i = 2, 3)$$
$$and \ \ 10 \leq x_i \leq 1000 (i = 4, \ldots, 8)$$
$$\textbf{Global optima: } x^* = (579.306685017979589, 1359.97067807935605,$$
$$5109.97065743133317, 182.01769963061534, 295.601173702746792,$$
$$217.982300369384632, 286.41652592786852, 395.601173702746735);$$
$$f(x^*) = 7049.24802052867$$

g11

$$\textbf{Definition: } f(x) = x_1^2 + (x_2 - 1)^2$$
$$\textbf{Subject to: } h_1(x) = x_2 - x_1^2 = 0$$
$$\textbf{Search Space: } -1 \leq x_i \leq 1 (i = 1, 2)$$
$$\textbf{Global optima: } x^* = (-0.707036070037170616, 0.500000004333606807);$$
$$f(x^*) = 0.7499$$

g12

$$\textbf{Definition: } f(x) = (100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2)/100$$
$$\textbf{Subject to: } g_1(x) = (x_1 - p)2 + (x_2 - q)2 + (x_3 - r)2 - 0.0625 \leq 0$$
$$\textbf{Search Space: } 0 \leq x_i \leq 10 \ \ (i = 1, 2, 3) \ \ p, q, r = 1, \ldots, 9.$$
$$\textbf{Global optima: } x^* = (5, 5, 5); f(x^*) = -1$$

g15

$$\begin{aligned}
\textbf{Definition: } & f(x) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3 \\
\textbf{Subject to: } & h_1(x) = x_1^2 + x_2^2 + +x_3^2 - 25 = 0 \\
& h_2(x) = 8x_1 + 14x_2 + 7x_3 - 56 = 0 \\
\textbf{Search Space: } & 0 \leq x_i \leq 10 \ \ (i = 1, 2, 3) \\
\textbf{Global optima: } & x^* = (3.51212812611795133, 0.216987510429556135, \\
& 3.55217854929179921) \ \ f(x^*) = 961.715022289961
\end{aligned} \tag{B.1}$$

g16

$$\begin{aligned}
\textbf{Definition: } & f(x) = 0.000117y_{14} + 0.1365 + 0.00002358y_{13} + 0.000001502y_{16} \\
& +0.0321y_{12} + 0.004324y_5 + 0.0001c_{15}c_{16} + 37.48y_2c_{12} - 0.0000005843y_{17} \\
\textbf{Subject to: } & g_1(x) = \frac{0.28}{0.72}y_5 - y_4 \leq 0, \ g_2(x) = x_3 - 1.5x_2 \leq 0 \\
& g_3(x) = 3496\frac{y_2}{c_{12}} - 21 \leq 0, \ g_4(x) = 110.6 + y_1 - \frac{62212}{c_{17}} \leq 0 \\
& g_5(x) = 213.1 - y_1 \leq 0, \ g_6(x) = y_1 - 405.23 \leq 0 \\
& g_7(x) = 17.505 - y_2 \leq 0, \ g_8(x) = y_2 - 1053.6667 \leq 0 \\
& g_9(x) = 11.275 - y_3 \leq 0, \ g_{10}(x) = y_3 - 35.03 \leq 0 \\
& g_{11}(x) = 214.228 - y_4 \leq 0, \ g_{12}(x) = y_4 - 665.585 \leq 0 \\
& g_{13}(x) = 7.458 - y_5 \leq 0, \ g_{14}(x) = y_5 - 584.463 \leq 0 \\
& g_{15}(x) = 0.961 - y_6 \leq 0, \ g_{16}(x) = y_6 - 265.916 \leq 0 \\
& g_{17}(x) = 1.612 - y_7 \leq 0, \ g_{18}(x) = y_7 - 7.046 \leq 0 \\
& g_{19}(x) = 0.146 - y_8 \leq 0, \ g_{20}(x) = y_8 - 0.222 \leq 0 \\
& g_{21}(x) = 107.99 - y_9 \leq 0, \ g_{22}(x) = y_9 - 273.366 \leq 0 \\
& g_{23}(x) = 922.693 - y_{10} \leq 0, \ g_{24}(x) = y_{10} - 1286.105 \leq 0 \\
& g_{25}(x) = 926.832 - y_{11} \leq 0, \ g_{26}(x) = y_{11} - 1444.046 \leq 0 \\
& g_{27}(x) = 18.766 - y_{12} \leq 0, \ g_{28}(x) = y_{12} - 537.141 \leq 0 \\
& g_{29}(x) = 1072.163 - y_{13} \leq 0, \ g_{30}(x) = y_{13} - 3247.039 \leq 0 \\
& g_{31}(x) = 8961.448 - y_{14} \leq 0, \ g_{32}(x) = y_{14} - 26844.086 \leq 0
\end{aligned} \tag{B.2}$$

$$g_{33}(x) = 0.063 - y_{15} \leq 0,\ g_{34}(x) = y_{15} - 0.386 \leq 0$$

$$g_{35}(x) = 71084.33 - y_{16} \leq 0,\ g_{36}(x) = -140000 + y_{16} \leq 0$$

$$g_{37}(x) = 2802713 - y_{17} \leq 0,\ g_{38}(x) = y_{17} - 12146108 \leq 0$$

$$Where,$$

$$y_1 = x_2 + x_3 + 41.6,\ c_1 = 0.024x_4 - 4.62$$

$$y_2 = \frac{12.5}{c_1} + 12,\ c_2 = 0.0003535x_2 1 + 0.5311x_1 + 0.08705y_2x_1$$

$$c_3 = 0.052x_1 + 78 + 0.002377y_2x_1,\ y_3 = \frac{c_2}{c_3},\ y_4 = 19y_3$$

$$c_4 = 0.04782(x_1 - y_3) + \frac{0.1956(x_1 - y_3)^2}{x_2} + 0.6376y_4 + 1.594y_3$$

$$c_5 = 100x_2,\ c_6 = x_1 - y_3 - y_4,\ c_7 = 0.950 - \frac{c_4}{c_5},\ y_5 = c_6c_7$$

$$y_6 = x_1 - y_5 - y_4 - y_3,\ c_8 = (y_5 + y_4)0.995$$

$$y_7 = \frac{c_8}{y_1},\ y_8 = c_8 3798,\ c_9 = y_7 - \frac{0.0663y_7}{y_8} - 0.3153,\ y_9 = \frac{96.82}{c_9} + 0.321y_1$$

$$y_{10} = 1.29y_5 + 1.258y_4 + 2.29y_3 + 1.71y_6,\ y_{11} = 1.71x_1 - 0.452y_4 + 0.580y_3$$

$$c_{10} = \frac{12.3}{752.3},\ c_{11} = (1.75y_2)(0.995x_1),\ c_{12} = 0.995y_{10} + 1998,\ y_{12} = c_{10}x_1 + \frac{c_{11}}{c_{12}}$$

$$y_{13} = c_{12} - 1.75y_2,\ y_{14} = 3623 + 64.4x_2 + 58.4x_3 + \frac{146312}{y_9 + x_5}$$

$$c_{13} = 0.995y_{10} + 60.8x_2 + 48x_4 - 0.1121y_{14} - 5095$$

$$y_{15} = \frac{y_{13}}{c_{13}},\ y_{16} = 148000 - 331000y_{15} + 40y_{13} - 61y_{15}y_{13}$$

$$c_{14} = 2324y_{10} - 28740000y_2,\ y_{17} = 14130000 - 1328y_{10} - 531y_{11} + \frac{c_{14}}{c_{12}}$$

$$c_{15} = y_{13}y_{15} - y_{13}0.52,\ c_{16} = 1.104 - 0.72y_{15}$$

$$c_{17} = y_9 + x_5$$

Search Space: $704.4148 \leq x_1 \leq 906.3855, 68.6 \leq x_2 \leq 288.88, 0 \leq x_3 \leq 134.75,$

$193 \leq x_4 \leq 287.0966\ and\ 25 \leq x_5 \leq 84.1988$

Global optima: $x^* = (705.174537070090537, 68.5999999999999943,$

$102.899999999999991, 282.324931593660324, 37.5841164258054832)$

$$f(x^*) = -1.90515525853479$$

g24

$$\begin{aligned}
\textbf{Definition: } & f(x) = -x_1 - x_2 \\
\textbf{Subject to: } & g_1(x) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leq 0 \\
& g_2(x) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leq 0 \\
\textbf{Search Space: } & 0 \leq x_1 \leq 3 and 0 \leq x_2 \leq 4 \\
\textbf{Global optima: } & x^* = (2.32952019747762, 3.17849307411774 \\
& f(x^*) = 5.50801327159536
\end{aligned}$$

B.2 Engineering design problems

Welded Beam Design

The problem is to design a welded beam for minimum cost of fabrication, subject to some constraints. The objective is to find the minimum fabrication cost, considering four design variables: x_1, x_2, x_3, x_4 and constraints of shear stress, bending stress in the beam, buckling load on the bar and end deflection on the beam.

$$\begin{aligned}
\textbf{Definition: } & f(x) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2) \\
\textbf{Subject to: } & g_1(x) = -(13000 - \tau(\overrightarrow{x})) \leq 0 \\
& g_2(x) = -(30000 - \sigma(\overrightarrow{x})) \leq 0 \\
& g_3(x) = x_1 - x_4 \leq 0 \\
& g_4(x) = 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \leq 0 \\
& g_5(x) = 0.125 - x_1 \leq 0 \\
& g_6(x) = \delta(\overrightarrow{x}) - 0.25 \leq 0 \\
& g_7(x) = -(P_c(x) - 6000) \leq 0 \\
& Where \\
& \tau = \sqrt{(\tau')^2 + \frac{2\tau'\tau''}{2R} + (\tau'')^2} \\
& \tau' = \frac{6000}{\sqrt{2}x_1x_2} \\
& \tau'' = \frac{MR}{J}
\end{aligned} \tag{B.3}$$

$$M = 6000(14 + \frac{x_2}{2})$$
$$R = \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2}$$
$$J = 2\{\sqrt{2}x_1x_2[\frac{x_2^2}{12} + (\frac{x_1 + x_2}{2})^2]\}$$
$$\sigma(\overrightarrow{x}) = \frac{504000}{x_3^2x_4}$$
$$\delta(\overrightarrow{x}) = \frac{2.1952}{x_3^3x_4}$$
$$P_c = 64746.022(1 - 0.0282346x_3)x_3x_4^3$$
$$\textbf{Search Space: } 0.1 \leq x_1, x_4 \leq 2,\ 0.1 \leq x_2, x_3 \leq 10$$
$$\textbf{Dimnesion: } \quad n = 4$$

Pressure Vessel Design

The main aim of Pressure Vessel Design problem is to minimize the total cost (f(x)), including the cost of the material, forming and welding. It is a four variable design problem; x_1 thickness of the shell, x_2 thickness of the head, x_3 inner radius and x_4 length of the cylindrical section of the vessel, not including the head. Among the four variables, x_1 and x_2 are integer multiples of 0.0625in that are the available thicknesses of rolled steel plates, and x_3 and x_4 are continuous variables. The problem can be formulated as follows

$$\textbf{Definition: } f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3.$$
$$\textbf{Subject to: } g_1(x) = -x_1 + 0.0193x_3 \leq 0$$
$$g_2(x) = -x_2 + 0.00954x_3 \leq 0$$
$$g_3(x) = -\Pi x_3^2x_4 - \frac{4}{3}\Pi x_3^3 + 1296000 \leq 0$$
$$g_4(x) = x_4 - 240 \leq 0$$
$$\textbf{Search Space: } 1 \leq x_1, x_2 \leq 99,\ 10 \leq x_3, x_4 \leq 200$$
$$\textbf{Dimnesion: } \quad n = 4$$

Speed Reducer design

The well known speed reducer test problem represents the design of a simple gear box such as might be used in a light airplane between the engine and propeller to allow each to rotate at its most efficient speed. The objective is to minimize the speed reducer weight while satisfying a number of constraints imposed by gear and

shaft design practices. There are seven design variables, $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. x_1 Width of the gear face (cm), x_5 Shaft 2 length between bearings (cm), x_2 Teeth module (cm) x6 Diameter of shaft 1 (cm), x_3 Number of pinion teeth (Integer), x_7 Diameter of shaft 2 (cm), x_4 Shaft 1 length between bearings (cm).

$$\textbf{Definition: } f_1(x) = 0.7854x_1x_2^2(10x_3^2/3 + 14.933x_3 - 43.0934)$$
$$-1.508x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$
$$\textbf{Subject to: } g_1(x) = \frac{27}{x_1x_2^2x_3} - 1 \leq 0$$
$$g_2(x) = \frac{397.5}{x_1x_2^2x_3^2} - 1 \leq 0$$
$$g_3(x) = \frac{1.93x_4^3}{x_2x_3x_6^4} - 1 \leq 0$$
$$g_4(x) = \frac{1.93x_5^3}{x_2x_3x_7^4} - 1 \leq 0$$
$$g_5(x) = \frac{\sqrt{(745x_4/x_2x_3)^2 + 1.69 \times 10^7}}{110.0x_6^3}$$
$$g_6(x) = \frac{\sqrt{(745x_5/x_2x_3)^2 + 15.75 \times 10^7}}{85.0x_7^3}$$
$$g_7(x) = \frac{x_2x_3}{40} - 1 \leq 0$$
$$g_8(x) = \frac{5x_2}{x_1} - 1 \leq 0$$
$$g_9(x) = \frac{x_1}{12x_2} - 1 \leq 0$$
$$g_{10}(x) = \frac{1.5x_6 + 1.9}{x_4} - 1 \leq 0$$
$$g_{11}(x) = \frac{1.1x_7 + 1.9}{x_5} - 1 \leq 0$$
$$\textbf{Search Space: } 2.6 \leq x_1 \leq 3.6,\ 0.7 \leq x_2 \leq 0.8,$$
$$17 \leq x_3 \leq 28, 7.3 \leq x_4, x_5 \leq 8.3, 2.9 \leq x_6 \leq 3.9, 5.0 \leq x_7 \leq 5.5$$
$$\textbf{Dimension: } n = 7$$

Three-bar Truss Design

The truss has to carry a certain load without elastic failure. Thus, in addition to the objective of designing the truss for minimum volume (which is equivalent to designing for minimum cost of fabrication), there are additional constraints of

minimizing stresses in each of the two members.

$$\textbf{Definition: } f_1(x) = (2\sqrt{2}x_1 + x_2)l$$

$$\textbf{Subject to: } g_1(x) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0$$

$$g_2(x) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2}P - \sigma \leq 0$$

$$g_3(x) = \frac{1}{\sqrt{2}x_2 + x_1}P - \sigma \leq 0$$

$$Where$$

$$l = 100cm,\ P = 2kN/cm^2\ and\ \sigma = 2kN/cm^2$$

$$\textbf{Search Space: } 0 \leq x_{1,2} \leq 1$$

$$\textbf{Dimnesion: } \quad n = 2$$

Himmelblau Nonlinear problem

This problem was proposed by Himmelblau and similar to problem g04 of the benchmark except for the second coefficient of the first constraint. There are five design variables

$$\textbf{Definition: } f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$$

$$\textbf{Subject to:}$$

$$g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0$$

$$g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0$$

$$g_3(x) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_2^3 - 110 \leq 0$$

$$g_4(x) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_2^3 + 90 \leq 0$$

$$g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0$$

$$g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0$$

$$\textbf{Search Space: } 78 \leq x_1 \leq 102, 33 \leq x_2 \leq 45\ \ and\ \ 27 \leq x_i \leq 45\ \ (i = 3, 4, 5)$$

$$\textbf{Dimnesion: } \quad n = 5$$

Appendix C

Multi Objective Unconstrained Benchmark Functions

Multi Objective Unconstrained Function are those function in which multiple objective functions are solved simultaneously with no additional constraints, and vary in levels of complexity. The well known benchmark functions are described below.

Table C.1: Summary of multiobjective unconstrained benchmark functions

Problem	Objective	Modality	Convexity	Connectivity
SCH	f_1	Unimodal	Convex	Connected
	f_2	Unimodal		
FON	f_1	Unimodal	Non Convex	Connected
	f_2	Unimodal		
POL	f_1	Multimodal	Non Convex	Disconnected
	f_2	Unimodal		
KUR	f_1	Unimodal	Non Convex	Disconnected
	f_2	Multimodal		
ZDT1	f_1	Unimodal	Convex	Connected
	f_2	Unimodal		
ZDT2	f_1	Unimodal	Non Convex	Connected
	f_2	Unimodal		
ZDT3	f_1	Unimodal	Convex	Disconnected
	f_2	Multimodal		
ZDT4	f_1	Unimodal	Non Convex	Connected
	f_2	Multimodal		
ZDT4	f_1	Multimodal	Non Convex	Connected
	f_2	Multimodal		

KUR

The KUR (Kursawe) function has the Pareto optimal set that is nonconvex as well as discontinuous, and, thus, the values of the decision variables that correspond to the true Pareto optimal solutions are difficult to obtain. The KUR problem has three disconnected Pareto-optimal regions, which may cause difficulty in finding nondominated solutions in all regions. It is a multimodal problem and has three disconnected curves in its Pareto front.

$$\textbf{Definition: } f_1(x) = \sum_{i=1}^{n-1}(-10exp(-0.2(\sqrt{x_i^2 + x_{i+1}^2}))$$
$$f_2(x) = \sum_{i=1}^{n}(|x_i|^{0.8} + 5\sin(x_i^3))$$
$$\textbf{Search Space: } -5 \leq x_i \leq 5, i = 1, 2, \ldots, n$$
$$\textbf{Dimension: } n = 3$$

FON

FON is a complex problem and has convex Pareto front.

$$\textbf{Definition: } f_1(x) = 1 - exp(-\sum_{i=1}^{n}(x_i - \frac{1}{\sqrt{3}})^2)$$
$$f_2(x) = 1 - exp(-\sum_{i=1}^{n}(x_i + \frac{1}{\sqrt{3}})^2)$$
$$\textbf{Search Space: } -4 \leq x_i \leq 4, i = 1, 2, \ldots, n$$
$$\textbf{Dimension: } n = 3$$

SCH

FON is also a complex problem and has convex Pareto front.

$$\textbf{Definition: } f_1(x) = x^2$$
$$f_2(x) = (x-2)^2$$
$$\textbf{Search Space: } -10^3 \leq x_i \leq 10^3, i = 1, 2, \ldots, n$$
$$\textbf{Dimension: } n = 1$$

POL

It is multimodal problem with many local optima. It has nonconvex and discon-

nected Pareto front. There are two disconnected curves in its Pareto front.

$$\begin{aligned}
\textbf{Definition: } f_1(x) &= [1 + (g_1 - h_1)^2 + (g_2 - h_2)^2] \\
f_2(x) &= [(x_1 + 3)^2 + (x_2 + 1)^2] \\
g_1 &= 0.5 \sin 1 - 2 \cos 1 + \sin 2 - 1.5 \cos 2 \\
g_2 &= 1.5 \sin 1 - \cos 1 + 2 \sin 2 - 0.5 \cos 2 \\
h_1 &= 0.5 \sin x_1 - 2 \cos x_1 + \sin x_2 - 1.5 \cos x_2 \\
h_2 &= 1.5 \sin x_1 - \cos x_1 + 2 \sin x_2 - 0.5 \cos x_2 \\
\textbf{Search Space: } & -\pi \leq x_i \leq \pi, i = 1, 2, \ldots, n \\
\textbf{Dimension: } & n = 2
\end{aligned}$$

ZDT1

The Pareto front of ZDT1 is convex and there is no dependency among the design variables. It has got the large number of variables.

$$\begin{aligned}
\textbf{Definition: } f_1(x) &= x_1 \\
f_2(x) &= g(x)[1 - \sqrt{\frac{x_1}{g(x)}}] \\
Where\ g(x) &= 1 + 9 \sum_{i=2}^{n} \frac{x_i}{n-1} \\
\textbf{Search Space: } & 0 \leq x_i \leq 1, i = 1, 2, \ldots, n \\
\textbf{Dimension: } & n = 30
\end{aligned}$$

ZDT2

The Pareto front of ZDT2 is concave (nonconvex) and there is no dependency among the design variables.

$$\begin{aligned}
\textbf{Definition: } f_1(x) &= x_1 \\
f_2(x) &= g(x)[1 - (\frac{x_1}{g(x)})^2] \\
Where\ g(x) &= 1 + 9 \sum_{i=2}^{n} \frac{x_i}{n-1} \\
\textbf{Search Space: } & 0 \leq x_i \leq 1, i = 1, 2, \ldots, n \\
\textbf{Dimension: } & n = 30
\end{aligned}$$

ZDT3

The Pareto front of ZDT3 is discontinuous and there is no dependency among the design variables. The Pareto front of this problem is made up of four disjoint curves and is considered to be multimodal.

$$\textbf{Definition: } f_1(x) = x_1$$
$$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}} - ((\frac{x_1}{g(x)})\sin(10\pi x_1))]$$
$$Where\ \ g(x) = 1 + 9\sum_{i=2}^{n}\frac{x_i}{n-1}$$
$$\textbf{Search Space: } 0 \leq x_i \leq 1, i = 1, 2, \ldots, n$$
$$\textbf{Dimension: } \quad n = 30$$

ZDT4

This is also a complex multimodal and has nonconvex Pareto front.

$$\textbf{Definition: } f_1(x) = x_1$$
$$f_2(x) = g(x)[1 - \sqrt{\frac{x_1}{g(x)}}]$$
$$Where\ \ g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)]$$
$$\textbf{Search Space: } 0 \leq x_i \leq 1, i = 1, 2, \ldots, n$$
$$\textbf{Dimension: } \quad n = 30$$

ZDT6

ZDT6 problem has nonconvex and nonuniformly spaced Pareto-optimal fronts, their images crowd in a corner of the Pareto front in the objective space. The adverse density of solutions across the Pareto-optimal front, together with the nonconvex nature of the front, makes it difficult for many multiobjective optimization algorithms to maintain a well-distributed nondominated set and converge to the true Pareto-optimal front. It is treated to be multimodal and has five disconnected

curves in Pareto front.

$$\textbf{Definition: } f_1(x) = 1 - exp(-4x_1)\sin^6(4\pi x_1)$$
$$f_2(x) = g(x)[1 - (\frac{f_1(x)}{g(x)})^2]$$
$$Where\ g(x) = 1 + 9[\sum_{i=2}^{n} \frac{x_i}{n-1}]^{0.25}$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1 \qquad -5 \leq x_i \leq 5, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } n = 10$$

UF1

$$\textbf{Definition: } f_1(x) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}[x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2$$
$$f_2(x) = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}[x_j - \sin(6\pi x_1 + \frac{j\pi}{n})]^2$$
$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \leq j \leq n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \leq j \leq n\}$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1 \qquad -1 \leq x_i \leq 1, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } n = 30$$

UF2

$$\textbf{Definition: } f_1(x) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1} y_j^2$$
$$f_2(x) = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2} y_j^2$$
$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \leq j \leq n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \leq j \leq n\}\ and$$
$$y_j = x_j - [0.3x_1^2\cos(24\pi x_1 + \frac{4j\pi}{n} + 0.6x_1)]\cos(6\pi x_1 + \frac{j\pi}{n})\ \ j \in J_1$$
$$y_j = x_j - [0.3x_1^2\cos(24\pi x_1 + \frac{4j\pi}{n} + 0.6x_1)]\sin(6\pi x_1 + \frac{j\pi}{n})\ \ j \in J_2$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1 \qquad -1 \leq x_i \leq 1, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } n = 30$$

UF3

$$\textbf{Definition: } f_1(x) = x_1 + \frac{2}{|J_1|}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$$

$$f_2(x) = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$$

$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \le j \le n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \le j \le n\}\ and$$

$$y_j = x_j - x_1^{0.5(1.0+\frac{3(j-2)}{n-2})} \qquad j = 2, 3, \ldots, n$$

$$\textbf{Search Space: } 0 \le x_i \le 1, i = 1, 2, \ldots, n$$

$$\textbf{Dimension: } n = 30$$

UF4

$$\textbf{Definition: } f_1(x) = x_1 + \frac{2}{|J_1|}(2\sum_{j\in J_1} h(y_j)$$

$$f_2(x) = 1 - x_1^2 + \frac{2}{|J_2|}2\sum_{j\in J_2} h(y_j)$$

$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \le j \le n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \le j \le n\}\ and$$

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}) \qquad j = 2, 3, \ldots, n$$

$$and \qquad h(t) = \frac{|t|}{1 + e^{2|t|}}$$

$$\textbf{Search Space: } 0 \le x_1 \le 1 \qquad -2 \le x_i \le 2, i = 2, 3, \ldots, n$$

$$\textbf{Dimension: } n = 30$$

UF5

$$\textbf{Definition: } f_1(x) = x_1 + (\frac{1}{2N} + \epsilon)|\sin(2N\pi x_1)| + \frac{2}{|J_1|}(2\sum_{j\in J_1} h(y_j)$$

$$f_2(x) = 1 - x_1 + (\frac{1}{2N} + \epsilon)|\sin(2N\pi x_1)| + \frac{2}{|J_2|}2\sum_{j\in J_2} h(y_j)$$

$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \le j \le n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \le j \le n\}\ and$$

(C.1)

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}) \qquad j = 2, 3, \ldots, n$$
$$\textit{and} \qquad h(t) = 2t^2 - \cos(4\pi t) + 1$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1 \qquad -1 \leq x_i \leq 1, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } \quad n = 30$$

UF6

Definition:

$$f_1(x) = x_1 + max\{0, 2(\frac{1}{2N} + \epsilon)|\sin(2N\pi x_1)|\} + \frac{2}{|J_1|}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_i\pi}{\sqrt{j}}) + 2)$$
$$f_2(x) = 1 - x_1 + max\{0, 2(\frac{1}{2N} + \epsilon)|\sin(2N\pi x_1)|\} + \frac{2}{|J_2|}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_2}\cos(\frac{20y_i\pi}{\sqrt{j}}) + 2)$$

$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \leq j \leq n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \leq j \leq n\}\ and$

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}) \qquad j = 2, 3, \ldots, n$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1 \qquad -1 \leq x_i \leq 1, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } \quad n = 30$$

UF7

$$\textbf{Definition: } f_1(x) = \sqrt[5]{x_1} + \frac{2}{|J_1|}\sum_{j\in J_1} y_j^2$$
$$f_2(x) = 1 - \sqrt[5]{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2} y_j^2$$

$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \leq j \leq n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \leq j \leq n\}\ and$

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}) \qquad j = 2, 3, \ldots, n$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1 \qquad -1 \leq x_i \leq 1, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } \quad n = 30$$

UF8

$$\textbf{Definition: } f_1(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_2(x) = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_3(x) = \sin(0.5x_1\pi) + \frac{2}{|J_3|}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$Where$$

$$J_1 = \{j|3 \ \leq j \leq n \ and \ j-1 \ is \ a \ multiplication \ of \ 3\},$$

$$J_2 = \{j|3 \ \leq j \leq n \ and \ j-2 \ is \ a \ multiplication \ of \ 3\},$$

$$J_3 = \{j|3 \ \leq j \leq n \ and \ j \ is \ a \ multiplication \ of \ 3\},$$

$$\textbf{Search Space: } 0 \leq x_1 \leq 1, \quad 0 \leq x_2 \leq 1, \quad -2 \leq x_i \leq 2, i = 3, 4, \ldots, n$$

$$\textbf{Dimension: } n = 30$$

UF9

Definition:

$$f_1(x) = 0.5[max\{0, (1+\epsilon)(1-4(2x_1-1)^2)\} + 2x_1]x_2 + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_2(x) = 0.5[max\{0, (1+\epsilon)(1-4(2x_1-1)^2)\} - 2x_1 + 2]x_2 + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_3(x) = 1 - x_2 + \frac{2}{|J_3|}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$Where$$

$$J_1 = \{j|3 \ \leq j \leq n \ and \ j-1 \ is \ a \ multiplication \ of \ 3\},$$

(C.2)

$$J_2 = \{j|3 \ \leq j \leq n \ and \ j-2 \ is \ a \ multiplication \ of \ 3\},$$
$$J_3 = \{j|3 \ \leq j \leq n \ and \ j \ is \ a \ multiplication \ of \ 3\},$$
$$and \ \epsilon = 0.1; \ \epsilon \ can \ take \ any \ other \ positive \ value$$

Search Space: $0 \leq x_1 \leq 1, \quad 0 \leq x_2 \leq 1, \quad -2 \leq x_i \leq 2, i = 3, 4, \ldots, n$

Dimension: $n = 30$

UF10

Definition:
$$f_1(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|}\sum_{j\in J_1}[4y_j^2 - \cos(8\pi y_j) + 1]$$
$$f_2(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_2|}\sum_{j\in J_2}[4y_j^2 - \cos(8\pi y_j) + 1]$$
$$f_3(x) = \sin(0.5x_1\pi) + \frac{2}{|J_3|}\sum_{j\in J_3}[4y_j^2 - \cos(8\pi y_j) + 1]$$

$$Where$$
$$J_1 = \{j|3 \ \leq j \leq n \ and \ j-1 \ is \ a \ multiplication \ of \ 3\},$$
$$J_2 = \{j|3 \ \leq j \leq n \ and \ j-2 \ is \ a \ multiplication \ of \ 3\},$$
$$J_3 = \{j|3 \ \leq j \leq n \ and \ j \ is \ a \ multiplication \ of \ 3\},$$
$$y_j = x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}) \qquad j = 3, \ldots, n$$

Search Space: $0 \leq x_1 \leq 1, \quad 0 \leq x_2 \leq 1, \quad -2 \leq x_i \leq 2, i = 3, 4, \ldots, n$

Dimension: $n = 30$

Appendix D

Multi Objective Constrained Benchmark Functions

Constrained Multi Objective problems involves optimization of multiple objectives simultaneously along with their associated constraints. Often this kind of problems has conflicting objectives and complex constraints, due to which optimizing algorithms finds it difficult to optimize. Here we have chosen the well known standard numerical benchmark functions from CEC 2009 [134] and engineering design problems from [159, 144].

D.1 Numerical Benchmark functions

BNH

The BNH function has a continuous convex Pareto-optimal set.

$$\begin{aligned} \textbf{Definition: } & f_1(x) = 4x_1^2 + 4x_2^2 \\ & f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 \\ \textbf{Subject to: } & g_1(x) = (x_1 - 5)^2 + x_2^2 - 25 \leq 0 \\ & g_2(x) = (x_1 - 8)^2 + (x_2 + 3)^2 - 7.7 \leq 0 \\ \textbf{Search Space: } & 0 \leq x_1 \leq 5, 0 \leq x_2 \leq 3 \\ \textbf{Dimension: } & n = 2 \end{aligned}$$

SRN

This function has continuous linear (can be considered convex or nonconvex)

Pareto-optimal sets.

$$\begin{aligned}
\textbf{Definition: } & f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 2)^2 \\
& f_2(x) = 9x_1 - (x_2 - 1)^2 \\
\textbf{Subject to: } & g_1(x) = x_1^2 + x_2^2 - 225 \leq 0 \\
& g_2(x) = x_1 - 3x_2 + 10 \leq 0 \\
\textbf{Search Space: } & -20 \leq x_1 \leq 20, -20 \leq x_2 \leq 20 \\
\textbf{Dimension: } & n = 2
\end{aligned}$$

OSY

This function has continuous linear (can be considered convex or nonconvex) Pareto-optimal sets. It is considered to be the difficult problem since it has a discontinuous Pareto-front and has high dimensional and a highly constrained problem.

$$\begin{aligned}
\textbf{Definition: } & f_1(x) = -[25(x_1 - 2)^2 + (x_2 - 2)^2 + (x_3 - 1)^2 + (x_4 - 4)^2 + (x_5 - 1)^2] \\
& f_2(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \\
\textbf{Subject to: } & g_1(x) = -(x_1 + x_2 - 2) \leq 0 \\
& g_2(x) = -(6 - x_1 - x_2) \leq 0 \\
& g_3(x) = -(2 - x_2 + x_1) \leq 0 \\
& g_4(x) = -(2 - x_1 + 3x_2) \leq 0 \\
& g_5(x) = -(4 - (x_3 - 3)^2 - x_4) \leq 0 \\
& g_6(x) = -((x_5 - 3)^2 - x_6 - 4) \leq 0 \\
\textbf{Search Space: } & 0 \leq x_{1,2,6} \leq 10, 1 \leq x_{3,5} \leq 5, 0 \leq x_4 \leq 6 \\
\textbf{Dimension: } & n = 6
\end{aligned}$$

CF1

$$\textbf{Definition: } f_1(x) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - x_1^{o.5(1.0+\frac{3(j-2)}{n-2})})^2$$

$$f_2(x) = 1 - x_1 + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - x_1^{o.5(1.0+\frac{3(j-2)}{n-2})})^2$$

$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \le j \le n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \le j \le n\}$$

$$\textbf{Subject to: } g_1(x) = f_1 + f_2 - a|\sin[N\pi(f1 - f2 + 1)]| - 1 \ge 0$$

$$Where\ N\ is\ an\ integer\ and\ a \ge \frac{1}{2N}$$

$$\textbf{Search Space: } 0 \le x_i \le 1, i = 1, 2, \ldots, n$$

$$N = 10,\ \ a = 1$$

$$\textbf{Dimension: } \quad n = 10$$

CF2

$$\textbf{Definition: } f_1(x) = x_1 + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - \sin(6\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_2(x) = 1 - \sqrt{x_1} + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - \sin(6\pi x_1 + \frac{j\pi}{n}))^2$$

$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \le j \le n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \le j \le n\}$$

$$\textbf{Subject to: } g_1(x) = \frac{t}{1 + e^{4|t|}} \ge 0$$

$$t = f_1 + f_2 - a\sin[N\pi(f1 - f2 + 1)] - 1$$

$$Where\ N\ is\ an\ integer\ and\ a \ge \frac{1}{2N}$$

$$\textbf{Search Space: } 0 \le x_1 \le 1,\ -1 \le x_i \le 1, i = 2, 3, \ldots, n$$

$$N = 10,\ \ a = 1$$

$$\textbf{Dimension: } \quad n = 10$$

CF3

$$\textbf{Definition: } f_1(x) = x_1 + \frac{2}{|J_1|}(4\sum_{j\in J_1} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$$

$$f_2(x) = 1 - x_1^2 + \frac{2}{|J_2|}(4\sum_{j\in J_2} y_j^2 - 2\prod_{j\in J_1}\cos(\frac{20y_j\pi}{\sqrt{j}}) + 2)$$

$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \le j \le n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \le j \le n\}\ and$$

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n$$

$$\textbf{Subject to: } g_1(x) = f_2 + f_1^2 - a\sin[N\pi(f_1^2 - f_2 + 1)] - 1 \ge 0$$

$$\textbf{Search Space: } 0 \le x_1 \le 1,\ -2 \le x_i \le 2, i = 2, 3, \ldots, n$$

$$N = 2,\ \ a = 1$$

$$\textbf{Dimension: } \quad n = 10$$

CF4

$$\textbf{Definition: } f_1(x) = x_1 + \sum_{j\in J_1} h_j(y_j)$$

$$f_2(x) = 1 - x_1 + \sum_{j\in J_2} h_j(y_j)$$

$$Where\ J_1 = \{j|j\ is\ odd\ and\ 2 \le j \le n\}\ and\ J_2 = \{j|j\ is\ even\ and\ 2 \le j \le n\}\ and$$

$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), j = 2, \ldots, n$$

$$h_2 = |t|\ \ if\ \ t < \frac{3}{2}(1 - \frac{\sqrt{2}}{2})$$

$$h_2 = 0.125 + (t-1)^2\ \ otherwise$$

$$h_j = t^2\ \ for\ \ j = 3, 4, \ldots, n$$

$$\textbf{Subject to: } g_1(x) = \frac{t}{1 + e^{4|t|}} \ge 0$$

$$t = x_2 - \sin(6\pi x_1 + \frac{2\pi}{n}) - 0.5x_1 + 0.25$$

$$\textbf{Search Space: } 0 \le x_1 \le 1,\ -2 \le x_i \le 2, i = 2, 3, \ldots, n$$

$$\textbf{Dimension: } \quad n = 10$$

CF5

$$\textbf{Definition: } f_1(x) = x_1 + \sum_{j \in J_1} h_j(y_j)$$
$$f_2(x) = 1 - x_1 + \sum_{j \in J_2} h_j(y_j)$$
$$Where\ J_1 = \{j | j\ is\ odd\ and\ 2 \leq j \leq n\}\ and\ J_2 = \{j | j\ is\ even\ and\ 2 \leq j \leq n\}\ and$$
$$y_j = x_j - 0.8x_1 \cos(6\pi x_1 + \frac{j\pi}{n}), if\ \ j \in J_1$$
$$y_j = x_j - 0.8x_1 \sin(6\pi x_1 + \frac{j\pi}{n}), if\ \ j \in J_2$$
$$h_2 = |t|\ \ if\ \ t < \frac{3}{2}(1 - \frac{\sqrt{2}}{2})$$
$$h_2 = 0.125 + (t-1)^2\ \ otherwise$$
$$h_j = 2t^2 - \cos(4\pi t) + 1\ \ for\ \ j = 3, 4, \ldots, n$$
$$\textbf{Subject to: } g_1(x) = x_2 - 0.8x_1 \sin(6\pi x_1 + \frac{2\pi}{n}) - 0.5x_1 + 0.25$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1,\ -2 \leq x_i \leq 2, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } \quad n = 10$$

CF6

$$\textbf{Definition: } f_1(x) = x_1 + \sum_{j \in J_1} y_j^2$$
$$f_2(x) = (1 - x_1)^2 + \sum_{j \in J_2} y_j^2$$
$$Where\ J_1 = \{j | j\ is\ odd\ and\ 2 \leq j \leq n\}\ and\ J_2 = \{j | j\ is\ even\ and\ 2 \leq j \leq n\}\ and$$
$$y_j = x_j - 0.8x_1 \cos(6\pi x_1 + \frac{j\pi}{n}), if\ \ j \in J_1$$
$$y_j = x_j - 0.8x_1 \sin(6\pi x_1 + \frac{j\pi}{n}), if\ \ j \in J_2 \tag{D.1}$$

$$\textbf{Subject to: } g_1(x) = x_2 - 0.8x_1 \sin(6\pi x_1 + \frac{2\pi}{n})$$
$$-sign(0.5(1-x_1) - (1-x_1)^2)\sqrt{|0.5(1-x_1) - (1-x_1)^2|} \geq 0$$
$$g_2(x) = x_4 - 0.8x_1 \sin(6\pi x_1 + \frac{4\pi}{n})$$
$$-sign(0.25\sqrt{(1-x_1)} - 0.5(1-x_1)^2)\sqrt{|0.25\sqrt{(1-x_1)} - 0.5(1-x_1)|} \geq 0$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1,\ -2 \leq x_i \leq 2, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } \quad n = 10$$

CF7

$$\textbf{Definition: } f_1(x) = x_1 + \sum_{j \in J_1} h_j(y_j)$$
$$f_2(x) = (1-x_1)^2 + \sum_{j \in J_2} h_j(y_j)$$

Where $J_1 = \{j | j \text{ is odd and } 2 \leq j \leq n\}$ *and* $J_2 = \{j | j \text{ is even and } 2 \leq j \leq n\}$ *and*

$$y_j = x_j - \cos(6\pi x_1 + \frac{j\pi}{n}), if \ \ j \in J_1$$
$$y_j = x_j - \sin(6\pi x_1 + \frac{j\pi}{n}), if \ \ j \in J_2$$
$$h_2(t) = h_4(t) = t^2,$$
$$h_j(t) = 2t^2 - \cos(4\pi t) + 1 \ \ for \ \ j = 3, 5, 6, \ldots, n$$

$$\textbf{Subject to: } g_1(x) = x_2 - \sin(6\pi x_1 + \frac{2\pi}{n})$$
$$-sign(0.5(1-x_1) - (1-x_1)^2)\sqrt{|0.5(1-x_1) - (1-x_1)^2|} \geq 0$$
$$g_2(x) = x_4 - \sin(6\pi x_1 + \frac{4\pi}{n})$$
$$-sign(0.25\sqrt{(1-x_1)} - 0.5(1-x_1)^2)\sqrt{|0.25\sqrt{(1-x_1)} - 0.5(1-x_1)|} \geq 0$$
$$\textbf{Search Space: } 0 \leq x_1 \leq 1,\ -2 \leq x_i \leq 2, i = 2, 3, \ldots, n$$
$$\textbf{Dimension: } \quad n = 10$$

CF8

$$\textbf{Definition: } f_1(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_2(x) = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_3(x) = \sin(0.5x_1\pi) + \frac{2}{|J_3|}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$Where$$

$$J_1 = \{j|3 \le j \le n,\ and\ j-1\ is\ a\ multiplication\ of\ 3\}$$

$$J_2 = \{j|3 \le j \le n,\ and\ j-2\ is\ a\ multiplication\ of\ 3\}$$

$$J_3 = \{j|3 \le j \le n,\ and\ j\ is\ a\ multiplication\ of\ 3\}$$

$$\textbf{Subject to: } g_1(x) = \frac{f_1^2+f_2^2}{1-f_3^2} - a|\sin[N\pi(\frac{f_1^2+f_2^2}{1-f_3^2}+1)]| - 1 \ge 0$$

$$\textbf{Search Space: } 0 \le x_1 \le 1,\ , 0 \le x_2 \le 1,\ -4 \le x_i \le 4, i = 3,4,\ldots,n$$

$$\textbf{Dimension: } n = 10$$

CF9

$$\textbf{Definition: } f_1(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|}\sum_{j\in J_1}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_2(x) = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{|J_2|}\sum_{j\in J_2}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$f_3(x) = \sin(0.5x_1\pi) + \frac{2}{|J_3|}\sum_{j\in J_3}(x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}))^2$$

$$Where$$

$$J_1 = \{j|3 \le j \le n,\ and\ j-1\ is\ a\ multiplication\ of\ 3\}$$

$$J_2 = \{j|3 \le j \le n,\ and\ j-2\ is\ a\ multiplication\ of\ 3\}$$

$$J_3 = \{j|3 \le j \le n,\ and\ j\ is\ a\ multiplication\ of\ 3\}$$

$$\textbf{Subject to: } g_1(x) = \frac{f_1^2+f_2^2}{1-f_3^2} - a|\sin[N\pi(\frac{f_1^2+f_2^2}{1-f_3^2}+1)]| - 1 \ge 0$$

$$\textbf{Search Space: } 0 \le x_1 \le 1,\ , 0 \le x_2 \le 1,\ -2 \le x_i \le 2, i = 3,4,\ldots,n$$

$$\textbf{Dimension: } n = 10$$

CF10

$$\textbf{Definition: } f_1(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|}\sum_{j\in J_1}[4y_j^2 - \cos(8\pi y_j) + 1]$$

$$f_2(x) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_2|}\sum_{j\in J_2}[4y_j^2 - \cos(8\pi y_j) + 1]$$

$$f_3(x) = \sin(0.5x_1\pi) + \frac{2}{|J_3|}\sum_{j\in J_3}[4y_j^2 - \cos(8\pi y_j) + 1]$$

$$Where$$

$$J_1 = \{j|3 \leq j \leq n,\ and\ j-1\ is\ a\ multiplication\ of\ 3\}$$

$$J_2 = \{j|3 \leq j \leq n,\ and\ j-2\ is\ a\ multiplication\ of\ 3\}$$

$$J_3 = \{j|3 \leq j \leq n,\ and\ j\ is\ a\ multiplication\ of\ 3\}$$

$$and$$

$$y_j = x_j - 2x_2\sin(2\pi x_1 + \frac{j\pi}{n}), for\ \ j = 3,\ldots,n$$

$$\textbf{Subject to: } g_1(x) = \frac{f_1^2 + f_2^2}{1 - f_3^2} - a|\sin[N\pi(\frac{f_1^2 + f_2^2}{1 - f_3^2} + 1)]| - 1 \geq 0$$

$$\textbf{Search Space: } 0 \leq x_1 \leq 1,\ ,0 \leq x_2 \leq 1,\ -2 \leq x_i \leq 2, i = 3,4,\ldots,n$$

$$\textbf{Dimension: } n = 10$$

D.2 Engineering design problems

Speed Reducer design

The well known speed reducer test problem represents the design of a simple gear box such as might be used in a light airplane between the engine and propeller to allow each to rotate at its most efficient speed. The objective is to minimize the speed reducer weight while satisfying a number of constraints imposed by gear and shaft design. There are seven design variables, $(x_1, x_2, x_3, x_4, x_5, x_6, x_7)$. x_1 Width of the gear face (cm), x_5 Shaft 2 length between bearings (cm), x_2 Teeth module (cm) x6 Diameter of shaft 1 (cm), x_3 Number of pinion teeth (Integer), x_7 Diameter of shaft 2 (cm), x_4 Shaft 1 length between bearings (cm). The first objective of speed reducer problem is to find the minimum of a gear box volume f_1, (and, hence, its minimum weight). The second objective, f_2, is to minimize the stress in one of the two gear shafts. The design is subject to constraints imposed

by gear and shaft design practices. An upper and lower limit is imposed on each of the seven design variables There are 11 inequality constraints.

$$\textbf{Definition: } f_1(x) = 0.7854x_1x_2^2(10x_3^2/3 + 14.933x_3 - 43.0934)$$
$$-1.508x_1(x_6^2 + x_7^2) + 7.477(x_6^3 + x_7^3) + 0.7854(x_4x_6^2 + x_5x_7^2)$$
$$f_2(x) = \frac{\sqrt{(745x_4/x_2x_3)^2 + 1.69 \times 10^7}}{0.1x_6^3}$$
$$\textbf{Subject to: } g_1(x) = \frac{1}{x_1x_2^2x_3} - \frac{1}{27} \leq 0$$
$$g_2(x) = \frac{1}{x_1x_2^2x_3^2} - \frac{1}{397.5} \leq 0$$
$$g_3(x) = \frac{x_4^3}{x_2x_3x_6^4} - \frac{1}{1.93} \leq 0$$
$$g_4(x) = \frac{x_5^3}{x_2x_3x_7^4} - \frac{1}{1.93} \leq 0$$
$$g_5(x) = x_2x_3 - 40 \leq 0$$
$$g_6(x) = \frac{x_1}{x_2} - 12 \leq 0$$
$$g_7(x) = 5 - \frac{x_1}{x_2} \leq 0$$
$$g_8(x) = 1.9 - x_4 + 1.5x_6 \leq 0$$
$$g_9(x) = 1.9 - x_5 + 1.1x_7 \leq 0$$
$$g_{10}(x) = f_2(x) \leq 1100$$
$$g_{11}(x) = \frac{\sqrt{(745x_5/x_2x_3)^2} + 1.575 \times 10^8}{0.1x_7^3} \leq 850$$
$$\textbf{Search Space: } 2.6 \leq x_1 \leq 3.6,\ 0.7 \leq x_2 \leq 0.8,$$
$$17 \leq x_3 \leq 18, 7.3 \leq x_3, x_4 \leq 8.3, 2.9 \leq x_6 \leq 3.9, 5.0 \leq x_7 \leq 5.5$$
$$\textbf{Dimension: } \quad n = 7$$

Two-bar Truss Design

The truss has to carry a certain load without elastic failure. Thus, in addition to the objective of designing the truss for minimum volume (which is equivalent to designing for minimum cost of fabrication), there are additional objectives of minimizing stresses in each of the two members.

$$\textbf{Definition: } f_1(x) = x_1\sqrt{16+x_3^2} + x_2\sqrt{1+x_3^2}$$
$$f_2(x) = max(\sigma_1, \sigma_2)$$
$$\textbf{Subject to: } g_1(x) = max(\sigma_1, \sigma_2) - 10^5 \leq 0$$
$$Where$$
$$\sigma_1 = 20\sqrt{16+x_3^2}/x_1x_3$$
$$\sigma_2 = 80\sqrt{1+x_3^2}/x_2x_3$$
$$\textbf{Search Space: } 0 \leq x_{1,2} \leq 0.01,\ 0 \leq x_3 \leq 3$$
$$\textbf{Dimension: } n = 3$$

Welded Beam Design

A beam needs to be welded on another beam and must carry a certain load F. The overhang portion of the beam has a length of 14 inch and F=6000lb force is applied at the end of the beam. The objective of the design is to minimize the cost of fabrication and minimize the end deflection.

$$\textbf{Definition: } f_1(x) = 1.10471h^2l + 0.04811tb(14+l)$$
$$f_2(x) = 2.1952/t^3b$$
$$\textbf{Subject to: } g_1(x) = -(13600 - \tau(x)) \leq 0$$
$$g_2(x) = -(30000 - \sigma(x)) \leq 0$$
$$g_3(x) = -(b-h) \leq 0$$
$$g_4(x) = -(P_c(x) - 6000) \leq 0$$
$$Where$$
$$\tau = \sqrt{(\tau')^2 + (\tau'')^2 + l\tau'\tau''/\sqrt{0.25(l^2+(h+t)^2)}}$$
$$\tau' = 6000/\sqrt{2}hl$$
$$\tau'' = \frac{6000(14+0.5l)\sqrt{0.25(l^2+(h+t)^2)}}{2\sqrt{2}hl(l^2/12 + 0.25(h+t)^2)}$$
$$\sigma = 504000/t^2b$$
$$P_c = 64746.022(1 - 0.0282346t)tb^3$$
$$\textbf{Search Space: } 0.125 \leq h, b \leq 5,\ 0.1 \leq l, t \leq 10$$
$$\textbf{Dimension: } n = 4$$

Gear train design

A compound gear train is to be designed to achieve a specific gear ratio between the driver and driven shafts. The objective of the gear train design is to find the number of teeth in each of the four gears so as to minimize (i) the error between the obtained gear ratio and a required gear ratio of 1/6.931 and (ii) the maximum size of any of the four gears. Since the number of teeth must be integers, all four variables are strictly integers.

$$\textbf{Definition: } f_1(x) = [\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4}]^2$$
$$f_2(x) = max(x_1, x_2, x_3, x_4)$$
$$\textbf{Subject to: } g_1(x) = 12 \leq x_{1,2,3,4} \leq 60$$
$$\textbf{Search Space: } 12 \leq x_{1,2,3,4} \leq 60$$
$$\textbf{Dimension: } n = 4$$

References

[1] D.M. Himmelblau. *Applied Nonlinear Programming.* McGraw-Hill, New York., 1972.

[2] Beightler C. S., Phillips D. T., and Wilde D. J. *Foundations of Optimization.* Prentice Hall, Inc, Englewood Cliffs, NJ, USA., 1979.

[3] S. S. Rao. *Optimization: Theory and Applications.* Wiley, New York, USA., 1984.

[4] Jayadeva Chandra, S. and A. Mehra. *Numerical Optimization with Applications.* Alpha Science Intl Ltd., 2009.

[5] C. Mohan and K. Deep. *Optimization Techniques.* New Age, NewDelhi, 2009.

[6] P. H. Grasse. *Termitologia, Tome II.* Fondation des Societs. Paris, Masson., 1984.

[7] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Computer graphics*, 21:25–34, August 1987.

[8] Jean-Louis Deneubourg and Simon Goss. Collective patterns and decision making. *Ethology, Ecology and Evolution*, 1(4):295–311, December 1989.

[9] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chretien. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*, pages 356–363, Cambridge, MA, USA, 1990. MIT Press.

[10] G. Beni and J. Wang. Theoretical problems for the realization of distributed robotic systems. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 1914 – 1920, Vol.3, Sacramento, CA, USA, April 1991.

[11] R. Eberhart and J. Kenedy. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, pages 1114–1121, Piscataway, NJ, November 1995.

[12] R. Eberhart and J. Kenedy. A new optimizer using particle swarm theory. In *Proceedings of* 6^{th} *International Symposium on Micro Machine and Human Science (MHS)*, pages 39–43, Cape Cod, MA, November 1995.

[13] J. Robinson and Yahya Rahmath Samii. Particle swarm optimization in electromagnetic. *IEEE Transactions on Antenna and Propagation*, 52(2):397–400, 2004.

[14] D. H. Wolpert and W. G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation,*, 1(1):67–82, April 1997.

[15] RC. Eberhart, P. Simpson, and R. Dobbins. *Computational Intelligence PC Tools, Chapter 6. Academic Press Professional.* San Diego, CA, 1996.

[16] Y. Shi and R. Eberhart. A modified particle swarm optimizer. In *Proceedings of the IEEE international conference on Evolutionary Computation*, pages 69–73, IEEE Press, Piscataway, NJ, 1998.

[17] R. Eberhart and Y. Shi. Comparing inertia weights and constriction factors in particle swarm optimization. In *Proceedings of the IEEE Congress Evolutionary Computation 2000*, pages 84–88, La Jolla, CA , USA, July 2000.

[18] M. Clerc and J. Kennedy. The particle swarm: explosion, stability and convergence in a multi-dimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(1):58–73, 2002.

[19] C Maurice and J. Kennedy. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation*, 6(2):58–73, 2002.

[20] T Peram, K Veeramachaneni, and K. Mohan, C. Fitness-distance ratio based particle swarm optimization. In *Proceedings of IEEE Swarm Intelligence System*, pages 174–181, Indianapolis, Indiana, USA, July 2003.

[21] A. Ratnaweera and K. Halgamuge, S. Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficient. *IEEE Transactions on Evolutionary Computation*, 8(3):240–255, 2005.

[22] D. Bergh, F and Engelbrecht. Andries, P. A cooperative approach to particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):225–239, 2004.

[23] R Mendes, J Kennedy, and J. Neves. The fully informed particle swarm: Simpler maybe better. *IEEE Transactions on Evolutionary Computation*, 8(3):204–210, 2004.

[24] Stefan Janson and Martin Middendorf. A hierarchical particle swarm optimizer and its adaptive variant. *IEEE Transactions on Systems, Man, And CYbernetics-Part B: Cybernetics*, 35(6):1272–1282, 2005.

[25] Krohling Renato, A and S C. Leandro, dos. Particle swarm with exponential distribution (*psoe*). In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1428–1433, Sheraton Vaccouver Wall Center Hotel, Canada, July 2006.

[26] Srinivas Pasupuleti and Roberto Battiti. The gregarious particle swarm optimizer (g-pso). In *Proceedings of the* 8^{th} *annual conference on Genetic and Evolutionary Computation*, GECCO '06, pages 69–73, New York, NY, USA, 2006. ACM.

[27] Xueming Yang, Jinsha Yuan, Jiangye Yuan, and Huina Mao. A modified particle swarm optimizer with dynamic adaptation. *Applied Mathematics and Computation*, 189(2):1205 – 1213, 2007.

[28] Samrat L Sabat and Layak Ali. The combined effect comprehensive learning particle swarm optimizer. In *Proceedings of* 24^{th} *International Symposium on Automation and Robotics in Construction (ISARC 2007)*, pages 347–354, Kochi,IIT Madras, India, September 19–21 2007.

[29] S Nema, J Goulermas, G Sparrow, and P. Cook. A hybrid particle swarm branch-and-bound (hpb) optimizer for mixed discrete nonlinear programming. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(6):1411–1427, 2008.

[30] M. Senthil Arumugam, M. V. C. Rao, and Aarthi Chandramohan. A new and improved version of particle swarm optimization algorithm with global and local best parameters. *Journal of Knowledge and Information Systems (KAIS)*, 16:331–357, August 2008.

[31] Samrat L. Sabat and Layak Ali. Accelerated Effect Particle Swarm Optimizer. In *Proceedings of IEEE International Conference TENCON 2008*, pages 1–6, Hyderabad, India, Nov 2008.

[32] Samrat L Sabat and Layak Ali. The hyperspherical acceleration effect particle swarm optimizer. *Applied Soft Computing*, 9(13):906–917, 2008.

[33] M. Senthil Arumugam, M.V.C. Rao, and Alan W.C. Tan. A novel and effective particle swarm optimization like algorithm with extrapolation technique. *Applied Soft Computing*, 9(1):308 – 320, 2009.

[34] Hsieh Sheng, Ta, Tsung-Ying Sun, Chan-Cheng Liu, and Shang-Jeng Tsai. Efficient population utilization strategy for particle swarm optimizer. *IEEE Transactions on Systems, Man, and Cybernetics*, 39(2):444–456, 2009.

[35] Samrat L Sabat, Layak Ali, and Siba K. Udgata. Adaptive accelerated exploration particle swarm optimizer for global multimodal functions. In *Proceedings of World Congress on Nature and Biologically Inspired Computing*, pages 654 – 659, Coimbatore, India, December 2009.

[36] Samrat L. Sabat and Layak Ali. Integrated learning particle swarm optimizer. In *Proceedings of* 3^{rd} *Indian International Conference on Artificial Intelligence (IICAI-07*, pages 421–440, Pune, India, December 17-19 2007.

[37] Samrat L. Sabat, Layak Ali, and Siba K. Udgata. Integrated learning particle swarm optimizer for global optimization. *Applied Soft Computing*, 11(1):574 – 584, 2011.

[38] P.N. Suganthan S. Bhasker, A. Alphones and J.J. Liang. Design of yagi-uda antenna using comprehensive learning particle swarm optimization. *IEEE Transactions on Microwave Antenna Propapagation*, 152(5):340–345, 2005.

[39] N. Suganthan, P, N. Hansen, J. Liang, J, K. Deb, P. Chen, Y, A. Auger, and S. Tiwari. Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. Technical Report 2005005, Nanyang Technological University, Singapore and IIT Kanpur, India,, Technical Report, Nanyang Technological University, Singapore, AND KanGAL Report 2005005, IIT Kanpur, India, 2005.

[40] Daniel Molina Joaquin Derrac, Salvador Garcia and Francisco Herrera. A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, 1:3–8, 2011.

[41] www.graphpad.com/prism/prism.htm.

[42] Z. Michalewicz and M. Schoenauer. Evolutionary algorithms for constrained parameter optimization problems. *Journal of Evolutionary Computation*, 4(1):1–32, 1996.

[43] Carlos A and Coello Coello. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering*, 191(11-12):1245 – 1287, 2002.

[44] A.E.Smith and D.W. Coit. *Constraint Handling Techniques Penalty Functions*. Oxford University Press and Institute of Physics Publishing, 1998.

[45] Z. Michalewicz. A survey of constraint handling techniques in evolutionary computation methods. In *Proceedings of* 4^{th} *Annual Conference on Evolutionary Programming*, pages 135–155, MIT Press, San Diego, 1995.

[46] T.P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4(3):284–294, 2000.

[47] Konstantinos E. Parsopoulos and Michael N. Vrahatis. Particle swarm optimization method for constrained optimization problems. In *Proceedings of the Euro-International Symposium on Computational Intelligence 2002*, pages 214–220. IOS Press, 2002.

[48] Xiaohui Hu and Russell Eberhart. Solving constrained nonlinear optimization problems with particle swarm optimization. In *Proceedings of* 6^{th} *World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002*, pages 203–206, 2002.

[49] G. Coath and K. Halgamuge, S. A comparison of constrainthandling methods for the application of particle swarm optimization to constrained nonlinear optimization problems. In *Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2003*, pages 2419–25, Canberra, Australia, December 2003.

[50] X. Hu, R. Eberhart, and Y. Shi. Engineering optimization with particle swarm. In *Proceedings of the IEEE Swarm Intelligence Symposium (SIS 2003)*, pages 53–57, Indianapolis, Indiana, USA, April 2003.

[51] T. Ray and K. M. Liew. Society and civilization: an optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*, 7(4):386–396, 2003.

[52] S. He, E. Prempain, and Q. Wu. An improved particle swarm optimizer for mechanical design optimization problems. *Engineering Optimization*, 36(5):585–605, 2004.

[53] E.Mezura-Montes and C. A. C. Coello. A simple multimembered evolution strategy to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 9(1):1–7, 2005.

[54] A.Amirjanov. The development of a changing range genetic algorithm. *Computer Methods in Applied Mechanics and Engineering*, 195(1):2495–2508, 2006.

[55] A Krohling, R and Santos Coelho. Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization prob-

lems. *IEEE Transactions on Systems, Man, Cybernetics*, 36(6):1407–1416, 2006.

[56] B. Tessema and G. Yen. A self adaptive penalty function based algorithm for constrained optimization. In *Proceedings of IEEE Congress on Evolutionary Computation, 2006*, pages 246–253, Vancouver, BC, Canada, July 2006.

[57] F. Z. Huang, L. Wang, and Q. He. An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and Computation*, 186(1):340–356, 2007.

[58] Y Ho, P and K. Shimizu. Evolutionary constrained optimization using an addition of ranking method and a percentage-based tolerance value adjustment scheme. *Information Sciences*, 177(14–15):2985–3004, 2007.

[59] Q. He and L. Wang. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering Applications of Artificial Intelligence*, 20(1):89 – 99, 2007.

[60] Minghai Jiao and Jiafu Tang. A novel particle swarm optimization for constrained engineering optimization problems. In *Proceedings of the* 3^{rd} *International Symposium on Advances in Computation and Intelligence*, ISICA '08, Berlin, Heidelberg, 2008. Springer-Verlag.

[61] M.Zhang, W. Luo, and X. Wang. Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, 178(15):3043 – 3074, 2008.

[62] Y. Wang, Z. Cai, Y. Zhou, and Z. Fan. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization*, 37(4):395–413, 2009.

[63] E. Mezura-Montes and I. Flores-Mendoza, J. Improved particle swarm optimization in constrained numerical search spaces. volume 193 of *R. Chiong (Ed.), Nature-Inspired Algorithms for Optimization, Studies in Computational Intelligence Series*, pages 299–332. Springer-Verlag, 2009.

[64] E. Zahara and Y. T. Kao. Hybrid nelder-mead simplex search and particle swarm optimization for constrained engineering design problems. *Expert Systems with Applications*, 36(2):3880 – 3886, 2009.

[65] M.Jaberipour and E. Khorram. Two improved harmony search algorithms for solving engineering optimization problems. *Communications in Nonlinear Science and Numerical Simulation*, 15(11):3316 – 3331, 2010.

[66] L.S. Coelho. Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. *Expert Systems with Applications*, 37(2):1676 – 1683, 2010.

[67] Layak Ali, Samrat L. Sabat, and Siba K. Udgata. Particle swarm optimization technique with stochastic ranking for constrained optimization problems. In *Proceedings of* 4^{th} *Mahasarakham International Workshop on Artificial Intelligence, MIWAI 2010*, pages 672–679, Thailand.

[68] Samrat L. Sabat, Layak Ali, and Siba K. Udgata. Stochastic ranking particle swarm optimization for constrained engineering design problems. In *Proceedings of International Conference on Swarm, Evolutionary and Memetic Computing*, pages 672–679, Chennai, India, December 2010.

[69] J.Robinson and Yahya Rahmath Samii. Particle swarm optimization in electromagnetic. *IEEE Transactions on Antenna and Propagation*, 52(2):397–400, 2004.

[70] Carlos A. and Coello Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113 – 127, 2000.

[71] S. Koziel and Z. Michalewicz. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. *IEEE Transactions on Evolutionary Computation*, 7(1):19–44, 1999.

[72] B. Yang, Y. Chen, Z. Zhao, and Q. Han. A master-slave particle swarm optimization algorithm for solving constrained optimization problems. In *Proceedings of* 6^{th} *Congress on Intelligent Control and Automation*, pages 3208–3212, Dalian, 2006.

[73] T.Takahama and S. Sakai. Solving constrained optimization problems by the ϵ constrained particle swarm optimizer with adaptive velocity limit control. In *Proceedings of IEEE Congress on Evolution Computation*, pages 308–315, Vancouver, BC, Canada, July 2006.

[74] J. Liang, J. Runarsson, T. P. Mezura-Montes, M. Clerc, P.N. Suganthan, Coello Coello, and K. Deb. Problem definitions and evaluation criteria for the CEC 2006, special session on constrained real-parameter optimization. Technical report, Nanyang Technological University, Singapore, 2006.

[75] Valceres V. R. Silva, Peter J. Fleming, Jungiro Sugimoto, and Ryuichi Yokoyama. Multiobjective optimization using variable complexity modelling for control system design. *Applied Soft Computing*, 8(1):392 – 401, 2008.

[76] J. David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the* 1^{st} *International Conference on Genetic Algorithms*, pages 93–100, Hillsdale, NJ, USA, 1985. L. Erlbaum Associates Inc.

[77] S. Agrawal, B.K. Panigrahi, and M.K. Tiwari. Multiobjective particle swarm algorithm with fuzzy clustering for electrical power dispatch. *IEEE Transactions on Evolutionary Computation,*, 12(5):529 –541, oct. 2008.

[78] J.G. Herrero, A. Berlanga, and J.M.M. Lopez. Effective evolutionary algorithms for many-specifications attainment: Application to air traffic control tracking filters. *IEEE Transactions on Evolutionary Computation,*, 13(1):151 –168, feb. 2009.

[79] Tung-Kuan Liu, Chiu-Hung Chen, and Jyh-Horng Chou. Optimization of short-haul aircraft schedule recovery problems using a hybrid multiobjective genetic algorithm. *Expert Systems with Applications*, 37(3):2307 – 2315, 2010.

[80] R. Goldberg A. Abraham, L. C. Jain. *Evolutionary Multiobjective Optimization: Theoretical Advances and Applications.* Springer Verlag, London, 2005.

[81] D.A. Van Veldhuizen C.A.C. Coello, G.B. Lamont. *Evolutionary Algorithms for Solving Multi Objective Problems.* Springer Verlag, 2007.

[82] K. Deb. *Multiobjective Optimization Using Evolutionary Algorithms.* John Wiley and Sons, 2001.

[83] Y. Wang and Y. Yang. Particle swarm optimization with preference order ranking for multi objective optimization. *Information Sciences*, 179(12):1944–1959, 2009.

[84] S. K. Pal P. K. Tripathi, S. Bandyopadhyay. Multi objective particle swarm optimization with time variant inertia and acceleration coefficients. *Information Sciences*, 177(22):5033–5049, 2002.

[85] E.J. Hughes. Multi objective evolutionary guidance for swarms. In *Proceedings of IEEE Congress on Evolutionary Computation*, 2002.

[86] G. G. Roy A. Abraham P. Chakraborty, S. Das. On convergence of the multi-objective particle swarm optimizers. *Information Sciences*, 181(8):1411–1425, 2011.

[87] Carlos M. Fonseca and Peter J. Fleming. Genetic algorithms for multiobjective optimization:formulation, discussion and generalization. In *Proceedings of* 5^{th} *International Conference Genetic Algorithms*, pages 416–423, 1993.

[88] T. Tanino, M. Tanaka, and C. Hojo. An interactive multicriteria decision making method by using a genetic algorithm. In *Proceedings of International Conference on Systems Science and Systens Engineering*, pages 381–386, 1993.

[89] J. Horn, N. Nafpliotis, and D.E. Goldberg. A niched pareto genetic algorithm for multiobjective optimization. In *Proceedings of the First IEEE World Congress on Computational Intelligence*, volume 1, pages 82 –87, June 1994.

[90] E. Zitzler and E. Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transaction on Evolutionary Computation*, 3(4):257–271, 1999.

[91] J. Knowles and D. Corne. M-paes: a memetic algorithm for multiobjective optimization. In *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2000*, pages 325–332, 2000.

[92] J. Knowles and D. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *IEEE Transactions on Evolutionary Computation*, 8(2):149–172, 2000.

[93] W. Leung, Y and P. Wang, Y. Multi-objective programming using uniform design and genetic algorithm. *IEEE Transactions on Systems, Man, and Cybernetics,Part C*, 30(3):293–304, 2000.

[94] K. Deb. *Multiobjective Optimization Using Evolutionary Algorithms*. New York:Wiley, 2001.

[95] Jin Y., Olhofer M., and Sendhoff. B. Dynamic weighted aggregation for evolutionary multi-objective optimization: why does it work and how. In *Proceedings of Conferenceon Genetic and Evolutionary Computation, GECCO 2001*, pages 1042–1049, 2001.

[96] E. Zitzler, M. Laumanns, and L. Thiele. SPEA: Improving the Strength of Pareto Evolutionary Algorithm. Technical Report TIK-Report 103, Department of Electrical Engineering, Swiss Federal Institute of Technology (ETH), Zurich, 2001.

[97] E. Zitzler, M. Laumanns, and L. Thiele. SPEA2: Improving the Strength of Pareto Evolutionary Algorithm for Multi-Objective Optimization. In *Proceedings of Evolutionary Methods for Design Optimization and Control with Applications to Industrial Problems*, pages 95–100, Athens, Greece, 2002.

[98] Laumanns M., L. Thiele, K. Deb, and E. Zitzler. Combining convergence and diversity in evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 10(3):263–282, 2002.

[99] K.E. Parsopoulos and M.N. Vrahatis. Particle Swarm Optimization Method in Multi-Objective Problems. In *Proceedings of ACM Symposium on Applied Computing 2002, SAC 2002*, pages 603–607, Madrid, Spain, 2002.

[100] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation,*, 6(2):182 –197, April 2002.

[101] G. G. Yen and H. Lu. Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation. *IEEE Transactions on Evolutionary Computation*, 7(3):253–274, 2003.

[102] C.A.C. Coello, G.T. Pulido, and M.S. Lechuga. Handling Multiple Objectives With Particle Swarm Optimization. *IEEE Transactions on Evolutionary Computation*, 8(3):256–279, 2004.

[103] R Sarker and H Abbass. Differential evolution for solving multi-objective optimization problems. *Asia Pacific Journal of Operational Research*, 21(2):225–240, 2004.

[104] Tan and Y. J. Yang and C. K. Gohr. K., C. A distributed cooperative coevolutionary algorithm for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 10(5):527–549, 2006.

[105] M. Emmerich and K. Giannakoglou and B. Naujoks. M., T. Single- and multiobjective evolutionary optimization assisted by gaussian random field metamodels. *IEEE Transactions on Evolutionary Computation*, 10(4):421–439, 2006.

[106] J.J. Liang, A.K. Qin, P.N. Suganthan, and S. Baskar. Evaluation of Comprehensive Learning Particle Swarm Optimizer. *Springer;Lecture Notes in Computer Science*, 3316:230–235, 2004.

[107] V.L. Huang, P.N. Suganthan, and J.J. Liang. Comprehensive Learning Particle Swarm Optimizer for Solving Multi-Objective Optimization Problems. *International Journal of Intelligent Systems 2006*, 21:209–211.

[108] Zhang Q. and Li. H. Moea/d: a multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[109] Sanchis J., Martnez M., A., and Ferragud. X., B. Integrated multiobjective optimization and a priori preferences using genetic algorithms. *Information Sciences*, 178(4):931–951, 2008.

[110] Leong W.-F. and Yen. G., G. Pso-based multiobjective optimization with dynamic population size and adaptive local archives. *IEEE Transactions on Systems,Man and Cybernetics, Part B (Cybernetics)*, 38(5):1270–1293, 2008.

[111] Gong M., Jiao L., Du H., and Bo. L. Multiobjective immune algorithm with nondominated neighbor-based selection. *IEEE Transactions on Evolutionary Computation*, 16(2):225–255, 2008.

[112] H. Fang, Q. Wang, Y.-C. Tu, and F. Horstemeyer, M. An efficient non-dominated sorting method for evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 16(3):355–384, 2008.

[113] R. Storn and K. Price. Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[114] A. Kai Qin and Ponnuthurai N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 1785–1791, 2005.

[115] V.L. Huang, S.Z. Zhao, R. Mallipeddi, and P.N. Suganthan. Multi-objective optimization using self-adaptive differential evolution algorithm. In *Proceedings of IEEE Congress on Evolutionary Computation, 2009. CEC '09.*, pages 190 –194, May 2009.

[116] Zhao and P. N. Suganthan. S., Z. Multi-objective evolutionary algorithm with ensemble of external archives. *International Journal of Innovative Computing, Information and Control*, 6(1):1713–1726, 2010.

[117] Wang Y. and and X. Yuan. L.-H., Wu. Multi-objective self-adaptive differential evolution with elitist archive and crowding entropy-based diversity measure. *Applied Soft Computing*, 14(3):193–209, 2010.

[118] Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems Magazine*, 22.

[119] Selcen Phelps and Murat Koksalan. An interactive evolutionary metaheuristic for multiobjective combinatorial optimization. *Management Science*, 49(12):1726–1738, 2003.

[120] Knowles J., D. and Corne. D., W. Memetic algorithms for multiobjective optimization: issues, methods and prospects. In *Proceedings of Recent Advances in Memetic Algorithms, Springer*, pages 313–352, 2004.

[121] Huband S., Hingston P., White L., and Barone. L. An evolution strategy with probabilistic mutation for multi-objective optimisation. In *Proceedings of IEEE Congress on Evolutionary Computation, CEC 2003*, pages 2284–2291, 2003.

[122] Lara A. and and C. A. Coello Coello and O. Schutze. G., Sanchez. a new local search strategy for memetic multiobjective evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 14(1):112–132, 2010.

[123] Coello Coello and N. C. Corts. C., A. Solving multiobjective optimization problems using an artificial immune system. *Genetic Programming and Evolvable Machines*, 6(2):163–190, 2005.

[124] Multi-objective optimization with artificial weed colonies. *Information Processing Letters*, 181.

[125] K. Deb, M. Mohan, and S. Mishra. Evaluating the epsilon-domination based multiobjective evolutionary algorithm for a quick computation of pareto-optimal solutions. *IEEE Transactions on Evolutionary Computation*, 13(4):501–525, 2005.

[126] K. Deb, J. Sundar, U. B. Rao, N., and S. Chaudhuri. Reference point based multiobjective optimization using evolutionary algorithms. *International Journal of Computational Intelligence Research*, 2(3):273–286, 2006.

[127] M. Basseur and E. Zitzler. Handling uncertainty in indicator-based multiobjective optimization. *International Journal of Computational Intelligence Research*, 2(3):255–272, 2006.

[128] Beausoleil. R., P. Moss: Multiobjective scatter search applied to non-linear multiple criteria optimization. *European Journal of Operational Research*, 169(2):426–449, 2006.

[129] C. Igel, N. Hansen, and S. Roth. Covariance matrix adaptation for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 15(1):1–28, 2007.

[130] Bingul. Z. Adaptive genetic algorithms applied to dynamic multiobjective problem. *Applied Soft Computing*, 7(3):791–799, 2007.

[131] Junjie Yang, Jianzhong Zhou, Li Liu, and Yinghai Li. A novel strategy of pareto-optimal solution searching in multi-objective particle swarm optimization (mopso). *Computers & Mathematics with Applications*, 57(11–12):1995–2000, 2009.

[132] Yujia Wang and Yupu Yang. Particle swarm optimization with preference order ranking for multi-objective optimization. *Information Sciences*, 179(12):1944 – 1959, 2009.

[133] A. Elhossini, S. Areibi, and R. Dony. Strength pareto particle swarm optimization and hybrid ea-pso for multi-objective optimization. *IEEE Transactions on Evolutionary Computation*, 18(1):127–156, 2010.

[134] Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari. Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition. Technical report, Nanyang Technological University, Singapore, 2009.

[135] E. Zitzler and L. Thiele. Multiobjective evolutionary algorithms: A comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.

[136] www.mathworks.com/matlabcentral/fileexchange/19651-hypervolume-indicator.

[137] R. Sarker, H. A. Abbass, and S. Karim. An evolutionary algorithm for constrained multiobjective optimization problems. In *The Fifth AustraliasiaJapan Joint Workshop*, pages 19–21, University of Otago, Dunedin, New Zealand, November 2001.

[138] Hemant Kumar Singh, Tapabrata Ray, and Warren Smith. C-psa: Constrained pareto simulated annealing for constrained multi-objective optimization. *Information Sciences*, 180(13):2499 – 2513, 2010.

[139] T. Binh and Korn. U. Mobes: A multiobjective evolution strategy for constrained optimization problems. In *Proceedings of* 3^{rd} *International Conference on Genetic Algorithms (Mende l97)*, pages 176–182, 1997.

[140] F. Jimenez and L. Verdegay, J. Constrained multiobjective optimization by evolutionary algorithm. In *Proceeding of International Computer Science Conventions Symposium on Engineering of Intelligent Systems (ICSC-EIS-98)*, pages 266–271, 1998.

[141] A. Kurpati, S. Azarm, and J. Wu. Constraint handling improvements for multiobjective genetic algorithms. *Structural and Multidisciplinary Optimization*, 2002.

[142] Chafekar Deepti, Xuan Jiang, and Rasheed. Khaleed. Constrained multi-objective optimization using steady state genetic algorithms. *In Proceedings of Genetic and Evolutionary Computation-GECCO 2003,Part I, Lecture Notes in Computer Science,Springer*, 2003.

[143] C. Ji. A revised particle swarm optimization approach for multi-objective and multi-constraint optimization. In *Proceedings of the* 8^{th} *annual conference on Genetic and Evolutionary Computation*, GECCO '04, pages 1204–1211, Seattle, Washington, USA, 2004.

[144] A. Mahmoud M. S. Osman and A.A. Mousa. IT-CEMOP: An iterative co-evolutionary algorithm for multiobjective optimization problem with nonlinear constraints. *Applied Mathematics and Computation,*, 183(2):373–389, April 2006.

[145] M. J. Reddy and D. N. Kumar. Multi-objective particle swarm optimization for generating optimal trade-offs in reservoir operation. *Hydrological Process*, 21(1):2897–2909, 2007.

[146] M. Reyes-Sierra and C. A. C. Coello. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International Journal of Computational Intelligence Research*, 2(1):287–308, 2006.

[147] Zhuhong and Zhang. Immune optimization algorithm for constrained nonlinear multiobjective optimization problems. *Applied Soft Computing*, 7(3):840 – 857, 2007.

[148] H. Singh, A. Isaacs, T. Ray, and W. Smith. A simulated annealing algorithm for constrained multi-objective optimization problems. In *Proceedings of*

World Congress on Computational Intelligence (WCCI 08), pages 2780–2787, Hong Kong, 2008.

[149] Woldesenbet and Gary G. Yen and Biruk G. Tessema. Yonas, Gebre. Constraint handling in multiobjective evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 13(3):514 – 525, june 2009.

[150] Minzhong Liu, Xiufen Zou, Yu Chen, and Zhijian Wu. Performance assessment of dmoea-dd with cec 2009 moea competition test instances. In *Proceedings of IEEE Congress on Evolutionary Computation*, CEC'09, pages 2913–2918, Piscataway, NJ, USA, 2009. IEEE Press.

[151] Hai-Lin Liu and Xueqiang Li. The multiobjective evolutionary algorithm based on determined weight and sub-regional search. In *Proceedings of IEEE Congress on Evolutionary Computation*, CEC'09, pages 1928–1934, Piscataway, NJ, USA, 2009. IEEE Press.

[152] Saku Kukkonen and Jouni Lampinen. Performance assessment of generalized differential evolution with a given set of constrained multi-objective test problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, CEC'09, pages 1943–1950, Piscataway, NJ, USA, 2009. IEEE Press.

[153] Aleš Zamuda, Janez Brest, Borko Boškovic, and Viljem Žumer. Differential evolution with self-adaptation and local search for constrained multiobjective optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, CEC ′ 09, pages 195–202, Piscataway, NJ, USA, 2009. IEEE Press.

[154] Lin-Yu Tseng and Chun Chen. Multiple trajectory search for unconstrained/constrained multi-objective optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, CEC ′ 09, pages 1951–1958, Piscataway, NJ, USA, 2009. IEEE Press.

[155] Karthik Sindhya, Ankur Sinha, Kalyanmoy Deb, and Kaisa Miettinen. Local search based evolutionary multi-objective optimization algorithm for constrained and unconstrained problems. In *Proceedings of IEEE Congress on Evolutionary Computation*, CEC ′ 09, pages 2919–2926, Piscataway, NJ, USA, 2009. IEEE Press.

[156] Chih-Ming Chen, Ying-ping Chen, and Qingfu Zhang. Enhancing moea/d with guided mutation and priority update for multi-objective optimization. In *Proceedings of IEEE Congress on Evolutionary Computation*, CEC ′ 09, pages 209–216, Piscataway, NJ, USA, 2009. IEEE Press.

[157] Layak Ali, Samrat L. Sabat, and K. Udgata, S. Adaptive and accelerated exploration particle swarm optimizer (aaepso) for solving constrained multi-objective optimization problems. In *Proceedings of International Conference on Swarm, Evolutionary and Memetic Computing*, pages 155–162, Chennai, India, December 2010.

[158] Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari. Multiobjective optimization Test Instances for the CEC 2009 Special Session and Competition. Technical report, Nanyang Technological University, Singapore, 2009.

[159] A. Deb, K. Pratap and S. Moitra. Mechanical component design for multiple objectives using elitist non-dominated sorting ga: Kangal report no. 200002. Technical report, Kanpur Genetic Algorithms Laboratory (KanGAL),Indian Institute of Technology Kanpur, India, 2000.

[160] Hemant Kumar Singh, Tapabrata Ray, and Warren Smith. C-psa: Constrained pareto simulated annealing for constrained multi-objective optimization. *Information Sciences*, 180(13):2499 – 2513, 2010.

[161] Erin Maneri and Wodek Gawronski. Lqg controller design using gui: Application to antennas and radio-telescopes. *ISA Transactions*, 39(2):243 – 264, 2000.

[162] E. L. Ulungu, J. Teghem, P. H. Fortemps, and D. Tuyttens. MOSA method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.

[163] Kay Chen Tan, Tong Heng Lee, D. Khoo, and Eik Fun Khor. A multiobjective evolutionary algorithm toolbox for computer-aided multiobjective optimization. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 31(4):537–556, 2001.

[164] Luisa Gama Caldas and Leslie K Norford. A design optimization tool based on a genetic algorithm. *Automation in Construction*, 11(2):173 – 184, 2002.

[165] Johannes Furtler, Konrad J. Mayer, Werner Krattenthaler, and Ivan Bajla. Spot development tool for software pipeline optimization for vliw-dsps used in real-time image processing. *Real-Time Imaging*, 9(6):387 – 399, 2003.

[166] J Mehnen, B Naujoks, K Schmitt, and D Zibold. Reihe computational intelligence kea - a software package for development, analysis and application of multiple objective evolutionary algorithms. *Computational Intelligence*, (November), 2004.

[167] Kalyanmoy Deb and Shamik Chaudhuri. I-emo: An interactive evolutionary multi-objective optimization tool. In *Proceedings of First International Conference on Pattern Recognition and Machine Intelligence: (PReMI-2005*, pages 690–695. Springer, 2005.

[168] Jussi Hakanen, Juha Hakala, and Jussi Manninen. An integrated multiobjective design tool for process design. *Applied Thermal Engineering*, 26(13):1393 – 1399, 2006.

[169] Joshua B. Kollat and Patrick Reed. A framework for visually interactive decision-making and design using evolutionary multi-objective optimization (video). *Environmental Modelling and Software*, 22:1691–1704, December 2007.

[170] H. Pohlheim. *Genetic and Evolutionary Algorithm Toolbox (GEATbx) for Matlab*. Available online at: www.geatbx.com, 2008.

[171] Tahir Sag and Mehmet Cunkas. A tool for multiobjective evolutionary algorithms. *Advances in Engineering Software*, 40(9):902 – 912, 2009.

[172] F. Boschetti, A. de La Tour, E.A. Fulton, and L.R. Little. Interactive modelling for natural resource management. *Environmental Modelling and Software*, 25(10):1075 – 1085, 2010.

Publications relevant to the thesis:

I. Research Papers published in International Refereed Journals

1. Samrat L. Sabat, **Layak Ali**, S.K.Udgata. Integrated Learning Particle swarm optimizer, Applied Soft Computing, 11(1), 2011, pp. 574-584, ISSN: 1568-4946, I**F 2.415**

2. Samrat L Sabat, **Layak Ali**. Hyperspherical Accelerated Effect Particle swarm optimizer, Applied Soft Computing. 9(13),2009, pp. 906-917, ISSN: 1568-4946, **IF 2.415**

II. Research Papers published in Peer Reviewed Int. Conferences

1. Samrat Sabat, **Layak Ali** and Siba K. Udgata, Stochastic ranking particle swarm optimization for constrained engineering design problems, In proceedings of International Conference on Swarm, Evolutionary and Memetic Computing, 2010, India , 6466 LNCS, pp. 672-679.

2. **Layak Ali**, Samrat Sabat and Siba K. Udgata, Adaptive and Accelerated Exploration Particle Swarm Optimizer (AAEPSO) for solving Constrained Multiobjective Optimization Problems, In proceedings of International Conference on Swarm, Evolutionary and Memetic Computing, 2010, India, 6466 LNCS, pp. 155-162 .

3. **Layak Ali**, Samrat Sabat and Siba K. Udgata, Particle swarm optimization technique with stochastic ranking for constrained optimization problems, In proceedings of 4^{th} Mahasarakham International Workshop on Artificial Intelligence, MIWAI 2010 , Thailand.

4. Samrat L Sabat, **Layak Ali**, Siba K Udgata Adaptive Accelerated Exploration Particle Swarm Optimizer: In Proceedings of IEEE World Congress on Nature and Biologically Inspired Computing , NaBIC , December 9-11,2009, India, pp. 654-659. (**Best Paper Awarded**), ISBN: 978-1-4244-5053-4

5. Samrat L. Sabat, **Layak Ali** Accelerated exploration particle swarm optimizer, In proceedings of IEEE International Conference TENCON 2008,, Hyderabad Nov 19-21, 2008, pp 1-6. ISBN: 978-1-4244-2408-5

6. Samrat L. Sabat, M.Vijayaraju , **Layak Ali** MESFET Small Signal Model Parameter Extraction Using Particle Swarm Optimization, In proceedings of 19^{th} IEEE International Conference on Microelectronics (ICM)-2008, Sharjah , Dec 14-17,2008, pp.183-186, ISBN: 978-1-4244-2369-9

7. Samrat L. Sabat, **Layak Ali**, Combined Effect Comprehensive Learning Particle Swarm Optimizer: ISARC -2007, In proceedings of 24^{th} international conference on Automation and Robotics in Construction ISARC 2007, September 19-21, 2007, pp . 347-354

8. Samrat L. Sabat, **Layak Ali**, Integrated Learning Particle swarm optimizer: In proceedings of 3^{rd} Indian International Conference on Artificial Intelligence (IICAI-07), December 17-19, 2007, pp.421-440.

III. Research Papers Communicated to International Refereed Journals

1. **Layak Ali**, Samrat Sabat and Siba K. Udgata, Particle swarm optimization with stochastic ranking for constrained numerical and engineering benchmark problems, International Journal of Bio-Inspired Computation (IJBIC), Inderscience Publishers, ISSN: 1758-0374, **(under revision)**.

2. **Layak Ali**, Samrat L Sabat, Siba K Udgata, Particle swarm optimization with adaptive acceleration for Multi-objective Optimization Problem, Swarm and Evolutionary Computation, Elsevier, **(under revision)**.

3. **Layak Ali**, Samrat L Sabat, Siba Udgata An improved Particle swarm optimization algorithm for solving constrained multiobjective numerical and engineering design problems, communicated to Journal of Computational and Applied Mathematics.

IV. Research Papers Communicated to Peer Reviewed Int. Conferences

1. **Layak Ali**, Samrat L Sabat. Particle Swarm Optimization Universal Solver, communicated to 1st Taibah University International Conference on Computing and Information Technology, Al-Madinah Al-Munawwarah, Saudi Arabia. (**Accepted**)