<u>**ISTANBUL KULTUR UNIVERSITY ★ INSTITUTE OF SOCIAL SCIENCES**</u>

PARTICLE SWARM OPTIMIZATION AND DIFFERENTIAL EVOLUTION ALGORITHMS FOR CONTINUOUS FUNCTION OPTIMIZATION PROBLEMS

MA Thesis by
İpek EKER

0310010008

Department: Business Administration
Programme: Business Administration

Supervisor: Prof .Dr. Güneş GENÇYILMAZ

AUGUST 2005

ISTANBUL KULTUR UNIVERSITY ★ INSTITUTE OF SOCIAL SCIENCE

PARTICLE SWARM OPTIMIZATION AND DIFFERENTIAL EVOLUTION ALGORITHMS FOR CONTINUOUS FUNCTION OPTIMIZATION PROBLEMS

İpek EKER
0310010008

Date of submission :	**25 July 2005**
Date of defence examination:	**05 August 2005**

Supervisor (Chairman):	**Prof. Dr. Güneş GENÇYILMAZ**
Members of the Examining Committee	**Prof. Dr. Güneş GENÇYILMAZ**
	Asst. Prof. Dr. M. Fatih TAŞGETİREN
	Asst.Prof. Dr. Rıfat Gürcan ÖZDEMİR

AUGUST 2005

ACKNOWLEDGMENTS

This thesis is dedicated to my parents for their constant support and encourragement throughout preparing this dissertation. I am greatly indebted to my advisor Prof. Dr. Güneş Gençyılmaz for his guidance and constant support during the preparation and investigation of this thesis. I am also indebted to Asst. Prof. Dr. M. Fatih Taşgetiren who contributed with the idea of this research topic. I also appericate the support provided by İstanbul Kültür University for the opportunity to obtain the master degree in Business Administration.

İpek Eker

August, 2005

TABLE OF CONTENTS

LIST OF ABBREVIATIONS

GA	:	Genetic Algorithm
SA	:	Simulated Annealing
EP	:	Evolutionary Programming
ES	:	Evolutionary Strategy
GP	:	Genetic Programming
TS	:	Tabu Search
CA	:	Chaos Algorithm
ACO	:	Ant Colony Optimization
PSO	:	Particle Swarm Optimization
DE	:	Differential Evolution
lbest	:	local best
gbest	:	global best
RTS	:	Reactive Tabu Search
NP	:	Population size
D	:	Dimension

FES	:	Function Evaluations
Std. D.	:	Standard deviation
Trm	:	Termination
AL	:	Accuracy Level
SR	:	Success Rate
MAGA	:	Multiagent Genetic Algorithm

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

X_i^t : the i^{th} particle in the swarm at iteration t

x_{ij}^t : position value of the i^{th} particle with respect to the j^{th} dimension

X^t : the set of NP particles in the swarm at iteration t

V_i^t : the velocity of particle i at iteration t

v_{ij}^t : the velocity of particle i at iteration t with respect to the j^{th} dimension

w^t : inertia weight

c_1 : acceleration coefficient

c_2 : acceleration coefficient

P_i^t : the best position of the particle

f_i^{pb} : the fitness function of the personal best

p_{ij}^t : the position value of the i^{th} personal best with respect to the j^{th} dimension

G^t : the best position of the globally best particle

f^{gb} : the fitness function of the global best

g_j^t : the position value of the global best with respect to the j^{th} dimension

r_1 : a uniform random number between 0 and 1

r_2 : a uniform random number between 0 and 1

f_i^t : the fitness function value of the particle X^t

The basic elements of the DE algorithm is summarized as follows:

X_i^t : the i^{th} individual in the population at generation t

V_i^t : the i^{th} individual in the population at generation t

U_i^t : the i^{th} individual in the population at generation t

X^t : the set of NP individuals in the population at generation t

V^t : the set of NP individuals in the population at generation t

U^t : the set of NP individuals in the population at generation t

F : Mutant constant $F \in (0,2)$

CR : user-defined crossover constant in the range [0, 1]

r_{ij}^{t+1} : a uniform random number between 0 and 1

University	**:**	**Istanbul Kultur University**
Institute	**:**	**Institute of Social Sciences**
Department	**:**	**Business Administration**
Programme	**:**	**Business Administration**
Supervisor	**:**	**Prof. Dr. Güneş GENÇYILMAZ**
Degree Awarded and Date	**:**	**MA – August 2005**

ABSTRACT

PARTICLE SWARM OPTIMIZATION AND DIFFERENTIAL EVOLUTION ALGORITHMS FOR CONTINUOUS OPTIMIZATION PROBLEMS

Ipek EKER

This study presents Particle Swarm Optimization (PSO) and Differential Evolution (DE) algorithms to solve nonlinear continuous function optimization problems. The algorithms were tested using 14 newly proposed benchmark instances in Congress on Evolutionary Computation 2005.

Particle Swarm Optimization (PSO) and Differential Evolution (DE) are two of the latest metaheuristic methods. PSO is based on the metaphor of social interaction and communication such as bird flocking and fish schooling. PSO and DE were both first introduced to optimize various continuous nonlinear functions.

In a PSO algorithm, each member is called a *particle*, and each particle moves around in the multi-dimensional search space with a velocity constantly updated by the particle's experience, the experience of the particle's neighbors, and the experience of the whole swarm.

In the DE algorithm, the target population is perturbed with a mutant factor, and the crossover operator is then introduced to combine the mutated population with the target population so as to generate a trial population. Then the selection operator is applied to compare the fitness function value of both competing populations, namely, target and trial populations. The better individuals among these two populations become members of the population for the next generation. This process is repeated until a convergence occurs.

The computational results show that the particle swarm optimization is able to solve the test problems. Both algorithms are promising to solve benchmark problems. However, the differential evolution algorithm performed better for the larger size of problems than the particle swarm optimization algorithm.

Key Words : Particle Swarm Optimization, Differential Evolution, Continuous Optimization, Genetic Algorithms.

Üniversitesi	**:**	**İstanbul Kültür Üniversitesi**
Enstitüsü	**:**	**Sosyal Bilimler**
Anabilim Dalı	**:**	**İşletme**
Programı	**:**	**İşletme**
Tez Danışmanı	**:**	**Prof. Dr. Güneş GENÇYILMAZ**
Tez Türü ve Tarihi	**:**	**Yüksek Lisans – Ağustos 2005**

KISA ÖZET

SÜREKLİ FONKSİYON OPTİMİZASYON PROBLEMLERİNİN ÇÖZÜMÜ İÇİN PARÇACIK SÜRÜ OPTİMİZASYONU (PSO) VE DİFERANSİYAL EVRİM (DE) ALGORİTMALARI

İpek EKER

Bu çalışma, doğrusal olmayan sürekli fonksiyon optimizasyon problemlerinin çözümü için Parçacık Sürü Optimizasyonu (PSO) ve Diferansiyal Evrim (DE) algoritmalarını sunmaktadır. Algoritmaların performansı Evrimsel Hesap Kongresi (CEC2005) için yeni geliştirilen 14 fonksiyonu kullanarak test edildi.

Parçacık Sürü Optimizasyonu (PSO) ve Diferansiyel Evrim (DE), en son geliştirilen meta-sezgisel yöntemlerden ikisidir. PSO, kuşların ve balıkların yem arama gibi sosyal etkileşmeleri ve iletişimleri metaforuna dayanır. PSO ve DE, orijinal olarak çeşitli doğrusal olmayan sürekli fonksiyonları optimize etmek için geliştirildi.

PSO algoritmasında, her bir üye, "parçacık" olarak adlandırılır ve her parçacık, çoklu-boyutsal arama uzayında bir hız ile hareket eder. Bu hız, parçacığın kendi deneyimi, komşularının deneyimi ya da populasyondaki bütün

parçacıkların deneyimi ile sürekli olarak güncellenir.DE algoritmasında, hedef populasyon mutasyon faktörü ile farklılaştırılır ve daha sonra deneme populasyonu oluşturmak için çaprazlama operatörü kullanılır. Çaprazlama operatörünün amacı farklılaştırılan populasyonla hedef populasyonu birleştirerek deneme populasyonunu oluşturmaktır. Son olarak, seçme operatörü kullanılarak rekabet eden her iki populasyon özellikle hedef ve deneme populasyonlarının amaç fonksiyon değerleri karşılaştırılır. Seçme operatörü vasıtasıyla bu iki populasyon arasındaki daha iyi çözümler bir sonraki jenerasyona ait populasyonun üyeleri haline gelir. Bu proses yakınsaklık elde edilinceye dek tekrar edilir.

Deneysel sonuçlar her iki algoritmanın test problemlerini belli bir hata payıyla veya optimal olarak çözebildiğini göstermektedir. Her iki algoritma, test problemlerini çözmede umut vericidir. Ancak, diferansiyel evrim algoritması büyük çaplı problemler için parçacık sürü optimizasyonu algoritmasından daha iyi sonuçlar üretmektedir.

Anahtar Sözcükler : Parçacık Sürü Optimizasyonu, Diferansiyel Evrim, Sürekli Fonksiyonu Optimizasyon Problemleri, Genetik Algoritmalar.

CHAPTER 1: INTRODUCTION

1.1. Literature Survey on Global Optimization Algorithms

The development of global optimization algorithm is closely bound up the development of computer. Many global optimization problems wait for solving and some algorithms are also put forward along with the greatness and complexity of the structures in engineering, especially these years.

1.2. Frame Work of the Algorithms

Two stages must be experienced in the process of solving the global optimum. The first stage can be called the global covering. The global optimum may be located in arbitrary region in the feasible region for the optimization problems in engineering, so any parts of the region must be considered equally critical. The stage of uniform distributing in the region *A* is required in this stage. The last stage is called local fine searching. It requires a stage of non-uniform distributing in the neighborhood of known better points, because some parts of the feasible region may be deemed more interesting than others and more accurate solutions in these parts are wanted.

1.3. Classification of Global Optimization Algorithms

The classification is made by Leon early in 1966 in [1], who classifies these algorithms into three kinds according to the search techniques: Blind search, Local search and Non-local search. Subsequently, Dixon. Szegio and Gomulka present two basic approaches namely the deterministic and probabilistic algorithms in 1978 in [2, 3]. The former comprises grid search algorithms and trajectory algorithms, the latter comprises random search algorithms, clustering algorithms and sampling algorithms.

Thereafter, Archetti and Schoen also makes between deterministic and probabilistic algorithms in 1984 in [4]. According to accuracy, Torn make two classifications namely algorithms with guaranteed accuracy and algorithms without, the latter comprises direct algorithms and indirect algorithms in [5]. Zhang Xiangsun reviews the deterministic algorithms in detail in 1984 in [6] and Zhang Yunkang also reviews the probabilistic algorithms in detail in 1992 in [7].

1.4. General Classification

General classification of all global algorithms primarily should be divided into three classes according to the different searching methods: Analytic algorithm, Enumerate algorithm and Random search algorithm. Analytic properties of objective functions are exerted to seek the global optimum in this algorithm (such as first-order, second-order derivative), which is divided into Direct algorithms and Indirect algorithm. The next searching step of direct algorithms is determined by the grade of objective functions. "Mountain climbing" strategy is adopted in this algorithm, which searches one of the local optimum according to the steepest direction (such as Cluttering algorithm and Generalized descent algorithm). But it is difficult to search the global optimum. The indirect algorithm is that a group of equations is educed by the necessary conditions of extremum, then the group of equations is solved and the global optimum is found by comparison. But the equations are always non-linearity, which are difficult to be solved. So it is applied for some very simple optimization, such as algorithms approximating the level sets and algorithms approximating the objective function. Enumerate algorithm is mostly applied in the field of dynamic programming. Random search algorithm is composed of Blind search algorithms and Guide search algorithms. Blind search algorithm includes covering algorithm and Random search algorithm. A very large computing effort is needed, so it is only applied in simple optimization; Guide search algorithms are also called Heuristic search algorithms, which are studied more frequently in present years, which include Meta-heuristic algorithms [8], algorithms based on uniform design [9, 10, 11] and mixed heuristic algorithms [12-21]. Meta-heuristic algorithms are studied more nowadays, which include simulated annealing [22] (SA), evolution algorithms (which include Genetic Algorithm [23-30] (GA), Evolutionary Programming [31] (EP), Evolutionary Strategy [32] (ES) and Genetic Programming [33] (GP)), Tabu

Search Algorithm, Chaos Algorithm [34, 35], Ant Colony Optimization [36, 37] (ACO), and so on. The mixed heuristic algorithms are researched relatively less at present, which mostly aim at the shortages of the intelligent heuristic search algorithms, and whose results and efficiency are better than the simple heuristic search algorithms, so the algorithms are the hotspot of the optimization research. Additionally, the heuristic search algorithms mixed with some local algorithms are also one of the future optimization research tendencies. Meta-heuristic algorithms are introduced as follows.

Meta-heuristic algorithms are developed along with the development of biology, physics and artificial intelligence. Although the optimal mechanisms are different, they are the same in optimal technological processes, which are a kind of "neighbor region search". The process of the algorithms is as follows: (1) start from one (or one group) initial point; (2) search many neighbor solves by the neighbor functions under the control of the algorithms parameters; (3) renew the current state according to the accept rules; (4) then adjust the control parameters, so repeat this process as to satisfy the accept rules.

SA is a clustering optimal algorithm, whose principle is: a state in the neighbor region at present is sampled randomly, at the same time renew the probability according to controlling "temperature", so that the search process has the ability of avoiding local optimum, and get the global optimum finally. The initial temperature, the functions of withdrawing temperature, the renew mode of states and the sample stabilization are the key factors which affect the performance of SA.

GA is a combining algorithm, which especially has the concealed combining property. Its principle is: in the code space, the processes of select, crossover and mutation are implemented ceaselessly according to a probability, so as to the aim of the group's combining evaluation. The number of group and the operations of reproduce, crossover and mutation are the key factors, which affect the performance of GA.

TS is a clustering optimal algorithm, which avoids repeating the states according to the operational memory structures in the near future, and which implements global search rapidly combining the deprecate rule. The size of the tabu table and the structure and the number of the function in neighbor region are the key factors, which affect the performance of TS.

Chaos is a non-linearity phenomenon in nature. The movement process of chaos variables has inherent rule. The randomness, the property of covering all over and regularity are used to search the optimum. The operation of CA includes two steps. Firstly, in the whole space, all points are inspected in turn by the movement of chaos variables, and the better point is accepted the optimum at present. Secondly, after certain steps the optimum at present is near the global optimum, then the optimum at present become the center and are added a little chaos change, the global optimum is attained through careful search. Repeating the two steps upwards, until the global optimum is attained. CA is a Random search algorithm, which is researched relatively less presently.

ACO, a new type of simulated evolutionary algorithm, is proposed first by Italian scholars Marco Dorigo. It is used to solve some optimization problems through simulating the process of ants searching for food, which is carried out through searching the shortest route between the ant cave and the food according to the individual information interchange and cooperation with one another.

Particle Swarm Optimization (PSO), one of the latest metaheuristic algorithms, was first introduced by Kennedy and Eberhart 1995 in [38]. PSO is based on the metaphor of social interaction and communication such as bird flocking and fish schooling. Since PSO is population-based and socially cognitive in nature, the members in a swarm tend to follow the leader of the group, i.e., the one with the best performance. In a PSO algorithm, each member is called a “particle”, and each particle flies around in the multi-dimensional search space with a velocity, which is updated according to the particle’s current velocity, the particle’s own experience and the experience of the neighbors. Depending on the size of neighbors, two types of basic PSO algorithms were developed – PSO with a local neighborhood and PSO with global neighborhood of Kennedy et al. 2001 in [39]. In the former model, called the *lbest*, each particle moves towards its best previous position and towards the best particle in its restricted neighborhood. While in the latter model, called the *gbest*, each particle moves towards its best previous position and towards the best particle in the entire swarm.

Differential evolution (DE) is also one of the latest evolutionary optimization methods proposed by Storn and Price 1997 in [40]. It is a simple but powerful population based stochastic search method for solving global optimization problems. Like other evolutionary-type algorithms, DE is a population-based, stochastic global optimizer. In a DE algorithm, candidate solutions are represented as chromosomes based on floating-point numbers. The major difference between DE and genetic algorithm (GA) is that in DE some of the parents are generated through a mutation process before performing crossover operator whereas GA usually selects parents from current population, performs crossover, and then mutates the offspring. In the mutation process of a DE algorithm, the weighted difference between two randomly selected population members is added to a third member to generate a mutated solution. Then, the crossover operator is introduced to combine the mutated solution with the target solution so as to generate a trial solution. Then a selection operator is applied to compare the fitness function value of both competing solutions, namely, target and trial solutions to determine who can survive for next generation.

Regarding the application of optimization algorithms for the continuous functions, few works deal with the application to the global minimization of functions depending on continuous variables. The works related to the subject are in [41, 42, 43, 44, 45, 46, 47, 48]. In addition, a simple benchmark on a function with many suboptimal local minima is considered in [49], where a straightforward discretization of the domain is used. A novel algorithm for the global optimization of functions (C-RTS) is presented in [50], in which a combinatorial optimization method cooperates with a stochastic local minimizer. The combinatorial optimization component, based on RTS, locates the most promising boxes , where starting points for the local minimizer are generated. In order to cover a wide spectrum of possible applications with no user intervention, the method is designed with adaptive mechanisms: in addition to the reactive adaptation of the prohibition period , the box size is adapted to the local structure of the function to be optimized (boxes are larger in ``flat" regions, smaller in regions with a ``rough" structure).

This thesis is organized as follows. Chapter 2 and 3 develops the PSO and DE algorithms to solve the nonlinear continuous functions, respectively. Chapter 4 introduces 14 newly developed benchmark functions and the performance criteria employed in this study. Computational results for PSO and DE algorithms are shown in Chapter 5. Chapter 6 compares both algorithms. Finally, Chapter 7 summarizes the concluding remarks.

CHAPTER 2: PARTICLE SWARM OPTIMIZATION ALGORITHM

2.1. Particle Swarm Optimization Algorithm

PSO was first developed to optimize continuous nonlinear functions. Since PSO is easy to implement and is efficient to obtain quality solutions, it has attracted much researchers' attention in recent years. The application of PSO consists of neural network training in [51, 52, 53], power and voltage control in [54], optimal power system design in [55, 56], feature selection in [57], mass-spring system in [58], electromagnetics in [59, 60], analyze of human tremor in [61], register 3D-to-3D biomedical image in [62], play games in [63], clustering in [64], logic circuit design in [65], lot sizing problem in [66], supplier selection and ordering problems in [67], task assignment problem in [68], automated drilling in [69], and scheduling problems in [70, 71, 72]. More literature can be found in [39]. Besides the wide range of applications above, the nonlinear continuous function optimization is still considered the benchmark problem when exploring the properties and performance of PSO algorithms. Therefore, this thesis aims at employing PSO in optimizing 14 newly developed test problems in Congress on Evolutionary Computation 2005.

The *gbest* model of Kennedy et al. 2001 in [39] is followed in this study. According to the *gbest* model, each particle moves towards its best previous position and towards the best particle in the whole swarm. In the PSO algorithm, parameters were initialized and the initial population was generated randomly. Each particle will then be evaluated to compute the fitness function value. After evaluation, the PSO algorithm repeats the following steps iteratively: With its position, velocity, and fitness value, each particle updates its personal best (best value of each individual so far) if an improved fitness value was found. On the other hand, the best particle in the whole swarm with its position and fitness value was used to update the global best (best particle in the whole swarm). Then the velocity of the particle is updated by using its previous velocity, the experiences of the personal best, and the global best in order to determine the position of each particle. Evaluation is again performed to compute the fitness of the particles in the swarm. This process is terminated with a predetermined stopping criterion. The pseudo code of the PSO algorithm is given in Figure 2.1.

```
    Initialize parameters
    Initialize population
    Evaluate
    Do {
        Find the personal best
        Find the global best
        Update the velocity
        Update the position
        Evaluate
} While (Termination)
```

Figure 2.1 A Simple PSO Algorithm.

The basic elements of PSO algorithm is summarized as follows:

Particle: X_i^t denotes the *i*th particle in the swarm at iteration *t* and is represented by $X_i^t = \lfloor x_{i1}^t, x_{i2}^t, .., x_{iD}^t \rfloor$, where x_{ij}^t is the position value of the *i*th particle with respect to the *j*th dimension ($j = 1,2,...,D$).

Population: X^t is the set of NP particles in the swarm at iteration *t*, i.e., $X^t = \lfloor X_1^t, X_2^t, ..., X_{NP}^t \rfloor$.

Particle velocity: V_i^t is the velocity of particle *i* at iteration *t*. It can be defined as $V_i^t = \lfloor v_{i1}^t, v_{i2}^t, ..., v_{iD}^t \rfloor$, where v_{ij}^t is the velocity of particle *i* at iteration *t* with respect to the *j*th dimension.

Inertia weight and acceleration coefficients: w^t is a parameter to control the impact of the previous velocities on the current velocity as described in [73, 74]. It has an impact on the trade-off between the global and local exploration capabilities of the particle. At the beginning of the search, large inertia weight is used to enhance the global exploration while it is reduced for better local exploitation later on in the search. c_1 and c_2 are constant parameters called acceleration coefficients which control the maximum step size that the particle can do.

Personal best: P_i^t represents the best position of the particle with the best fitness value until iteration *t*, so the best position associated with the best fitness value of the particle obtained so far is called the *personal best*. For each particle in the swarm, P_i^t can be determined and updated at each iteration *t*. In a minimization problem with the objective function $f\left(X_i^t\right)$, the personal best P_i^t of the *i*th particle is obtained such that $f\left(P_i^t\right) \le f\left(P_i^{t-1}\right)$. To simplify, the fitness function of the personal best is denoted as $f_i^{pb} = f\left(P_i^t\right)$. For each particle, the personal best is defined as $P_i^t = \left\lfloor p_{i1}^t, p_{i2}^t, ..., p_{iD}^t \right\rfloor$ where p_{ij}^t is the position value of the *i*th personal best with respect to the *j*th dimension ($j = 1,2,...,D$).

Global best: G^t denotes the best position of the globally best particle achieved so far in the whole swarm. For this reason, the global best can be obtained such that $f\left(G^t\right) \le f\left(P_i^t\right)$ for $i = 1,2,..., NP$. To simplify, the fitness function of the global best is denoted as $f^{gb} = f\left(G^t\right)$. The global best is then defined as $G^t = \left\lfloor g_1^t, g_2^t, ..., g_D^t \right\rfloor$ where g_j^t is the position value of the global best with respect to the *j*th dimension ($j = 1,2,...,D$).

Termination criterion: It is a condition that terminates the search process. It might be a maximum number of function evaluations or a maximum CPU time that terminates the search.

2.2. Initial Population

A population of particles is constructed randomly for the PSO algorithm. The continuous values of positions are established randomly. The following formula is used to construct the initial continuous position values of the particle uniformly:

$$x_{ij}^0 = x_{\min} + \left(x_{\max} - x_{\min}\right) * r_1$$

where $x_{\min}$ and $x_{\max}$ are the search range of the continuous functions and r_1 is a uniform random number between 0 and 1. Initial velocities are generated by a similar formula as follows:

$$v_{ij}^0 = v_{\min} + (v_{\max} - v_{\min}) * r_2$$

where $v_{\max} = (x_{\max} - x_{\min})/2$ and $v_{\min} = -v_{\max}$, and r_2 is a uniform random number between 0 and 1. Continuous velocity values are restricted to some range, namely $v_{ij}^t = [v_{\min}, v_{\max}]$

During the reproduction of the PSO algorithm, it is possible to extend the search outside of the initial range of the search space. For this reason, the position values violating the initial range are restricted to the feasible range as follows:

$$x_{ij}^t = x_{\min} + (x_{\max} - x_{\min}) * r_1$$

The only exception was the problem 7 for which the optimal was outside the search range. The population size is taken as 100. As the formulation of 14 functions suggests that the objective is to minimize 14 continuous functions, the fitness function value is the objective function value of the particle X^t. That is, $f_i^t(X_i^t)$. For simplicity, $f_i^t(X_i^t)$ will be denoted as f_i^t.

2.3. Computational Procedure

The complete computational procedure of the PSO algorithm can be summarized as follows:

Step 1: Initialization

- Set $t = 0$, $NP = 100$.
- Generate NP particles randomly as explained before, $\{X_i^0, i = 1,2,...,\mathrm{NP}\}$ where $X_i^0 = [x_{i1}^0, x_{i2}^0, ..., x_{iD}^0]$.
- Generate the initial velocities for each particle randomly, $\{V_i^0, i = 1,2,...,\mathrm{NP}\}$ where $V_i^0 = [v_{i1}^0, v_{i2}^0, ..., v_{iD}^0]$.
- Evaluate each particle in the swarm using the objective function f_i^0 for $i = 1,2,..., NP$.

- For each particle in the swarm, set $P_i^0 = X_i^0$, where $P_i^0 = \left[p_{i1}^0 = x_{i1}^0, p_{i2}^0 = x_{i2}^0, ..., p_{iD}^0 = x_{iD}^0\right]$ together with its best fitness value, f_i^{pb} for $i = 1,2,..., \mathrm{NP}$.
- Find the best fitness value among the whole swarm such that $f_l = \min\{f_i^0\}$ for $i = 1,2,..., NP$ with its corresponding positions X_l^0 . Set global best to $G^0 = X_l^0$ such that $G^0 = \left[g_1 = x_{l,1}, g_2 = x_{l,2}, ..., g_D = x_{l,D}\right]$ with its fitness value $f^{gb} = f_l$.

Step 2: Update iteration counter

- $t = t + 1$

Step 3: Update inertia weight

- $w^t = \left(\left(\max\_fes - FES\right) / \max\_fes\right) * \left(w_0 - w_n\right) + w_n$

where $\max\_fes$, FES, w_0, and w_n are the maximum number of function evaluation, number of function evaluations, initial inertia weight, and final inertia weight respectively.

Step 4: Update velocity

- $v_{ij}^t = w^{t-1} v_{ij}^{t-1} + c_1 r_1 \left(p_{ij}^{t-1} - x_{ij}^{t-1}\right) + c_2 r_2 \left(g_j^{t-1} - x_{ij}^{t-1}\right)$

where c_1 and c_2 are acceleration coefficients and r_1 and r_2 are uniform random numbers between 0 and 1.

Step 5: Update position

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

Step 6: Update personal best

- Each particle is evaluated to see if the personal best will improve. That is, if $f_i^t < f_i^{pb}$ for $i = 1,2,..., \mathrm{NP}$ then personal best is updated as $P_i^t = X_i^t$ and $f_i^{pb} = f_i^t$.

Step 7: Update global best

- Find the minimum value of personal best. That is, $f_l^t = \min\{f_i^{pb}\}, i = 1,2,..., \mathrm{NP}; l \in \{i; i = 1,2,..., \mathrm{NP}\}$.
- If $f_l^t < f^{gb}$, then the global best is updated as $G^t = X_l^t$ and $f^{gb} = f_l^t$.

Step 8: Stopping criterion

- If the number of function evaluations exceeds the maximum number of function evaluations, then stop; otherwise go to step 2.

The flowchart of the computational procedure is given in Figure 2.2.

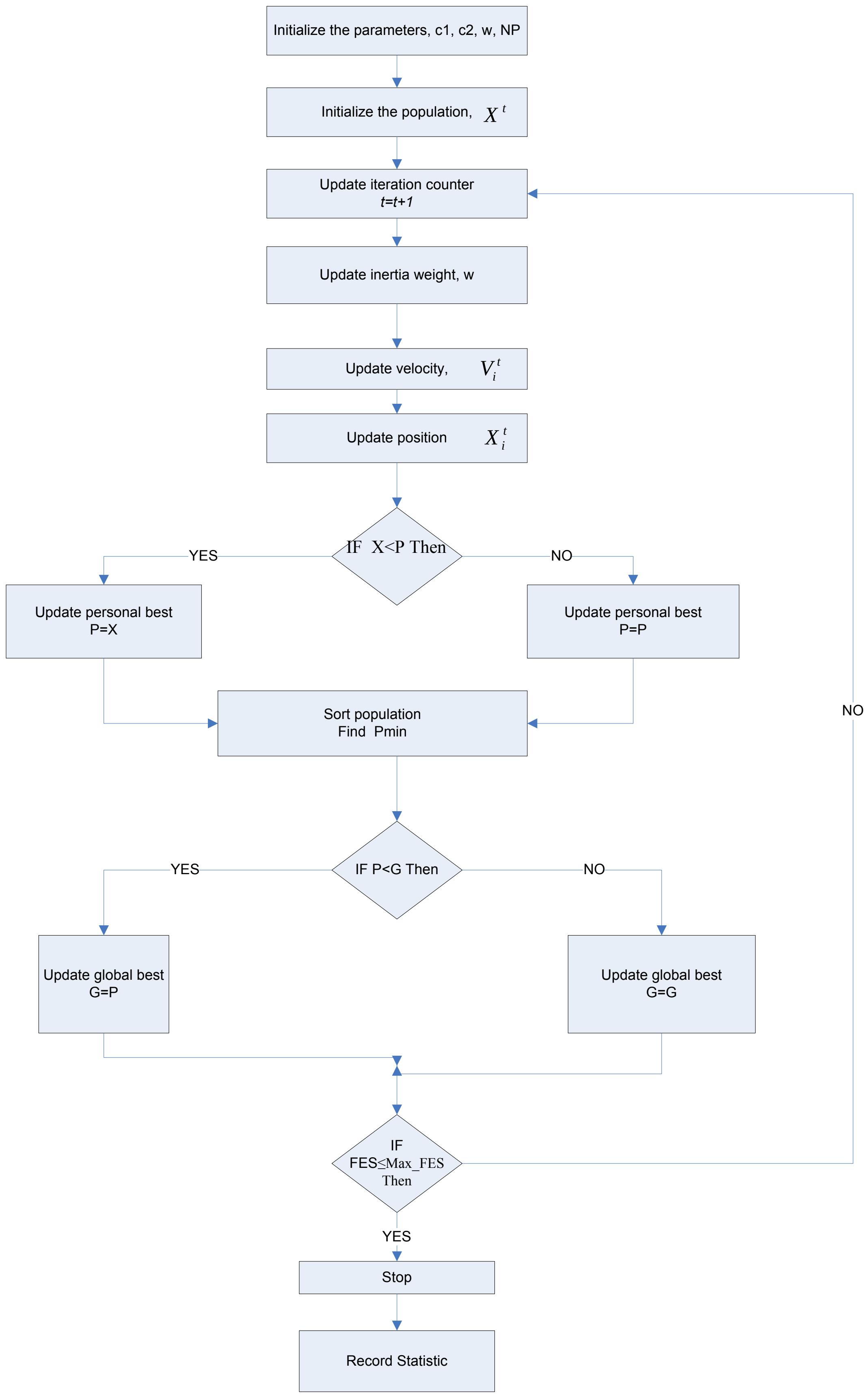
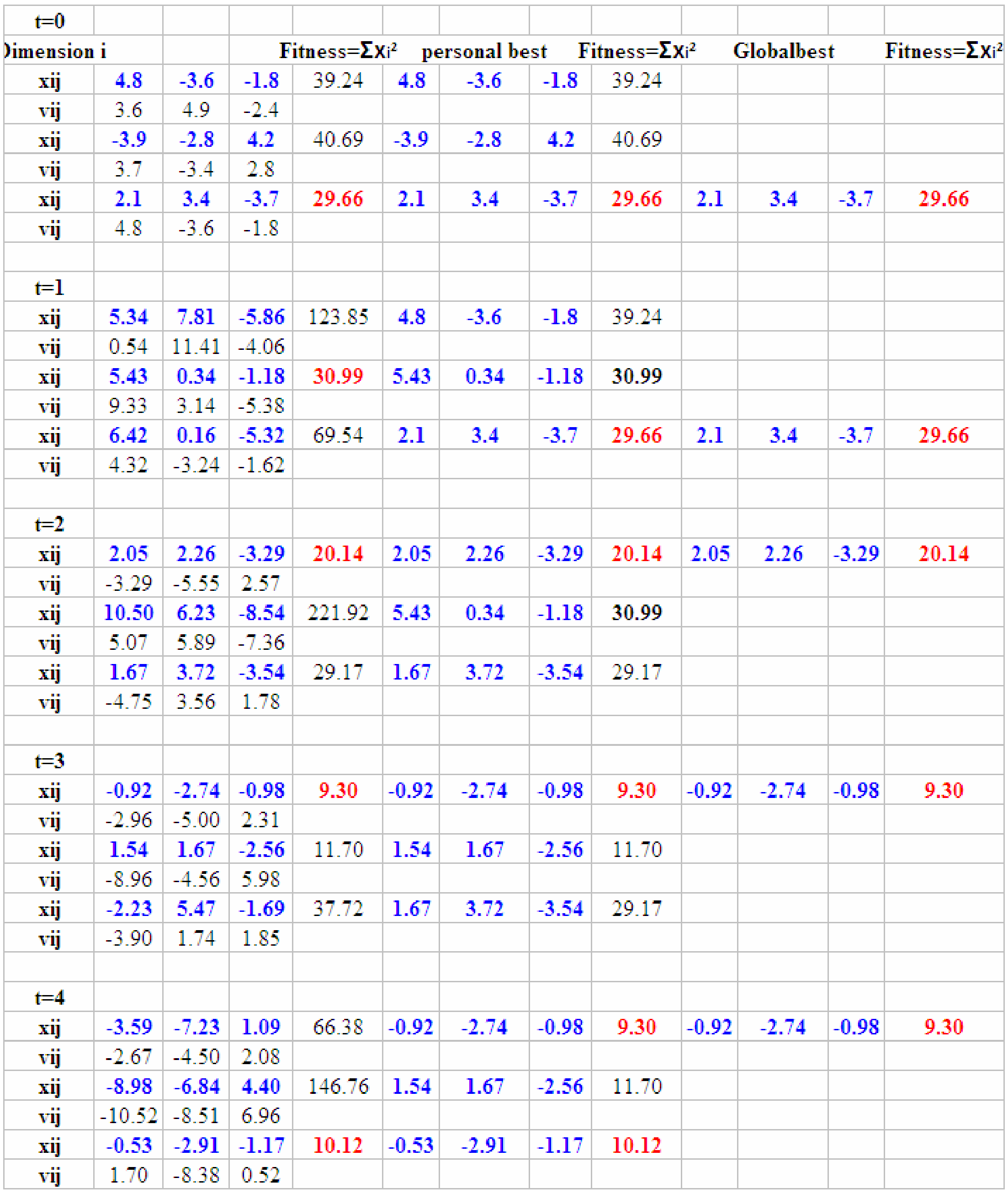


Figure 2.2 Flowchart of the PSO algorithm.

2.4. An Example for PSO Algorithm

In this section, an example of minimization of Sphere Function with 3 dimensions is given below:



t=0												
Dimension i			Fitness=Σx_i^2	personal best			Fitness=Σx_i^2	Globalbest			Fitness=Σx_i^2	
xij	4.8	-3.6	-1.8	39.24	4.8	-3.6	-1.8	39.24				
vij	3.6	4.9	-2.4									
xij	-3.9	-2.8	4.2	40.69	-3.9	-2.8	4.2	40.69				
vij	3.7	-3.4	2.8									
xij	2.1	3.4	-3.7	29.66	2.1	3.4	-3.7	29.66	2.1	3.4	-3.7	29.66
vij	4.8	-3.6	-1.8									
t=1												
xij	5.34	7.81	-5.86	123.85	4.8	-3.6	-1.8	39.24				
vij	0.54	11.41	-4.06									
xij	5.43	0.34	-1.18	30.99	5.43	0.34	-1.18	30.99				
vij	9.33	3.14	-5.38									
xij	6.42	0.16	-5.32	69.54	2.1	3.4	-3.7	29.66	2.1	3.4	-3.7	29.66
vij	4.32	-3.24	-1.62									
t=2												
xij	2.05	2.26	-3.29	20.14	2.05	2.26	-3.29	20.14	2.05	2.26	-3.29	20.14
vij	-3.29	-5.55	2.57									
xij	10.50	6.23	-8.54	221.92	5.43	0.34	-1.18	30.99				
vij	5.07	5.89	-7.36									
xij	1.67	3.72	-3.54	29.17	1.67	3.72	-3.54	29.17				
vij	-4.75	3.56	1.78									
t=3												
xij	-0.92	-2.74	-0.98	9.30	-0.92	-2.74	-0.98	9.30	-0.92	-2.74	-0.98	9.30
vij	-2.96	-5.00	2.31									
xij	1.54	1.67	-2.56	11.70	1.54	1.67	-2.56	11.70				
vij	-8.96	-4.56	5.98									
xij	-2.23	5.47	-1.69	37.72	1.67	3.72	-3.54	29.17				
vij	-3.90	1.74	1.85									
t=4												
xij	-3.59	-7.23	1.09	66.38	-0.92	-2.74	-0.98	9.30	-0.92	-2.74	-0.98	9.30
vij	-2.67	-4.50	2.08									
xij	-8.98	-6.84	4.40	146.76	1.54	1.67	-2.56	11.70				
vij	-10.52	-8.51	6.96									
xij	-0.53	-2.91	-1.17	10.12	-0.53	-2.91	-1.17	10.12				
vij	1.70	-8.38	0.52									

Figure 2.3 An example for PSO algorithm.

CHAPTER 3: DIFFERENTIAL EVOLUTION ALGORITHM

3.1. Differential Evolution Algorithm

Since DE was first introduced to solve the Chebychev polynomial fitting problem by Storn and Price 1995 in [75], it has been successfully applied in a variety of applications including digital filter design in [76, 77], neural network training in [78], pattern recognition in [79], communication in [80],aerodynamic design in [81], earthquake relocation in [82], microprocessor synthesis in [83], permutation flowshop sequencing problems in [84], multisensor fusion in [85], heat transfer in [86], system design in [87], cancer diagnosis in [88], and scheduling problems in [89]. A number of recent studies comparing DE with other heuristics, such as GA and PSO regarding real-world and artificial problems indicate superiority of DE in single-objective, noise free, numerical optimisation in [90, 91, 92, 93]. More introduction and literature surveys of DE can be found in [94, 95, 96, 97]. In addition, the advantages of DE such as simple concept, immediately accessible for practical applications, simple structure, ease of use, speed to get the solutions, and robustness has all led itself a good candidate to solve difficult nonlinear continuous functions. Therefore, this thesis aims at employing DE to optimize 14 newly developed benchmark suite in Congress on Evolutionary Computation 2005.

Currently, there exist several variants of DE. We follow the *DE/rand/1/bin* scheme of Storn and Price 1997 in [98]. The pseudo code of the DE algorithm is given in Figure 3.1.

```
Initialize parameters
Initialize target population
Evaluate
Do {
        Obtain the mutant population
        Obtain the trial population
        Evaluate trial population
        Selection
        While (Termination)
```

Figure 3.1 A Simple DE Algorithm

The basic elements of the DE algorithm is summarized as follows:

Target individual: X_i^t denotes the *i*th individual in the population at generation *t* and is represented as $X_i^t = [x_{i1}^t, x_{i2}^t, .., x_{iD}^t]$, where x_{ij}^t is the optimized parameter value of the *i*th individual with respect to the *j*th dimension ($j = 1,2,..., D$).

Mutant individual: V_i^t denotes the *i*th individual in the population at generation *t* and is represented as $V_i^t = [v_{i1}^t, v_{i2}^t, .., v_{iD}^t]$, where v_{ij}^t is the optimized parameter value of the *i*th individual with respect to the *j*th dimension ($j = 1,2,..., D$).

Trial individual: U_i^t denotes the *i*th individual in the population at generation *t* and is represented as $U_i^t = [u_{i1}^t, u_{i2}^t, .., u_{iD}^t]$, where u_{ij}^t is the optimized parameter value of the *i*th individual with respect to the *j*th dimension ($j = 1,2,..., D$).

Target population: X^t is the set of NP individuals in the population at generation *t*, i.e., $X^t = [X_1^t, X_2^t, ..., X_{NP}^t]$.

Mutant population: V^t is the set of NP individuals in the population at generation *t*, i.e., $V^t = [V_1^t, V_2^t, ..., V_{NP}^t]$.

Trial population: U^t is the set of NP individuals in the population at generation *t*, i.e., $U^t = [U_1^t, U_2^t, ..., U_{NP}^t]$.

Mutant constant: $F \in (0,2)$ is a real constant which affects the differential variation between two individuals.

Crossover constant: $CR \in (0,1)$ is a crossover constant which affects the diversity of population for the next generation.

Fitness function: In a minimization problem, the objective function is the continuous function value denoted as $f(X_i^t)$.

Termination criterion: It is a condition that the search process will be terminated. It might be a maximum number of function evaluations or maximum CPU time to terminate the search.

3.2. Initial Population

A population of individuals is constructed randomly for the DE algorithm. The continuous parameter values are established randomly. The following formula is used to construct the initial continuous parameter values of the individual uniformly:

$$x_{ij}^{0} = x_{\min} + (x_{\max} - x_{\min}) * r_1$$

where $x_{\min}$ and $x_{\max}$ are search range of the continuous functions and r_1 is a uniform random number between 0 and 1. During the reproduction of the DE algorithm, it is possible to extend the search outside of the initial range of the search space. For this reason, parameter values violating the initial range are restricted to the feasible range as follows:

$$x_{ij}^{t} = x_{\min} + (x_{\max} - x_{\min}) * r_1$$

The population size is taken as 100. As the formulation of 14 functions suggests that the objective is to minimize the 14 continuous functions, the fitness value is the continuous function value of the individual X^t. That is, $f_i^t(X_i^t)$. For simplicity, $f_i^t(X_i^t)$ will be denoted as f_i^t.

3.3. Computational Procedure

The complete computational procedure of the DE algorithm can be summarized as follows:

Step 1: Initialization

- Set t=0, NP =100.
- Generate NP individuals randomly as explained before, $\{X_i^0, i = 1,2,..., NP\}$ where. $X_i^0 = [x_{i1}^0,.., x_{iD}^0]$

- Evaluate each individual *i* in the population using the objective function $f_i^0\left(X_i^0\right)$ for $i = 1,2,..., NP$.

Step 2: Update generation counter

- $t = t + 1$

Step 3: Generate mutant population

- For each target individual, X_i^t, $i = 1,2,..., NP$, at generation *t*, a mutant individual, $V_i^{t+1} = \left[v_{i1}^{t+1}, v_{i2}^{t+1},.., v_{iD}^{t+1}\right]$, is determined such that:

  $$V_i^{t+1} = X_{best}^t + F\left(X_{a_i}^t - X_{b_i}^t\right)$$

  where X_{best}^t is the best individual so far in the population and a_i, and b_i are two randomly chosen individuals from the population such that ($a_i \neq b_i$). $F > 0$ is a mutant factor which affects the differential variation between two individuals.

Step 4: Generate trial population

- Following the mutation phase, the crossover (recombination) operator is applied to obtain the trial population. For each mutant individual, $V_i^{t+1} = \left[v_{i1}^{t+1}, v_{i2}^{t+1},.., v_{iD}^{t+1}\right]$, an integer random number between 1 and *D*, i.e, $D_i \in \left(1,2,.., D\right)$, is chosen, and a trial individual, $U^{t+1} = \left[U_1^{t+1}, U_2^{t+1},..., U_{NP}^{t+1}\right]$ is generated such that:

  $$u_{ij}^{t+1} = \begin{cases} v_{ij}^{t+1}, if \quad r_{ij}^{t+1} \leq CR \quad or \quad j = D_i \\ x_{ij}^t, \qquad Otherwise \end{cases}$$

  where the index *D* refers to a randomly chosen dimension (*j=1,2,..,D*), which is used to ensure that at least one parameter of each trial individual U_i^{t+1} differs from its counterpart in the previous generation U_i^t, CR is a user-defined crossover constant in the range [0, 1], and r_{ij}^{t+1} is a uniform random number between 0 and 1. In other words, the trial individual is made up with some parameters of mutant individual, or at least one of the parameters randomly selected, and some other parameters of target individual.

Step 5: Evaluate trial population

- Evaluate the trial population using the objective function $f_i^{t+1}\left(U_i^{t+1}\right)$ for $i = 1,2,..., NP$.

Step 6: Selection

- To decide whether or not the trial individual U_i^{t+1} should be a member of the target population for the next generation, it is compared to its counterpart target individual X_i^t at the previous generation. The selection is based on the survival of fitness among the trial population and target population such that:

$$X_i^{t+1} = \begin{cases} U_i^{t+1}, if \quad f\left(U_i^{t+1}\right) \le f\left(X_i^t\right) \\ X_i^t, otherwise \end{cases}$$

Step 7: Stopping criterion

- If the number of function evaluations (FES) exceeds the maximum number of function evaluations, then stop; otherwise go to step 2.

The flowchart of the computational procedure is given in Figure 3.2.

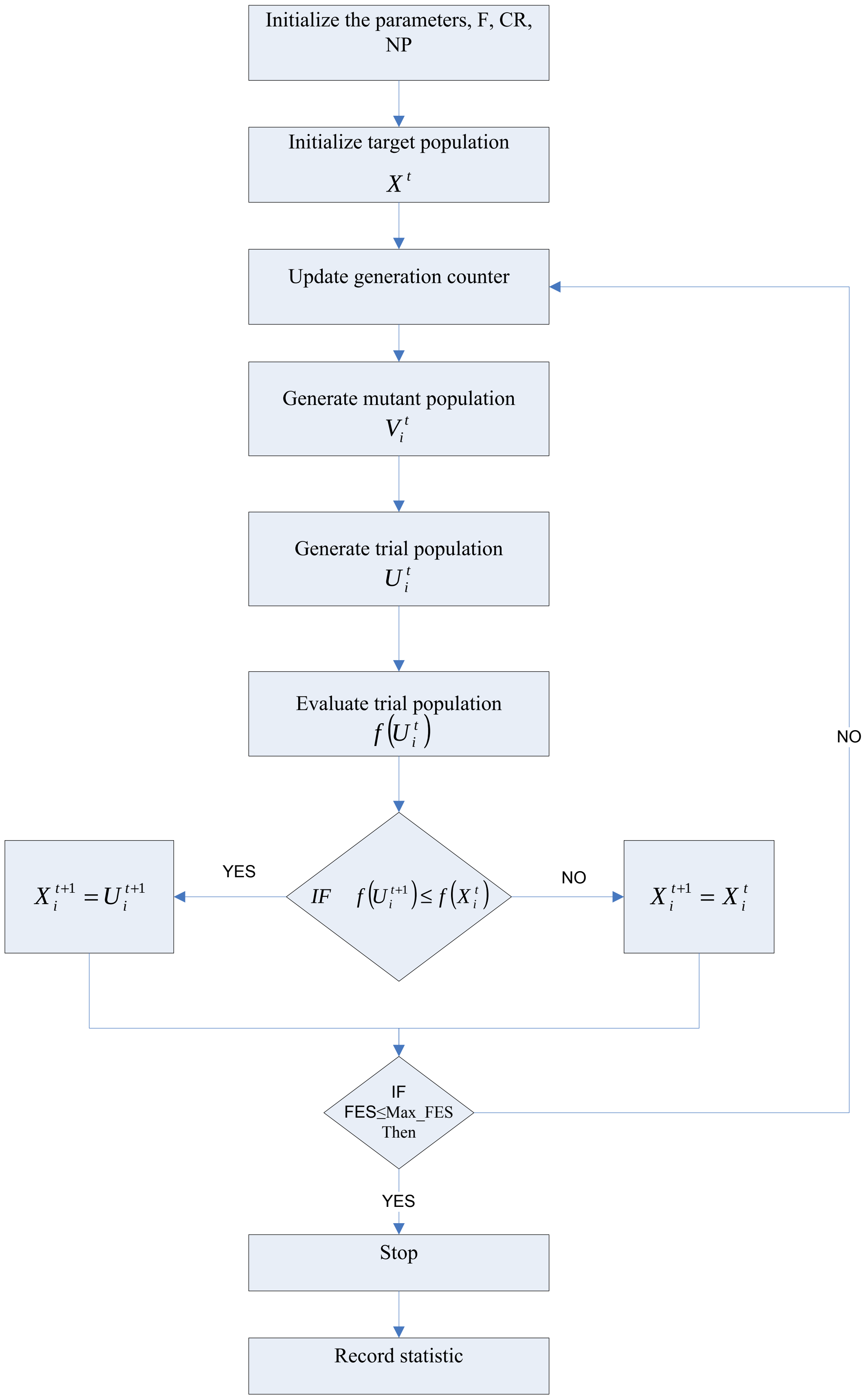
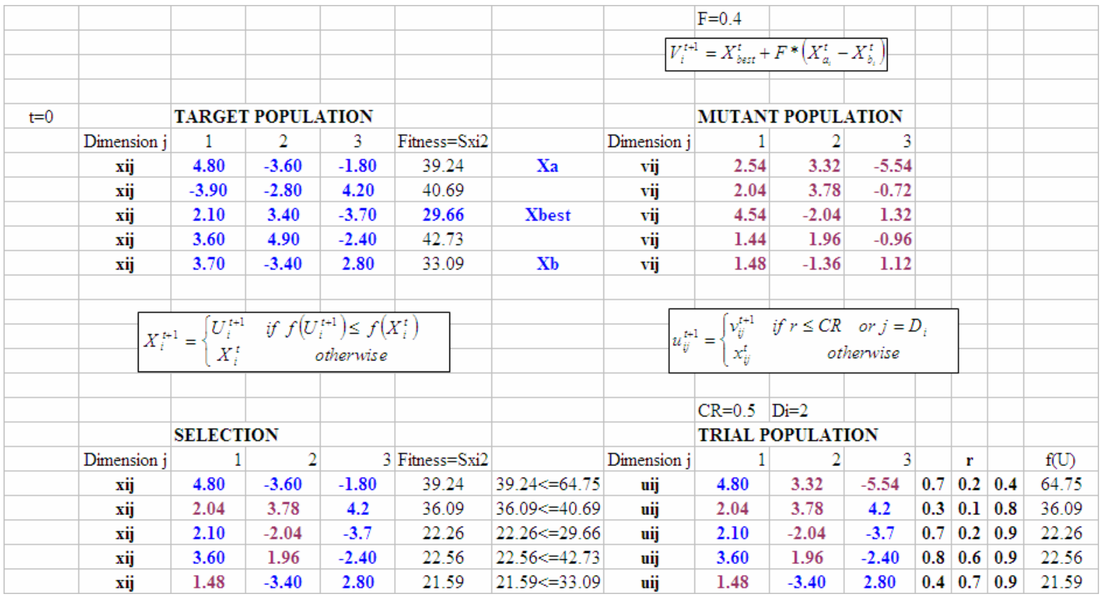

Initialize the parameters, F, CR, NP
Initialize target population X^t
Update generation counter
Generate mutant population V_i^t
Generate trial population U_i^t
Evaluate trial population $f(U_i^t)$
$X_i^{t+1} = U_i^{t+1}$
YES
$IF \quad f(U_i^{t+1}) \le f(X_i^t)$
NO
$X_i^{t+1} = X_i^t$
NO
IF FES≤Max_FES Then
YES
Stop
Record statistic

Figure 3.2 Flowchart of the DE algorithm.

3.4. An Example for DE Algorithm

In this section, an example of minimization of Sphere Function with 3 dimensions is given below:



F=0.4

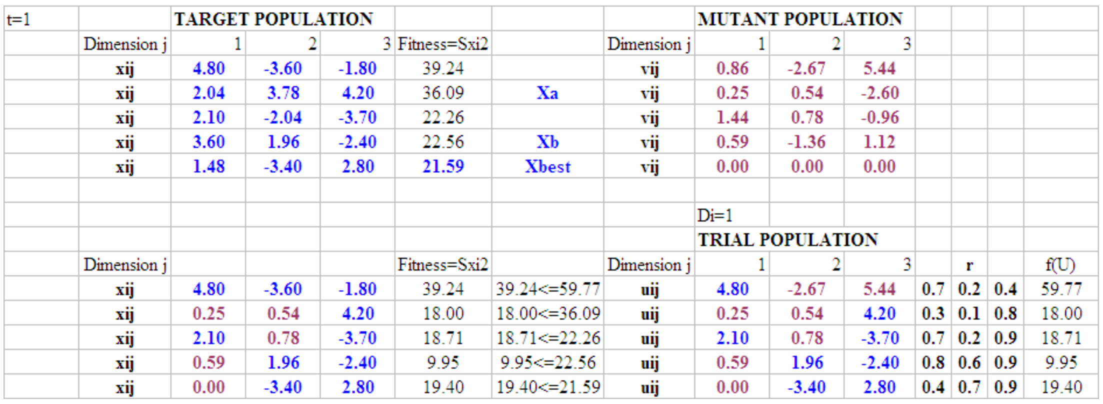
$$V_i^{t+1} = X_{best}^t + F * \left(X_{a_i}^t - X_{b_i}^t\right)$$

t=0		**TARGET POPULATION**						**MUTANT POPULATION**						
	Dimension j	1	2	3	Fitness=Sxi2		Dimension j	1	2	3				
	xij	**4.80**	**-3.60**	**-1.80**	39.24	**Xa**	**vij**	**2.54**	**3.32**	**-5.54**				
	xij	**-3.90**	**-2.80**	**4.20**	40.69		**vij**	**2.04**	**3.78**	**-0.72**				
	xij	**2.10**	**3.40**	**-3.70**	**29.66**	**Xbest**	**vij**	**4.54**	**-2.04**	**1.32**				
	xij	**3.60**	**4.90**	**-2.40**	42.73		**vij**	**1.44**	**1.96**	**-0.96**				
	xij	**3.70**	**-3.40**	**2.80**	33.09	**Xb**	**vij**	**1.48**	**-1.36**	**1.12**				

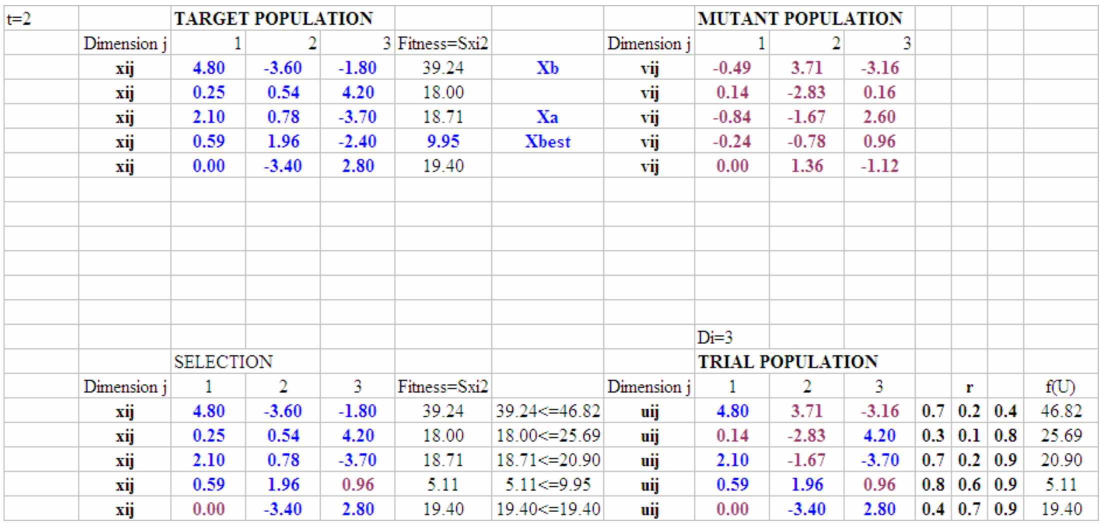
$$X_i^{t+1} = \begin{cases} U_i^{t+1} & \text{if } f\left(U_i^{t+1}\right) \le f\left(X_i^t\right) \\ X_i^t & \text{otherwise} \end{cases}$$

$$u_{ij}^{t+1} = \begin{cases} v_{ij}^{t+1} & \text{if } r \le CR \quad \text{or } j = D_i \\ x_{ij}^t & \text{otherwise} \end{cases}$$

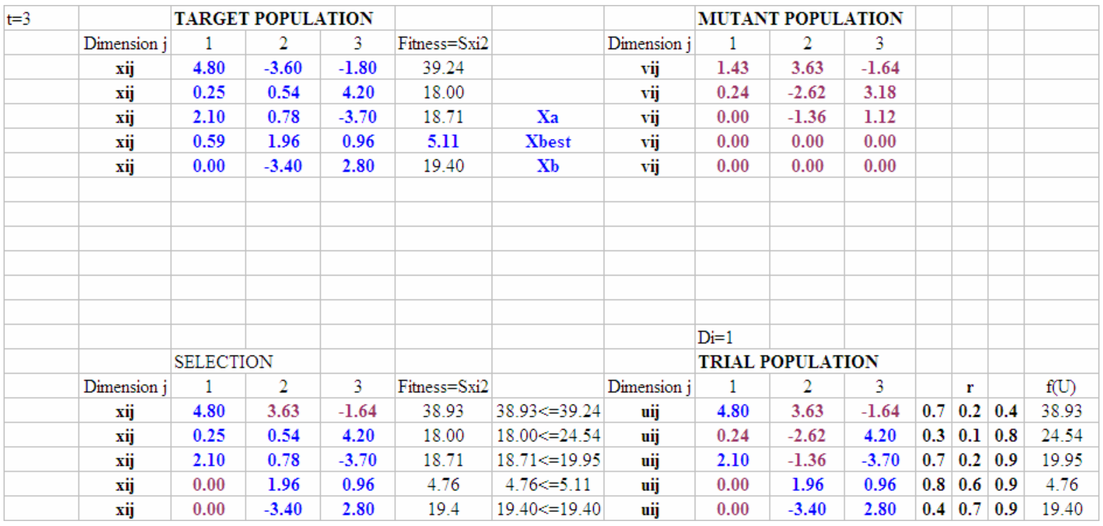
								CR=0.5	Di=2					
		SELECTION						**TRIAL POPULATION**						
	Dimension j	1	2	3	Fitness=Sxi2		Dimension j	1	2	3		**r**		f(U)
	xij	**4.80**	**-3.60**	**-1.80**	39.24	39.24<=64.75	**uij**	**4.80**	**3.32**	**-5.54**	**0.7**	**0.2**	**0.4**	64.75
	xij	**2.04**	**3.78**	**4.2**	36.09	36.09<=40.69	**uij**	**2.04**	**3.78**	**4.2**	**0.3**	**0.1**	**0.8**	36.09
	xij	**2.10**	**-2.04**	**-3.7**	22.26	22.26<=29.66	**uij**	**2.10**	**-2.04**	**-3.7**	**0.7**	**0.2**	**0.9**	22.26
	xij	**3.60**	**1.96**	**-2.40**	22.56	22.56<=42.73	**uij**	**3.60**	**1.96**	**-2.40**	**0.8**	**0.6**	**0.9**	22.56
	xij	**1.48**	**-3.40**	**2.80**	21.59	21.59<=33.09	**uij**	**1.48**	**-3.40**	**2.80**	**0.4**	**0.7**	**0.9**	21.59



t=1		**TARGET POPULATION**						**MUTANT POPULATION**						
	Dimension j	1	2	3	Fitness=Sxi2		Dimension j	1	2	3				
	xij	**4.80**	**-3.60**	**-1.80**	39.24		**vij**	**0.86**	**-2.67**	**5.44**				
	xij	**2.04**	**3.78**	**4.20**	36.09	**Xa**	**vij**	**0.25**	**0.54**	**-2.60**				
	xij	**2.10**	**-2.04**	**-3.70**	22.26		**vij**	**1.44**	**0.78**	**-0.96**				
	xij	**3.60**	**1.96**	**-2.40**	22.56	**Xb**	**vij**	**0.59**	**-1.36**	**1.12**				
	xij	**1.48**	**-3.40**	**2.80**	**21.59**	**Xbest**	**vij**	**0.00**	**0.00**	**0.00**				
								Di=1						
								TRIAL POPULATION						
	Dimension j				Fitness=Sxi2		Dimension j	1	2	3		**r**		f(U)
	xij	**4.80**	**-3.60**	**-1.80**	39.24	39.24<=59.77	**uij**	**4.80**	**-2.67**	**5.44**	**0.7**	**0.2**	**0.4**	59.77
	xij	**0.25**	**0.54**	**4.20**	18.00	18.00<=36.09	**uij**	**0.25**	**0.54**	**4.20**	**0.3**	**0.1**	**0.8**	18.00
	xij	**2.10**	**0.78**	**-3.70**	18.71	18.71<=22.26	**uij**	**2.10**	**0.78**	**-3.70**	**0.7**	**0.2**	**0.9**	18.71
	xij	**0.59**	**1.96**	**-2.40**	9.95	9.95<=22.56	**uij**	**0.59**	**1.96**	**-2.40**	**0.8**	**0.6**	**0.9**	9.95
	xij	**0.00**	**-3.40**	**2.80**	19.40	19.40<=21.59	**uij**	**0.00**	**-3.40**	**2.80**	**0.4**	**0.7**	**0.9**	19.40



t=2		**TARGET POPULATION**						**MUTANT POPULATION**						
	Dimension j	1	2	3	Fitness=Sxi2		Dimension j	1	2	3				
	xij	**4.80**	**-3.60**	**-1.80**	39.24	**Xb**	**vij**	**-0.49**	**3.71**	**-3.16**				
	xij	**0.25**	**0.54**	**4.20**	18.00		**vij**	**0.14**	**-2.83**	**0.16**				
	xij	**2.10**	**0.78**	**-3.70**	18.71	**Xa**	**vij**	**-0.84**	**-1.67**	**2.60**				
	xij	**0.59**	**1.96**	**-2.40**	**9.95**	**Xbest**	**vij**	**-0.24**	**-0.78**	**0.96**				
	xij	**0.00**	**-3.40**	**2.80**	19.40		**vij**	**0.00**	**1.36**	**-1.12**				
								Di=3						
		SELECTION						**TRIAL POPULATION**						
	Dimension j	1	2	3	Fitness=Sxi2		Dimension j	1	2	3		**r**		f(U)
	xij	**4.80**	**-3.60**	**-1.80**	39.24	39.24<=46.82	**uij**	**4.80**	**3.71**	**-3.16**	**0.7**	**0.2**	**0.4**	46.82
	xij	**0.25**	**0.54**	**4.20**	18.00	18.00<=25.69	**uij**	**0.14**	**-2.83**	**4.20**	**0.3**	**0.1**	**0.8**	25.69
	xij	**2.10**	**0.78**	**-3.70**	18.71	18.71<=20.90	**uij**	**2.10**	**-1.67**	**-3.70**	**0.7**	**0.2**	**0.9**	20.90
	xij	**0.59**	**1.96**	**0.96**	5.11	5.11<=9.95	**uij**	**0.59**	**1.96**	**0.96**	**0.8**	**0.6**	**0.9**	5.11
	xij	**0.00**	**-3.40**	**2.80**	19.40	19.40<=19.40	**uij**	**0.00**	**-3.40**	**2.80**	**0.4**	**0.7**	**0.9**	19.40



t=3		TARGET POPULATION						MUTANT POPULATION						
	Dimension j	1	2	3	Fitness=Sxi2		Dimension j	1	2	3				
	xij	4.80	-3.60	-1.80	39.24		vij	1.43	3.63	-1.64				
	xij	0.25	0.54	4.20	18.00		vij	0.24	-2.62	3.18				
	xij	2.10	0.78	-3.70	18.71	Xa	vij	0.00	-1.36	1.12				
	xij	0.59	1.96	0.96	5.11	Xbest	vij	0.00	0.00	0.00				
	xij	0.00	-3.40	2.80	19.40	Xb	vij	0.00	0.00	0.00				
								Di=1						
		SELECTION						TRIAL POPULATION						
	Dimension j	1	2	3	Fitness=Sxi2		Dimension j	1	2	3		r		f(U)
	xij	4.80	3.63	-1.64	38.93	38.93<=39.24	uij	4.80	3.63	-1.64	0.7	0.2	0.4	38.93
	xij	0.25	0.54	4.20	18.00	18.00<=24.54	uij	0.24	-2.62	4.20	0.3	0.1	0.8	24.54
	xij	2.10	0.78	-3.70	18.71	18.71<=19.95	uij	2.10	-1.36	-3.70	0.7	0.2	0.9	19.95
	xij	0.00	1.96	0.96	4.76	4.76<=5.11	uij	0.00	1.96	0.96	0.8	0.6	0.9	4.76
	xij	0.00	-3.40	2.80	19.4	19.40<=19.40	uij	0.00	-3.40	2.80	0.4	0.7	0.9	19.40

Figure 3.3 An example for DE algorithm.

CHAPTER 4: BENCHMARK SUITE

4.1. Introduction

In order to solve continuous function optimization problems, several optimization algorithms have been presented in the literature with their results based on a small subset of the standard test problems such as Sphere, Schwefel, Rosenbrock, Rastrigin, and so on. Often, confusing results limited to the test problems were reported in the literature in such a way that the same algorithm working for a set of functions may not work for some other functions. For these reasons, these algorithms should be evaluated more systematically by determining a common termination criterion, size of problems, initialization scheme, running time and so on. The special session on real-parameter optimization in CEC2005 aimed at developing new benchmark functions to be publicly available to the researchers for evaluating their algorithms. The problem definition files, codes and evaluation criteria are obtained from [99].

4.2. Properties of Benchmark Functions

Many real-world problems can be formulated as optimization which can be converted to the following form:

Minimize $f(x), x = [x_1, x_2, .., x_D]$ where $x \in [x_{\min}, x_{\max}]$

Many novel algorithms are introduced to solve the above global optimization problem. In order to compare and evaluate different algorithms, various benchmark functions with various properties have been proposed. Many of these popular benchmark functions possess some properties that have been exploited by some algorithms to achieve excellent results. According to Liang et. al. [100], some of these issues are:

1. Global optimum having the same parameter values for different variables/dimensions: Most of the popular benchmark functions have the same parameter values for different dimensions at the global optimum because of their symmetry. For example, Rastrigin's functions' and Griewank's functions' global optima are $[0,0,0,...,0]$ and Rosenbrock's functions' global optimum are $[1,1,1,...,1]$. In this situation, if there exist some operators to copy one dimension's value to the other dimensions, the global optimum may be found rapidly. For example, the neighborhood competition operator in [101] is defined as follows:

$$l = \left(m_1,...,m_{i_1-1},m_{i_1},m_{i_1-1},...,m_{i_2-1},m_{i_2},m_{i_2+1},...,m_D\right)$$
$$= \left(m_1,...,m_{i_1-1},m_{i_2},m_{i_2-1},...,m_{i_1+1},m_{i_1},m_{i_2+1},...,m_D\right)$$

where *m* is the best solution in the population and l is the new generated solution, i_1 and i_2 are two integer random numbers and $1 < i_1 < i_2 < D$, D is the dimension size of the problem. Hence, if the algorithm has found the globally optimal coordinates for some dimensions, they will be easily copied to the other dimensions. However, this operator might not be useful if the global optimum does not have the same value for many dimensions. In other words, if the global optimum is shifted to make the optimum to have different values for different dimensions, the performance of the MAGA algorithm in [101] significantly deteriorated. When we solve the real-world problems, global optimum is unlikely to have the same value for different dimensions.

2. Global optimum at the origin [101] : In this case, the global optimum o is equal to $[0,0,0,...,0]$. Zhong et. al. [101] proposed the following function $[l*(1-sRadius), l*(1+sRadius)]$, where l is the search center and $sRadius$ is the local search radius, to perform the local search. It can be observed that the local search range is much smaller when l is near the origin than when l is far from the origin. This operator is not effective if the global optimum is not at the origin. Hence,

this operator is specifically designed to exploit this common property of many benchmark functions.

3. Global optimum lying in the center of the search range: Some algorithms have the potential to converge to the center of the search range. The mean-centric crossover operator is just a good example for this type. When the initial population is randomly generated uniformly, the mean-centric method will have a trend to lead the population to the center of the search range.

4. Global optimum on the bounds: This situation is encountered in some multi-objective optimization algorithms as some algorithms set the dimensions moving out of the search range to the bounds [102]. If the global optimum is on the bounds, as in some multi-objective benchmark functions, the global optimum will be easily found. However, if there are some local optima near the bounds, it will be easy to fall into the local optima and fail to find the global optimum.

5. Local optima lying along the coordinate axes or no linkage among the variables/dimensions: Most of the benchmark functions, especially high dimensional functions, always have their symmetrical grid structure and local optima are always along the coordinate axes. In this case, the information of the local optima could be used to locate the global optimum. Further, for some functions it is possible to locate the global optimum by using just D one-dimensional searches for a D dimensional problem. Some co-evolutionary algorithms [103] and the one dimensional mutation operator[101, 104] just use these properties to locate the global optimum rapidly.

By analyzing these problems, Liang et al. [100] recommend that the researchers should use the following methods to avoid these problems when they use the benchmark functions suffering from these problems, to test a novel algorithm.

1. Shift the global optimum to a random position as shown below to make the global optimum to have different parameter values for different dimensions for benchmark functions suffering from problems 1 to 3: $F(x) = f(x - o_{new} + o_{old})$ where $F(x)$ is the new function, $f(x)$ is the old function, o_{old} is the old global optimum and o_{new} is

the new setting global optimum which has different values for different dimensions and not in the center of the search range.

2. For issue 4, considering there are real problems which have the global optimum on the bounds, it is an acceptable method for bounds handling to set the population to the near bounds when they are out of the search range. However, Liang et. al. [100] suggest using different kinds of benchmark functions to test the algorithms. For example, However, Liang et. al. [100] suggested that one can use some problems with the global optimum on bounds, not on bounds and some problems with local optima on bounds. One may not just test one algorithm that uses this bounds handling method, on benchmark functions with global optimum on bounds, and conclude the algorithm to be good.

3. Rotate the functions with issue 5 as below:
$F(x) = f(R * x)$ where R is an orthogonal rotation matrix obtained using Salmon's method [105]. In this way, local optima can be avoided lying along the coordinate axes and retain the benchmark functions' properties at the same time.

When a novel algorithm is tested, except for the global optimum's position need be shifted, functions having different properties should be included such as continuous functions, non-continuous functions, global optimum on the bounds, global optimum not on the bounds, unrotated functions, rotated functions, function with no clear structure in the fitness landscape, narrow global basin of attraction and so on.
The first five functions are unimodal functions whereas the remaining nine functions are multimodal where seven of them are basic functions and two of them are the expanded functions. These functions are summarized below:

- Unimodal Functions:
    - Shifted Sphere Function
    - Shifted Schwefel's Problem 1.2
    - Shifted Rotated High Conditioned Elliptic Function
    - Shifted Schwefel's Problem 1.2 with noise in Fitness
    - Schwefel's Problem 2.6 with Global Optimum on Bounds

- Multimodal Functions:
  - Basic Functions:
    - Shifted Rosenbrock’s Function
    - Shifted Rotated Griewank’s Function without Bounds
    - Shifted Rotated Ackley’s Function with Global Optimum on Bounds
    - Shifted Rastrigin’s Function
    - Shifted Rotated Rastrigin’s Function
    - Shifted Rotated Weierstrass Function
    - Schwefel’s Problem 2.13
  - Expanded Functions:
    - Expanded Extended Griewank’s plus Rosenbrock’s Function(F8F2)
    - Expanded Rotated Extended Scaffer’s F6

These test functions were designed to test an optimizer’s ability to find a global optimum under a variety of circumstances such as:

- Function landscape is highly conditioned
- Function landscape is rotated
- Optimum lies in a narrow basin
- Optimum lies on a bound
- Optimum lies beyond the initial bounds
- Function is not continuous everywhere
- Bias is added to the function evaluation

4.3. Benchmark Suite

Test functions employed in this study are given in detail below:

1. Shifted Sphere Function:

$$f(x) = \sum_{i=1}^{D} z_i^2 + f\_bias \qquad z = x - o \qquad x = [x_1, x_2, ..., x_D]$$

D: Dimension

$o = [o_1, o_2, ..., o_D]$: The shifted global optimum, to avoid the global optimum from the origin

Properties:

- Unimodal
- Shifted
- Separable
- Scalable
- $x \in [-100,100]^D$, global optimum: $x^* = o$, $f(x^*) = f\_bias(1) = -450$

Data Files:

Name	:	sphere_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum
Name	:	fbias_data.txt
Variable	:	f_bias 1*25 vector

2. Shifted Schwefel's Problem 1.2

$$f(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} z_j \right)^2 + f\_bias \quad z = x - o \qquad x = [x_1, x_2, ..., x_D]$$

D: Dimension

$o = [o_1, o_2, ..., o_D]$: The shifted global optimum

Properties:

- Unimodal
- Shifted
- Non-separable
- Scalable
- $x \in [-100,100]^D$, global optimum: $x^* = o$, $f(x^*) = f\_bias(2) = -450$

Data Files:

Name	:	schwefel_102_data.txt
Variable	:	o 1*100 vector the shifted global optimum

3. Shifted Rotated High Conditioned Elliptic Function

$$f(x)=\sum_{i=1}^{D}\left(10^6\right)^{\frac{i-1}{D-1}}z_i^2+f\_bias \quad z=(x-o)*M \quad x=[x_1,x_2,...,x_D]$$

D: Dimension

$o=[o_1,o_2,...,o_D]$: The shifted global optimum

M: orthogonal matrix

Properties:

- Unimodal
- Shifted
- Rotated
- Non-separable
- Scalable
- $x\in[-100,100]^D$, global optimum: $x^*=o$, $f(x^*)=f\_bias(3)=-450$

Data File:

Name	:	high_cond_elliptic_rot_data.txt
Variable	:	o 1*100 vector the shifted global optimum
Name	:	elliptic_M_D10.txt
Variable	:	M 10*10 matrix
Name	:	elliptic_M_D30.txt
Variable	:	M 30*30 matrix
Name	:	elliptic_M_D50.txt
Variable	:	M 50*50 matrix

4. Shifted Schwefel's Problem 1.2 with Noise in Fitness

$$f(x)=\left(\sum_{i=1}^{D}\left(\sum_{j=1}^{i}z_j\right)^2\right)*\left(1+0.4\left|N(0,1)\right|\right)+f\_bias \quad z=x-o$$

$$x=[x_1,x_2,...,x_D]$$

D: Dimension

$o=[o_1,o_2,...,o_D]$: The shifted global optimum

Properties:

- Unimodal
- Shifted

- Non-separable
- Scalable
- Noise in fitness
- $x \in [-100,100]^D$, global optimum: $x^* = o$, $f(x^*) = f\_bias(4) = -450$

Data File:

Name	:	schwefel_102_data.txt	
Variable	:	o	1*100 vector the shifted global optimum

5. Schwefel's Problem 2.6 with Global Optimum on Bounds

$f(x) = \max\{|x_1 + 2x_2 - 7|, |2x_1 + x_2 - 5|\}$, $i = 1,...,n$, $x^* = [1,3]$, $f(x^*) = 0$

Extend to D dimensions:

$f(x) = \max\{|A_i x - B_i|\} + f\_bias$, $i = 1,...,D$, $x^* = [1,3]$, $x = [x_1, x_2, ..., x_D]$

D: Dimension

A is a *D***D* matrix, a_{ij} are integer random numbers in the range $[-500,500]$, $\det(A) \neq 0$ A_i is the *i*th row of A.

$B_i = A_i * o$, o is a *D**1 vector, o_i are random number in the range $[-100,100]$

After loading the data file, set $o_i = -100$, for $i = 1,2,...,\lceil D/4 \rceil$, $o_i = 100$ for $i = \lfloor 3D/4 \rfloor,...,D$

Properties:

- Unimodal
- Non-separable
- Scalable
- If the initialization procedure initializes the population at the bounds, this problem will be solved easily.
- $x \in [-100,100]^D$, global optimum: $x^* = o$, $f(x^*) = f\_bias(5) = -310$

Data File:

Name	:	schwefel_206_data.txt	
Variable	:	o	1*100 vector the shifted global optimum
		A	100*100 matrix

In schwefel_206_data.txt, the first line is o (1*100 vector), and line2-line101 is A (100*100 matrix)

6. Shifted Rosenbrock's Function

$$f(x)=\sum_{i=1}^{D-1}\left(100\left(z_i^2-z_{i+1}\right)^2+\left(z_i-1\right)^2\right)+f\_bias \quad z=x-o+1$$

$$x=[x_1,x_2,...,x_D]$$

D: Dimension

$o=[o_1,o_2,...,o_D]$: The shifted global optimum

Properties:

- Multi-modal
- Shifted
- Non-separable
- Scalable
- Having a very narrow valley from local optimum to global optimum
- $x\in[-100,100]^D$, global optimum: $x^*=o$, $f(x^*)=f\_bias(6)=390$

Data File:

Name	:	rosenbrock_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum

7. Shifted Rotated Griewank's Function without Bounds

$$f(x)=\sum_{i=1}^{D}\frac{z_i^2}{4000}-\prod_{i=1}^{D}\cos\left(\frac{z_i}{\sqrt{i}}\right)+1+f\_bias,\ z=(x-o)*M, \qquad x=[x_1,x_2,...,x_D]$$

D: Dimension

$o=[o_1,o_2,...,o_D]$: The shifted global optimum

M' : linear transformation matrix, condition number =3

$M=M'(1+0.3|N(0,1)|)$

Properties:

- Multi-modal
- Rotated
- Shifted
- Non-separable
- Scalable
- No bounds for variables x

- Initialize population in $[0,600]^D$, Global optimum $x^* = o$ is outside of the initialization range, $f(x^*) = f\_bias(7) = -180$

Data File:

Name	:	griewank_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum
Name	:	griewank_M_D10.txt
Variable	:	M 10*10 matrix
Name	:	griewank_M_D30.txt
Variable	:	M 30*30 matrix
Name	:	griewank_M_D50.txt
Variable	:	M 50*50 matrix

8. Shifted Rotated Ackley's Function with Global Optimum on Bounds

$$f(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi z_i)\right) + e + f\_bias,$$

$z = (x - o) * M$, $\quad x = [x_1, x_2, ..., x_D]$

D: Dimension

$o = [o_1, o_2, ..., o_D]$: The shifted global optimum;

After loading the data file, set $o_{2j-1} = -32 o_{2j}$ are randomly distributed in the search range, for $j = 1,2,...,\lfloor D/2 \rfloor$

M : linear transformation matrix, condition number =100

Properties:

- Multi-modal
- Rotated
- Shifted
- Non-separable
- Scalable
- A's condition number Cond(A) increases with the number of variables as $O(D^2)$
- Global optimum on the bound
- If the initialization procedure initializes the population at the bounds, this problem will be solved easily.

- $x \in [-32,32]^D$, global optimum: $x^* = o$, $f(x^*) = f\_bias(8) = -140$

Data File:

Name	:	ackley_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum
Name	:	ackley_M_D10.txt
Variable	:	M 10*10 matrix
Name	:	ackley_M_D30.txt
Variable	:	M 30*30 matrix
Name	:	ackley_M_D50.txt
Variable	:	M 50*50 matrix

9. Shifted Rastrigin's Function

$$f(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right) + f\_bias, \quad z = x - o \qquad x = [x_1, x_2, ..., x_D]$$

D: Dimension

$o = [o_1, o_2, ..., o_D]$: The shifted global optimum

Properties:

- Multi-modal
- Shifted
- Separable
- Scalable
- Local optima's number is huge
- $x \in [-5,5]^D$, global optimum: $x^* = o$, $f(x^*) = f\_bias(9) = -330$

Data File:

Name	:	rastrigin_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum

10. Shifted Rotated Rastrigin's Function

$$f(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right) + f\_bias, \ z = (x - o) * M, \qquad x = [x_1, x_2, ..., x_D]$$

D: Dimension

$o = [o_1, o_2, ..., o_D]$: The shifted global optimum

M : linear transformation matrix, condition number =2

Properties:

- Multi-modal
- Shifted
- Rotated
- Non-separable
- Scalable
- Local optima's number is huge
- $x \in [-5,5]^D$, global optimum: $x^* = o$, $f(x^*) = f\_bias(10) = -330$

Data File:

Name	:	rastrigin_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum
Name	:	rastrigin_M_D10.txt
Variable	:	M 10*10 matrix
Name	:	rastrigin_M_D30.txt
Variable	:	M 30*30 matrix
Name	:	rastrigin_M_D50.txt
Variable	:	M 50*50 matrix

11. Shifted Rotated Weierstrass Function

$$f(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k_{\max}}\left[a^k \cos\left(2\pi b^k (z_i + 0.5)\right)\right]\right) - D\sum_{k=0}^{k_{\max}}\left[a^k \cos\left(2\pi b^k \cdot 0.5\right)\right] + f\_bias, \quad a = 0.5,$$

$b = 3$, $k_{\max} = 20$,

$z = (x - o)*M$, $x = [x_1, x_2, ..., x_D]$

D: Dimension

$o = [o_1, o_2, ..., o_D]$: The shifted global optimum

M : linear transformation matrix, condition number =5

Properties:

- Multi-modal
- Shifted
- Rotated
- Non-separable
- Scalable
- Continuous but differentiable only on a set of points

- $x \in [-0.5, 0.5]^D$, global optimum: $x^* = o$, $f(x) = f\_bias(11) = 90$

Data File:

Name	:	weierstrass_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum
Name	:	weierstrass_M_D10.txt
Variable	:	M 10*10 matrix
Name	:	weierstrass_M_D30.txt
Variable	:	M 30*30 matrix
Name	:	weierstrass_M_D50.txt
Variable	:	M 50*50 matrix

12. Schwefel's Problem 2.13

$$f(x) = \sum_{i=1}^{D} (A_i - B_i(x))^2 + f\_bias, \ x = [x_1, x_2, ..., x_D]$$

$$A_i = \sum_{j=1}^{D} \left(a_{ij} \sin \alpha_j + b_{ij} \cos \alpha_j \right), \ B_i(x) = \sum_{j=1}^{D} \left(a_{ij} \sin x_j + b_{ij} \cos x_j \right), \text{ for } i = 1, ..., D$$

D: Dimension

A, B are two *D***D* matrix, a_{ij}, b_{ij} are integer random numbers in the range $[-100, 100]$,

$\alpha = [\alpha_1, \alpha_2, ..., \alpha_D]$, α_j are random numbers in the range $[-\pi, \pi]$.

Properties:

- Multi-modal
- Shifted
- Non-separable
- Scalable
- $x \in [-\pi, \pi]^D$, global optimum: $x^* = \alpha$, $f(x^*) = f\_bias(12) = -460$

Data File:

Name	:	schwefel_213_data.txt
Variable	:	alpha 1*100 vector the shifted global optimum
		a 100*100 matrix
		b 100*100 matrix

In schwefel_213_data.txt, line1-line100 is a (100*100 matrix), and line101-line200 is b (100*100 matrix), the last line is alpha (α) (1*100 vector)

Expanded Functions:

Use a two dimensional function $F(x, y)$ as a starting function.

The corresponding expanded function

$$EF(x_1, x_2, ..., x_D) = F(x_1, x_2) + F(x_2, x_3) + ... + F(x_{D-1}, x_D) + F(x_D, x_1)$$

13. Shifted Expanded Griewank's plus Rosenbrock's Function (F8F2)

F8: Griewank's Function: $F8(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

F2: Rosenbrock's Function: $F2(x) = \sum_{i=1}^{D-1} \left(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2\right)$

$$F8F2(x_1, x_2, ..., x_D) = F8(F2(x_1, x_2)) + F8(F2(x_2, x_3)) + ... + F8(F2(x_{D-1}, x_D)) + F8(F2(x_D, x_1))$$

Shift to

$$f(x) = F8(F2(z_1, z_2)) + F8(F2(z_2, z_3)) + ... + F8(F2(z_{D-1}, z_D)) + F8(F2(z_D, z_1)) + f\_bias$$

$z = x - o + 1$, $x = [x_1, x_2, ..., x_D]$

D: Dimension

$o = [o_1, o_2, ..., o_D]$: The shifted global optimum

Properties:

- Multi-modal
- Shifted
- Non-separable
- Scalable
- $x \in [-5,5]^D$, global optimum: $x^* = o$, $f(x^*) = f\_bias(13) = -130$

Data File:

Name	:	EF8F2_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum

14. Shifted Rotated Expanded Scaffer's F6 Function

$$F(x,y) = 0.5 + \frac{\left(\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5\right)}{\left(1+0.001\left(x^2+y^2\right)\right)^2}$$

Expanded to

$f(x) = EF(z_1, z_2, ..., z_D) = F(z_1, z_2) + F(z_2, z_3) + ... + F(z_{D-1}, z_D) + F(z_D, z_1) + f\_bias$, $z = (x-o)*M$, $x = [x_1, x_2, ..., x_D]$

D: Dimension

$o = [o_1, o_2, ..., o_D]$: The shifted global optimum

M : linear transformation matrix, condition number =3

Properties:

- Multi-modal
- Shifted
- Non-separable
- Scalable
- $x \in [-100,100]^D$, Global optimum $x^* = o$, $f(x^*) = f\_bias(14) = -300$

Data File:

Name	:	E_ScafferF6_func_data.txt
Variable	:	o 1*100 vector the shifted global optimum
Name	:	E_ScafferF6_M_D10.txt
Variable	:	M 10*10 matrix
Name	:	E_ScafferF6_M_D30.txt
Variable	:	M 30*30 matrix
Name	:	E_ScafferF6_M_D50.txt
Variable	:	M 50*50 matrix

4.4. Evaluation Criteria

14 newly designed functions as indicated before are given in Suganthan et. al.[99] for different level of dimensions ranging from 10 to 50. The population size was 100 for all functions. For the evaluation purposes, Suganthan et. al. [99] provided several criterion measures explained below in order to make a fair comparison of different competing algorithms:

Problems: 14 minimization problems

Dimensions: D=10, 30, 50

Number of replications: 25

Maximum number of function evaluations (Max_FES): Max_FES is set to 10000*D where it is increased with the dimension size, i.e., Max_FES_10D= 100000; for 30D= 300000; for 50D= 500000.

Initialization: Uniform random initialization within the search range is used except for the problem 7 for which initialization ranges are specified.

Global Optimum: All problems, except for 7, have the global optimum within the given bounds and there is no need to perform search outside of the given bounds for these problems. Problem 7 is exception without any search bounds and with the global optimum outside of the specified initialization range.

Termination: Search is terminated before reaching Max_FES.

4.5. Performance Criteria

Following performance measures are used consistent with Suganthan et. al.[99].

- Record the error value $\left(f(x) - f(x^*)\right)$ after 1e3, 1e4, 1e5 FES and at termination for each run.
- For each function, sort the error values in 25 runs from the smallest (best) to the largest (worst)
- Present the following: 1st (best/smallest), 7th, 13th (median), 19th, 25th (worst/largest) values, mean and standard deviation for the 25 runs
- Record the success rate needed in each run to achieve the fixed accuracy level. Fixed accuracy level for each function is given in Table 4.1.

Table 4.1. : Fixed Accuracy Level for Each Function

F	1	2	3	4	5	6	7
AL	-450+1e-6	-450+1e-6	-450+1e-6	-450+1e-6	-310+1e-6	390+1e-2	-180+1e-2

F	8	9	10	11	12	13	14
AL	-140+1e-2	-330+1e-2	-330+1e-2	90+1e-2	-460+1e-2	-130+1e-2	-300+1e-2

A successful run is defined as a run during which the algorithm achieves the fixed accuracy level within the Max_FES for the particular problem-dimension.

4.6. Success Rate for Each Problem

Success Rate= (number of successful runs according to the table above) / total runs
Success rate is computed for each problem-dimension size separately.

4.7. Convergence Graphs

Convergence Graphs for each problem size are given. The semi-log graphs will show $\log\left(f(x) - f(x^*)\right)$ vs FES for each dimension size.

4.8. Algorithm Complexity

Algorithm complexity is computed as follows:

a. Run the test program below:
```
for i= 1: 1000000
x= (double) 5.55;
x= x+x; x= x./2; x= sqrt(x); x= ln(x); x= exp(x); y= x/x;
end
```
Computing time for the above= T0;

b. Evaluate the computing time just for Function 3. For 200000 evaluations of a certain dimension D, it gives T1;

c. The complete computing time for the algorithm with 200000 evaluations of the same D dimensional benchmark function 3 is T2. Execute step c 5 times and get T2 values. $\hat{T}2 = Mean(T2)$

The complexity of the algorithm is reflected by: $\hat{T}2$, T1,T0, and $\left(\hat{T}2 - T1\right)/T0$

The algorithm complexities are calculated on 10, 30 and 50 dimensions, to show the algorithm complexity’s relationship with dimension. In addition, it provides sufficient details on the computing system and the programming language used. In step c, we execute the complete algorithm 5 times to accommodate variations in execution time due to adaptive nature of some algorithms.

CHAPTER 5: COMPUTATIONAL RESULTS

5.1 Computational Results for the Particle Swarm Optimization Algorithm

In order to solve continuous function optimization problems, several optimization algorithms have been presented in the literature with their results based on a small subset of the standard test problems such as Sphere, Schwefel, Rosenbrock, Rastrigin, and so on. Often, confusing results limited to the test problems were reported in the literature in such a way that the same algorithm working for a set of functions may not work for some other functions. For these reasons, these algorithms should be evaluated more systematically by determining a common termination criterion, size of problems, initialization scheme, running time. The special session on real-parameter optimization in Congress on Evolutionary Computation (CEC2005) aimed at developing new benchmark functions to be publicly available to the researchers for evaluating their algorithms. The problem definition files, codes and evaluation criteria are made available in http://www.ntu.edu.sg/home/EPNSugan [99].

The traditional PSO algorithm was coded in C and run on an Intel P4 1.33 GHz PC with 256MB memory. Regarding the PSO parameters, the acceleration coefficients were taken as $c_1 = c_2 = 2$ consistent with the literature. Initial inertia weight and final inertia weight were set to $w_0 = 0.9$ and $w_n = 0.4$ respectively. The inertia weight is linearly decreased by the following equation:

$$w^t = ((\max\_FES - FES)/\max\_FES) * (w_0 - w_n) + w_n$$

The population size was taken as 100. The maximum number of function evaluations is fixed at 10000**D* where *D* is the size of dimension varying from 10 to

50. The PSO algorithm was run for the 14 benchmark functions recently developed. The performance evaluation of the PSO algorithm is also conducted through the guidelines described in the evaluation criteria in the webpage above. 25 replications are conducted for each benchmark function to record the error values, $f(x) - f(x^*)$, after 1e3 FES, 1e4 FES, 1e5 FES and at the termination.

The mean error values and standard deviations are given in Table 5.1. In addition, the error values achieved at different FES levels are given in details in Appendices A.1, A.2, and A.3. The Appendices list the test function, 1st (best/smallest), 7th, 13th (median), 19th, 25th (worst/largest), mean and standard deviation of the error values found at 1e3 FES, 1e4 FES, 1e5 FES and at termination for the 25 runs. The convergence graphs are also given in Figures 5.1 to 5.6.

Since these benchmarks are newly designed, unfortunately there exist no results for comparison purposes. For this reason, the computational results are presented along with the convergence graphs to be compared with the DE algorithm in the next Chapter.

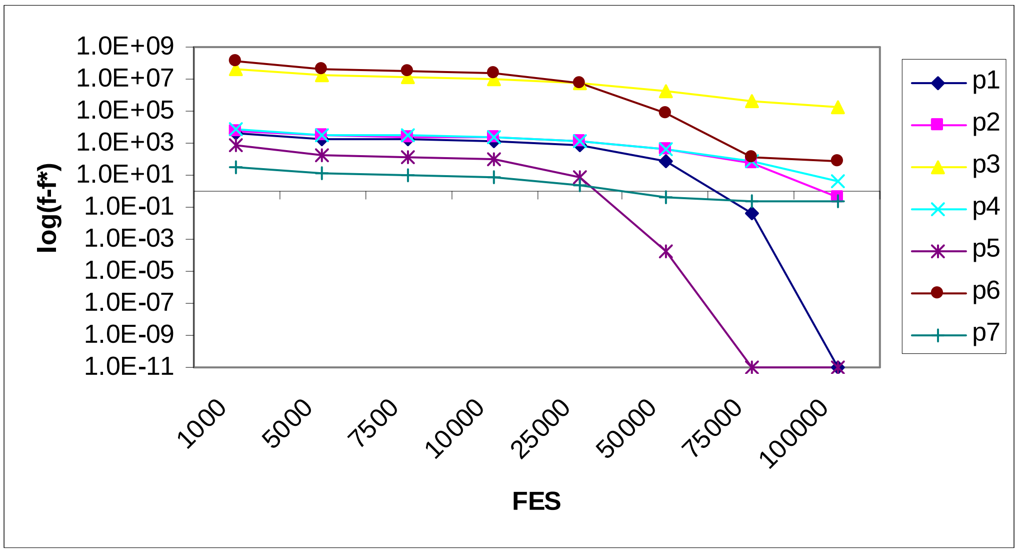


Figure 5.1 Convergence Graph of PSO for D=10 for functions 1-7

As seen in Figure 5.1, the PSO algorithm converges to the optimal solution easily in each run whereas near-optimal solutions are obtained for the functions 2, 4, and 7. In addition, the PSO algorithm was not able to find reasonable near-optimal solutions for the functions 3 and 6. In other words, the PSO algorithm failed for the functions 3 and 6.

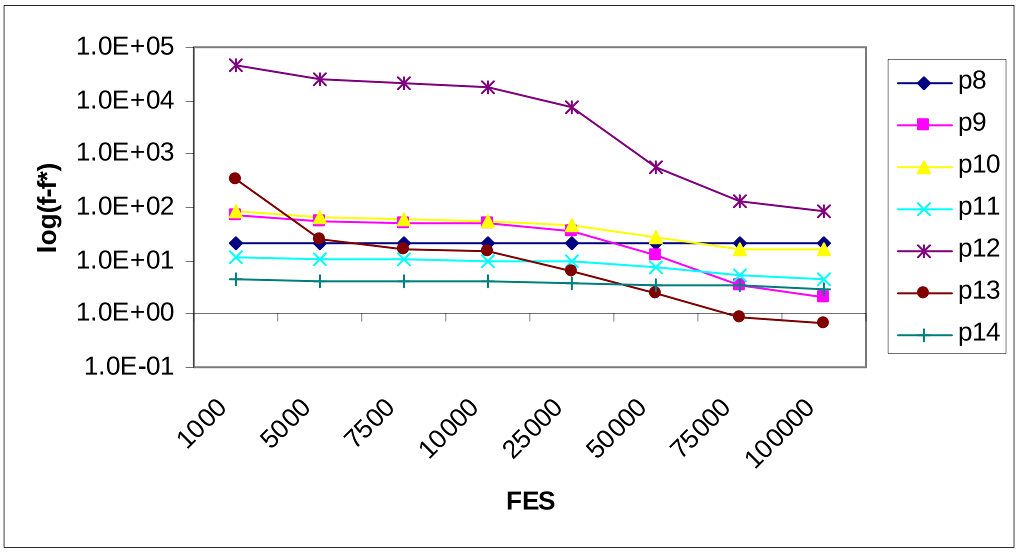


Figure 5.2 Convergence Graph of PSO for D=10 for functions 8-14

As seen in Figure 5.2, the PSO algorithm performed relatively good by generating near-optimal solutions for the functions 9, 11, 13, and 14 whereas it fails for the functions 8, 10, and 12.

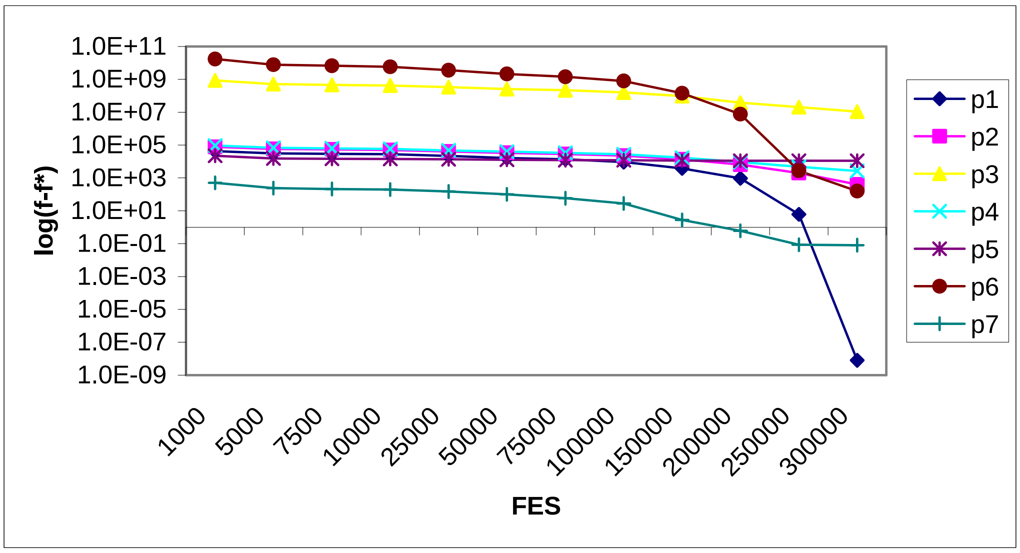


Figure 5.3 Convergence Graph of PSO for D=30 for functions 1-7

As seen in Figure 5.3, the PSO algorithm performed relatively good results by generating near-optimal solutions for only the functions 1 and 7 whereas it fails for the rest of the 5 other functions. Another interpretation of these results is that the PSO algorithm could not be able to generate near-optimal solutions when the dimension size is increased.

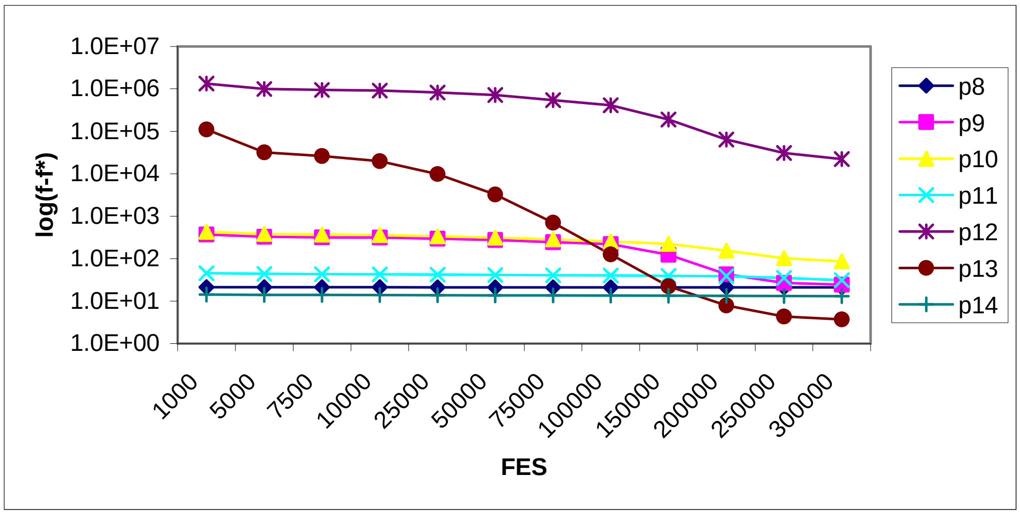


Figure 5.4 Convergence Graph of PSO for D=30 for functions 8-14

As seen in Figure 5.4, the PSO algorithm could not be able to generate satisfactory solutions for the eight functions except for the function 13. Another interpretation of these results is again that the PSO algorithm could not be able to generate near-optimal solutions when the dimension size is increased.

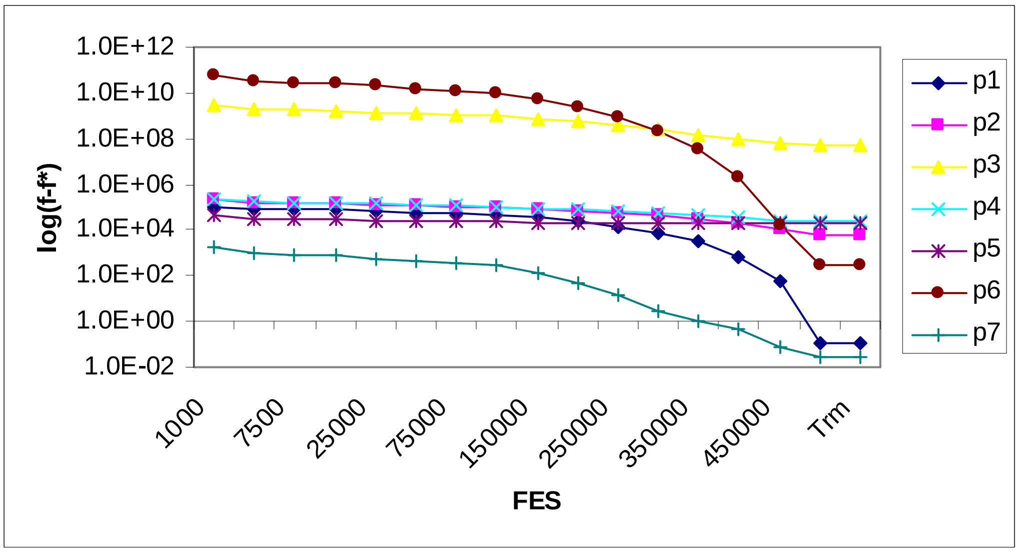


Figure 5.5 Convergence Graph of PSO for D=50 for functions 1-7

As seen in Figure 5.5, the PSO algorithm performed satisfactory results only for the functions 1 and 7 whereas it fails for the rest of the 5 other functions. Another interpretation of these results is again that the PSO algorithm could not be able to generate near-optimal solutions when the dimension size is increased.

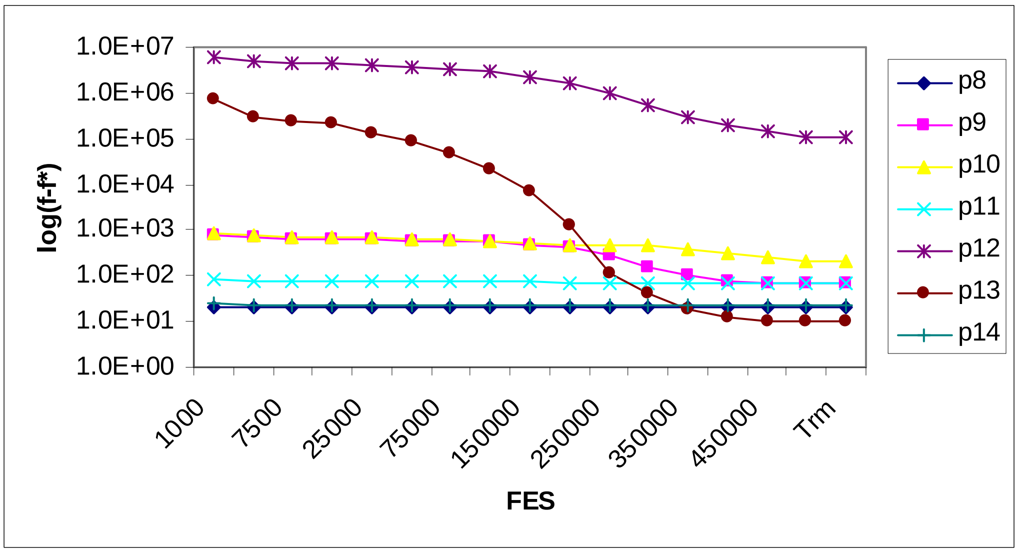


Figure 5.6 Convergence Graph of PSO for D=50 for functions 8-14

As seen in Figure 5.6, the PSO algorithm could not be able to generate satisfactory solutions for the eight functions except for the function 13. Another interpretation of these results is again that the PSO algorithm could not be able to generate near-optimal solutions when the dimension size is increased.

From these results, we can conclude that the PSO algorithm is affected by the dimension size. In other words, the performance of the PSO algorithm gets worsened as the dimension size increases.

5.2 Computational Results for the Differential Evolution Algorithm

The traditional DE algorithm was coded in C and run on an Intel P4 1.33 GHz PC with 256MB memory. Regarding the DE parameters, mutation (MR) and crossover rates (CR) are taken as 0.9 respectively. The population size was 100. The maximum number of function evaluations is fixed at 10000**D* where *D* is the size of dimension and varied from 10 to 50. The DE algorithm was run for the 14 benchmark functions recently developed. The performance evaluation of the DE algorithm is also conducted through the guidelines described in [99]. 25 replications are conducted for each benchmark function to record the error values, $f(x) - f(x^*)$, after 1e3 FES, 1e4 FES, 1e5 FES and at the termination.

Table 5.1. : Mean Error and standard deviation values achieved at the termination for PSO Algorithm

Func.		D=10	D=30	D=50
1	**Mean**	**0.00000E+00**	**8.00000E-09**	**1.16368E-01**
	Std D.	0.00000E+00	3.79693E-08	5.62614E-01
2	**Mean**	**4.31888E-01**	**4.01956E+02**	**6.53038E+03**
	Std D.	1.96785E+00	3.40265E+02	3.35448E+03
3	**Mean**	**1.98704E+05**	**1.07647E+07**	**4.84217E+07**
	Std D.	2.12806E+05	9.33441E+06	3.11780E+07
4	**Mean**	**4.25415E+00**	**2.70271E+03**	**2.27204E+04**
	Std D.	1.98131E+01	1.48463E+03	7.01268E+03
5	**Mean**	**0.00000E+00**	**1.10620E+04**	**1.98764E+04**
	Std D.	0.00000E+00	3.83930E+03	3.63886E+03
6	**Mean**	**6.92385E+01**	**1.57711E+02**	**3.03737E+02**
	Std D.	9.08055E+01	2.13877E+02	3.42479E+02
7	**Mean**	**2.41916E-01**	**8.19353E-02**	**2.47858E-02**
	Std D.	1.29032E-01	8.93249E-02	2.36381E-02
8	**Mean**	**2.03441E+01**	**2.09311E+01**	**2.11326E+01**
	Std D.	8.13829E-02	6.64438E-02	3.99925E-02
9	**Mean**	**1.99013E+00**	**2.43572E+01**	**6.62258E+01**
	Std D.	1.21875E+00	5.51904E+00	1.11524E+01
10	**Mean**	**1.64690E+01**	**8.72283E+01**	**2.14250E+02**
	Std D.	7.04173E+00	3.91043E+01	9.29100E+01
11	**Mean**	**4.62658E+00**	**3.11684E+01**	**6.61378E+01**
	Std D.	1.45265E+00	5.19378E+00	5.93583E+00
12	**Mean**	**8.47897E+01**	**2.21836E+04**	**1.08151E+05**
	Std D.	1.65819E+02	1.60610E+04	5.76638E+04
13	**Mean**	**6.62293E-01**	**3.70427E+00**	**9.64705E+00**
	Std D.	2.06914E-01	9.19095E-01	2.69821E+00
14	**Mean**	**2.96542E+00**	**1.30484E+01**	**2.27563E+01**
	Std D.	5.20650E-01	2.44407E-01	2.80591E-01

The mean error values and standard deviations are given in Table 5.2. In addition, the error values achieved at different FES levels are given in details in Appendices B.1, B.2, and B.3. The Appendices presents the test functions, 1st (best/smallest), 7th, 13th (median), 19th, 25th (worst/largest) values, mean and standard deviation of the error values found at 1e3 FES, 1e4 FES, 1e5 FES and at termination. The complexity of the DE algorithm will be given in detail in the next Chapter. The convergence graphs are also given in Figures 5.7 to 5.12.

Since these benchmarks are newly designed, unfortunately there exist no results for comparison purposes. For this reason, the computational results are presented along with the convergence graphs to be compared with the PSO algorithm in the next Chapter.

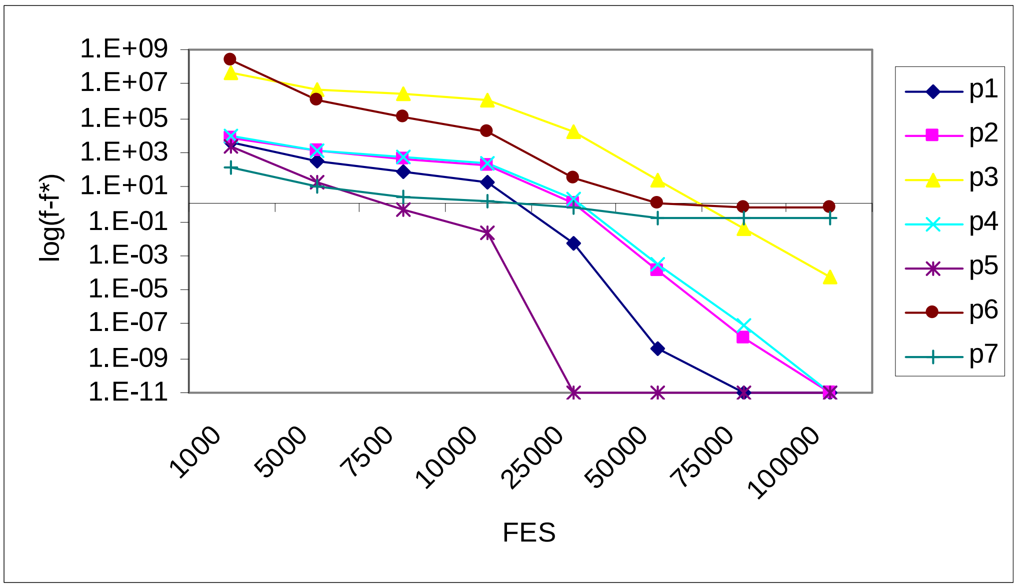


Figure 5.7 Convergence Graph of DE for D=10 for functions 1-7

As seen in Figure 5.7, the DE algorithm performed very good by generating optimal solutions for the functions 1, 2, 4, and 5. In addition, it generated results near to the optimal solutions for functions 3, 6 and 7.

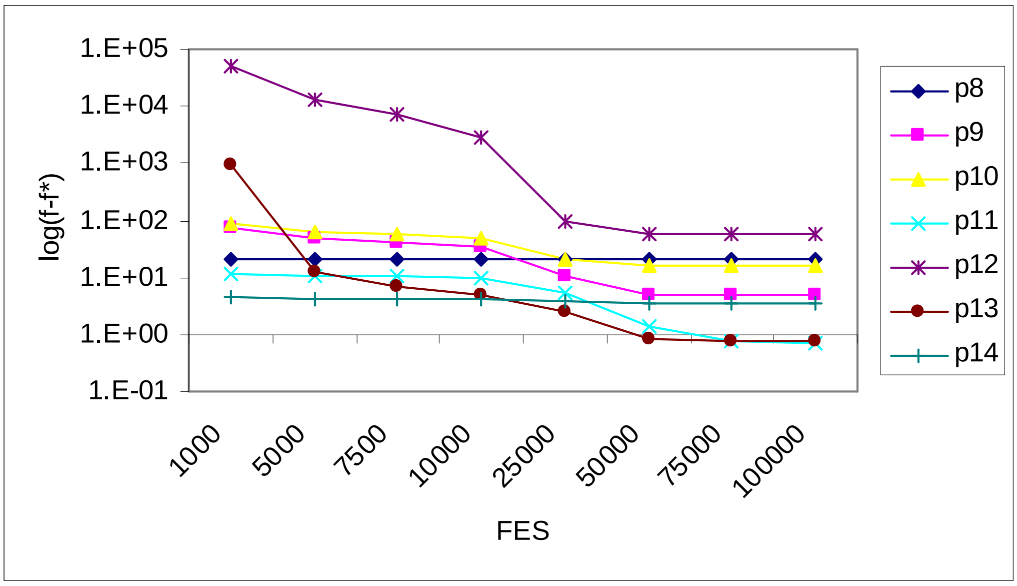


Figure 5.8 Convergence Graph of DE for D=10 for functions 8-14

As seen in Figure 5.8, the DE algorithm performed relatively good results by generating near-optimal solutions for the functions 11, 13 and 14 whereas it fails for the rest of the 4 other functions.

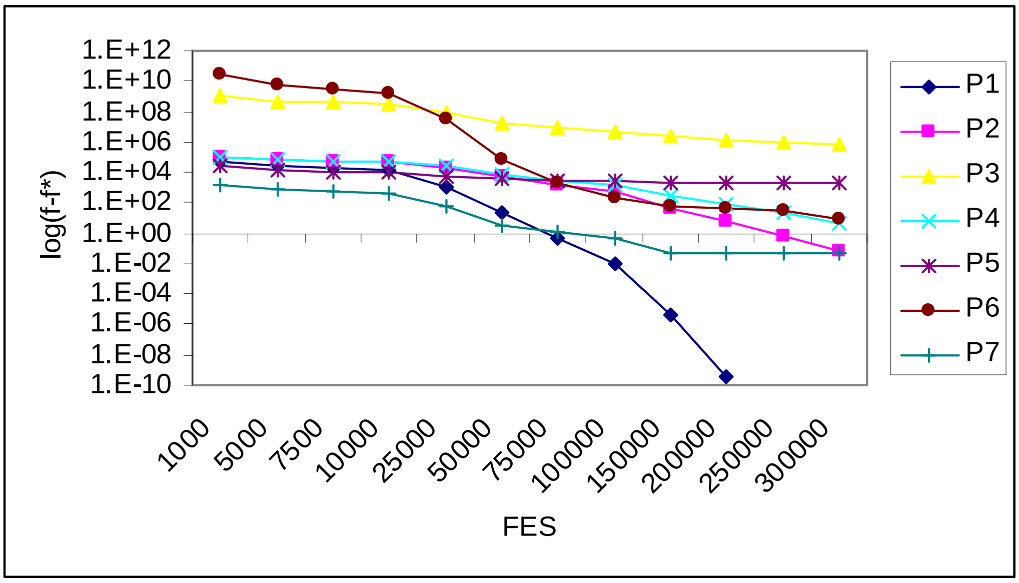


Figure 5.9 Convergence Graph of DE for D=30 for functions 1-7

As seen in Figure 5.9, the DE algorithm generated the optimal solution for the first function. It also generated near-optimal results for the functions 2 and 7. However, it fails for the functions 3, 4, 5 and 6.

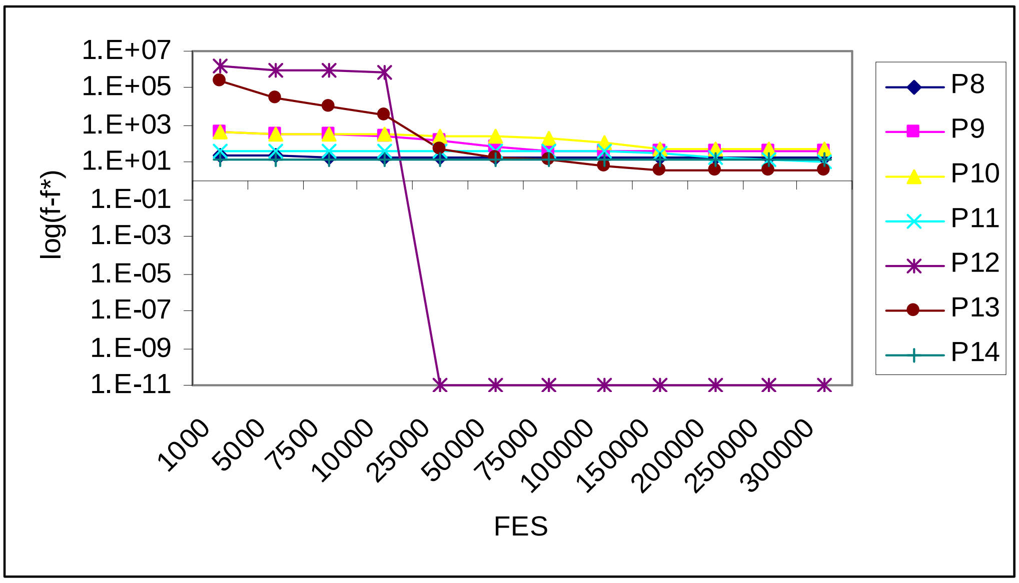


Figure 5.10 Convergence Graph of DE for D=30 for functions 8-14

As seen in Figure 5.10, the DE algorithm generated the optimal solution for the function 12. It also generated near-optimal result for the functions 13. However, it fails for the functions 8, 9, 10, 11, and 14.

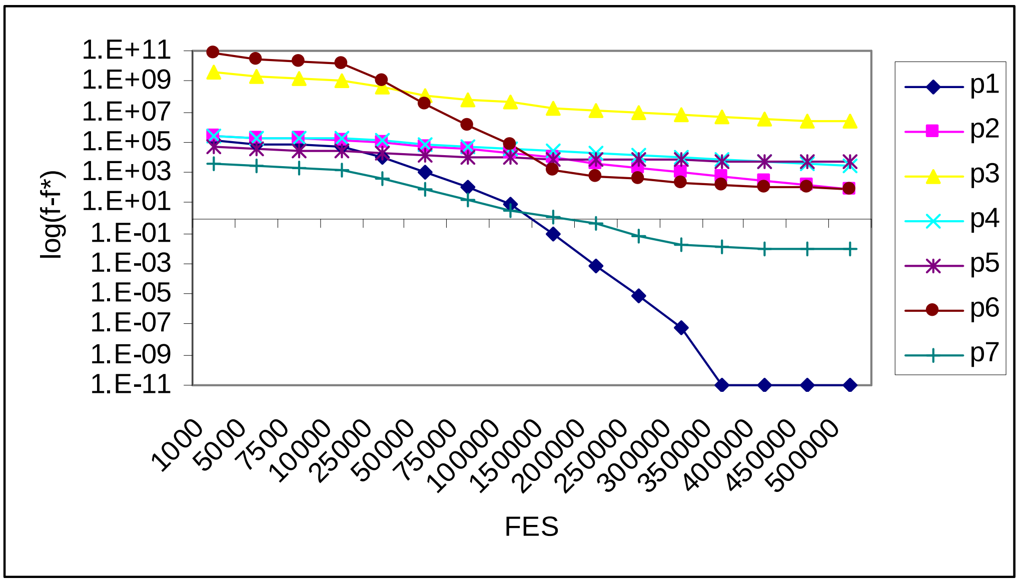


Figure 5.11 Convergence Graph of DE for D=50 for functions 1-7

As seen in Figure 5.11, the DE algorithm generated the optimal solution for the first function 1. It also generated near-optimal result for the function 7. However, it fails for the functions 2, 3, 4, 5 and 6.

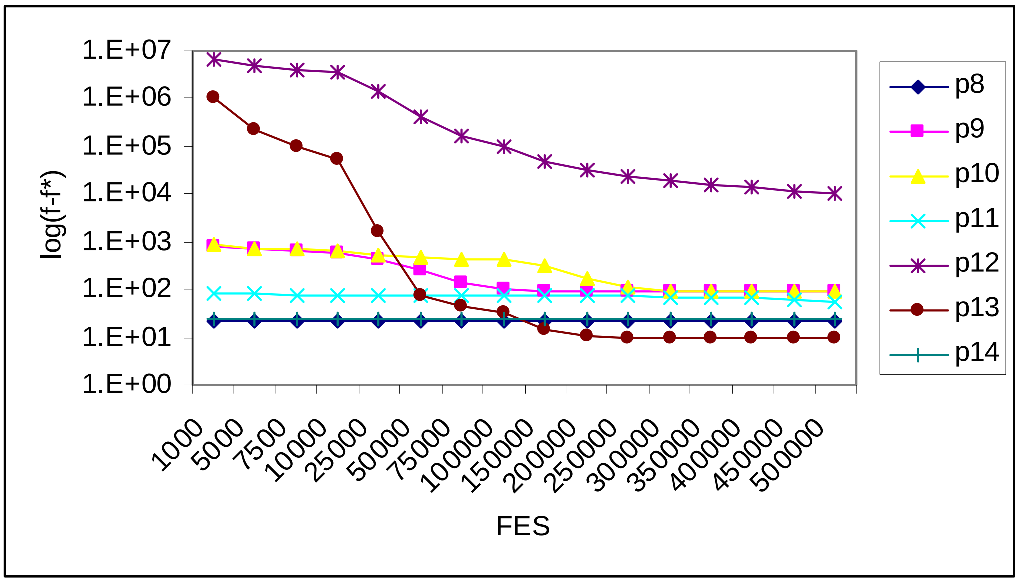


Figure 5.12 Convergence Graph of DE for D=50 for functions 8-14

As seen in Figure 5.12, the DE algorithm generated the near-optimal solution only for the function 13. It fails for the functions 8, 9, 10, 11, 12 and 14. Another interpretation of these results is that the DE algorithm could not be able to generate near-optimal solutions when the dimension size is increased.

From these results, we can conclude that the DE algorithm is affected by the dimension size. In other words, the performance of the DE algorithm gets worsened as the dimension size increases.

Table 5.2. : Mean Error and standard deviation values achieved at the termination for the DE Algorithm

Func.		D=10	D=30	D=50
1	**Mean**	**0.00000E+00**	**0.00000E+00**	**0.00000E+00**
	Std D.	0.00000E+00	0.00000E+00	0.00000E+00
2	**Mean**	**0.00000E+00**	**6.18957E-02**	**6.70521E+01**
	Std D.	0.00000E+00	4.21461E-02	2.92809E+01
3	**Mean**	**5.46328E-05**	**7.34622E+05**	**2.18206E+06**
	Std D.	1.01191E-04	3.83248E+05	9.15385E+05
4	**Mean**	**0.00000E+00**	**4.06987E+00**	**3.00509E+03**
	Std D.	0.00000E+00	4.20637E+00	2.18711E+03
5	**Mean**	**0.00000E+00**	**1.89052E+03**	**5.49715E+03**
	Std D.	0.00000E+00	1.88807E+03	2.49397E+03
6	**Mean**	**6.37853E-01**	**9.45834E+00**	**9.13380E+01**
	Std D.	1.49164E+00	5.37511E+00	8.49118E+01
7	**Mean**	**1.50829E-01**	**4.16067E-02**	**1.07342E-02**
	Std D.	6.40648E-02	6.68922E-02	1.58607E-02
8	**Mean**	**2.03384E+01**	**2.09600E+01**	**2.11374E+01**
	Std D.	7.50173E-02	5.14799E-02	3.69487E-02
9	**Mean**	**5.09419E+00**	**4.08729E+01**	**9.27698E+01**
	Std D.	2.09729E+00	1.12049E+01	2.10559E+01
10	**Mean**	**1.65177E+01**	**5.02670E+01**	**9.14884E+01**
	Std D.	6.98306E+00	1.33351E+01	1.68417E+01
11	**Mean**	**7.09222E-01**	**1.17990E+01**	**5.32926E+01**
	Std D.	9.72805E-01	4.16057E+00	1.83731E+01
12	**Mean**	**5.89551E+01**	**0.00000E+00**	**1.06185E+04**
	Std D.	2.68501E+02	0.00000E+00	8.53809E+03
13	**Mean**	**7.66957E-01**	**3.49835E+00**	**9.58925E+00**
	Std D.	3.34237E-01	1.17383E+00	3.20361E+00
14	**Mean**	**3.44692E+00**	**1.33659E+01**	**2.31630E+01**
	Std D.	5.73178E-01	1.99834E-01	1.74935E-01

CHAPTER 6: COMPARISON OF PSO AND DE ALGORITHMS

6.1 Comparison of PSO and DE Algorithms

In this chapter, PSO and DE algorithms have been compared according to best mean, standard deviation of the error values achieved at the termination as well as their success ratio. Error values achieved within the maximum number of function evaluation and the success ratio achieved within the fixed accuracy levels are presented in Tables 6.1 to 6.3. The fixed accuracy levels obtained from Suganthan et. al. [99] for the 14 test functions as given in Chapter 4 are 1e-6 for functions 1 to 5 and 1e-2 for functions 6 to 14.

Table 6.1. : Error values achieved in the Max_FES and Success Rate (PSO and DE for D=10)

Func.		1st(Min)	7th	13th(Median)	19th	25th(Max)	Mean	Std.	SR(%)
1	PSO	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	100
	DE	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	100
2	PSO	1.76600E-05	2.65480E-04	9.93690E-04	1.26971E-02	9.86478E+00	4.31888E-01	1.96785E+00	100
	DE	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	100
3	PSO	2.50113E+04	8.09600E+04	1.24667E+05	2.06380E+05	8.50952E+05	1.98704E+05	2.12806E+05	0
	DE	4.10000E-07	2.02000E-06	7.09000E-06	6.23900E-05	3.83810E-04	**5.46328E-05**	1.01191E-04	44
4	PSO	1.03240E-04	1.14938E-03	2.06002E-02	1.41357E-01	9.92933E+01	4.25415E+00	1.98131E+01	0
	DE	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	100
5	PSO	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	100
	DE	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	100
6	PSO	3.84650E-01	3.66355E+00	8.44764E+00	1.42832E+02	2.88304E+02	6.92385E+01	9.08055E+01	0
	DE	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	3.98658E+00	**6.37853E-01**	1.49164E+00	100
7	PSO	6.01463E-02	1.56842E-01	2.16706E-01	3.42226E-01	5.35337E-01	2.41916E-01	1.29032E-01	0
	DE	5.08014E-02	1.03748E-01	1.50057E-01	2.07027E-01	2.58674E-01	**1.50829E-01**	6.40648E-02	0
8	PSO	2.01574E+01	2.02826E+01	2.03568E+01	2.04105E+01	2.04617E+01	2.03441E+01	8.13829E-02	0
	DE	2.01870E+01	2.02973E+01	2.03631E+01	2.03872E+01	2.05102E+01	**2.03384E+01**	7.50173E-02	0
9	PSO	0.00000E+00	9.94959E-01	1.98992E+00	2.98488E+00	3.97990E+00	**1.99013E+00**	1.21875E+00	92
	DE	1.98992E+00	2.98488E+00	4.97480E+00	6.96471E+00	8.95463E+00	5.09419E+00	2.09729E+00	0
10	PSO	5.28933E+00	1.25393E+01	1.58296E+01	1.88197E+01	3.25179E+01	**1.64690E+01**	7.04173E+00	0
	DE	7.79445E+00	1.14447E+01	1.52435E+01	1.89897E+01	3.63416E+01	1.65177E+01	6.98306E+00	0
11	PSO	2.56900E+00	3.60895E+00	4.63681E+00	5.43349E+00	8.49518E+00	4.62658E+00	1.45265E+00	0
	DE	7.93340E-04	2.43548E-03	5.85140E-03	1.50192E+00	3.00221E+00	**7.09222E-01**	9.72805E-01	80
12	PSO	9.48824E-01	1.04495E+01	2.02180E+01	3.96839E+01	7.12255E+02	8.47897E+01	1.65819E+02	0
	DE	0.00000E+00	0.00000E+00	2.00000E-08	1.00030E+01	1.34735E+03	**5.89551E+01**	2.68501E+02	96
13	PSO	3.89080E-01	5.06936E-01	6.38670E-01	7.38544E-01	1.22706E+00	**6.62293E-01**	2.06914E-01	0
	DE	3.81196E-01	5.33667E-01	7.16250E-01	9.19178E-01	1.56296E+00	7.66957E-01	3.34237E-01	0
14	PSO	1.57958E+00	2.61841E+00	3.12565E+00	3.29433E+00	3.62920E+00	**2.96542E+00**	5.20650E-01	0
	DE	1.00236E+00	3.45017E+00	3.58983E+00	3.75966E+00	3.92519E+00	3.44692E+00	5.73178E-01	0
Avg.	**PSO**	**1.78876E+03**	**5.78677E+03**	**8.91019E+03**	**1.47582E+04**	**6.08666E+04**	**1.42079E+04**	**1.52211E+04**	**28**
	DE	**2.24332E+00**	**2.77264E+00**	**3.21739E+00**	**4.48089E+00**	**1.01849E+02**	**7.61552E+00**	**2.00780E+01**	**51.43**

As seen in Table 6.1, the DE algorithm was superior to the PSO algorithm for all the performance measures taken when overall average of the performance measures for 14 functions. In other words, the DE algorithm was better than the PSO algorithm in terms of min, median, max, mean, standard deviation and success ratio. When comparing the success ratio of both algorithms, the DE algorithm outperformed the PSO algorithm with a 51.43 percent which is almost twice as much as a success ratio of 28% for the PSO algorithm.

Table 6.2. : Error values achieved in the Max_FES and Success Rate (PSO and DE for D=30)

Func.		1st(Min)	7th	13th(Median)	19th	25th(Max)	**Mean**	Std.	SR(%)
1	PSO	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	1.90000E-07	8.00000E-09	3.79693E-08	100
	DE	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	100
2	PSO	9.19895E+01	1.44785E+02	2.84253E+02	5.35763E+02	1.37277E+03	4.01956E+02	3.40265E+02	0
	DE	1.00082E-02	2.69799E-02	4.70619E-02	8.74604E-02	1.58316E-01	**6.18957E-02**	4.21461E-02	0
3	PSO	2.14739E+06	3.51459E+06	5.23473E+06	1.76596E+07	3.46473E+07	**1.07647E+07**	9.33441E+06	0
	DE	2.52866E+05	3.63513E+05	6.71938E+05	9.74700E+05	1.45497E+06	7.34622E+05	3.83248E+05	0
4	PSO	8.49725E+02	1.54421E+03	2.65265E+03	3.59317E+03	6.77614E+03	2.70271E+03	1.48463E+03	0
	DE	4.44201E-01	1.56985E+00	2.26303E+00	4.37445E+00	1.58945E+01	**4.06987E+00**	4.20637E+00	0
5	PSO	1.80210E+01	8.76067E+03	1.08879E+04	1.35322E+04	1.80297E+04	1.10620E+04	3.83930E+03	0
	DE	1.08400E-05	9.65160E-03	2.19951E+03	2.91858E+03	5.93939E+03	**1.89052E+03**	1.88807E+03	0
6	PSO	1.28023E+01	2.84509E+01	9.26050E+01	1.63450E+02	1.08048E+03	**1.57711E+02**	2.13877E+02	0
	DE	7.91052E-01	6.02897E+00	1.01827E+01	1.26779E+01	1.80863E+01	**9.45834E+00**	5.37511E+00	0
7	PSO	4.92000E-06	1.03743E-02	4.04980E-02	1.50584E-01	3.08387E-01	8.19353E-02	8.93249E-02	64
	DE	0.00000E+00	0.00000E+00	1.00000E-08	5.39371E-02	2.89677E-01	**4.16067E-02**	6.68922E-02	100
8	PSO	2.07623E+01	2.08980E+01	2.09448E+01	2.09789E+01	2.10350E+01	**2.09311E+01**	6.64438E-02	0
	DE	2.08070E+01	2.09520E+01	2.09705E+01	2.09984E+01	2.10302E+01	2.09600E+01	5.14799E-02	0
9	PSO	1.39294E+01	1.98992E+01	2.48740E+01	2.88538E+01	3.28336E+01	**2.43572E+01**	5.51904E+00	0
	DE	2.18891E+01	3.28336E+01	4.07933E+01	4.77579E+01	6.06924E+01	4.08729E+01	1.12049E+01	0
10	PSO	4.46290E+01	6.76721E+01	7.70619E+01	9.74682E+01	2.44440E+02	8.72283E+01	3.91043E+01	0
	DE	2.34153E+01	4.49232E+01	5.00901E+01	5.49342E+01	7.95865E+01	**5.02670E+01**	1.33351E+01	0
11	PSO	2.10586E+01	2.74924E+01	3.04811E+01	3.50665E+01	3.94895E+01	3.11684E+01	5.19378E+00	0
	DE	5.37358E+00	8.52891E+00	1.09034E+01	1.56893E+01	1.87345E+01	**1.17990E+01**	4.16057E+00	0
12	PSO	2.21992E+03	9.09589E+03	1.75403E+04	3.40146E+04	6.06273E+04	2.21836E+04	1.60610E+04	0
	DE	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	0
13	PSO	2.12003E+00	3.06190E+00	3.71451E+00	4.08763E+00	6.58410E+00	3.70427E+00	9.19095E-01	0
	DE	1.97830E+00	2.68041E+00	2.95265E+00	4.04031E+00	5.77030E+00	**3.49835E+00**	1.17383E+00	0
14	PSO	1.24395E+01	1.29290E+01	1.30828E+01	1.32259E+01	1.34684E+01	**1.30484E+01**	2.44407E-01	0
	DE	1.29578E+01	1.32526E+01	1.33953E+01	1.35254E+01	1.36599E+01	1.33659E+01	1.99834E-01	0
Avg	**PSO**	**1.53621E+05**	**2.52451E+05**	**3.76169E+05**	**1.26512E+06**	**2.48111E+06**	**7.71527E+05**	**6.68314E+05**	**11.71**
	DE	**1.80681E+04**	**2.59746E+04**	**4.81635E+04**	**6.98423E+04**	**1.04367E+05**	**5.26191E+04**	**2.75126E+04**	**14.29**

As seen in Table 6.2, the DE algorithm again outperformed the PSO algorithm for all the performance measures taken when overall average of the performance measures for 14 functions for the dimension size of 30. In other words, the DE algorithm was better than the PSO algorithm in terms of min, median, max, mean, standard deviation and success ratio. When comparing the success ratio of both algorithms, the DE algorithm outperformed the PSO algorithm since DE's success ratio was 14.29 percent whereas PSO's success ratio was 11.71 percent.

Table 6.3. : Error values achieved in the Max_FES and Success Rate (PSO and DE for D=50)

Func.		1st(Min)	7th	13th(Median)	19th	25th(Max)	**Mean**	Std.	SR(%)
1	PSO	2.00000E-08	6.80000E-07	3.72000E-06	1.92900E-05	2.81544E+00	1.16368E-01	5.62614E-01	100
	DE	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	0.00000E+00	**0.00000E+00**	0.00000E+00	100
2	PSO	2.73281E+03	4.12783E+03	5.65296E+03	7.55080E+03	1.50197E+04	6.53038E+03	3.35448E+03	0
	DE	2.67056E+01	5.11244E+01	5.74726E+01	7.93419E+01	1.40927E+02	**6.70521E+01**	2.92809E+01	0
3	PSO	7.18257E+06	2.71240E+07	3.66474E+07	6.29808E+07	1.21738E+08	4.84217E+07	3.11780E+07	0
	DE	9.83717E+05	1.49749E+06	2.10192E+06	2.56510E+06	4.52991E+06	**2.18206E+06**	9.15385E+05	0
4	PSO	1.01778E+04	1.82279E+04	2.25180E+04	2.79352E+04	3.50883E+04	2.27204E+04	7.01268E+03	0
	DE	6.42150E+02	1.61209E+03	2.26646E+03	3.58110E+03	1.04788E+04	**3.00509E+03**	2.18711E+03	0
5	PSO	1.25707E+04	1.71181E+04	1.95252E+04	2.21800E+04	2.88438E+04	1.98764E+04	3.63886E+03	0
	DE	5.38846E+02	3.66471E+03	5.61799E+03	7.33205E+03	1.04086E+04	**5.49715E+03**	2.49397E+03	0
6	PSO	4.24331E+01	1.09911E+02	1.71295E+02	3.04782E+02	1.36723E+03	3.03737E+02	3.42479E+02	0
	DE	2.13773E+01	4.01439E+01	8.45228E+01	9.32595E+01	3.88045E+02	**9.13380E+01**	8.49118E+01	0
7	PSO	1.02000E-03	6.85393E-03	1.99886E-02	3.07924E-02	1.05730E-01	2.47858E-02	2.36381E-02	72
	DE	1.00000E-07	6.60000E-07	2.66000E-06	1.66393E-02	4.86660E-02	**1.07342E-02**	1.58607E-02	96
8	PSO	2.10415E+01	2.11087E+01	2.11392E+01	2.11633E+01	2.11908E+01	**2.11326E+01**	3.99925E-02	0
	DE	2.10276E+01	2.11233E+01	2.11432E+01	2.11634E+01	2.11989E+01	2.11374E+01	3.69487E-02	0
9	PSO	4.57681E+01	5.87025E+01	6.96471E+01	7.26319E+01	8.75563E+01	**6.62258E+01**	1.11524E+01	0
	DE	6.66622E+01	7.95966E+01	8.65613E+01	9.65108E+01	1.42279E+02	9.27698E+01	2.10559E+01	0
10	PSO	1.01273E+02	1.39716E+02	1.84071E+02	2.89274E+02	4.04977E+02	2.14250E+02	9.29100E+01	0
	DE	5.59457E+01	8.37425E+01	9.01580E+01	9.74667E+01	1.33335E+02	**9.14884E+01**	1.68417E+01	0
11	PSO	5.32597E+01	6.31906E+01	6.70882E+01	7.08982E+01	7.45000E+01	6.61378E+01	5.93583E+00	0
	DE	2.44855E+01	3.74256E+01	5.70049E+01	7.11363E+01	7.44238E+01	**5.32926E+01**	1.83731E+01	0
12	PSO	1.80026E+04	6.22675E+04	1.02842E+05	1.39305E+05	2.27342E+05	1.08151E+05	5.76638E+04	0
	DE	6.30067E+02	4.08555E+03	8.10495E+03	1.60856E+04	3.08875E+04	**1.06185E+04**	8.53809E+03	0
13	PSO	5.33924E+00	7.66812E+00	9.28354E+00	1.16804E+01	1.55820E+01	9.64705E+00	2.69821E+00	0
	DE	5.19514E+00	7.35488E+00	8.78431E+00	1.15155E+01	1.59468E+01	**9.58925E+00**	3.20361E+00	0
14	PSO	2.20372E+01	2.26008E+01	2.27769E+01	2.28943E+01	2.32639E+01	**2.27563E+01**	2.80591E-01	0
	DE	2.26743E+01	2.30478E+01	2.32035E+01	2.32611E+01	2.34932E+01	2.31630E+01	1.74935E-01	0
Avg	**PSO**	**5.16167E+05**	**1.94473E+06**	**2.62846E+06**	**4.51275E+06**	**8.71759E+06**	**3.46998E+06**	**2.23215E+06**	**12.29**
	DE	**7.04123E+04**	**1.07657E+05**	**1.51310E+05**	**1.85185E+05**	**3.27330E+05**	**1.57259E+05**	**6.63413E+04**	**14**

As seen in Table 6.3, the DE algorithm again outperformed the PSO algorithm for all the performance measures taken when overall average of the performance measures for 14 functions for the dimension size of 50. In other words, the DE algorithm was better than the PSO algorithm in terms of min, median, max, mean, standard deviation and success ratio. When comparing the success ratio of both algorithms, the DE algorithm outperformed the PSO algorithm since DE's success ratio was 14 percent whereas PSO's success ratio was 12.29 percent.

Finally, the computational complexity of each algorithm is considered. The algorithm complexity, which is defined in chapter 4, is computed for 10, 30, and 50 dimensions by using function 3 in order to show the algorithm complexity relationship with increasing dimensions. The computational complexity of each algorithm is given. Table 6.4 and Table 6.5 show the complexity of the PSO and DE algorithms, respectively.

Table 6.4. : Complexity of the PSO Algorithm

PSO			
	D=10	D=30	D=50
T0	551	551	551
T1	1442	4526	7481
T2	7312	25671	50476
Complexity	10.65	38.37	78.03

Table 6.5. : Complexity of the DE Algorithm

DE			
	D=10	D=30	D=50
T0	551	551	551
T1	1442	4526	7481
T2	7150	25094	49497
Complexity	10.36	37.33	76.25

As seen from Table 6.4 and 6.5, the time complexity of both algorithm show similar behavior in terms of CPU times

Table 6.6 : Average Success Rates of PSO and DE at D=10,30,50

Dimension		10	30	50
PSO	Avg. SR.(%)	28	11.72	12.29
DE	Avg. SR.(%)	51.43	14.29	14

To sum up all the results, overall average success ratio for both algorithms with different dimensions are given in Table 6.6. The DE algorithm perfoms better than the PSO algorithm according to average success ratio. However, as the dimension size increases, the DE algorithm also deteriorates.

CHAPTER 7: CONCLUSION

7.1 Conclusions

In this research, latest metaheuristic approaches so called the particle swarm optimization and differential evolution algorithms are presented to solve continuous function optimization. The benchmark suite is taken from Suganthan et. al. [99]. According to the results, the DE algorithm performed better than the PSO algorithm in general for the functions considered. Main contribution of this research is the development of both algorithm for newly designed benchmark problems. Since these benchmarks are newly designed, unfortunately there exist no results for comparison purposes. For this reason, the computational results are presented to be compared with the DE algorithm only.

For the future work, these algorithms can be extended to other versions of the PSO and DE algorithms such as multi-swarm parallel algorithms with master-slave or island models to obtain better results. In addition, there exist different types of local search algorithms that can be embedded in these algorithms. These algorithms are Nelder and Mead algorithm, Solis and Wets algorithm, Pattern Search etc. Including these local searches in the PSO and DE algorithms may lead to better results.

REFERENCES

[1] **Leon, A.,** 1966. "A classified bibliography on optimization." Recent advances in optimization techniques.

[2] **Dixon, L.C.W. and Szego, G.P.,** 1978. **"**The global optimization problem: an introduction." Towards Global Optimization 2. , 1-15.

[3] **Gomulka, J.**, 1978. "A user's experience with Torn's clustering algorithm." Towards Global Optimization 2. , 63-70.

[4] **Archetti, F. and Schoen, F.**, 1984. "A Survey on the global optimization problem: general theory and computational approaches." *Annals of Operations Research 1,* J.C.Baltzer A.G. Publishing. pp. 87-110.

[5] **Torn, A., Zilinskas, A.,** 1989. Global Optimisation. Lecture notes in Computer Science, vol. 350. Springer-Verlag, Berlin.

[6] **Xiangsun, Z.,** 1984. "Survey on the global optimum deterministic algorithms*", Chinese journal of operation research,* **Vol.3**, No.2, 1-13

[7] **Yunkang, Z.,** 1992. "Survey on the global optimum probabilistic algorithms*", Chinese journal of operation research,* **Vol.11,** No.2, 28-41

[8] **Ling, W.,** 2000. "Meta-heuristic algorithms: A review, Control and Decision," Beijin, **Vol.15,** No.3,257-262

[9] **Peng, Z., and Chuan H.,** 2000. "A global optimization method based on uniform design", *Proceeding of the 1st International Conference on Mechanical Engineering,*Shanghai.

[10] **Qisheng, G.,** 1999. “The genetic algorithm and application based on uniform design”, *Information and Control*, Beijin, **Vol.28,** No.3, 236-239

[11] **Qisheng, G.,** 2000. “Dimulsted annealing concurrent algorithm of parameter design”, *Theory and practice of system engineering*, Beijin, No.8, 41-44

[12] **Shaojun, L.,** 2000. “Study on Genetic-Alopex algorithms for seeking the global optimization”, *Information and Control*, Beijin, **Vol.29,** No.4, 304-308

[13] **Zhiyuan, W.,** 1998. “Annealing accuracy penalty function based nonlinear constrained optimization method with genetic algorithms”, *Control and Decision*, Beijin, **Vol.13**, No.2, 136-140

[14] **Ling, W.,** 1998. “Study on GASA hybrid strategy and its convergence behaviour”, *Control and Decision*, Beijin, **Vol.13**, No.6, 669-672

[15] **Ting, H.,** 2000. “Research on SAA-based new hybrid evolutionary algorithm and its application”, *Control and Decision*, Beijin, **Vol.15**, No.4, 504-506

[16] **Chenzhong, L.,** 2000. “Evolutionary algorithms with chaotic mutation”, *Control and Decision*, Beijin, **Vol.15**, No5, 557-560

[17] **Zicai, W.,** 1999. “Simulated annealing algorithm based on chaotic variable”, *Control and Decision*, Beijin, **Vol.14**, No.4, 381-384

[18] **Ling, W.**, 1999. “Genetic algorithm combined with a chaotic sequence”, *Theory and practice of system engineering*, Beijin, No.11, 1-7

[19] **Ling, W.,** 2000. “A kind of chaotic neural network optimization algorithm based on annealing strategy”, *Control Theory and Applications*, Beijin, **Vol.17,** No.1, 139-142

[20] **Wei, P.**, 1999. “A hybrid genetic algorithm for function optimization”, *Journal of Software*, Beijin,**Vol.10,** No.8, 819-823

[21] **Junling, W.,** 1998. “A heuristic search method by analogy”, *Journal of computer science*, Beijin, **Vol.25,** No. 5, 33-37

[22] **Lishan, K.,** 1994. “non-numeric merged algorithms- Simulated annealing algorithm”, the science press, Beijin.

[23] **Zhenjun, P.,** 1998. Evolvement computation, the press of Qinghua University , Beijin.

[24] **Yang, J.M.,** 1997. “A Continuous Genetic Algorithm for Global Optimization”, Editor: Thomas Back, *Proceedings of the Seventh International Conference on Genetic Algorithms.* Morgan Kaufmann Publishers, Inc. San Francisco, California. 230-237

[25] **Lishan, K.,** 1994. “non-numeric merged algorithms- Genetic algorithm”, *the science press,* Beijin.

[26] **Guoliang**, **C.,** 1996. “Genetic algorithm and its application”, the press of people’s post, Beijin.

[27] **Gen, M., and Cheng, R.,** 2000. “Genetic algorithms and engineering design”, the science press, Beijin.

[28] **Yugeng, X.,** 1996. “Survey on genetic algorithm”, *Control Theory and Applications,* Beijin, **Vol.13,** No.6, 697-708

[29] **Yanfeng, S.,** 1996. “The application of genetic algorithm in optimization”, *Control and Decision*, Beijin, **Vol.11**, No.4, 425-431

[30] **Cong, D.,** 1998. “Generalized genetic algorithm”, *Exploration of Nature,* Beijin, **Vol.17**, No.63, 33-37

[31] **Fogel, L. J., Owens, A. J., and Walsh, M. J.,** 1966. Artifical Intelligence through Simulated Evolution. John Wiley, New York.

[32] **Schwefel, H. P.,** 1981. Numerical Optimization of Computer Models. John Wiley, Chichester, UK.

[33] **Koza, J. R.,** 1994. Genetic Programming: On the Programming of Computer by Means of Natural Selection, MIT Press, Cambridge.

[34] **Bing, L.,** 1997. “Chaos optimization method and its application”, *Control Theory and Applications,* Beijin, **Vol.14,** No.4, 613-615

[35] **Chenzhong, L.,** 2000. “Chaos search method for nonlinear constrained optimization”, *Theory of system engineering and its applications*, No.8, 54-57

[36] **Zhang, J.,** 1999. “A new evolution algorithm-Ant Colony Algorithm”, *Theory of system engineering and its applications,* Beijin, No.3, 84-87

[37] **Zhang, J.,** 2000. “A self-adaptive ant colony algorithm”, *Control Theory and Applications,* Beijin, **Vol.17,** No.1, 1-3

[38] **Kennedy, J. and Eberhart, R. C.,** 1995. “Particle Swarm Optimization,” *Proc. of IEEE International Conference on Neural Networks,* Piscataway, NJ, USA, pp. 1942-1948.

[39] **Kennedy, J., Eberhart, R. C., and Shi, Y.,** 2001. Swarm Intelligence, Morgan Kaufmann, San Mateo, CA.

[40] **Storn, R. and Price, K.,** 1997. “Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Space,” *Journal of Global* Optimization, vol. 11, pp. 341-359.

[41] **De Jong, K.A.**, 1975. “An Analysis of the Behavior of a Class of Genetic Adaptative Systems”, *Ph.D. Thesis,* University microfilms no. 76-9381, University of Michigan, Ann Arbor, MI.

[42] **Baker, J.E.,** 1985. “Reducing Bias and Inefficiency in the Selection Algorithm.” *In Proceedings of the Second International Conference on Genetic Algorithms and their Applications,* New Jersey, USA.

[43] **Goldberg, D.E.**, 1989. Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, New York.

[44] **Mühlenbein H. and Schlierkamp-Voosen, D.,** 1993. “Analysis of Selection, Mutation and Recombination in Genetic Algorithms.” Technical Report 93/94, GMD.

[45] **Chipperfield, A.J., P.J. Fleming, H. Pohleim, and C.M. Fonseca.,** 1994. “Genetic Algorithm Toolbox User’s Guide.” ACSE Research Report No. 512, University of Sheffield.

[46] **Reeves, C.R. (ed.),** 1995. “Modern Heuristic Techniques for Combinatorial Problems.” In *Advanced Topics in Computer Science*. McGraw-Hill, Chap. 4.

[47] **Michalewicz, Z.,** 1996. *Genetic Algorithms+Data Structures=Evolution Programs*.Third ed., Springer- Verlag, New York.

[48] **Siarry, P., G. Berthiau, F. Durbin, and J. Haussy**, 1997. “Enhanced Simulated Annealing for Globally Minimizing Functions of Many Continuous Variables.” *ACM Transactions on Mathematical Software,* 23(2), 209–228.

[49] **Battiti, R., and Tecchiolli, G.,** 1994. “The reactive tabu search”, *ORSA Journal on Computing*, 6(2):126—140.

[50] **Battiti, R., and Tecchiolli, G.,** 1995. “The continuous reactive tabu search: blending combinatorial optimization and stochastic search for global optimization”, *Annals of Operations Research (in press).*

[51] **Salerno, J.,** 1997. “Using the Particle Swarm Optimization Technique to Train a Recurrent Neural Model,” *Proc. of the IEEE International Conference on Tools with Artificial Intelligence,* pp. 45-49.

[52] **Ismail, A. and Engelbrecht, A. P.** , 1999. “Training Product Units in Feedforward Neural Networks Using Particle Swarm Optimization,” *Proc. of the International Conference on Artificial Intelligence,* Durban, South Africa, pp. 36-40.

[53] **Gudise, V. G. and Venayagamoorthy, G. K.** , 2003. “Comparison of Particle Swarm Optimization and Backpropagation as Training Algorithms for Neural Networks,” *Proc. of the IEEE Swarm Intelligent Symposium 2003,* Indianapolis, Indiana, USA, pp. 110-117.

[54] **Yoshida, H., Kawata, K., Fukuyama, Y., and Nakanishi, Y.** , 2000. “A Particle Swarm Optimization for Reactive Power and Voltage Control Considering Voltage Security Assessment,” *IEEE Transactions on Power Systems,* **vol. 15,** pp.1232-1239.

[55] **Abido, M. A.** , 2002. “Optimal Power Flow Using Particle Swarm Optimization,” *Electrical Power and Energy Systems,* **vol. 24,** pp. 563-571.

[56] **Abido, M. A.** , 2002. “Optimal Design of Power System Stabilizers Using Particle Swarm Optimization,” *IEEE Transactions on Energy Conversion,* **vol. 17,** pp. 406-413.

[57] **Agrafiotis, D. K. ad Cedeno, W.** , 2002. “Feature Selection for Structure-Activity Correlation Using Binary Particle Swarms,” *Journal of Medicinal Chemistry,* **vol. 45,** pp. 1098-1107.

[58] **Brandstatter, B. and Baumgartner, U.** , 2002. “Particle Swarm Optimization – Mass-Spring System Analogon,” *IEEE Transactions on Magnetics,* **vol. 38,** pp. 997-1000.

[59] **Ciuprina, G., Ioan, D., and Munteanu, I.** , 2002. “Use of Intelligent-Particle Swarm Optimization in Electromagnetics,” *IEEE Transactions on Magnetics,* **vol. 38,** pp. 1037-1040.

[60] **Robinson, J. and Rahmat-Samii, Y. ,** 2004. “Particle Swarm Optimization in Electromagnetics,” *IEEE Transactions on Antennas and Propagation,* **vol. 52,** pp. 397-407.

[61] **Eberhart, R. C. , and Hu, X. ,** 1999. “Human tremor analysis using particle swarm optimization,” *Proc. Congress on Evolutionary Computation 1999,* Washington, DC, pp 1927-1930. Piscataway, NJ: IEEE Service Center.

[62] **Wachowiak, M. P. , Smolikova`, R. , Zheng, Y. , Zurada, J. M. , and Elmaghraby, A. S. ,** 2004. “An approach to multimodal biomedical image registration utilizing particle swarm optimization,” *IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).*

[63] **Messerschmidt, L. , Engelbrecht, A. P. ,** 2004. “Learning to play games using a PSO-based competitive learning approach,” *IEEE Transactions on Evolutionary Computation (accepted for special issue on PSO).*

[64] **Van der Merwe, D. W. and Engelbrecht, A. P.** , 2003. “Data Clustering Using Particle Swarm Optimization,” *Proc. of IEEE Congress on Evolutionary Computation 2003,* Canbella, Australia, pp. 215-220.

[65] **Coello Coello, C. A., Luna, E. H. n., and Aguirre, A. H. n.** , 2003. “Use of Particle Swarm Optimization to Design Combinational Logic Circuits,” *Lecture Notes in Computer Science (LNCS),* **no. 2606,** pp. 398-409.

[66] **Tasgetiren, M. F. and Liang, Y.-C.** , 2003. “A Binary Particle Swarm Optimization Algorithm for Lot Sizing Problem,” *Journal of Economic and Social Research,* **vol. 5, no. 2,** pp. 1-20.

[67] **L.-W. Yeh,** 2003. Optimal Procurement Policies for Multi-product Multi-supplier with Capacity Constraint and Price Discount, *Master thesis,* Department of Industrial Engineering and Management, Yuan Ze University, Taiwan, R.O.C.

[68] **Salman, A., Ahmad, I., and Al-Madani, S.**, 2003. “Particle Swarm Optimization for Task Assignment Problem,” *Microprocessors and Microsystems,* **vol. 26,** pp. 363-371.

[69] **Onwubolu, G. C. and Clerc, M.,** 2004. “Optimal Path for Automated Drilling Operations by a New Heuristic Approach Using Particle Swarm Optimization,” *International Journal of Production Research,* **vol. 4,** pp. 473-491.

[70] **Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., and Gencyilmaz, G.**, 2004. “Particle Swarm Optimization Algorithm for Makespan and Maximum Lateness Minimization in Permutation Flowshop Sequencing Problem,” *Proc. of the Fourth International Symposium on Intelligent Manufacturing Systems,* Sakarya, Turkey, pp. 431-441.

[71] **Tasgetiren, M. F., Sevkli, M., Liang, Y.-C., and Gencyilmaz, G.**, 2004. “Particle Swarm Optimization Algorithm for Single Machine Total Weighted Tardiness Problem,” *Proc. of IEEE Congress on Evolutionary Computation 2004,* Portland, Oregon, USA, pp. 1412-1419.

[72] **Tasgetiren, M. F., Sevkli, M., Liang, Y.-C., and Gencyilmaz, G.**, 2004. “Particle Swarm Optimization Algorithm for Permutation Flowshop Sequencing Problem,” *Lecture Notes in Computer Science (LNCS),* **no. 3172,** pp. 382-390.

[73] **Y. Shi and R. C. Eberhart,** 1998. ”Parameter Selection in Particle Swarm Optimization”, Evolutionary Programming VII (1998), *Lecture Notes in Computer Science 1447,* pp. 591–600, Springer, 1998.

[74] **Y. Shi and R. C. Eberhart,** 1998. ”A modified Particle Swarm Optimiser”, *IEEE International Conference on Evolutionary Computation,* Anchorage, Alaska.

[75] **Storn, R. and Price, K.**, 1995. “Differential Evolution – a Simple and Efficient Adaptive Scheme for Global Optimization over Continuous Spaces,” *Technical Report,* **TR-95-012, ICSI.**

[76] **Storn, R.**, 1999. “Designing Digital Filters with Differential Evolution.” In D. Corne, M. Dorigo, and F. Glover (eds.), New Ideas in Optimization, London: McGraw-Hill, UK, 109-125.

[77] **Storn, R.**, 1996. “Differential Evolution Design of an IIT-Filter with Requirements for Magnitude and Group Delay,” *Proc. of IEEE International Conference on Evolutionary Computation,* pp. 268-273.

[78] **Masters, T. and Land, W.**, 1997. “A New Training algorithm for the General Regression Neural Network,” *Proc. of the 1997 IEEE International Conference on Systems, Man, and Cybernetics,* pp. 1990-1994.

[79] **J. Ilonen, J.-K. Kamarainen and J. Lampinen,** “Differential Evolution Training Algorithm for Feed-Forward Neural Networks,” In: Neural Processing Letters Vol. 7, No. 1 93-105. 2003.

[80] **R. Storn,** 1996. “Differential evolution design of an IIR-filter,” *In: Proceedings of IEEE Int. Conference on Evolutionary Computation ICEC'96.* IEEE Press, New York. 268-273.

[81] **Rogalsky, T., S. Kocabiyik and R. W. Derksen.** ,2000. “Differential Evolution in Aerodynamic Optimization.” *Canadian Aeronautics and Space Journal,* **46(4),** 183-190.

[82] **Ruzek, B. and M. Kvasnicka.** , 2001. “Differential Evolution Algorithm in the Earthquake Hypocenter Location.” *Pure and Applied Geophysics,* **158(4),** 667-693.

[83] **Rae, A. and S. Parameswaran.** , 2001. “Synthesising Application-Specific Heterogenous Multiprocessors Using Differential Evolution.” *IEICE Transactions on Fundamentals of Electronics,* Communications and Computer Sciences E84-A(12), 3125-3131.

[84] **Tasgetiren, M. F., Liang, Y.-C., Sevkli, M., and Gencyilmaz, G.**, 2004. “Differential Evolution Algorithm for Permutation Flowshop Sequencing Problem with Makespan Criterion,” *Proc. of the Fourth International Symposium on Intelligent Manufacturing Systems,* Sakarya, Turkey, pp. 442-452.

[85] **Joshi, R. and Sanderson, A. C.**, 1999. “Minimal Representation Multisensor Fusion Using Differential Evolution,” *IEEE Transactions on Systems, Man, and Cybernetics,* Part A, **vol. 29,** pp. 63-76.

[86] **Babu, B. V. and Sastry, K. K. N.**, 1999. "Estimation of Heat Transfer Parameters in a Trickle-Bed Reactor Using Differential Evolution and Orthogonal Collocation," *Computers and Chemical Engineering,* **vol. 23,** pp. 327-339.

[87] **Storn, R.**, 1999. "System Design by Constraint Adaptation and Differential Evolution," *IEEE Transactions on Evolutionary Computation,* **vol. 3,** pp. 22-34.

[88] **Abbass, H. A.**, 2002. "An Evolutionary Artificial Neural Networks Approach for Breast Cancer Diagnosis," *Artificial Intelligence in Medicine,* **vol. 25,** pp. 265-281.

[89] **Rüttgers, M.**, 1997. "Design of a New Algorithm for Scheduling in Parallel Machine Shops," *Proc. of the Fifth European Congress on Intelligent Techniques and Soft Computing,* pp. 2182-2187.

[90] **Ursem, R., Vadstrup, R.,** 2003. "Parameter identification of induction motors using differential evolution," *In: Proceedings of the Fifth Congress on Evolutionary Computation (CEC-2003),* IEE Press, Piscataway, NJ, USA, pp. 790–796.

[91] **Thomsen, R.,** 2003. "Flexible ligand docking using differential evolution," *In: Proceedings of the Fifth Congress on Evolutionary Computation, (CEC-2003,*. IEE Press, Piscataway, NJ, USA, pp. 2354–2361.

[92] **Krink, T., Filipiˇc, B., Fogel, G., Thomsen, R.,** 2004. "Noisy optimisation problems—a particular challenge for differential evolution?" *In: Proceedings of the Sixth Congress on Evolutionary Computation (CEC-2004),* IEE Press, Piscataway, NJ, USA, pp. 332–339.

[93] **VesterstrZm, J., Thomsen, R.,** 2004. "A comparative study of differential evolution, particle swarm optimization and evolutionary algorithms on numerical benchmark problems," *In: Proceedings of the Sixth Congress on Evolutionary Computation (CEC-2004),* IEE Press, Piscataway, NJ, USA, pp. 1980–1987.

[94] **Corne, D., Dorigo, M., and Glover, F. (eds.)** , 1999. "Part Two: Differential Evolution," New Ideas in Optimization, McGraw-Hill, pp. 77-158.

[95] **Lampinen, J.**, 2001. "A Bibliography of Differential Evolution Algorithm". *Technical Report*, Lappeenranta University of Technology, Department of Information Technology, Laboratory of Information Processing.

[96] **Babu, B. V. and Onwubolu, G. C. (eds.)**, 2004. New Optimization Techniques in Engineering, Springer Verlag.

[97] **Price, K., Storn, R., and Lampinen, J.**, 2005. Differential Evolution – A Practical Approach to Global Optimization, Springer-Verlag.

[98] **Storn, R. and Price, K.**, 1997. "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Space," *Journal of Global Optimization*, **vol. 11,** pp. 341-359.

[99] The problem definition files, codes and evaluation criteria are made available in *http://www.ntu.edu.sg/home/EPNSugan*

[100] **Liang, J. J., Suganthan, P. N., and Deb, K.,** 2005. "Novel composition test functions for numerical global optimization", *IEEE Swarm Intelligence Symposium,* pp. 68-75.

[101] **Zhong, W. C. , Liu, J. , Xue, M. Z. and Jiao, L. C. ,** 2004. "A multiagent genetic algorithm for global numerical optimization," *IEEE Trans. On Systems, Man and Cybernetics (Part B) ,* **vol. 34**, pp. 1128-1141.

[102] **Coello Coello, C. A. , Pulido, G. T. and Lechuga, M. S. ,** 2004. "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation,* **8(3),** pp. 256-279.

[103] **Van den Bergh, F. and Engelbrecht, A. P. ,** 2004. "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation,* **8(3),** pp. 225-239.

[104] **Leung, Y. W. and Wang, Y. P.,** 2001. "An orthogonal genetic algorithm with quantization for global numerical optimization," *IEEE Transactions on Evolutionary Computation,* **5(1),** pp. 41-53.

[105] **Salomon, R.,** 1996. "Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions," *Biosystems,* **vol. 39,** pp. 263-278.

APPENDIX A

Table A.1. : Error Values Achieved at 1e3 FES, 1e4 FES, 1e5 FES and at Termination for PSO Algorithm for D=10

FES		1	2	3	4	5	6	7
1e3	1st	1.47948E+03	3.21335E+03	9.68577E+06	2.28773E+03	2.13117E+02	2.13503E+07	1.06912E+01
	7th	2.75122E+03	4.86396E+03	2.50273E+07	4.73202E+03	5.60503E+02	7.49290E+07	2.35460E+01
	13th	3.99646E+03	5.51322E+03	4.22674E+07	7.41025E+03	6.33911E+02	1.26762E+08	2.96474E+01
	19th	4.23627E+03	6.41568E+03	5.39285E+07	8.83893E+03	9.41594E+02	1.97655E+08	4.49104E+01
	25th	6.24589E+03	9.45018E+03	1.29522E+08	1.28319E+04	3.34479E+03	3.15349E+08	6.35104E+01
	Mean	**3.67483E+03**	**5.57230E+03**	**4.40712E+07**	**7.06576E+03**	**8.21209E+02**	**1.36002E+08**	**3.34451E+01**
	Std D.	1.16968E+03	1.48813E+03	2.46624E+07	2.67136E+03	6.09282E+02	7.29025E+07	1.33728E+01
1e4	1st	4.83387E+02	1.42596E+03	3.92467E+06	9.03847E+02	2.49714E+01	3.30793E+06	3.30480E+00
	7th	1.07941E+03	2.04405E+03	7.82193E+06	2.02561E+03	6.31861E+01	1.99488E+07	6.30282E+00
	13th	1.30838E+03	2.23742E+03	9.91559E+06	2.54786E+03	1.01348E+02	2.53140E+07	7.45523E+00
	19th	1.72260E+03	2.51689E+03	1.31614E+07	2.90955E+03	1.27797E+02	3.51203E+07	8.56092E+00
	25th	2.22004E+03	3.21198E+03	2.08326E+07	3.94517E+03	1.74643E+02	5.77292E+07	1.34647E+01
	Mean	**1.35595E+03**	**2.24009E+03**	**1.05080E+07**	**2.49749E+03**	**9.50043E+01**	**2.71476E+07**	**7.51574E+00**
	Std D.	4.79773E+02	4.22656E+02	4.05245E+06	7.94431E+02	4.03227E+01	1.49223E+07	2.24194E+00
1e5	1st	0.00000E+00	1.76600E-05	2.50113E+04	1.03240E-04	0.00000E+00	3.84650E-01	6.01463E-02
	7th	0.00000E+00	2.65480E-04	8.09600E+04	1.14938E-03	0.00000E+00	3.66355E+00	1.56842E-01
	13th	0.00000E+00	9.93690E-04	1.24667E+05	2.06002E-02	0.00000E+00	8.44764E+00	2.16706E-01
	19th	0.00000E+00	1.26971E-02	2.06380E+05	1.41357E-01	0.00000E+00	1.42832E+02	3.42226E-01
	25th	0.00000E+00	9.86478E+00	8.50952E+05	9.92933E+01	0.00000E+00	2.88304E+02	5.35337E-01
	Mean	**0.00000E+00**	**4.31888E-01**	**1.98704E+05**	**4.25415E+00**	**0.00000E+00**	**6.92385E+01**	**2.41916E-01**
	Std D.	0.00000E+00	1.96785E+00	2.12806E+05	1.98131E+01	0.00000E+00	9.08055E+01	1.29032E-01

FES		8	9	10	11	12	13	14
1e3	1st	2.04495E+01	5.53821E+01	6.69360E+01	9.61917E+00	2.57417E+04	3.15227E+01	3.88626E+00
	7th	2.06269E+01	6.48798E+01	7.50317E+01	1.08404E+01	3.25720E+04	1.46080E+02	4.22190E+00
	13th	2.07588E+01	6.97970E+01	8.53185E+01	1.12605E+01	4.50141E+04	2.65613E+02	4.34066E+00
	19th	2.08260E+01	7.71770E+01	8.94315E+01	1.19421E+01	4.98899E+04	4.76715E+02	4.43381E+00
	25th	2.09419E+01	9.11886E+01	1.03635E+02	1.26457E+01	7.60014E+04	1.34741E+03	4.51309E+00
	Mean	**2.07317E+01**	**7.08314E+01**	**8.34598E+01**	**1.13093E+01**	**4.47763E+04**	**3.48797E+02**	**4.28668E+00**
	Std D.	1.37932E-01	9.09859E+00	1.08567E+01	8.33069E-01	1.38980E+04	3.05521E+02	1.75099E-01
1e4	1st	2.03398E+01	3.19782E+01	4.66520E+01	8.09263E+00	5.23562E+03	7.29153E+00	3.79794E+00
	7th	2.04698E+01	4.31703E+01	5.06742E+01	9.90019E+00	1.40645E+04	1.06259E+01	3.91413E+00
	13th	2.05270E+01	5.12512E+01	5.64420E+01	1.01939E+01	1.83108E+04	1.30385E+01	3.99351E+00
	19th	2.05660E+01	5.57051E+01	5.96499E+01	1.04326E+01	2.19604E+04	1.81350E+01	4.02913E+00
	25th	2.06885E+01	5.92326E+01	7.49138E+01	1.10542E+01	2.98589E+04	3.44424E+01	4.16541E+00
	Mean	**2.05206E+01**	**4.93295E+01**	**5.67747E+01**	**1.00724E+01**	**1.79268E+04**	**1.45014E+01**	**3.98123E+00**
	Std D.	8.29416E-02	8.02532E+00	6.63481E+00	6.71169E-01	6.29392E+03	6.11292E+00	9.40904E-02
1e5	1st	2.01574E+01	0.00000E+00	5.28933E+00	2.56900E+00	9.48824E-01	3.89080E-01	1.57958E+00
	7th	2.02826E+01	9.94959E-01	1.25393E+01	3.60895E+00	1.04495E+01	5.06936E-01	2.61841E+00
	13th	2.03568E+01	1.98992E+00	1.58296E+01	4.63681E+00	2.02180E+01	6.38670E-01	3.12565E+00
	19th	2.04105E+01	2.98488E+00	1.88197E+01	5.43349E+00	3.96839E+01	7.38544E-01	3.29433E+00
	25th	2.04617E+01	3.97990E+00	3.25179E+01	8.49518E+00	7.12255E+02	1.22706E+00	3.62920E+00
	Mean	**2.03441E+01**	**1.99013E+00**	**1.64690E+01**	**4.62658E+00**	**8.47897E+01**	**6.62293E-01**	**2.96542E+00**
	Std D.	8.13829E-02	1.21875E+00	7.04173E+00	1.45265E+00	1.65819E+02	2.06914E-01	5.20650E-01

Table A.2. : Error Values Achieved at 1e3 FES, 1e4 FES, 1e5 FES and at Termination for PSO Algorithm for D=30

FES		1	2	3	4	5	6	7
1e3	1st	3.26609E+04	3.72956E+04	3.61002E+08	7.16017E+04	1.61519E+04	7.46561E+09	3.67427E+02
	7th	4.00946E+04	6.95113E+04	6.72351E+08	8.45589E+04	1.97316E+04	1.40431E+10	4.55083E+02
	13th	4.34139E+04	7.34467E+04	8.66443E+08	9.35033E+04	2.18862E+04	1.67137E+10	5.03242E+02
	19th	4.63498E+04	8.61529E+04	1.06733E+09	1.07016E+05	2.49962E+04	2.16686E+10	5.71819E+02
	25th	5.05507E+04	1.03745E+05	1.37145E+09	1.19937E+05	3.02311E+04	2.61949E+10	6.47344E+02
	Mean	**4.27186E+04**	**7.73655E+04**	**8.70683E+08**	**9.44374E+04**	**2.21673E+04**	**1.73596E+10**	**5.05097E+02**
	Std D.	4.72393E+03	1.37158E+04	2.51636E+08	1.37182E+04	3.31320E+03	5.28564E+09	8.17832E+01
1e4	1st	1.81602E+04	3.26952E+04	2.62316E+08	4.28805E+04	7.64483E+03	2.60587E+09	1.36284E+02
	7th	2.65052E+04	4.64983E+04	3.76459E+08	5.12360E+04	1.21514E+04	5.02081E+09	1.68826E+02
	13th	2.82050E+04	5.15790E+04	4.16373E+08	5.91561E+04	1.39636E+04	5.85878E+09	2.01139E+02
	19th	2.98952E+04	5.45213E+04	4.86897E+08	6.22452E+04	1.76609E+04	6.96080E+09	2.18251E+02
	25th	3.44228E+04	6.57085E+04	6.47755E+08	7.70740E+04	1.89739E+04	8.14435E+09	2.59205E+02
	Mean	**2.76529E+04**	**5.06327E+04**	**4.26770E+08**	**5.83211E+04**	**1.43102E+04**	**5.75436E+09**	**1.96106E+02**
	Std D.	3.98255E+03	8.03226E+03	9.10987E+07	8.75533E+03	3.26674E+03	1.37951E+09	3.41709E+01
1e5	1st	5.12188E+03	1.77790E+04	8.31522E+07	2.02395E+04	6.57582E+02	2.99734E+08	1.51934E+01
	7th	8.58031E+03	2.00173E+04	1.27527E+08	2.39511E+04	9.19801E+03	6.54437E+08	2.21534E+01
	13th	9.60244E+03	2.07185E+04	1.53377E+08	2.70384E+04	1.15765E+04	8.46761E+08	2.77779E+01
	19th	1.01966E+04	2.36774E+04	1.88294E+08	3.13016E+04	1.41361E+04	9.25307E+08	3.19367E+01
	25th	1.32968E+04	3.42341E+04	2.50549E+08	3.76261E+04	1.82480E+04	1.40014E+09	4.45827E+01
	Mean	**9.26168E+03**	**2.24131E+04**	**1.58325E+08**	**2.73436E+04**	**1.18024E+04**	**7.96579E+08**	**2.80197E+01**
	Std D.	1.80831E+03	4.12476E+03	4.36007E+07	5.06482E+03	4.07392E+03	2.58387E+08	7.61752E+00
Trm	1st	0.00000E+00	9.19895E+01	2.14739E+06	8.49725E+02	1.80210E+01	1.28023E+01	4.92000E-06
	7th	0.00000E+00	1.44785E+02	3.51459E+06	1.54421E+03	8.76067E+03	2.84509E+01	1.03743E-02
	13th	0.00000E+00	2.84253E+02	5.23473E+06	2.65265E+03	1.08879E+04	9.26050E+01	4.04980E-02
	19th	0.00000E+00	5.35763E+02	1.76596E+07	3.59317E+03	1.35322E+04	1.63450E+02	1.50584E-01
	25th	1.90000E-07	1.37277E+03	3.46473E+07	6.77614E+03	1.80297E+04	1.08048E+03	3.08387E-01
	Mean	**8.00000E-09**	**4.01956E+02**	**1.07647E+07**	**2.70271E+03**	**1.10620E+04**	**1.57711E+02**	**8.19353E-02**
	Std D.	3.79693E-08	3.40265E+02	9.33441E+06	1.48463E+03	3.83930E+03	2.13877E+02	8.93249E-02

FES		8	9	10	11	12	13	14
1e3	1st	2.10547E+01	3.26795E+02	3.69252E+02	4.18514E+01	7.10021E+05	3.02854E+04	1.39278E+01
	7th	2.11412E+01	3.58184E+02	4.09542E+02	4.48355E+01	1.25821E+06	6.51057E+04	1.40790E+01
	13th	2.11902E+01	3.75054E+02	4.34987E+02	4.54701E+01	1.33873E+06	8.17115E+04	1.41698E+01
	19th	2.12538E+01	3.85413E+02	4.52458E+02	4.66307E+01	1.50883E+06	1.38745E+05	1.42488E+01
	25th	2.13254E+01	4.21830E+02	4.71608E+02	4.71774E+01	1.73783E+06	2.76638E+05	1.43932E+01
	Mean	**2.11965E+01**	**3.72422E+02**	**4.30582E+02**	**4.54119E+01**	**1.34176E+06**	**1.11227E+05**	**1.41733E+01**
	Std D.	6.86168E-02	2.35558E+01	2.72296E+01	1.42015E+00	2.26488E+05	6.48907E+04	1.23281E-01
1e4	1st	2.10274E+01	2.82904E+02	3.15219E+02	4.03378E+01	7.03553E+05	5.98967E+03	1.35924E+01
	7th	2.10897E+01	3.00134E+02	3.43194E+02	4.17414E+01	8.36977E+05	1.53346E+04	1.38242E+01
	13th	2.11288E+01	3.16519E+02	3.66039E+02	4.33941E+01	9.16778E+05	1.89644E+04	1.39211E+01
	19th	2.11559E+01	3.28077E+02	3.80618E+02	4.42463E+01	9.79474E+05	2.41404E+04	1.39874E+01
	25th	2.11869E+01	3.55479E+02	4.00969E+02	4.46045E+01	1.10252E+06	3.86288E+04	1.40973E+01
	Mean	**2.11206E+01**	**3.15133E+02**	**3.61031E+02**	**4.28959E+01**	**9.10468E+05**	**1.98403E+04**	**1.38910E+01**
	Std D.	4.59126E-02	1.90067E+01	2.35740E+01	1.41244E+00	1.17351E+05	6.96427E+03	1.27295E-01
1e5	1st	2.08904E+01	1.90559E+02	1.97211E+02	3.51733E+01	2.40193E+05	3.81373E+01	1.30270E+01
	7th	2.09677E+01	2.05704E+02	2.37464E+02	3.98904E+01	3.20629E+05	7.69832E+01	1.33430E+01
	13th	2.10060E+01	2.13758E+02	2.63121E+02	4.03886E+01	3.86757E+05	1.07775E+02	1.34803E+01
	19th	2.10232E+01	2.32828E+02	2.82439E+02	4.09082E+01	4.79624E+05	1.59699E+02	1.36382E+01
	25th	2.10996E+01	2.66556E+02	2.99724E+02	4.20882E+01	6.41161E+05	3.96238E+02	1.38246E+01
	Mean	**2.09969E+01**	**2.20419E+02**	**2.56298E+02**	**4.01637E+01**	**4.12067E+05**	**1.26483E+02**	**1.34869E+01**
	Std D.	4.72351E-02	2.17877E+01	2.98414E+01	1.61163E+00	1.16896E+05	7.77277E+01	1.95144E-01
Trm	1st	2.07623E+01	1.39294E+01	4.46290E+01	2.10586E+01	2.21992E+03	2.12003E+00	1.24395E+01
	7th	2.08980E+01	1.98992E+01	6.76721E+01	2.74924E+01	9.09589E+03	3.06190E+00	1.29290E+01
	13th	2.09448E+01	2.48740E+01	7.70619E+01	3.04811E+01	1.75403E+04	3.71451E+00	1.30828E+01
	19th	2.09789E+01	2.88538E+01	9.74682E+01	3.50665E+01	3.40146E+04	4.08763E+00	1.32259E+01
	25th	2.10350E+01	3.28336E+01	2.44440E+02	3.94895E+01	6.06273E+04	6.58410E+00	1.34684E+01
	Mean	**2.09311E+01**	**2.43572E+01**	**8.72283E+01**	**3.11684E+01**	**2.21836E+04**	**3.70427E+00**	**1.30484E+01**
	Std D.	6.64438E-02	5.51904E+00	3.91043E+01	5.19378E+00	1.60610E+04	9.19095E-01	2.44407E-01

Table A.3. : Error Values Achieved at 1e3 FES, 1e4 FES, 1e5 FES and at Termination for PSO Algorithm for D=50

FES		1	2	3	4	5	6	7
1e3	1st	7.99289E+04	1.50482E+05	1.52688E+09	1.36997E+05	3.10210E+04	3.56895E+10	1.49758E+03
	7th	9.58299E+04	2.11843E+05	2.52811E+09	2.22054E+05	3.83738E+04	4.84342E+10	1.66213E+03
	13th	1.02728E+05	2.29513E+05	2.79278E+09	2.49396E+05	4.09088E+04	6.23082E+10	1.76736E+03
	19th	1.13268E+05	2.46106E+05	3.69842E+09	2.77223E+05	4.40263E+04	6.95955E+10	1.90823E+03
	25th	1.24383E+05	2.71331E+05	4.49212E+09	3.68676E+05	5.05454E+04	9.19626E+10	2.12386E+03
	Mean	**1.03584E+05**	**2.29645E+05**	**2.93190E+09**	**2.46886E+05**	**4.12608E+04**	**6.00769E+10**	**1.79297E+03**
	Std D.	1.17906E+04	2.71755E+04	7.35651E+08	4.96471E+04	4.34388E+03	1.41053E+10	1.68187E+02
1e4	1st	5.32826E+04	1.04767E+05	1.12401E+09	1.13852E+05	2.31834E+04	1.94511E+10	4.71562E+02
	7th	6.99214E+04	1.35100E+05	1.32787E+09	1.44872E+05	2.57065E+04	2.37890E+10	6.99388E+02
	13th	7.53881E+04	1.45078E+05	1.91380E+09	1.63151E+05	2.69704E+04	2.75951E+10	7.53584E+02
	19th	7.92484E+04	1.63923E+05	1.96641E+09	1.72928E+05	2.91061E+04	3.01735E+10	8.03569E+02
	25th	9.05838E+04	1.89837E+05	2.36448E+09	1.99026E+05	3.28848E+04	3.77934E+10	8.77140E+02
	Mean	**7.41496E+04**	**1.48924E+05**	**1.71664E+09**	**1.58780E+05**	**2.75222E+04**	**2.75653E+10**	**7.31137E+02**
	Std D.	7.86464E+03	2.16255E+04	3.75210E+08	2.12488E+04	2.71337E+03	5.16234E+09	1.08171E+02
1e5	1st	3.10871E+04	5.31299E+04	5.55355E+08	7.43091E+04	1.94062E+04	4.64346E+09	1.95477E+02
	7th	4.05697E+04	8.25054E+04	8.70376E+08	9.39969E+04	2.07428E+04	7.36180E+09	2.37282E+02
	13th	4.33032E+04	9.40284E+04	9.47788E+08	1.04543E+05	2.17470E+04	9.04666E+09	2.67250E+02
	19th	4.95761E+04	1.00028E+05	1.06812E+09	1.18148E+05	2.42312E+04	1.05491E+10	3.15088E+02
	25th	5.27993E+04	1.17661E+05	1.21389E+09	1.30751E+05	3.00622E+04	1.48500E+10	3.53566E+02
	Mean	**4.42244E+04**	**9.32410E+04**	**9.60764E+08**	**1.05018E+05**	**2.28238E+04**	**9.21952E+09**	**2.76141E+02**
	Std D.	6.35030E+03	1.39237E+04	1.52366E+08	1.60333E+04	2.78956E+03	2.50171E+09	4.69342E+01
Trm	1st	2.00000E-08	2.73281E+03	7.18257E+06	1.01778E+04	1.25707E+04	4.24331E+01	1.02000E-03
	7th	6.80000E-07	4.12783E+03	2.71240E+07	1.82279E+04	1.71181E+04	1.09911E+02	6.85393E-03
	13th	3.72000E-06	5.65296E+03	3.66474E+07	2.25180E+04	1.95252E+04	1.71295E+02	1.99886E-02
	19th	1.92900E-05	7.55080E+03	6.29808E+07	2.79352E+04	2.21800E+04	3.04782E+02	3.07924E-02
	25th	2.81544E+00	1.50197E+04	1.21738E+08	3.50883E+04	2.88438E+04	1.36723E+03	1.05730E-01
	Mean	**1.16368E-01**	**6.53038E+03**	**4.84217E+07**	**2.27204E+04**	**1.98764E+04**	**3.03737E+02**	**2.47858E-02**
	Std D.	5.62614E-01	3.35448E+03	3.11780E+07	7.01268E+03	3.63886E+03	3.42479E+02	2.36381E-02

FES		8	9	10	11	12	13	14
1e3	1st	2.12328E+01	6.39437E+02	7.17584E+02	7.79921E+01	4.31863E+06	2.78433E+05	2.36851E+01
	7th	2.13198E+01	7.34444E+02	8.01193E+02	7.99720E+01	5.48904E+06	5.22436E+05	2.39080E+01
	13th	2.13472E+01	7.52918E+02	8.23799E+02	8.18222E+01	6.22901E+06	7.52823E+05	2.40274E+01
	19th	2.13757E+01	7.68611E+02	8.48158E+02	8.25800E+01	6.86861E+06	9.04863E+05	2.41006E+01
	25th	2.14093E+01	8.27755E+02	9.09537E+02	8.40386E+01	7.41755E+06	1.27489E+06	2.41817E+01
	Mean	**2.13445E+01**	**7.46962E+02**	**8.21172E+02**	**8.13983E+01**	**6.12130E+06**	**7.17402E+05**	**2.39839E+01**
	Std D.	4.50871E-02	4.38789E+01	4.38956E+01	1.71280E+00	8.43808E+05	2.57650E+05	1.49434E-01
1e4	1st	2.11240E+01	5.18483E+02	6.33732E+02	7.34549E+01	3.47085E+06	9.38962E+04	2.32017E+01
	7th	2.12328E+01	6.25779E+02	7.02869E+02	7.69962E+01	4.19723E+06	1.62513E+05	2.35572E+01
	13th	2.12615E+01	6.45175E+02	7.25085E+02	7.77338E+01	4.63587E+06	2.15659E+05	2.36576E+01
	19th	2.12898E+01	6.67905E+02	7.35191E+02	7.88030E+01	4.87297E+06	2.56635E+05	2.37189E+01
	25th	2.13195E+01	7.24445E+02	7.99969E+02	8.06125E+01	5.64425E+06	4.25250E+05	2.38053E+01
	Mean	**2.12529E+01**	**6.43071E+02**	**7.17591E+02**	**7.77269E+01**	**4.55114E+06**	**2.19146E+05**	**2.36111E+01**
	Std D.	4.81156E-02	3.97923E+01	3.61348E+01	1.46475E+00	5.64677E+05	7.48052E+04	1.51861E-01
1e5	1st	2.11240E+01	4.67534E+02	5.38989E+02	7.25566E+01	2.02089E+06	4.44730E+03	2.30449E+01
	7th	2.11681E+01	5.29000E+02	5.88400E+02	7.38230E+01	2.87356E+06	1.25456E+04	2.32017E+01
	13th	2.11910E+01	5.48867E+02	5.99826E+02	7.47650E+01	3.07729E+06	1.78642E+04	2.33826E+01
	19th	2.12065E+01	5.69679E+02	6.12452E+02	7.58665E+01	3.38339E+06	2.96162E+04	2.34807E+01
	25th	2.12357E+01	6.03010E+02	6.50944E+02	7.78157E+01	3.79779E+06	5.50305E+04	2.35869E+01
	Mean	**2.11848E+01**	**5.44062E+02**	**5.95392E+02**	**7.48973E+01**	**3.07672E+06**	**2.20874E+04**	**2.33599E+01**
	Std D.	2.95916E-02	3.38632E+01	2.77647E+01	1.39268E+00	4.18751E+05	1.19801E+04	1.61707E-01
Trm	1st	2.10415E+01	4.57681E+01	1.01273E+02	5.32597E+01	1.80026E+04	5.33924E+00	2.20372E+01
	7th	2.11087E+01	5.87025E+01	1.39716E+02	6.31906E+01	6.22675E+04	7.66812E+00	2.26008E+01
	13th	2.11392E+01	6.96471E+01	1.84071E+02	6.70882E+01	1.02842E+05	9.28354E+00	2.27769E+01
	19th	2.11633E+01	7.26319E+01	2.89274E+02	7.08982E+01	1.39305E+05	1.16804E+01	2.28943E+01
	25th	2.11908E+01	8.75563E+01	4.04977E+02	7.45000E+01	2.27342E+05	1.55820E+01	2.32639E+01
	Mean	**2.11326E+01**	**6.62258E+01**	**2.14250E+02**	**6.61378E+01**	**1.08151E+05**	**9.64705E+00**	**2.27563E+01**
	Std D.	3.99925E-02	1.11524E+01	9.29100E+01	5.93583E+00	5.76638E+04	2.69821E+00	2.80591E-01

APPENDIX B

Table B.1. : Error Values Achieved at 1e3 FES, 1e4 FES, 1e5 FES, and at Termination for the DE Algorithm for D=10

FES		1	2	3	4	5	6	7
1e3	1st	1.85424E+03	3.01951E+03	1.90610E+07	5.26307E+03	1.15819E+03	5.02058E+07	5.58453E+01
	7th	3.59667E+03	5.04843E+03	3.02341E+07	7.50755E+03	1.72110E+03	1.11428E+08	1.04098E+02
	13th	4.29586E+03	6.09160E+03	4.51607E+07	8.48740E+03	2.26924E+03	1.93328E+08	1.16678E+02
	19th	5.00820E+03	7.77914E+03	6.24066E+07	9.40238E+03	2.83959E+03	2.92390E+08	1.47886E+02
	25th	6.60096E+03	9.57528E+03	9.00851E+07	1.35384E+04	5.75893E+03	5.35614E+08	2.48523E+02
	Mean	**4.23316E+03**	**6.29732E+03**	**4.63044E+07**	**8.64123E+03**	**2.44959E+03**	**2.18675E+08**	**1.23055E+02**
	Std D.	1.25635E+03	1.73963E+03	1.95728E+07	1.90330E+03	1.09490E+03	1.28321E+08	4.04756E+01
1e4	1st	3.09619E+00	6.46120E+01	1.13580E+05	1.34154E+02	3.03966E-03	3.28694E+03	1.04623E+00
	7th	1.44383E+01	1.26493E+02	6.82542E+05	1.83790E+02	8.20477E-03	7.95177E+03	1.24116E+00
	13th	1.87169E+01	1.86524E+02	9.17870E+05	2.58114E+02	1.61937E-02	1.27184E+04	1.39226E+00
	19th	2.16114E+01	2.21809E+02	1.10022E+06	3.61044E+02	2.02880E-02	2.18604E+04	1.46352E+00
	25th	3.38957E+01	6.97320E+02	2.52789E+06	4.37839E+02	1.05464E-01	4.11938E+04	1.77393E+00
	Mean	**1.88167E+01**	**1.94335E+02**	**1.00105E+06**	**2.67162E+02**	**1.88126E-02**	**1.55603E+04**	**1.37673E+00**
	Std D.	6.52482E+00	1.16798E+02	5.43200E+05	9.98847E+01	1.98726E-02	1.01946E+04	1.74155E-01
1e5	1st	0.00000E+00	0.00000E+00	4.10000E-07	0.00000E+00	0.00000E+00	0.00000E+00	5.08014E-02
	7th	0.00000E+00	0.00000E+00	2.02000E-06	0.00000E+00	0.00000E+00	0.00000E+00	1.03748E-01
	13th	0.00000E+00	0.00000E+00	7.09000E-06	0.00000E+00	0.00000E+00	0.00000E+00	1.50057E-01
	19th	0.00000E+00	0.00000E+00	6.23900E-05	0.00000E+00	0.00000E+00	0.00000E+00	2.07027E-01
	25th	0.00000E+00	0.00000E+00	3.83810E-04	0.00000E+00	0.00000E+00	3.98658E+00	2.58674E-01
	Mean	**0.00000E+00**	**0.00000E+00**	**5.46328E-05**	**0.00000E+00**	**0.00000E+00**	**6.37853E-01**	**1.50829E-01**
	Std D.	0.00000E+00	0.00000E+00	1.01191E-04	0.00000E+00	0.00000E+00	1.49164E+00	6.40648E-02

FES		8	9	10	11	12	13	14
1e3	1st	2.04153E+01	6.17860E+01	6.21051E+01	9.68019E+00	1.57966E+04	7.07425E+01	3.97909E+00
	7th	2.06656E+01	7.06628E+01	8.11575E+01	1.14143E+01	4.72954E+04	3.73213E+02	4.31508E+00
	13th	2.07916E+01	7.49743E+01	9.09541E+01	1.16786E+01	5.14369E+04	6.16887E+02	4.37866E+00
	19th	2.08319E+01	8.13895E+01	9.64874E+01	1.22943E+01	6.01442E+04	1.20720E+03	4.41749E+00
	25th	2.09123E+01	8.93128E+01	1.02935E+02	1.30496E+01	8.13749E+04	2.72287E+03	4.51682E+00
	Mean	**2.07451E+01**	**7.57054E+01**	**8.74951E+01**	**1.16931E+01**	**5.28403E+04**	**9.12928E+02**	**4.35079E+00**
	Std D.	1.31186E-01	7.77410E+00	1.07634E+01	8.09603E-01	1.44527E+04	6.91357E+02	1.18453E-01
1e4	1st	2.03216E+01	2.02641E+01	3.34415E+01	7.73432E+00	7.49916E+02	2.32086E+00	3.58983E+00
	7th	2.05162E+01	3.29482E+01	4.42874E+01	9.11807E+00	1.42068E+03	3.89395E+00	3.93832E+00
	13th	2.05686E+01	3.74767E+01	5.22460E+01	9.96183E+00	2.23430E+03	5.13630E+00	4.03647E+00
	19th	2.05919E+01	4.16618E+01	5.45106E+01	1.03578E+01	3.35670E+03	5.86467E+00	4.08288E+00
	25th	2.06642E+01	4.52072E+01	6.31435E+01	1.11138E+01	8.52400E+03	7.46224E+00	4.32129E+00
	Mean	**2.05474E+01**	**3.58943E+01**	**5.00818E+01**	**9.78996E+00**	**2.91231E+03**	**4.99484E+00**	**3.99999E+00**
	Std D.	7.82286E-02	7.37228E+00	7.99405E+00	8.82700E-01	2.08499E+03	1.35427E+00	1.82657E-01
1e5	1st	2.01870E+01	1.98992E+00	7.79445E+00	7.93340E-04	0.00000E+00	3.81196E-01	1.00236E+00
	7th	2.02973E+01	2.98488E+00	1.14447E+01	2.43548E-03	0.00000E+00	5.33667E-01	3.45017E+00
	13th	2.03631E+01	4.97480E+00	1.52435E+01	5.85140E-03	2.00000E-08	7.16250E-01	3.58983E+00
	19th	2.03872E+01	6.96471E+00	1.89897E+01	1.50192E+00	1.00030E+01	9.19178E-01	3.75966E+00
	25th	2.05102E+01	8.95463E+00	3.63416E+01	3.00221E+00	1.34735E+03	1.56296E+00	3.92519E+00
	Mean	**2.03384E+01**	**5.09419E+00**	**1.65177E+01**	**7.09222E-01**	**5.89551E+01**	**7.66957E-01**	**3.44692E+00**
	Std D.	7.50173E-02	2.09729E+00	6.98306E+00	9.72805E-01	2.68501E+02	3.34237E-01	5.73178E-01

Table B.2. : Error Values Achieved at 1e3 FES, 1e4 FES, 1e5 FES, and at Termination for the DE Algorithm for D=30

FES		1	2	3	4	5	6	7
	1st	3.88765E+04	6.34975E+04	4.19986E+08	6.74118E+04	1.84970E+04	1.13668E+10	1.09900E+03
	7th	4.97149E+04	7.33956E+04	8.27105E+08	8.60348E+04	2.21715E+04	1.87084E+10	1.34025E+03
	13th	5.45561E+04	8.31323E+04	1.04649E+09	1.03477E+05	2.39906E+04	2.26591E+10	1.44405E+03
1e3	19th	6.04220E+04	9.42252E+04	1.20097E+09	1.13990E+05	2.56209E+04	3.12496E+10	1.57084E+03
	25th	6.54650E+04	1.22545E+05	1.34340E+09	1.48995E+05	2.85592E+04	4.41784E+10	1.90925E+03
	Mean	**5.44088E+04**	**8.59290E+04**	**1.00926E+09**	**9.98429E+04**	**2.39570E+04**	**2.60007E+10**	**1.45692E+03**
	Std D.	7.20866E+03	1.74239E+04	2.39042E+08	1.86456E+04	2.63151E+03	9.42864E+09	1.95042E+02
	1st	7.37041E+03	2.50505E+04	1.43161E+08	3.43311E+04	7.59217E+03	8.00479E+08	1.91022E+02
	7th	1.17346E+04	4.66858E+04	2.87888E+08	4.82081E+04	9.70474E+03	1.21021E+09	3.46103E+02
	13th	1.28905E+04	5.00936E+04	3.13126E+08	5.11596E+04	1.02530E+04	1.49848E+09	4.21542E+02
1e4	19th	1.47663E+04	5.40392E+04	3.84733E+08	5.52670E+04	1.15160E+04	1.78987E+09	5.45858E+02
	25th	1.81025E+04	6.45800E+04	5.20868E+08	6.71006E+04	1.31915E+04	4.59954E+09	7.43594E+02
	Mean	**1.27943E+04**	**4.91997E+04**	**3.26878E+08**	**5.08241E+04**	**1.04777E+04**	**1.65636E+09**	**4.44259E+02**
	Std D.	2.62879E+03	9.25481E+03	8.82836E+07	8.59486E+03	1.34589E+03	8.34416E+08	1.41394E+02
	1st	8.83870E-04	1.75557E+02	1.84217E+06	3.61277E+02	1.33228E+01	3.78990E+01	1.85172E-01
	7th	5.89966E-03	3.60806E+02	3.24895E+06	9.14246E+02	7.03626E+02	6.80512E+01	3.63210E-01
	13th	9.70001E-03	4.29233E+02	4.18139E+06	1.27675E+03	2.82633E+03	1.23097E+02	5.09019E-01
1e5	19th	1.15735E-02	6.19901E+02	5.42530E+06	1.85842E+03	3.25818E+03	1.99468E+02	7.65749E-01
	25th	2.69409E-02	8.46557E+02	9.51084E+06	3.56928E+03	6.61989E+03	1.62802E+03	9.68942E-01
	Mean	**9.85310E-03**	**4.82823E+02**	**4.64793E+06**	**1.45545E+03**	**2.56002E+03**	**2.31275E+02**	**5.40389E-01**
	Std D.	6.55476E-03	1.89156E+02	2.02291E+06	8.24726E+02	1.93257E+03	3.50876E+02	2.32872E-01
	1st	0.00000E+00	1.00082E-02	2.52866E+05	4.44201E-01	1.08400E-05	7.91052E-01	0.00000E+00
	7th	0.00000E+00	2.69799E-02	3.63513E+05	1.56985E+00	9.65160E-03	6.02897E+00	0.00000E+00
	13th	0.00000E+00	4.70619E-02	6.71938E+05	2.26303E+00	2.19951E+03	1.01827E+01	1.00000E-08
Trm	19th	0.00000E+00	8.74604E-02	9.74700E+05	4.37445E+00	2.91858E+03	1.26779E+01	5.39371E-02
	25th	0.00000E+00	1.58316E-01	1.45497E+06	1.58945E+01	5.93939E+03	1.80863E+01	2.89677E-01
	Mean	**0.00000E+00**	**6.18957E-02**	**7.34622E+05**	**4.06987E+00**	**1.89052E+03**	**9.45834E+00**	**4.16067E-02**
	Std D.	0.00000E+00	4.21461E-02	3.83248E+05	4.20637E+00	1.88807E+03	5.37511E+00	6.68922E-02

FES		8	9	10	11	12	13	14
1e3	1st	2.10371E+01	3.51505E+02	3.86808E+02	4.30150E+01	1.21493E+06	7.96154E+04	1.36177E+01
	7th	2.11788E+01	3.81350E+02	4.53468E+02	4.48355E+01	1.35209E+06	1.57959E+05	1.41172E+01
	13th	2.12375E+01	4.08661E+02	4.74526E+02	4.54247E+01	1.49368E+06	2.18095E+05	1.41871E+01
	19th	2.12775E+01	4.20831E+02	4.93344E+02	4.61805E+01	1.66345E+06	3.04821E+05	1.42367E+01
	25th	2.13002E+01	4.74661E+02	5.22231E+02	4.80001E+01	2.02808E+06	4.92661E+05	1.44037E+01
	Mean	**2.12195E+01**	**4.04524E+02**	**4.69685E+02**	**4.55424E+01**	**1.51555E+06**	**2.28311E+05**	**1.41610E+01**
	Std D.	6.80106E-02	2.94053E+01	3.16319E+01	1.28294E+00	2.01895E+05	1.06164E+05	1.48381E-01
1e4	1st	2.09520E+01	2.33280E+02	2.84955E+02	4.06643E+01	4.32073E+05	1.46393E+03	1.36177E+01
	7th	2.10683E+01	2.59949E+02	3.18478E+02	4.25440E+01	6.48983E+05	2.67125E+03	1.38431E+01
	13th	2.11071E+01	2.80316E+02	3.42835E+02	4.30718E+01	6.71627E+05	3.83519E+03	1.39494E+01
	19th	2.11398E+01	2.91904E+02	3.51147E+02	4.35359E+01	7.95226E+05	5.17356E+03	1.40068E+01
	25th	2.11875E+01	3.13804E+02	3.81147E+02	4.43857E+01	1.05816E+06	7.88898E+03	1.41080E+01
	Mean	**2.10956E+01**	**2.76174E+02**	**3.34933E+02**	**4.28564E+01**	**7.10077E+05**	**4.00502E+03**	**1.39081E+01**
	Std D.	5.95784E-02	2.17571E+01	2.61489E+01	9.64590E-01	1.36521E+05	1.86825E+03	1.32341E-01
1e5	1st	2.08070E+01	2.19159E+01	3.00719E+01	1.75625E+01	0.00000E+00	2.44370E+00	1.30431E+01
	7th	2.09824E+01	3.28623E+01	6.86397E+01	3.83301E+01	0.00000E+00	4.40461E+00	1.35254E+01
	13th	2.10002E+01	4.09024E+01	1.14043E+02	3.96573E+01	0.00000E+00	5.26838E+00	1.36172E+01
	19th	2.10286E+01	4.77876E+01	1.95522E+02	4.06290E+01	0.00000E+00	9.04498E+00	1.36717E+01
	25th	2.10878E+01	6.07723E+01	2.28969E+02	4.27992E+01	0.00000E+00	1.55170E+01	1.37656E+01
	Mean	**2.09986E+01**	**4.09120E+01**	**1.26306E+02**	**3.82544E+01**	**0.00000E+00**	**6.64955E+00**	**1.35707E+01**
	Std D.	5.40783E-02	1.12045E+01	6.52610E+01	5.23822E+00	0.00000E+00	3.19829E+00	1.60776E-01
Trm	1st	2.08070E+01	2.18891E+01	2.34153E+01	5.37358E+00	0.00000E+00	1.97830E+00	1.29578E+01
	7th	2.09520E+01	3.28336E+01	4.49232E+01	8.52891E+00	0.00000E+00	2.68041E+00	1.32526E+01
	13th	2.09705E+01	4.07933E+01	5.00901E+01	1.09034E+01	0.00000E+00	2.95265E+00	1.33953E+01
	19th	2.09984E+01	4.77579E+01	5.49342E+01	1.56893E+01	0.00000E+00	4.04031E+00	1.35254E+01
	25th	2.10302E+01	6.06924E+01	7.95865E+01	1.87345E+01	0.00000E+00	5.77030E+00	1.36599E+01
	Mean	**2.09600E+01**	**4.08729E+01**	**5.02670E+01**	**1.17990E+01**	**0.00000E+00**	**3.49835E+00**	**1.33659E+01**
	Std D.	5.14799E-02	1.12049E+01	1.33351E+01	4.16057E+00	0.00000E+00	1.17383E+00	1.99834E-01

Table B.3. : Error Values Achieved at 1e3 FES, 1e4 FES, 1e5 FES and at Termination for the DE Algorithm for D=50

FES		1	2	3	4	5	6	7
1e3	1st	1.09919E+05	1.58505E+05	2.07215E+09	2.01008E+05	3.60708E+04	6.24047E+10	2.70670E+03
	7th	1.24213E+05	2.15679E+05	3.34100E+09	2.49303E+05	4.19097E+04	7.05402E+10	3.24870E+03
	13th	1.32942E+05	2.51152E+05	4.15515E+09	2.69409E+05	4.37841E+04	7.54364E+10	3.42658E+03
	19th	1.42118E+05	2.72774E+05	4.62201E+09	3.15076E+05	4.60202E+04	9.19463E+10	3.55218E+03
	25th	1.55557E+05	3.38458E+05	5.17139E+09	3.67871E+05	4.96220E+04	1.23848E+11	3.87867E+03
	Mean	**1.32369E+05**	**2.47597E+05**	**4.02174E+09**	**2.80243E+05**	**4.39236E+04**	**8.17463E+10**	**3.40663E+03**
	Std D.	1.30691E+04	4.56185E+04	8.47471E+08	4.69922E+04	3.58509E+03	1.70617E+10	2.88949E+02
1e4	1st	3.98083E+04	1.08123E+05	7.87258E+08	1.24568E+05	2.00128E+04	6.18604E+09	1.06680E+03
	7th	4.43059E+04	1.36973E+05	1.07956E+09	1.53487E+05	2.50087E+04	9.91087E+09	1.34262E+03
	13th	4.65329E+04	1.49204E+05	1.17726E+09	1.61514E+05	2.60129E+04	1.20288E+10	1.50405E+03
	19th	5.08941E+04	1.56631E+05	1.46263E+09	1.79026E+05	2.73082E+04	1.53635E+10	1.67263E+03
	25th	6.81645E+04	1.93490E+05	1.82079E+09	2.08217E+05	3.01435E+04	1.82119E+10	2.08144E+03
	Mean	**4.79236E+04**	**1.47232E+05**	**1.26728E+09**	**1.63019E+05**	**2.60249E+04**	**1.23105E+10**	**1.51560E+03**
	Std D.	6.01150E+03	2.03001E+04	2.65941E+08	1.99857E+04	2.39857E+03	3.46091E+09	2.61940E+02
1e5	1st	3.91216E+00	1.05025E+04	9.72750E+06	1.36500E+04	4.43186E+03	7.58733E+03	1.89023E+00
	7th	5.40717E+00	1.82658E+04	2.42321E+07	3.05233E+04	7.12124E+03	2.55062E+04	2.40129E+00
	13th	7.11906E+00	1.99615E+04	3.64855E+07	3.30409E+04	9.76172E+03	3.59316E+04	3.05294E+00
	19th	9.80515E+00	2.11416E+04	4.16241E+07	3.99110E+04	1.03183E+04	8.55643E+04	3.79685E+00
	25th	2.13350E+01	3.28744E+04	7.62817E+07	5.55904E+04	1.24092E+04	4.16202E+05	7.23378E+00
	Mean	**8.63916E+00**	**2.00253E+04**	**3.73887E+07**	**3.53287E+04**	**9.03571E+03**	**6.28620E+04**	**3.50589E+00**
	Std D.	4.72937E+00	4.71459E+03	1.68236E+07	9.48968E+03	2.21534E+03	8.09670E+04	1.51754E+00
Trm	1st	0.00000E+00	2.67056E+01	9.83717E+05	6.42150E+02	5.38846E+02	2.13773E+01	1.00000E-07
	7th	0.00000E+00	5.11244E+01	1.49749E+06	1.61209E+03	3.66471E+03	4.01439E+01	6.60000E-07
	13th	0.00000E+00	5.74726E+01	2.10192E+06	2.26646E+03	5.61799E+03	8.45228E+01	2.66000E-06
	19th	0.00000E+00	7.93419E+01	2.56510E+06	3.58110E+03	7.33205E+03	9.32595E+01	1.66393E-02
	25th	0.00000E+00	1.40927E+02	4.52991E+06	1.04788E+04	1.04086E+04	3.88045E+02	4.86660E-02
	Mean	**0.00000E+00**	**6.70521E+01**	**2.18206E+06**	**3.00509E+03**	**5.49715E+03**	**9.13380E+01**	**1.07342E-02**
	Std D.	0.00000E+00	2.92809E+01	9.15385E+05	2.18711E+03	2.49397E+03	8.49118E+01	1.58607E-02

FES		8	9	10	11	12	13	14
	1st	2.11750E+01	7.39051E+02	7.59304E+02	7.53534E+01	5.39738E+06	3.04674E+05	2.36851E+01
	7th	2.13133E+01	7.74119E+02	8.65020E+02	7.97841E+01	6.42235E+06	9.05402E+05	2.38686E+01
	13th	2.13442E+01	7.99986E+02	8.88168E+02	8.14681E+01	6.91974E+06	1.06373E+06	2.39823E+01
1e3	19th	2.13676E+01	8.31190E+02	9.14221E+02	8.26106E+01	7.23733E+06	1.34206E+06	2.40377E+01
	25th	2.13887E+01	8.72940E+02	9.52398E+02	8.52554E+01	8.34791E+06	1.82215E+06	2.41530E+01
	Mean	**2.13327E+01**	**8.03630E+02**	**8.81999E+02**	**8.12444E+01**	**6.86581E+06**	**1.07915E+06**	**2.39449E+01**
	Std D.	4.94163E-02	3.42247E+01	4.49513E+01	2.23697E+00	7.18680E+05	3.54193E+05	1.40011E-01
	1st	2.11554E+01	5.37857E+02	5.77085E+02	7.40360E+01	2.74711E+06	1.98473E+04	2.33465E+01
	7th	2.12198E+01	5.93322E+02	6.24283E+02	7.67951E+01	3.11405E+06	3.29731E+04	2.36235E+01
	13th	2.12656E+01	6.18468E+02	6.48086E+02	7.78537E+01	3.39804E+06	5.31764E+04	2.37075E+01
1e4	19th	2.12915E+01	6.26707E+02	6.60825E+02	7.85654E+01	3.69401E+06	7.25397E+04	2.37938E+01
	25th	2.13343E+01	6.50201E+02	7.11174E+02	7.98899E+01	4.19302E+06	1.11901E+05	2.39553E+01
	Mean	**2.12563E+01**	**6.09454E+02**	**6.47203E+02**	**7.76677E+01**	**3.41839E+06**	**5.45568E+04**	**2.36952E+01**
	Std D.	5.06526E-02	2.99158E+01	3.30762E+01	1.39661E+00	3.99081E+05	2.55809E+04	1.59301E-01
	1st	2.10799E+01	7.02114E+01	3.75240E+02	7.17453E+01	1.24834E+04	2.46534E+01	2.29297E+01
	7th	2.11601E+01	8.80808E+01	4.02222E+02	7.41917E+01	6.09260E+04	3.06321E+01	2.33554E+01
	13th	2.11858E+01	9.69369E+01	4.17646E+02	7.48307E+01	8.93236E+04	3.50081E+01	2.34080E+01
1e5	19th	2.12055E+01	1.16184E+02	4.35728E+02	7.62536E+01	1.30699E+05	3.75569E+01	2.35048E+01
	25th	2.12311E+01	1.46614E+02	4.84708E+02	7.72227E+01	1.99062E+05	4.37736E+01	2.36513E+01
	Mean	**2.11808E+01**	**1.01463E+02**	**4.20275E+02**	**7.50219E+01**	**9.48683E+04**	**3.44406E+01**	**2.33992E+01**
	Std D.	3.33359E-02	2.02831E+01	2.47438E+01	1.35699E+00	4.33802E+04	4.87466E+00	1.52068E-01
	1st	2.10276E+01	6.66622E+01	5.59457E+01	2.44855E+01	6.30067E+02	5.19514E+00	2.26743E+01
	7th	2.11233E+01	7.95966E+01	8.37425E+01	3.74256E+01	4.08555E+03	7.35488E+00	2.30478E+01
	13th	2.11432E+01	8.65613E+01	9.01580E+01	5.70049E+01	8.10495E+03	8.78431E+00	2.32035E+01
Trm	19th	2.11634E+01	9.65108E+01	9.74667E+01	7.11363E+01	1.60856E+04	1.15155E+01	2.32611E+01
	25th	2.11989E+01	1.42279E+02	1.33335E+02	7.44238E+01	3.08875E+04	1.59468E+01	2.34932E+01
	Mean	**2.11374E+01**	**9.27698E+01**	**9.14884E+01**	**5.32926E+01**	**1.06185E+04**	**9.58925E+00**	**2.31630E+01**
	Std D.	3.69487E-02	2.10559E+01	1.68417E+01	1.83731E+01	8.53809E+03	3.20361E+00	1.74935E-01