Toward Enhancing Metaheuristic Optimization Algorithms Using Center-based Sampling Strategies for Solving Single- and Multi- objective Large-scale Problems

Hanan Hiba

A thesis submitted in fulfillment for the degree of
Doctor of Philosophy in Electrical and Computer Engineering

in

Department of Electrical, Computer, and Software Engineering
Faculty of Engineering and Applied Sciences
University of Ontario Institute of Technology (Ontario Tech University)
Oshawa, Ontario, Canada
December, 2020

Thesis Examination Information

Submitted by: Hanan Hiba

Doctor of Philosophy in Electrical and Computer Engineering

Thesis title:
Toward Enhancing Metaheuristic Optimization Algorithms Using Center-based Sampling Strategies for Solving Single- and Multi- objective Large-scale Problems.

An oral defense of this thesis took place on December 8, 2020 in front of the following examining committee:

Examining Committee:

Chair of Examining Committee	Dr. Ying Wang
Research Supervisor	Prof. Shahryar Rahnamayan
Examining Committee Member	Dr. Masoud Makrehchi
Examining Committee Member	Dr. Ramiro Liscano
University Examiner	Dr. Sean Bohun
External Examiner	Dr. Ebrahim Bagheri, Ryerson University

The above committee determined that the thesis is acceptable in form and content and that a satisfactory knowledge of the field covered by the thesis was demonstrated by the candidate during an oral examination. A signed copy of the Certificate of Approval is available from the School of Graduate and Postdoctoral Studies.

Abstract

Over the last decade, metaheuristic algorithms have become well-established approaches utilized for solving complex real-world optimization problems. Most metaheuristic algorithms have used stochastic strategies in their initialization as well as during the new candidate solution generation process where there is no a priori knowledge about the solution, which is a common assumption for any black-box optimization problem.

In recent years, researchers have introduced a new concept called center-based sampling that can be used in any search component of the optimization process, but so far, it has mainly been utilized for population initialization. This concept clarifies that in a search space, the center point has a higher probability value to be closer to an unknown solution compared to a uniformly generated random point, especially when the dimension increases. Thus, this novel concept helps the optimizer to find a better solution efficiently.

In this thesis, a comprehensive study has been conducted on the effect of center-based sampling to solve an optimization problem using three different levels of investigation. These levels are as follows: 1) no specific algorithm and no specific landscape (i.e., Monte-Carlo-based simulation); 2) a specific landscape but no specific algorithm (random search vs. center-based random search), and finally, 3) a specific algorithm and specific landscape (proposing three different schemes for using center-based sampling for solving Large-scale Global Optimization (LSGO) problems). Also, a center-based sampling for multi-objective optimization is proposed. Furthermore, in this thesis, I seek to investigate the properties and capabilities of center-based sampling during optimization, which can be extended to utilize it in machine learning techniques, as well.

The proposed methods are evaluated on discrete and continuous Large-scale Global

Optimization (LSGO) benchmark functions. The experimental results confirm that center-based sampling has a crucial impact on improving the convergence rate of optimization/search algorithms when solving high-dimensional optimization problems.

keywords: Center-based Sampling, Large-scale Optimization, Monte-Carlo Simulation, Differential Evolution, High-dimensional Optimization.

Declaration of Authorship

Signed: Hanan Hiba

Date: December 15, 2020

Statement of Contributions

The work described in Chapter4 has been published as:

Hiba, H., Mahdavi, S., Rahnamayan, S. (2017). Differential evolution with center-based mutation for large-scale optimization. In 2017 IEEE Symposium Series on Computational Intelligence (SSCI) (pp. 1-8). IEEE.

Hiba, H., El-Abd, M., Rahnamayan, S. (2019, June). Improving SHADE with Center-based Mutation for Large-scale Optimization. In 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 1533-1540). IEEE.

Hiba, H., Ibrahim, A., Rahnamayan, S. (2019, June). Large-scale optimization using center-based differential evolution with dynamic mutation scheme. In 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 3189-3196). IEEE.

The work described in Chapter5 has been published as:

Hiba, H., Bidgoli, A. A., Ibrahim, A., Rahnamayan, S. (2019, June). CGDE3: An Efficient Center-based Algorithm for Solving Large-scale Multi-objective Optimization Problems. In 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 350-358). IEEE.

Part of the work described in Chapter 3 and Chapter 4 has been submitted to Swarm and Evolutionary Computation, Elsevier journal as:

Hanan Hiba et al. “A Comprehensive Investigation on Novel Center-based Sampling for Large-scale Global Optimization.”.

I performed the majority of the experiments and writing of the manuscript.

Acknowledgements

I would like to express my sincere gratitude to my supervisor, Prof. Shahryar Rahnamayan, for his assistance, invaluable guidance, support, and tremendous inspiration along my Ph.D. journey. You always provided me the challenging ideas, the best advice, and a suitable direction. Thanks for believing in me, and thanks for allowing me to explore many interesting research problems that eventually assist me in achieving my goals. Without your help, guidance, patience, and encouragement, this work would not be possible.

I would like to express my appreciation to the Libyan Ministry of Higher Education and Scientific Research for their generous financial support during my Ph.D. study.

A special thanks go to my beloved parents Dr. Mohamed Faiz Heiba and Mrs. Jamila Shehab, my brothers and my family, for their unconditional support and love during my whole life.

I would like to gratefully thank my husband, who inspired me to continue my journey, my lovely children Omar, Abdulrahman, Basmah, and Noor for encouraging me with their best wishes.

Special thanks to Dr. Sedigheh Mahdavi, Dr. Azam Asilian Bidgoli, Dr. Amin Ibrahim, and Dr. Mohammed El-Abd for sharing their wealth of expertise in the optimization field and for their collaboration.

My gratitude to the committee members: Dr. Masoud Makrehchi, Dr. Ramiro Liscano, university examiner Dr.Sean Bohun, and external examiner Dr.Ebrahim Bagheri for generously offering their time to provide valuable feedback on my thesis.

Many thanks go to my friends, Zakiya Alfoghi, Mashael Nasser, Sharareh Kianiharchegani, Khiria Aldwib, and all my present and past friends who always encouraged me through my journey.

I would like to thank all my colleagues and faculty staff at the Faculty of Engineering and Applied Science (FEAS) for their support in many ways.

To my grandfather and grandmother. . .

Contents

List of Figures

List of Tables

List of Symbols

X	Decision variable vector
D	The dimension of the search space
$\|\|x_d\|\|$	The length of any point vector
$E[\|\|x_d\|\|]$	The length of the mean point vector
D_{max_d}	The farthest point
D_{min_d}	The nearest point
a	The length of the cube
P_c	Probability of the closeness of the center point
P	Stands for probability function
P_R	Probability of the closeness of the random candidate solution
d	Euclidean distance function
S	Unknown solution
NP	Population size
F	Mutation scale factor
CR	Crossover rate
X_i^t	The ith vector of the population
X_{min}	The maximum bounds
X_{max}	The minimum bounds
x	The base vector in mutation scheme
y	The number of difference vectors

z	The type of crossover approach
I_{rate}	The improved accuracy rate
x_{center}	The average of three vectors
μ	The mean of the normal distribution
σ	The standard deviation
MAX_NFC	The maximum number of fitness function evaluations
$iter$	The number of iteration
M	The number of objectives
IGD	The inverse generational distance

Chapter 1

Introduction

1.1 Motivation

Many papers have focused on enhancing population-based algorithms by improving their mutation and crossover schemes in recent years. Using the center-based sampling concept is a novel technique that has been applied in different steps of population-based algorithms. In [1], the center of mass crossover operator (CMX) with a multi-parent combination operator was proposed for a real coded genetic algorithm. A base operator was created by selecting a set of parents, then calculating a set of virtual mates by mirroring each parent through a center of mass, using a two-parent recombination operator. They utilized an operator for the real vector recombination, namely, blended crossover base operator (BLX-α) due to its simplicity [2].

Center-based sampling theory has been introduced by Rahnamayan and Wang [3] in which they investigated the probability of closeness of the center to a solution in a black-box problem by using Monte-Carlo simulations. They measured the Euclidean distances of the points to the unknown solution for various dimensions. As a result, they indicated that this probability is growing exponentially toward almost one when the dimensionality of the search space increases. Therefore, I aim to extend the investigation using different methods to demonstrate that center-based sampling is crucial for black-box problems. Besides, this research seeks to interest the research community in utilizing the simple but powerful capabilities of center-based sampling in their proposed optimization and learning approaches.

Moreover, the center-based sampling concept has been utilized in the initialization level of meta-heuristic algorithms to accelerate the convergence speed for large scale optimization problems. Such as center-point-based Simulated Annealing algorithm [4] and Co-operative Co-evolution algorithm [5]. In addition, a center-based Differential Evolution was proposed by Esmailzadeh and Rahnamayan [6]. They generated random points in the center-based region between the candidate solution (x) and the opposite candidate solution ($\hat{x}$). All the research results that utilized the center point as an initial point have shown that the enhanced algorithms are promising in improving the acceleration rate of large-scale optimization algorithm. Furthermore, some research utilized center-based sampling in the operation-level schemes for population-based algorithms to solve low dimensional problems in an attempt to enhance its performance. Fan et al. proposed a trigonometric mutation operation (TMO) scheme for low denominational problems [7]. They changed the base vector of the mutation scheme into the center point of the geometric triangle, and they utilized the weight terms for the difference vector. Their algorithm has significantly increased the convergence speed of the DE algorithm compared to the classical DE algorithm.

The main motivation behind designing a new gravity center-based mutation scheme is mainly inspired by the previously mentioned research, specifically [1, 3]. Since the center-based concept is promising for enhancing population-based algorithms, a modified DE algorithm is introduced by defining a new mutation scheme based on the gravity center of three randomly selected parents. The base vector is replaced in the mutation operation using the average of three candidate solutions as a mean value of the normal distribution. The performance of the proposed algorithm is evaluated on CEC 2013 LSGO benchmark functions, and 15 discrete benchmark functions with dimensions 500 and 1000 [8]. Experimental results from the current study confirm that the proposed algorithm demonstrates a better performance on the majority of the benchmark functions. To the best of our knowledge, using a center-based sampling scheme during the optimization of large-scale problems is the first attempt to enhance the acceleration of population-based algorithms; however, the previous ones have utilized center-based sampling only during the population initialization to solve high dimensional problems.

The main focus of this thesis is a deep understanding of high-dimensional search space and its properties and center-based sampling, as well as introducing the methods for enhancing population-based algorithms by using the center-based concept in several different ways. The main performance measures in the optimization algorithm are convergence speed and solution accuracy. Hence, it is vital to find a technique that creates adequate performance measures to balance the criteria mentioned above. The main goals of this thesis are:

1) Understanding in detail the properties of high-dimensional space and center-based sampling.

2) Proposing center-based sampling schemes for population-based algorithms.

3) Utilizing a center-based scheme in various schemes for the population-based algorithms.

4) Investigating the power of designed center-based schemes with commonly used complex benchmark problems.

This study focuses on using the center-based concept for population-based algorithms at the operational level. Figure 1.1 shows the categorization of optimization methods and the scope of this thesis.

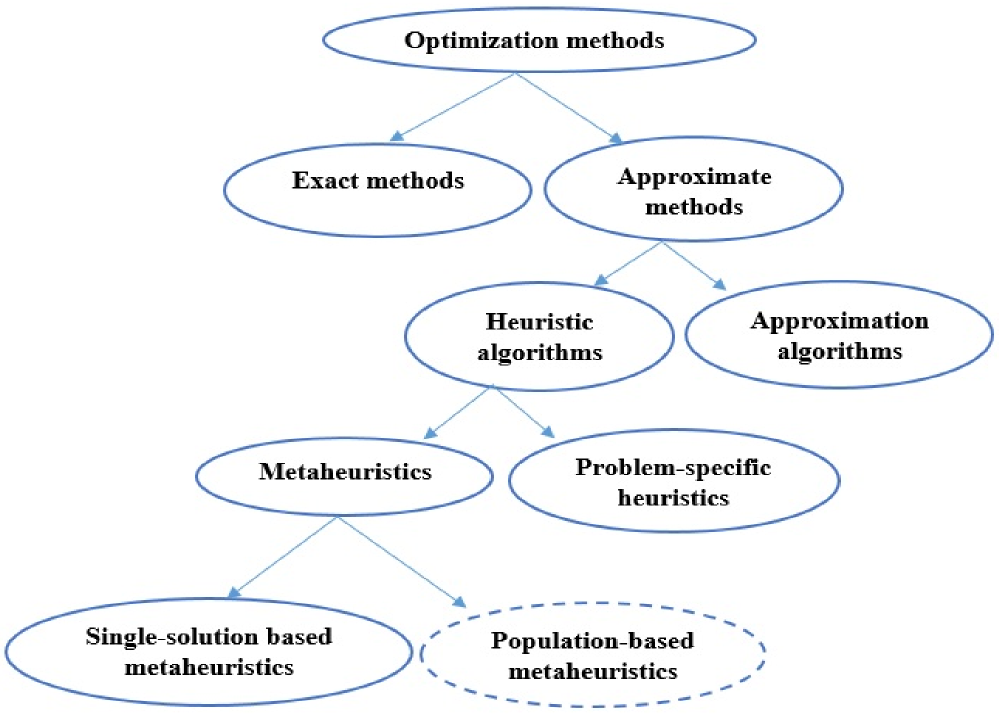


Figure 1.1: Categorization of optimization methods. Population-based meta-heuristics are the main focus of this thesis.

1.2 Research Contributions

The main contributions of this thesis are as follows:

1) Utilizing operation level center-based sampling schemes during the optimization.

2) Providing a comprehensive investigation for the closeness of the center to an unknown solution by using a Monte-Carlo-based investigation and Random Search algorithm (i.e., landscape and algorithm independent studies).

3) Proposing center-based for DE algorithm using different mutation schemes.

4) Enhancing the DE algorithm and the SHADE algorithm using a center-based mutation scheme.

5) Improving the GDE3 algorithm for multi-objective optimization using a center-based mutation scheme.

6) Proposing Dynamic center-based DE algorithm which divides the population into two portions for both center-based and classical mutations.

The following papers have contributed to the research outcomes of the current thesis:

- Hanan Hiba et al. "Differential evolution with center-based mutation for large-scale optimization." 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2017 [9].

This is the first paper that proposes a center-based mutation scheme, which is based on the utilization of the center of gravity as a base vector. This mutation scheme aims to generate the candidate solution using the center of three randomly selected candidate solutions. This new scheme is evaluated on CEC 2013 LSGO benchmark functions on the dimension 1000 and fifteen shifted discrete benchmark functions on dimensions 500 and 1000. Experimental results confirm that the new scheme achieves a great success rate compared to the classical DE over most of the test problems in terms of convergence rate and solution accuracy.

- Hanan Hiba et al. "Differential evolution with self-adaptive mutation scaling factor." 2017 IEEE Symposium Series on Computational Intelligence (SSCI). IEEE, 2017 [10].

In this paper, the DE algorithm is proposed that uses a newly designed mutation scaling factor to dynamically adapt the movement of the individuals in the search space toward the optimal value during the evolutionary process. However, the DE superiority is highly dependent on its control parameters and the search operators (i.e., mutation and crossover schemes). Therefore, to obtain the optimal performance, tuning the parameters is essential. The numerical experiments are conducted on thirty CEC 2014 benchmark functions on four different dimensions; 10, 30, 50, and 100. The obtained results demonstrate that the proposed algorithm is highly competitive and shows better performance than the classical DE algorithm.

- Hanan Hiba et al. “Improving SHADE with Center-based Mutation for Large-scale Optimization.” 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2019 [11].

In this paper, a center-based mutation for the Success-History Based Parameter Adaptation for Differential Evolution (SHADE) algorithm (CSHADE) is proposed. In this mutation scheme, the base vector for SHADE’s mutation is replaced with a center-based sampled candidate solution using the normal distribution. The proposed method is evaluated on CEC-2010 and CEC-2013 LSGO benchmark functions with dimension 1000. The experimental results show that CSHADE outperforms SHADE algorithm over the majority of benchmark functions in terms of solution accuracy.

- Hanan Hiba et al. “Large-scale Optimization Using Center-based Differential Evolution with Dynamic Mutation Scheme.” 2019 IEEE Congress on Evolutionary Computation (CEC). IEEE, 2019 [12].

In this paper, five different dynamic center-based DE mutation schemes (DCDE) is proposed to solve large-scale optimization problems. In each generation, the proposed dynamic center-based mutation strategies linearly divide the population into two different groups. Then, the first sub-population group utilizes a center-based mutation scheme, and the second sub-population employs the classical DE mutation. The proposed dynamic schemes are benchmarked on CEC 2013 large-scale optimization problems. The experimental results show that the overall performance of the proposed dynamic center-based mutation schemes better than the compared algorithms in solving LSGO problems.

- Hanan Hiba et al. “CGDE3: An Efficient Center-based Algorithm for Solving Large-scale Multi-objective Optimization Problems.” In 2019 IEEE Congress on Evolutionary Computation (CEC), pp. 350-358. IEEE, 2019 [13].

In this paper, a center-based mutation for Third Generalized Differential Evolution (CGDE3) algorithm is proposed to solve large-scale multi-objective optimization problems; in fact, this time center-based sampling scheme is employed during the optimization process, not just during the population initialization phase. The CGDE3 algorithm utilizes five randomly selected candidate solutions from its current population to generate a new trial vector for its mutation scheme. The proposed method enhances the GDE3 algorithm by improving its exploration ability using extra center-based sampling during the evolution process. This algorithm is tested on CEC 2017 competition benchmarks on evolutionary multi-objective optimization with dimensions of 100, 500, and 1000. Experimental results confirm that CGDE3 outperforms GDE3 over all three studied large-scale dimensions.

- Hanan Hiba et al. “A Comprehensive Investigation on Novel Center-based Sampling for Large-scale Global Optimization.” This is a journal paper submitted to Swarm and Evolutionary Computation, Elsevier journal.

In this paper, experimental investigations are proposed using LSGO benchmark test problems. These experiments include investigation with various approaches. First, calculating the probability of closeness of the center to the unknown solution and the closeness of the random point to the unknown solution using Monte-Carlo simulation with three different distance measures: Euclidean, Manhattan, and Cosine where it is not based on any specific algorithm or landscape. Second, utilizing a random search algorithm where a specific landscape is provided without a specific algorithm. Finally, providing a distinct optimization algorithm and landscape in which a center-based mutation scheme is applied in three different cases. In the first case, the center-based mutation is proposed for different schemes of the classical Differential Evolution (DE) algorithm. In the second case, a center-based mutation scheme is applied for the Success history-based parameter adaptation technique of the DE (SHADE) algorithm. Finally, the center-based DE and DE mutations contribute dynamically to improve the classical DE algorithm.

1.3 Organization of the thesis

This thesis consists of six chapters and three appendices, which are organized as follows:

Chapter 2 presents the main challenges of large-scale optimization, some properties for high dimension search space, mathematical calculation of closeness probability of center to an unknown solution, a background review that has relevance to the research including the concept of center-based, and optimization algorithms which are enhanced by the center-based sampling concept.

Chapter 3 provides an investigation into the closeness of the center to an unknown solution using Monte-Carlo simulation and conducting some experiments using a random search and adaptive random search algorithms.

Chapter 4 provides an introduction about the Differential Evolution algorithm (DE) and the Discrete Differential Evolution (DDE) algorithms. This chapter proposes a Monte-Carlo investigation into the center of three points. Also, it explains the details of the presented case studies for single-objective DE and its experimental results. In case study one, a center-based mutation DE strategy is proposed. Next, case study two proposes a center-based SHADE algorithm. Finally, the dynamic center-based mutation for the DE algorithm is investigated in case study three.

Chapter 5 provides an efficient center-based algorithm for solving large-scale multi-objective optimization problems, a background review about Third Generalized Differential Evolution (GDE3), and the first proposed center-based multi-objective algorithm (CGDE3). It presents the experimental results achieved through the use of center-based mutation scheme for CGDE3 algorithm.

Chapter 6 provides a summary of the thesis contribution and future directions. Finally, the appendices provide a definition of utilized benchmark functions in the conducted experiments.

Chapter 2

Background Review

This chapter provides an introduction about 1) large-scale optimization challenges, 2) unforeseen properties for high dimensional search space, 3) a review of the mathematical computation for the probability of center to be closer to the solution, and 4) Enhanced optimization algorithms by center-based Sampling.

2.1 Large-scale Optimization Challenges

Most of the real-world optimization problems deal with a big number of decision variables, these problems are known as Large-scale Global Optimization (LSGO) problems. A lot of research in the LSGO area was reported in various leading journals as well as several international conferences and workshops. For instance, seven benchmark test set functions provided in the CEC-2008 special session on LSGO [14, 15], twenty high-dimensional global optimization functions in CEC-2010 [15, 16] and a set of fifteen scalable benchmark functions in CEC-2013 [15, 17] were provided to evaluate the performance of proposed optimization algorithms. Also, there is a lot of large-scale machine learning problems, high dimensional feature space from deep learning, which they need an improved optimization technique for big data analytic [18, 19]; all of these confirmations reveal that companies and researchers are faced with practical complex large-scale problems which have been modeled as optimization problems.

A global optimization problem is defined as follows:

$$min/maxF(\vec{x}) = f(x_1, x_2, ..., x_n), \vec{x} \in R^n, \tag{2.1}$$

where n is the number of variables in large-scale setting (in general, $n > 100$), R^n indicates the decision space with n dimensions, and $\vec{x} = (x_1, x_2, ..., x_n) \in R^n$ is the decision variable vector, $f : x \rightarrow R$ stands for a real-valued continuous nonlinear objective function mapping from n dimensional space to one dimensional fitness value $F(\vec{x})$ [15].

Since large-scale optimization is an essential challenge in science and engineering fields, many algorithms have been proposed to solve them. Mahdavi et al. [15] recognized two main categories of approaches. First, Non-decomposition-based methods that solve LSGO problems as a whole so that they are designed with the specific effective operators (e.g., evolutionary computation) or they are combined with other optimization methods to further enhance their performance to explore complex search spaces (e.g. local-based approaches). Second, Cooperative Co-evolution (CC) algorithms which work based on the divide-and-conquer approach. They decompose LSGO problems into multiple single-variable or low dimensional sub-components [15, 20, 21] by utilizing static or dynamic grouping strategies.

In fact, the performance of standard meta-heuristic algorithms deteriorates when solving high dimensional problems [16, 17, 22] for the following main reasons. First, the volume of the search space increases exponentially when the number of the decision variable increase. Second, the properties of the problem landscape might change toward a harder condition and shape when the dimension increases. For example, some of the multi-modal functions turn to highly multi-modal problems when the dimension increases. Also, interaction among the parameters can turn to more complex ones [15]. Therefore, exploring the entire search space by the optimization algorithm becomes exponentially difficult [23–27].

2.2 Unforeseen properties for high dimensional search space

I seek to highlight some properties for high dimensional search space due to the fact that the large-scale search space does not follow the same rules of low-scale one. Thus, the following properties will help to understand what happens when the search space grows.

Property 1) A large number of variables causes a phenomenon called the curse of dimensionality. Grid sampling in high dimensional space requires exponentially increasing number of points. For example, assuming 2 bins in each dimension:

1) d=2, $2^2 = 4$ cells.

2) d=10, 2^{10}=1024 cells.

3) d=100, $2^{100} = 1.27 \times 10^{30}$ cells.

This means that when the dimension increases the search space becomes more complex and impossible to search it as a whole. Thus, the ability of an optimization algorithm to find the optimal solution degrades when the dimension increases [28].

Property 2) The theorem by Beyer et al. [28, 29] states that the variance of the ratio between the length of any point vector (denoted by $||x_d||$) with the length of the mean point vector (denoted by $E[||x_d||]$) converges to zero as the dimension is increasing. The proof shows that there is not much difference between the distance of the farthest point D_{max_d} and the nearest point D_{min_d}. Therefore, the difference between two points ($D_{max_d} - D_{min_d}$) does not increase with the dimensionality as fast as D_{min_d}. Given $\lim_{d\to\infty} var(\frac{||x_d||}{E[||x_d||]}) = 0$, then $\frac{D_{max_d} - D_{min_d}}{D_{min_d}} \to 0$. In other words, as the dimensionality increases, the points become uniformly distributed, and the ratio of ($D_{max_d} - D_{min_d}$) to D_{min_d} converges to zero. So, the scalability of measuring the Euclidean distance is generally poor.

Property 3) Every point is extreme in at least one dimension in high dimensional distributions. In other words, the point will be far away in at least one dimension and be far from another point when the dimension increases, which requires more dispersed sub-spaces. The probability for a point to be farther away from the mean than 3σ in a single dimension is $\approx (0.0027) = 1 - 0.9973$. This is due to the fast decay of the tails of the probability which can be quantified using (Mills inequality) [30]. For d independently normally distributed dimensions, the combined probability of a point appearing to be normal in every single dimension is $\approx 0.9973^d$. For instance, for d= 10, 100, 1000, the probability for a point to be farther away from the mean would be 97.33%, 76.31%, 6.696%, respectively.

Property 4) The volume of high dimensional hypersphere approaches to zero. In high dimensions, the vast majority of the volume of a solid hypersphere is concentrated in a thin shell near its surface [31]. Figure 2.1 shows the volume of a unit sphere sharply decreases when the dimension increases.

$$\lim_{d\to\infty} vol = (hypersphere(d)) = 0. \tag{2.2}$$

Also, most of the volume of the high-dimensional cube is located in its corners. Consider that a unit length hypercube is a and its volume is $V = a^n$. If we assume that there is a small cube inside the original one, its length is 0.999, and its volume is $V' = (0.999)^n$

$$\lim_{n\to\infty} (\frac{V'}{V}) = \frac{(0.999)^n}{a^n} = (\frac{0.999}{1})^n \approx 0. \tag{2.3}$$

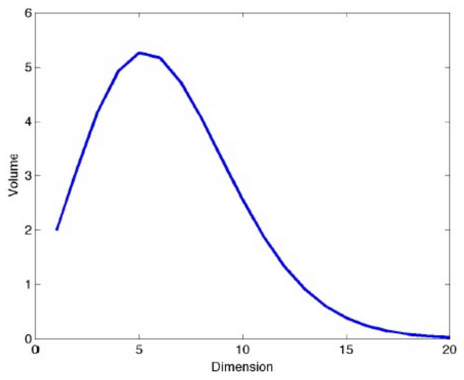


Figure 2.1: The volume of a unit sphere when the dimension of the sphere increases [28].

Hence, most of the volume goes to the surface.

Property 5) An interesting phenomenon is noted for the unit-radius sphere and the unit-length cube in higher dimensions. Consider the difference between the volume of a cube with unit-length sides, and the volume of a unit-radius sphere as the dimension (d) of the space increases. As this happens, the volume of the unit cube is always one, and the maximum possible distance between two points grows as $\sqrt{d}$. In comparison, the volume of a unit-radius sphere goes to zero, and the maximum possible distance between two points stays at two as the dimension of a unit-radius sphere increases [31]. It is interesting to note that the volume of a unit sphere goes to zero as the dimension of the sphere increases. Also, the volume of a high-dimensional sphere is essentially all contained in a thin slice at the surface.

Furthermore, in the case of a cube in high dimensions resembling a spiny solid, assume that a cube with side length 1 inside a sphere with radius 1. The distance from the center to a vertex of the cube (the length of the diagonal of the cube) is $\frac{\sqrt{2}}{2}$, and the apothem is $\frac{1}{2}$ as seen in Figure 2.2.

For $4d$ space, the distance from the center to a vertex of the cube (i.e., the length of the

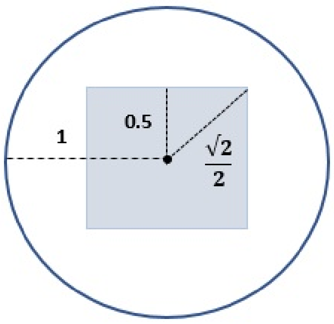


Figure 2.2: The unit sphere and cube in 2-dimensional space

.

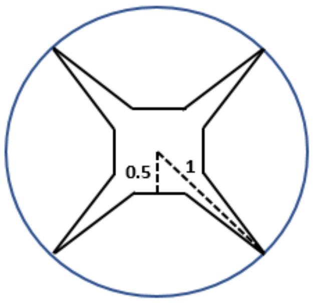


Figure 2.3: Projections of the 4-dimensional unit sphere and unit cube, centered at the origin (4 out of 16 vertices of the hypercube are shown).

cross-diagonal) is 1, so the vertices of the cube touch the surface of the sphere. However, the length of the apothem is still $\frac{1}{2}$ as seen in Figure 2.3.

For $d > 4$, the distance from the center to a vertex is $\frac{\sqrt{d}}{2}$ >1, and thus the vertices of the hypercube extend far outside the sphere as seen in Figure 2.4.

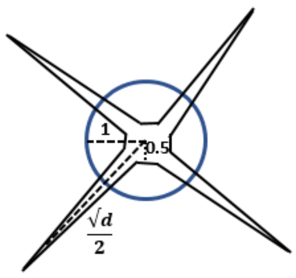


Figure 2.4: Projections of the d-dimensional unit sphere and unit cube, centered at the origin (4 of the 2^D vertices of the hypercube are shown).

.

2.3 Mathematical investigation for the probability of the center to be closer to the solution

Rahnamayan and Wang provided mathematical proof for center-based sampling method [32] in addition to the Monte-Carlo simulations. They explained that the probability of the closeness of the center point (P_c) to unknown solution increases with the dimension of the search space and approaches to one for higher dimensions. Their calculation was based on the following two scenarios where the solution is located: 1) on the border or 2) in the corner (worst-case scenario) of the search space. For the worst case scenario, the random points in the search space have a greater chance to be closer to an unknown solution than the center.

Figure 2.5 shows this situation for a $1D$ search space. As illustrated, the solution is located on the boundary. Thus, in this case, P_c can be calculated as follows:

$$P_{c(D=1)} = 1 - \frac{\frac{a}{2}}{a} = 0.50 \tag{2.4}$$

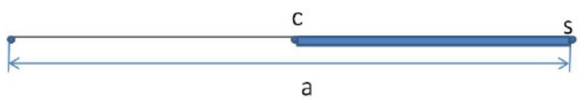


Figure 2.5: For 1D search space, the solution is located on the boundary. All points on the illustrated line segment (shadowed region, which is $\frac{a}{2}$) are closer to the solution, s, than the center point, c [32].

Figure 2.6 shows the situation in which the solution is on the boundary case for 2D. P_c is calculated as follows (i.e., a half-circle inside the square), where a is the length of the square.

$$P_{c(D=2)} = 1 - \frac{\frac{\pi \times (\frac{a}{2})^2}{2}}{a^2} = 0.61 \tag{2.5}$$

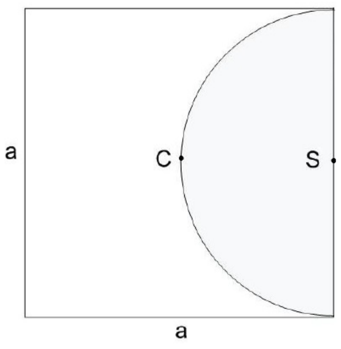


Figure 2.6: For 2D search space, the solution is located on the border. All points inside the illustrated circle (shadowed region) are closer to the solution, s, than the center point, c. [32].

For other dimensions, they worked with hypercubes as a search space, and hyperspheres as a sub-space. For hypercubes, the edge size is equal to a, and for hyperspheres, the radius r is equal to $\frac{a}{2}$. The volume formula for the general case of Hypercube is: $n-hypercube-volume = a^n$,

and the general case of Hypersphere is:

$n{-}hypersphere{-}volume = \frac{\pi^{\frac{n}{2}}}{\Gamma(\frac{n}{2}+1)} \times r^n$,

where Γ is the Gamma function that is calculated as: $\Gamma(n) = (n-1)!$

$$P_{c(D=3)} = 0.74 \text{ (i.e., 2-sphere inside 3-cube)} \tag{2.6}$$

$$P_{c(D=4)} = 0.85 \text{ (i.e., 3-sphere inside 4-cube)} \tag{2.7}$$

$$P_{c(D=5)} = 0.92 \text{ (i.e., 4-sphere inside 5-cube)} \tag{2.8}$$

$$P_{c(D=6)} = 0.96 \text{ (i.e., 5-sphere inside 6-cube)} \tag{2.9}$$

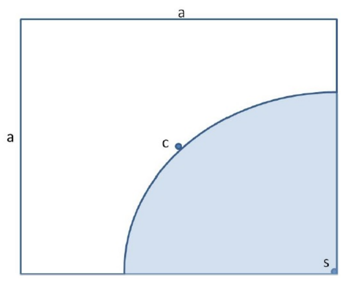


Figure 2.7: For 2D search space, the solution is located on the corner.[32].

For N dimension (i.e., $(N-1)$-sphere inside N-cube) the P_c will be calculated as following:

$$P_{c(D=N)} = 1 - \frac{\frac{V_N(\frac{a}{2})}{2}}{a^N} \tag{2.10}$$

Therefore, for a very big N (very high dimensions), the fraction will approach zero:

$$\frac{\frac{V_N(\frac{a}{2})}{2}}{a^N} \approx 0; \tag{2.11}$$

and So:

$$P_c \approx 1. \tag{2.12}$$

Figure 2.7 demonstrates the solution on the corner scenario. The hyperspheres radius would be $\frac{\sqrt{2}\times a}{2}$ instead of $\frac{a}{2}$. However, the figure shows $\frac{1}{4}$ for the hyperspheres instead of $\frac{1}{2}$. Therefore, the P_c can be calculated as follows:

$$P_{c(D=N)} = 1 - \frac{\frac{V_N(\frac{\sqrt{2}\times a}{2})}{4}}{a^N} \tag{2.13}$$

$$P_{r(D=2)} = \frac{1}{4} \times \frac{\pi \times (\frac{\sqrt{2}\times a}{2})^2}{a^2} \tag{2.14}$$

$$P_{r(D=2)} = \frac{1}{4} \times \frac{\pi \times \frac{1}{2}a^2}{a^2} \tag{2.15}$$

$$P_{r(D=2)} = \frac{1}{8} \times \pi = 0.39 \tag{2.16}$$

In this case, the P_c will be:

$$P_{c(D=2)} = 1 - 0.39 = 0.61 \tag{2.17}$$

$$P_{c(D=3)} = 0.63 \tag{2.18}$$

$$P_{c(D=4)} = 0.69 \tag{2.19}$$

$$P_{c(D=5)} = 0.77 \tag{2.20}$$

$$P_{c(D=6)} = 0.84 \tag{2.21}$$

Similarly, for very high dimensions, the fraction will approach zero:

$$\frac{\frac{V_N(\frac{\sqrt{2}\times a}{2})}{4}}{a^N} \approx 0\ ; \tag{2.22}$$

Therefore, the probability of center closeness to the solution will approach one.

$$P_c \approx 1. \tag{2.23}$$

These results show that even for the worst-case scenario when the dimension increases, the probability of closeness to unknown solution for the center approaches 1. It seems for similar dimensions, the corner case P_c is smaller compared to the boundary case.

2.4 Center-based Sampling

Rahnamayan and Wang [3] proposed a center-based sampling concept in 2009. They investigated the probability of closeness to an unknown solution for a center point and a uniform random point. By using Monte-Carlo simulation, they measured the Euclidean distances of the points to the unknown solution for dimension 1 to 1000 [5, 6, 33]. To get precise results, they split the search space interval ([a,b]) into partitions of 10^{-3} step-sizes (they moved from a corner to the opposite corner, using the defined step-size). In each dimension (1, 2, 3, ...,D), for each fixed-point, x, they repeated the following steps 10^6 times (i.e. trials):

1) Generating uniformly a random solution (s) and a uniform-random point (r) in the search space.

2) Measuring the Euclidean distance of the fixed-point, x, and the uniform-random point, from the created solution.

3) Based on the smallest distance value, the suitable distance variables were updated for calculating the probability of closeness and the average distance, in the last of the 10^6 times.

They discovered that the probability of points being closer to an unknown solution became greater towards the center of the search space compared to a uniformly generated random point over the entire space.

As can be seen from Figure 2.8, the center of the interval [a,b] for dimensions 1 and n is formulated as follows:

For D=1:

$$c = \frac{(a+b)}{2} \tag{2.24}$$

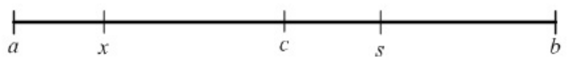


Figure 2.8: The visual illustration (in 1D) of uniform-random point, x, and the unknown-solution, s, in the interval [a,b], where c indicates corresponding center of the search space, c=(a+b)/2 [6].

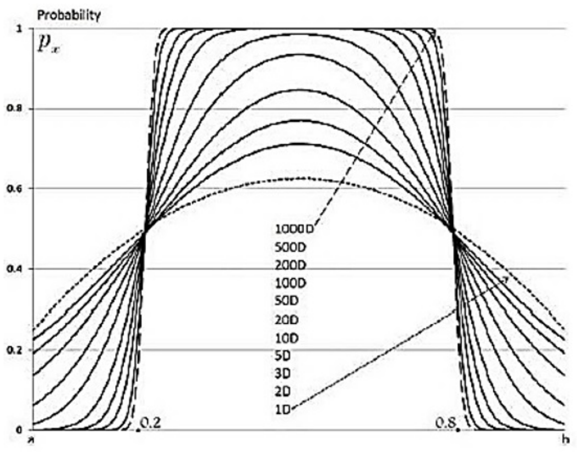


Figure 2.9: The graphs of Monte-Carlo simulations which present the probability of closeness of candidate-solution to an unknown solution in the interval [a,b], for different dimensions [3].

For D=n:

$$c_i = \frac{(a_i + b_i)}{2} \tag{2.25}$$

Where $i = 1, ..., n$, and D is the dimension of the problem.

Their simulation results demonstrated that when the dimension increases, the probability of closeness to an unknown solution for the center point increases sharply. The interesting phenomenon is that as the dimension of the problem increases, the probability of closeness to the solution improves as well and approaches to almost one for the higher dimensions as seen in Figure 2.9.

The phenomenon of center can be intuitively explained according to Figure 2.8. The

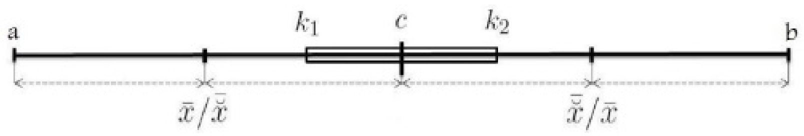


Figure 2.10: Illustration of the solution's region for 1-D search space in which a random point is closer to the unknown solution than x and $\hat{x}$. k1 and k2 are the centers of intervals [x/$\hat{x}$, r] and [r, $\hat{x}$/x] [34].

entire search interval of $[a, b]$ is divided into sub-intervals of $[a, c]$ and $[c, b]$ by the center point c. The candidate solution x and the unknown solution s, can each be in different sub-intervals, or they can both be in the same sub-interval. Since we are dealing with uniform random, the chances of either of the previous cases are 50%. When x and s are in the different sub-interval, c is in between x and s, this is irrelevant which sub-intervals belongs to, the relation between the distance of x and c to s is $|x - s| \geq |c - s|$. Therefore, in this case, c is definitely closer to s, than x to s. It means for 50% of situations for sure, c is closer to the solution. When x and s are in the same sub-interval, then x and c are competing together for closeness to s. Therefore, center-based sampling has a higher chance to be closer to s overall [6, 33].

Rahnamayan et al. explained intuitively that the opposite of a candidate solution based on the center of the search space acts better than a random point in term of closeness to an unknown solution [34]. They considered the interval for one dimensional space which is bounded by $[a, b]$ and has a center point c as seen in Figure 2.10. By assuming that the random solution $x \in [a, c]$, and its opposite solution $\hat{x} \in [c, b]$. Then, the average values of x and the opposite, $\hat{x}$ are located in a center of the sub-intervals $[a, c]$ and $[c, b]$, respectively. The average of both random guesses will be located at $\bar{r}$=c. As a result of considering these mean values they confirmed that a uniformly distributed solution $s \in [a, b]$ will (on average) be closer to the independent randomly generated points within the region $[k_1, k_2]$, where k_1

and k_2 are the centers of intervals $[x/\hat{x},\, r]$ and $[r,\, \hat{x}/x]$. In fact, they explained the power of opposition-based searching by using the power of center-based sampling.

2.5 Enhanced Optimization Algorithms by Center-based Sampling

In recent years, many papers have focused on enhancing population-based algorithms by improving their mutation and crossover schemes. Using the center-based sampling concept is a novel technique that has been applied in different steps of population-based algorithms. In [1], a center of mass crossover operator (CMX) with a multi-parent combination operator was proposed for a real coded genetic algorithm. A base operator was created by selecting a set of parents, then calculating a set of virtual mates by mirroring each parent through a center of mass, using a two-parent recombination operator. They utilized an operator for the real vector recombination, namely, blended crossover base operator (BLX-α) due to its simplicity [2].

Fan et al. developed the Trigonometric Mutation Operation (TMO) algorithm by designing a new mutation operation [7, 35]. In TMO, they used the center point of the geometric triangle as a base vector for the mutation operation and the weight terms $(p_2 -$

$p_1), (p_3 - p_2), (p_1 - p_3)$. Their mutation operation is defined as:

$$\begin{aligned} V_{i,G+1} =& (\frac{x_{r1,G} + x_{r2,G} + x_{r3,G}}{3}) + \\ & (p_2 - p_1).(x_{r1,G} - x_{r2,G}) + \\ & (p_3 - p_2).(x_{r2,G} - x_{r3,G}) + \\ & (p_1 - p_3).(x_{r3,G} - x_{r1,G}), \end{aligned} \tag{2.26}$$

where $r1 \neq r2 \neq r3 \neq i$ and parameters are calculated as follows.

$$\begin{aligned} p_1 &= |f(x_{r1,G})|/p' \\ p_2 &= |f(x_{r2,G})|/p' \\ p_3 &= |f(x_{r3,G})|/p' \\ p' &= |f(x_{r1,G})| + |f(x_{r2,G})| + |f(x_{r3,G})| \end{aligned} \tag{2.27}$$

Furthermore, Xu et al. conducted a hybridization technique on local search with DE in [36]. In this method, they randomly generated a population of size NP, then ranked the individuals from best to worst according to the objective values. They calculated the centroid of the top Q individuals as:

$$\bar{X} = \sum_{i=1}^{Q} \frac{X_i}{Q} \tag{2.28}$$

Then, the new individuals were generated by calculating the difference between NP and Q as ($NP - Q$). Then, the $NP - Q$ and the top Q individuals were merged to form the initial population for the DE algorithm. Motivated by this technique, Khanum et al. proposed a centroid population initialization for Adaptive Differential Evolution with Optional External Archive (JADE) algorithm in [37]. They generated $3 \times NP$ individuals randomly, then they selected three individuals and computed their centroid X_i as calculated by Eq. 2.29 until they got NP centroids to utilize them as an initial population.

$$X_i = \frac{X_{r1} + X_{r2} + X_{r3}}{3} \tag{2.29}$$

Esmailzadeh and Rahnamayan proposed a center-based concept for enhancing the differential evolution algorithm [6]. They generated random points in the center-based region between the candidate solution (x) and the opposite candidate solution ($\hat{x}$). Besides, they utilized a Center-point concept in the Simulated Annealing algorithm (CSA) at the initialization step to accelerate its convergence speed [4]. They used the center point concept since it is a unique point that has the highest probability of being closer to the unknown solution compared to any other random point generated in the entire search space. Their results confirmed that the proposed algorithm is superior when solving large-scale optimization problems.

Ali et al. introduced two new schemes embedded in a centroid-based mutation operation in DE [38]. In the first scheme, DE with the centroid-based mutation (CMO) was utilized. The CMO and the classical mutation scheme DE/rand/1 were used stochastically based on the centroid mutation probability P_c. The mutation scheme for CMO was defined as:

$$V_{i,G+1} = (\frac{X_{min,G} + X_{r1,G} + X_{r2,G}}{3}) + F \,.\, (X_{r1,G} - X_{r2,G}), \tag{2.30}$$

where $X_{min,G}$ was the current best candidate solution and $X_{r1,G}$ and $X_{r2,G}$ were distinct random individuals different from the best and current individuals. In the second algorithm, they used DE with local search (DELS), in which a centroid-based local neighborhood search was applied to explore the neighborhood of the best individual.

Chen et al. proposed a centroid strategy for the Artificial Bee Colony algorithm in [39]. They computed the centroid position in Artificial Bee Colony algorithm to enhance the ability to explore the search as follows:

$$C = \frac{\sum_{i=1}^{SN} X_i}{SN}, \tag{2.31}$$

where C is the position of the centroid, X_i is the position of the i employed bee, and SN is the number of the food source. They implemented the centroid position for improving ABC algorithms as well as combined the proposed strategy with other bee colony algorithms to strengthen the search capabilities toward the global.

Liu et al. introduced a random-based sampling and neighborhood mutation scheme for the DE algorithm, named NRDE [33]. In this scheme, they computed the center of the sub-population ($centerp$) and the center of the population ($centerg$) as follows:

$$centerp = \frac{1}{S} \sum_{X_i'' \in S} X_i'' \tag{2.32}$$

$$centerg = \frac{1}{NP} \sum_{X_i' \in NP} X_i', \tag{2.33}$$

where s was the size of the sub-population, and NP was the size of all populations. Then, they implemented two mutation schemes as follows:

$$V_i' = p * (centerg + centerp) \tag{2.34}$$

$$V_i'' = X_{r1}'' + 0.5 * (centerg - X_i'' + centerp - X_i'') \tag{2.35}$$

The p value was a uniformly distributed random number between 0 and 1. Therefore, two candidate solutions were generated in NRDE. Then, to enhance the efficiency of local search, the best vector from two new trial vectors was selected to replace the target vector in the sub-population.

Furthermore, Mahdavi et al. [5] introduced the utilization of the center-based concept in a cooperative co-evolutionary algorithm for large-scale optimization just for the initialization. They proposed three different schemes called center-based normal distribution sampling (CNS), central golden region (CGR), and hybrid random-center normal distribution sampling (HRCN) to enhance the performance of the algorithm. Their results confirmed that the CGR scheme is the best performing scheme compared with the others. Moreover, they simulated the closeness probability of center point to unknown solution in the corner for the worst-case scenario. They observed that the probability of the closeness of the center point in the worst-case scenario has 100% chance to be closer to the optimum solution for dimensions higher than 200 as seen in Figure 2.11.

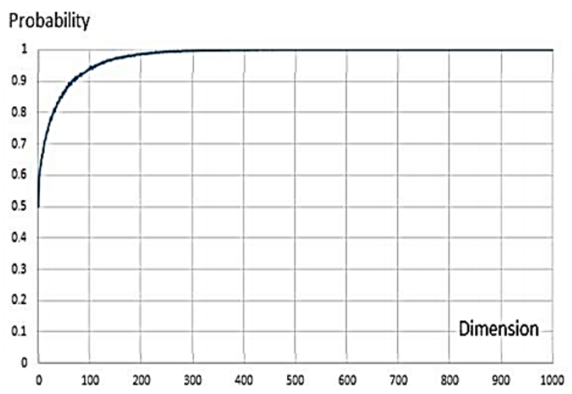


Figure 2.11: Probability of center-point closeness to the optimum solution in the corner, worst case scenario (compared to a uniformly generated random point) versus dimension of search space [5].

Salehinejad and Rahnamayan proposed centroid-based population initialization for the micro version of the DE algorithm [40]. In this method, they initialized the population within a central percentile of the boundaries a and b. As can be seen from Figure 2.12, the lower $\bar{x}_d^{min}$ and upper $\bar{x}_d^{max}$ boundaries of the centroid interval are calculated as:

$$\bar{x}_d^{min} = x_d^{min} + \frac{1-C}{2}(x_d^{max} - x_d^{min}) \tag{2.36}$$

and

$$\bar{x}_d^{max} = x_d^{max} - \frac{1-C}{2}(x_d^{max} - x_d^{min}), \tag{2.37}$$

where C is the selected centroid portion from the whole interval (0, 1). Their experimental results show that a small population size enhanced the performance of the classical DE algorithm for large-scale problems.

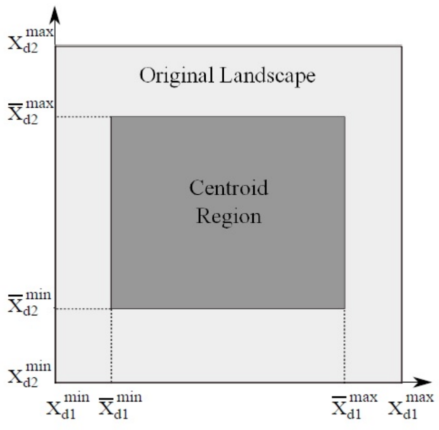


Figure 2.12: Centroid boundaries on a two dimensional search space. Dimensions are denoted by d1 and d2. The original search space (light grey square) refers to the original boundaries of the dimensions. The centroid region (dark grey square) refers to the centroid boundaries of the dimensions [40].

2.6 Summary

As mentioned in this chapter, large-scale problems were considered as a challenge in the optimization field. Therefore, the main properties for the high dimension search space were mentioned. In the related work, many experiments were explained that the center point is unique in the search space, significantly when the dimension increases. However, some more gaps need in-depth research on using center-based sampling to enhance the metaheuristic algorithms. Therefore, the next chapter will address an investigation using Monte-Carlo simulation for the closeness of the center to the unknown solution, which is different from the previously mentioned in [3]. In this simulation, the closeness of the center to the unknown solution will be investigated using different methods in terms of the hyper-cube size and the closeness of the center to the unknown solution in the worst-case scenario using three distance measures. Furthermore, some experiments will be conducted using the Random Search algorithm.

Chapter 3

Investigation of Closeness of the Center to an Unknown Solution in Black-box Optimization

This chapter investigates the center-based sampling advantages, and is divided into two sections:

Section one presents the first investigation in which there is no specific landscape and no specific algorithm (Monte-Carlo simulation). The Monte-Carlo simulation is applied for measuring two distances; the distance between the solution and the uniform random point; and the distance between the solution and center point; using three distance measures which are Euclidean distance, Manhattan distance, and Cosine distance.

Section two presents the second investigation in which there is a specific landscape but no specific algorithm (random search). Two schemes were provided for this case; in the first scheme, the center of the population is considered. In the second scheme, the center of the search space (CS) is considered.

The main reason behind section 3.1 investigation is to demonstrate that the center solution has a better chance to be closer to the unknown solution than any random solution in the black-box problems.

3.1 Monte-Carlo-Based Investigation

In this section, the effect of the closeness of the center to an unknown solution has been investigated without any specific algorithm or specific optimization problem (landscape). Three kinds of Monte-Carlo-based experiments were conducted. A D-dimensional search space with a uniform randomly generated solution (s) in the space is simulated in these experiments. The boundary for each variable (dimension) is [0,1]. The goal was to calculate the probability of closeness of the center (or a candidate solution in the region around the center called center-based candidate solution (c)) to an unknown solution compared to a random candidate solution (r). In the course of running 100,000 trials, three points were produced: a center-based candidate solution, a random point (as a regular-candidate solution (r)), and an assumed solution (s).

The probability of closeness of c to s was calculated and compared to the closeness of r to s as follows:

$$P_C = P[\, d(c,s) \leq d(r,s)] \tag{3.1}$$

$$P_R = P[\, d(r,s) < d(c,s)], \tag{3.2}$$

$$P_C + P_R = 1, \tag{3.3}$$

where P stands for probability function and d is the distance function.

For this purpose, two distances were calculated: 1) the distance between the r and s; 2) the distance between the c and s. The simulator counts the number of times that the center-based candidate solution is closer to the assumed solution. Then, the ratio of the calculated number to all iterations was considered as the probability of closeness of center-based candidate solution. In the following subsections, the results of three kinds of different simulations are discussed:

3.1.1 Investigation on the size of the hyper-cube region around the center

A D-dimensional search space with variables in the interval [0,1] is considered in this simulation. At each iteration, two random points are produced in this space as a candidate solution (r) and an unknown solution (s). In order to investigate the effect of the closeness of candidate solution to the center, a region around the center of search space (0.5,0.5) is considered to produce a center-based candidate solution in this region (c). Then, the closeness of r and c to s is compared. This simulation was repeated using different sizes of the regions. By starting with the size of a region equal to the whole search space, then, each time the region is decreased and shrank toward the center with a predefined step size of 0.01 to show how getting close to the center increases the probability of closeness to the unknown solution. Figure 3.1 represents the region considered around the center in 2-dimensional space from which the center-based candidate solution is randomly selected. The simulation with various distance measures is repeated using Euclidean distance, Manhattan distance, and Cosine dissimilarity.

The following equations were utilized for measuring the distances of

The Euclidean distance: The distance between x and y, if $x = (a, b)$ and $y = (c, d)$ was calculated as:

$$Euclidean - distance(x, y) = \sqrt{(a-c)^2 + (b-d)^2} \tag{3.4}$$

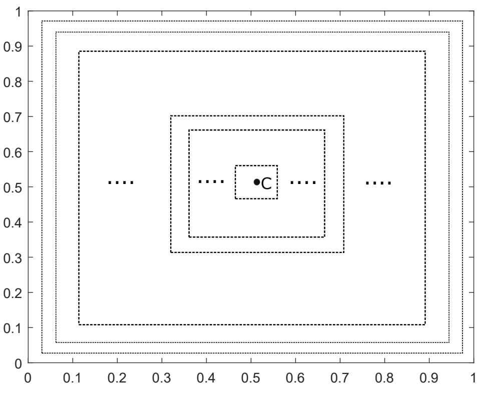


Figure 3.1: Different size of the region around the center considered to producing the center-based candidate solution.

The Manhattan distance between x and y was calculated as:

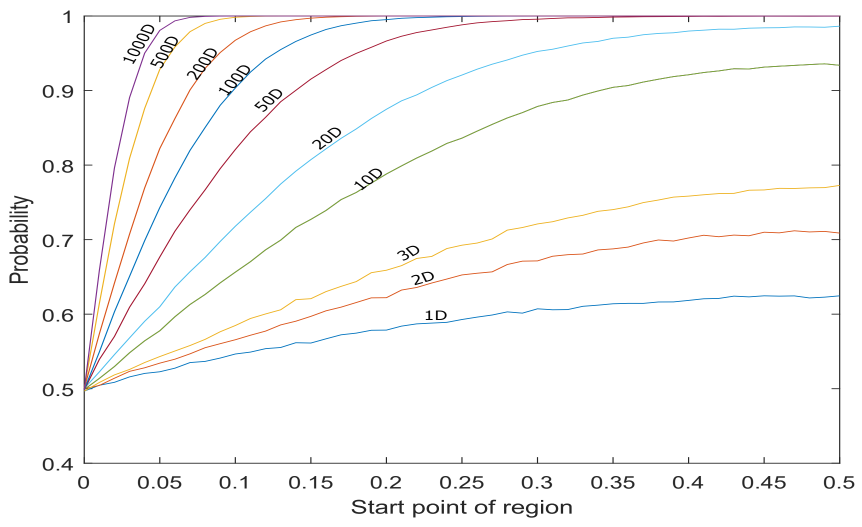
$$Manhattan - distance(x, y) = |a - c| + |b - d| \tag{3.5}$$

The Cosine dissimilarity calculates the cosine of the angle between two vectors x and y and its computing as:

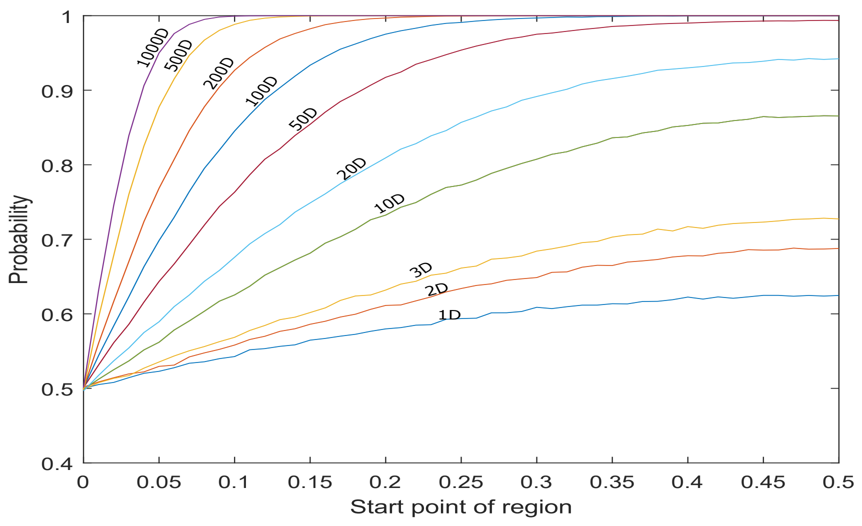
$$Dissimilarity(x, y) = 1 - \frac{\sum x_i \times y_i}{\sqrt{\sum x_i^2} \times \sqrt{\sum y_i^2}}, \tag{3.6}$$

where i is the points of each vector from 1...n.

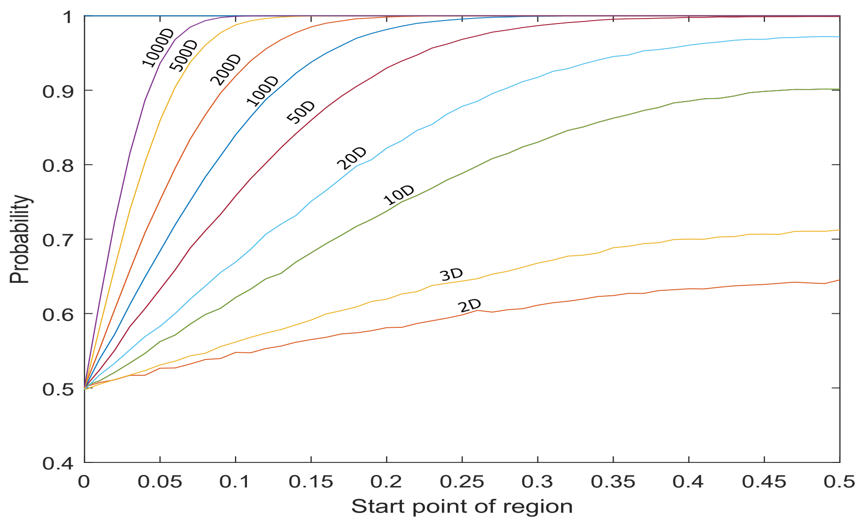
Figure 3.2 illustrates the probability of produced candidates in different sizes of the region (horizontal axis) and different dimensions (plots in different colors) calculated using three kinds of distance measures. The horizontal axis indicates the starting point of the region. As it is presented, by decreasing the size of the region, the probability of closeness of randomly selected candidate solution in the region to unknown solution increases comparing to a candidate solution in the whole space. Furthermore, the plots clarify that the closeness of the candidate solution in the region around the center increases when the



(a) Euclidean distance



(b) Manhattan distance



(c) Cosine distance

Figure 3.2: The graphs of Monte-Carlo simulations which present the probability of closeness of candidate-solution in a region around the center with different size to an unknown solution, in different dimensions.

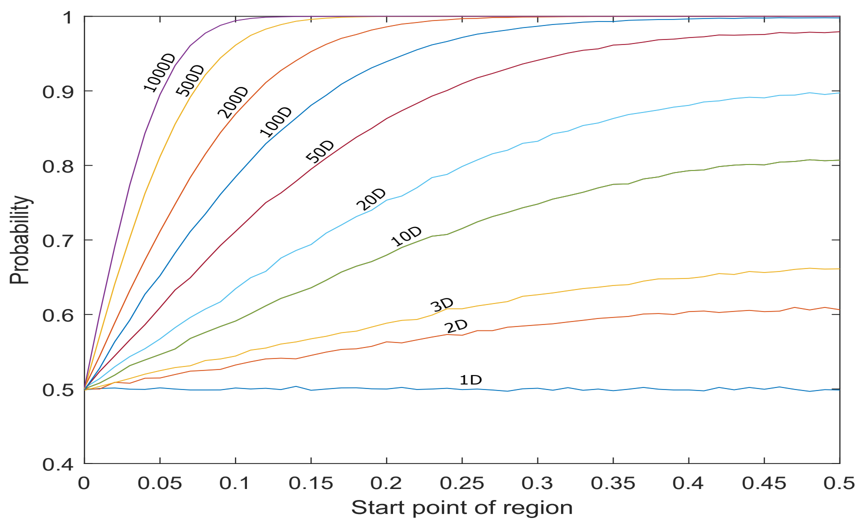
dimension goes up. The first selected region is equal to the whole search space, so the probabilities of the closeness of r and c to s are the same (0.5). As the region starts to shrink, the corresponding probability related to c increases. For large-scale search spaces ($D > 100$), the change in the probability gets sharper. The value of probability increases to one very fast. It is confirmed that by increasing the dimension of the search space, selecting the candidate solution close to the center has a better chance to be closer to an unknown solution; thus, the probability of closeness to an unknown solution will increase. Even though different types of distance measures were utilized, it confirmed this fact. Note that, the cosine similarity can not be defined in one-dimensional space because the angle between every two vectors in this space is zero.

In addition, the resulted probability based on three distance measures will be compared in terms of growing faster among three different distance measures. For instance, Figure 3.5a shows the probability of dimension 50, reaching one from the starting point 0.30 of the region for the Euclidean distance. On the other hand, the probability of dimension 50 reaching one from the starting point 0.52 and 0.36 of the region for Manhattan distance Figure 3.5b and Cosine dissimilarity Figure 3.5c, respectively. Therefore, the probability of the Euclidean distance rises faster in comparison with Manhattan distance and Cosine dissimilarity.

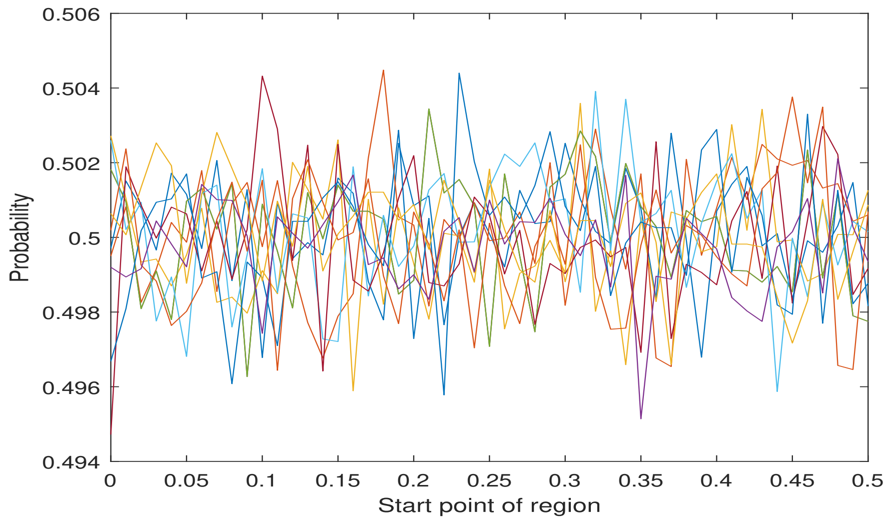
3.1.2 How close is a center-based candidate solution to a solution located in the worst position (worst-case-scenario)?

In this simulation, similar to the previous one, two random candidate solutions are selected, one from the whole search space (r) and one from a region around the center (c). However, it is assumed the solution is located in the corner of the search space (i.e., the farthest point from the center with variables equal to zero, in order to investigate the worst-case scenario). In fact, this case can be considered as the worst-case scenario of the distance between the center and the solution, so that if center-based sampling works for the worst-case scenario, it should be even better for other cases with respect to the position of the solution. Figure 3.3 represents the changes in probabilities using Euclidean and Manhattan measures. Since the inner multiplication cannot be defined using a vector with zero variables, cosine dissimilarity is not applicable for this simulation. In one-dimensional space using Euclidean distance, the probability of closeness for a point in each region around the center is the same as a point in the whole space (interval [0,1] in this example). Thus, the value of probability by changing the size of the region remains unchanged in one-dimensional space. From the simulation experiment, it can be observed that the center still has a better chance to be closer to the worst-case solution. Since for the Manhattan distance, changing the region size and the dimension doesn't affect increasing the probability, the center point is only better (closer to the solution in the worse position, corner) than half of the random points in the search space. Therefore, the probability would be the same for both candidate solutions (center-based and random ones). Fluctuating the plots happens because calculating the probability using a discrete stochastic simulation leads to different probability values.

The simulation results show that center-based sampling over Euclidean and Manhattan distances have similar behavior, but it is different on the Cosine dissimilarity. The fluctuations around 0.5 is because of the randomness property, nothing else. It happens because the Cosine distance is based on angle differences, not vector magnitude. This similarity metric is mainly used when the vector magnitude should not be used, like in Natural Language Processing (NLP) context applications.



(a) Euclidean distance



(b) Manhattan distance

Figure 3.3: The graphs of Monte-Carlo simulations which present the probability of closeness of candidate-solution in a region around the center with different sizes to the worst case solution in the corner, in different dimensions.

3.1.3 How much is the quality of the farthest candidate solution in a region around the center is good?

In this simulation, the closeness of the worst-case of center-based candidate solution to the solution was investigated. The first simulation was repeated but with a difference in producing the center-based candidate solution. Instead of selecting a random point in the region around the center, a fixed point on the farthest position in the region to the center was considered. Other settings of the simulation were similar to the previous one. Each time, two points in the whole search space were selected randomly as the random candidate solution (r) and the solution (s). The point in the corner of a region was assumed as the center-based candidate solution (c). The simulation was repeated on different sizes of regions and dimensions. In fact, by changing the size of the region, the center-based solution moved on the diameter of the search space (a point with the same value on all variables). The place of P_c in each region is shown in Figure 3.4 in red color. Figure 3.5 demonstrates the probability of closeness of c to s by different kinds of distance measures.

As can be seen, by decreasing the size of the region, the probability of closeness to the solution increases. The changes in probability have been accelerated in a high-dimensional problem. For example, in dimension 1000, the probability increases very sharply from zero to one when c is in point 0.2. An interesting discovery about this point is that in all dimensions, the probability of closeness of c in the regions, which are between 0.2 and 0.8, is more than 50% (more than the corresponding value for the random point); this region is called a golden region. The experiment indicates that while selecting the farthest point in the region (worst case) as a candidate solution is more beneficial than a random point in the space, it guarantees the effectiveness of all other points in the region compared to the random points. In other words, every point in the golden region definitely has a better chance to be closer to the solution. The same results with the Manhattan distance measure were received. Using the cosine distance, since c moves on the diameter of the space by changing the size of the region, the angle between this point and the assumed solution is not changed totally. So, the size of the region doesn't affect the probability of the closeness to the solution. On the other hand, increasing the dimension of the space leads to a larger value for probability. Note that cosine dissimilarity for one-dimensional space cannot be defined in this simulation, either.

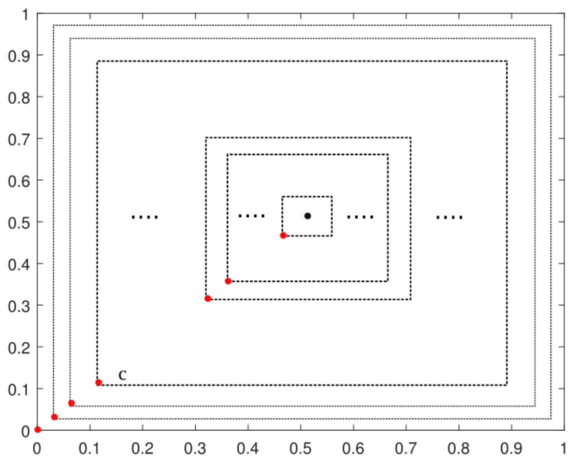
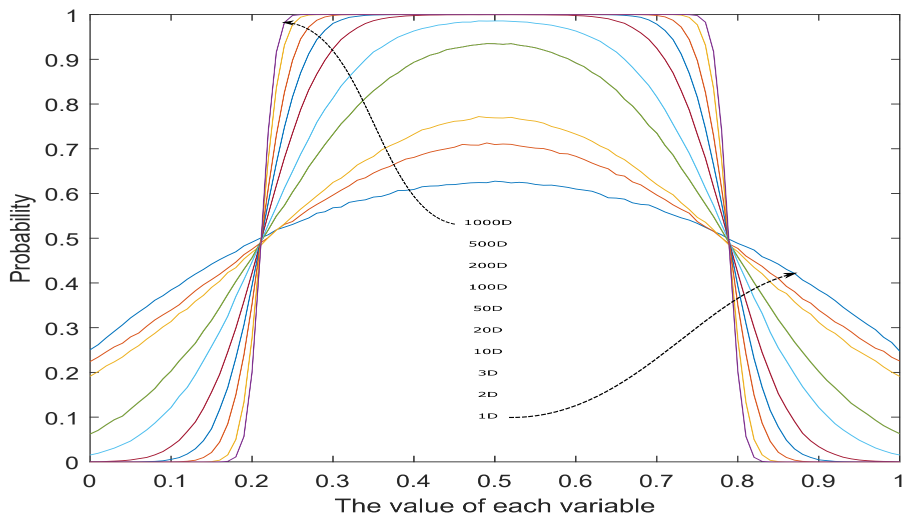


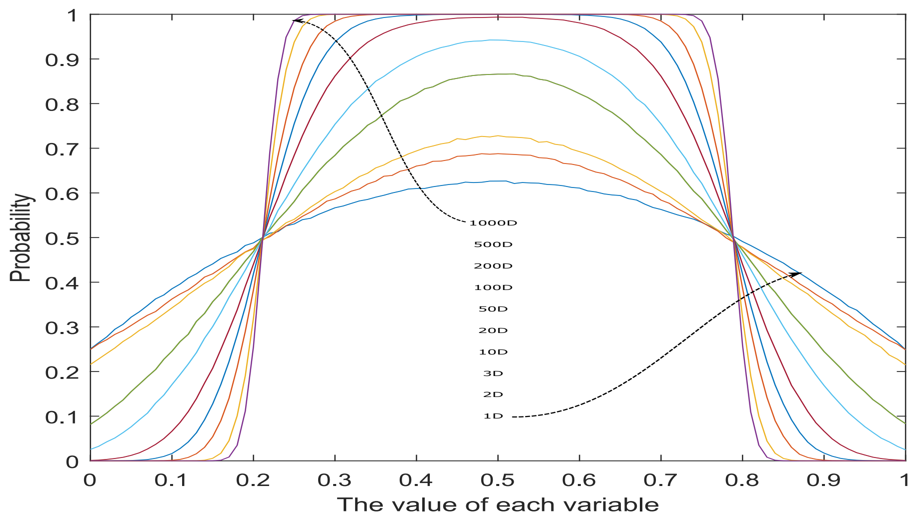
Figure 3.4: Demonstration of worst candidate solution in the region around the center by red color point.

3.2 Center-based Random Search (CRSA) and Center Adaptive Random Search Algorithms (CARSA)

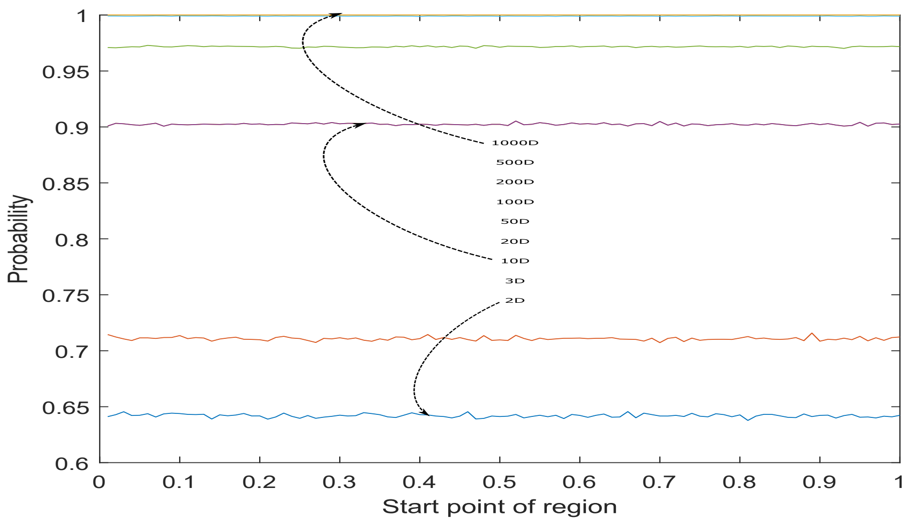
In this section, a center-based concept for Random Search Algorithm is utilized. The main goal was to investigate the impact of center-based sampling without a specific algorithm with a defined landscape. First, a background review of the Random Search technique is explained. Second, center-based Random Search schemes are proposed. Finally, the experimental results are analyzed.



(a) Euclidean distance



(b) Manhattan distance



(c) Cosine distance

Figure 3.5: The graphs of Monte-Carlo simulations which present the probability of closeness of candidate-solution to an unknown solution in the interval [a,b], for different dimensions.

3.2.1 Random Search Algorithm (RSA) and Adaptive Random Search Algorithm (ARSA)

A huge number of optimization problems can be handled using random search techniques, especially when the function to be optimized has several local minima, and it is difficult to find the global optimum. Random search techniques were introduced by Anderson in 1953 [41, 42], then later were proposed by Rastrigin [43] and Karnopp [44]. Random Search is a simple technique and a direct search method which does not require derivatives to search a continuous domain [45].

Adaptive Random Search is an extended version of the Random Search Algorithm, and it was designed to address the limitations of the fixed step size in the Localized Random Search Algorithm. The technique of the Adaptive Random Search Algorithm approximates the best step size required to achieve the global optimum in the search space continually. This can be reached by adopting smaller or larger step sizes only if they result in an enhancement in the search performance. The specific technique is to test a larger step in each iteration and select the larger step if it improves the result. Very large step size is tested also in the same manner. This technique is intended to allow the algorithm to escape local optima. Smaller step sizes are selected if there is no improvement for the next iterations [45].

3.2.1.1 The Proposed CRS and CARS Algorithms

In this case, two schemes for the Center-based Random Search Algorithm were introduced. In the first scheme, the Center-based Random Search Algorithm called CRSA is applied.

The mean of all individuals in the population was calculated and added to the population. Then, the minimum value was considered as the best value. This was repeated until the termination was satisfied.

In the second scheme, the center of the search space for both algorithms (Center Random Search and Center Adaptive Random Search Algorithm) was applied for the benchmark functions, called CSRSA and CSARSA. To explain more, the minimum and the maximum boundaries were shrunk to half so that the search would be focused more on the center of each problem boundary. For example, if the search space boundary was at $[100, -100]$, then the search space of the benchmark functions would start from $[50, -50]$.

3.2.2 Experimental Results

In this section, the benchmark functions utilized for all of our experiments are explained in the benchmark functions section. Also, in order to investigate the performance of the proposed Center Random Search Algorithms versus the classical Random Search Algorithms, comprehensive experiments have been conducted, and the numerical results are analyzed.

The proposed center-based schemes were applied and tested on CEC 2013 LSGO [17] benchmark functions with dimension 1000. It included five different groups of function types: fully separable functions (f_1-f_3), partially separable functions with a separable sub-component (f_4-f_7), partially separable functions with non-separable sub-components (f_8-f_{11}), overlapping functions (f_{12}-f_{14}) and one fully non-separable function (f_{15}).

In this thesis, the maximum number of evaluations was set to $3000D$, and the population size was set to 50. Moreover, a Wilcoxon's signed rank test [46] with a significance level of 95% was performed to have a statistical comparison for the best values achieved by algorithms. The symbols ”+”, ”=” and ”-” indicate that performance of the proposed algorithm was better than, similar to, or worse than the compared algorithms. In the last row of each table, *“w/t/l”* denotes *“the number of wins/ the number of ties/ the number of loses”* for the proposed algorithm in comparison with the original algorithm.

The results of the first scheme RSA versus CRSA on CEC 2013 LSGO are summarized in Table 3.1. As it can be observed from this table, CRSA accomplished significantly better than RSA on 10 functions (f_1-f_3,f_5-f_7,f_{11}-f_{13} and f_{15}) whereas CRSA had comparable results with RSA for five functions (f_4, f_8-f_{10} and f_{14}).

In addition, Table 3.2 summarizes the results of the second scheme CSRSA and CSARSA versus RSA and ARSA. As can be seen, CSRSA outperformed RSA on 14 functions (f_1-f_3 and f_5-f_{15}), tied on one function (f_4), no losses. Furthermore, center adaptive Random Search Algorithm CSARSA had better performance than ARSA on 11 functions (f_1-f_6, f_9 and f_{11}-f_{14}), 4 ties (f_7-f_8, f_{10} and f_{15}), zero loss.

Figure 3.6 shows the performance of the CRSA algorithm versus the RSA algorithm. They clearly show that the proposed algorithm of the first scheme (CRSA) started from less values which significantly improved the most functions of CEC 2013 LSGO. As can be seen, CRSA had better performance since the center of the search space had helped the algorithm to find more promising regions that promoted the location of the global optimum.

Table 3.1 Results of Random Search and Center Random Search Algorithms (Scheme 1) on the CEC 2013 benchmark functions for dimension 1000; Symbols '$+$', '$-$' and '$=$' denote the proposed algorithms are better than, worse than, or similar to the compared algorithm respectively.

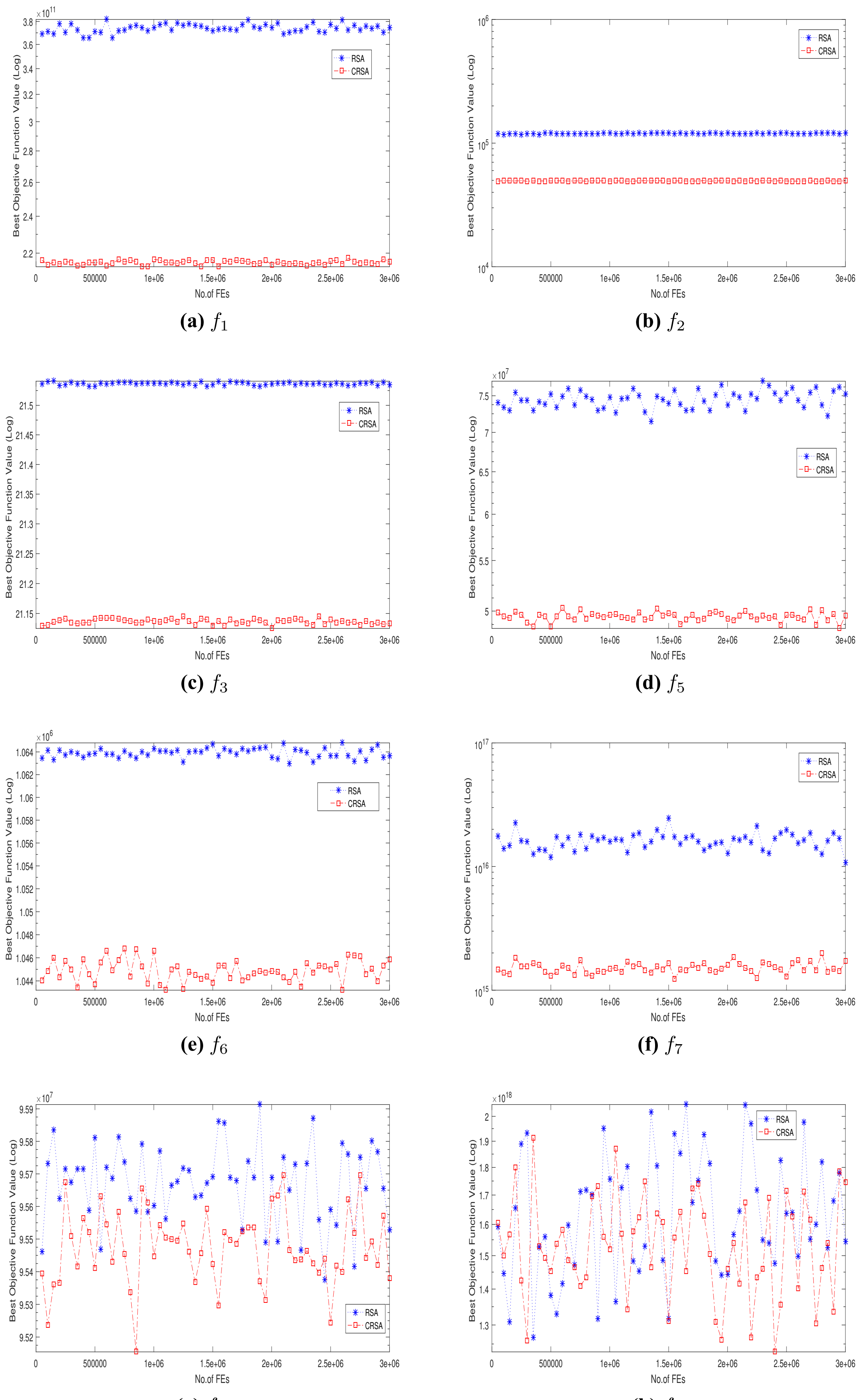
Function2013		RSA	CRSA	
f_1	Mean	3.742e+11	**2.156e+11**	+
	Std	1.655e+10	6.499e+09	
f_2	Mean	1.202e+05	**4.954e+04**	+
	Std	3.866e+03	9.138e+02	
f_3	Mean	2.153e+01	**2.113e+01**	+
	Std	1.178e-02	2.682e-02	
f_4	Mean	3.500e+13	3.509e+13	=
	Std	1.242e+13	1.197e+13	
f_5	Mean	7.527e+07	**4.961e+07**	+
	Std	7.948e+06	1.939e+06	
f_6	Mean	1.064e+06	**1.046e+06**	+
	Std	2.743e+03	3.545e+03	
f_7	Mean	1.072e+16	**1.714e+15**	+
	Std	9.706e+15	9.120e+14	
f_8	Mean	2.156e+18	2.203e+18	=
	Std	5.782e+17	7.510e+17	
f_9	Mean	5.898e+09	5.862e+09	=
	Std	8.583e+08	6.956e+08	
f_{10}	Mean	9.553e+07	9.538e+07	=
	Std	4.411e+05	6.082e+05	
f_{11}	Mean	7.133e+17	**1.668e+17**	+
	Std	4.486e+17	1.478e+17	
f_{12}	Mean	8.362e+12	**1.819e+12**	+
	Std	2.667e+11	3.730e+10	
f_{13}	Mean	1.387e+18	**1.251e+17**	+
	Std	1.417e+18	5.215e+16	
f_{14}	Mean	1.544e+18	1.744e+18	=
	Std	7.256e+17	1.049e+18	
f_{15}	Mean	8.815e+17	**3.170e+15**	+
	Std	3.168e+17	5.672e+14	
	w/t/l		10\5\0	

Table 3.2 Results of Random Search Vs. Center Random Search Algorithms, and Adaptive Random Search Vs. Center Adaptive Random Search Algorithms for center of the search space (CS) (Scheme 2) on the CEC 2013 benchmark functions for dimension 1000; Symbols '$+$', '$-$' and '$=$' denote the proposed algorithms are better than, worse than, or similar to the compared algorithm respectively.

Function		RSA	CRSA[CS]		Adaptive RSA	CARSA[CS]	
f_1	Mean	3.742e+11	**2.342e+11**	$+$	3.606e+11	**2.398e+11**	$+$
	Std	1.655e+10	8.582e+09		1.143e+11	3.447e+10	
f_2	Mean	1.202e+05	**6.059e+04**	$+$	7.960e+04	**5.474e+04**	$+$
	Std	3.866e+03	1.860e+03		3.939e+04	1.192e+04	
f_3	Mean	2.153e+01	**2.127e+01**	$+$	2.139e+01	**2.123e+01**	$+$
	Std	1.178e-02	1.330e-02		1.891e-01	1.054e-01	
f_4	Mean	3.500e+13	3.165e+13	$=$	1.240e+14	**5.208e+13**	$+$
	Std	1.242e+13	1.099e+13		9.285e+13	3.275e+13	
f_5	Mean	7.527e+07	**4.859e+07**	$+$	8.629e+07	**5.803e+07**	$+$
	Std	7.948e+06	2.567e+06		3.484e+07	9.492e+06	
f_6	Mean	1.064e+06	**1.044e+06**	$+$	1.058e+06	**1.043e+06**	$+$
	Std	2.743e+03	3.702e+03		1.287e+04	9.181e+03	
f_7	Mean	1.072e+16	**7.253e+14**	$+$	1.865e+18	4.114e+15	$=$
	Std	9.706e+15	3.456e+14		8.949e+18	6.763e+15	
f_8	Mean	2.156e+18	**1.435e+18**	$+$	4.421e+18	3.002e+18	$=$
	Std	5.782e+17	4.463e+17		4.724e+18	2.467e+18	
f_9	Mean	5.898e+09	**4.186e+09**	$+$	1.032e+10	**5.321e+09**	$+$
	Std	8.583e+08	5.308e+08		9.451e+09	1.555e+09	
f_{10}	Mean	9.553e+07	**9.426e+07**	$+$	9.477e+07	9.440e+07	$=$
	Std	4.411e+05	8.621e+05		7.659e+05	6.812e+05	
f_{11}	Mean	7.133e+17	**2.820e+16**	$+$	7.290e+19	**1.702e+17**	$+$
	Std	4.486e+17	1.212e+16		1.973e+20	4.623e+17	
f_{12}	Mean	8.362e+12	**2.958e+12**	$+$	6.340e+12	**2.570e+12**	$+$
	Std	2.667e+11	7.803e+10		3.364e+12	6.390e+11	
f_{13}	Mean	1.387e+18	**4.628e+16**	$+$	3.575e+20	**3.581e+17**	$+$
	Std	1.417e+18	1.912e+16		1.729e+21	6.870e+17	
f_{14}	Mean	1.544e+18	**7.742e+16**	$+$	1.893e+19	**7.970e+17**	$+$
	Std	7.256e+17	4.228e+16		4.069e+19	1.822e+18	
f_{15}	Mean	8.815e+17	**1.040e+16**	$+$	2.497e+18	9.044e+15	$=$
	Std	3.168e+17	2.858e+15		4.895e+18	1.171e+16	
	w/t/l		14/1/0			11/4/0	

3.3 Summary

This chapter introduced a Monte-Carlo investigation for closeness to the unknown solution using three distance measurements where there is no solution and no landscape. All results confirmed that when the dimension increases, the probability of the center to be closer to the solution has 100% to be closer to the solution. Also, some experiments using a Random Search algorithm were conducted for large-scale problems, where a specific landscape is provided without a specific algorithm. The results confirmed that the center Random Search algorithm outperformed the classical Random Search algorithm for most problems. In the next chapter, the Differential Evolution algorithm will be introduced. Furthermore, three experiments using center-based mutation DE for single solution optimization will be conducted.



(a) f_1 **(b)** f_2 **(c)** f_3 **(d)** f_5 **(e)** f_6 **(f)** f_7 **(g)** f_{10} **(h)** f_{14}

Figure 3.6: Convergence plots of some functions for CEC 2013 benchmark problems set with D=1000. The results were averaged over 51 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations (FEs).

Chapter 4

The Proposed Center-based for Single-objective Differential Evolution Algorithm

In this section, three case studies of the DE that use the center-based concept are proposed to enhance the comparing algorithm. For each case, the proposed method is explained and its experimental results are provided. In the first case, center-based mutation strategies for the DE algorithm with binomial and exponential crossover are conducted to indicate the effect of using center-based sampling for each mutation strategy. In the second case, improving SHADE with the center-based mutation is proposed. SHADE is an enhanced version of DE, which uses an external archive for F and CR parameters. Finally, the dynamic center-based mutation for the DE algorithm is investigated in case 3. The population size NP of the classical DE algorithm is divided into two portions: The first portion is utilizes the classical DE. The second portion uses the center-DE to observe center-based effectiveness during the exploration or exploitation stages.

4.1 Introduction

Population-based algorithms are one of the well-known optimization approaches which have been successful in solving a variety of real-world optimization problems. These algorithms are mainly meta-heuristic approaches for exploring complex search spaces. Their main goal is exploring the search space effectively to find the optimal solutions in a given time budget. The major difficulties of the population-based algorithms are a premature convergence into local optima because of losing the population diversity in the early stages of the optimization process [47, 48], and also their slow convergence, in general.

Several main population-based meta-heuristic algorithms such as Genetic algorithm (GA) [49], Differential Evolution Algorithm (DE) [50, 51], Particle Swarm Optimization

(PSO) [52, 53], Ant Colony Optimization (ACO) [54, 55], and Artificial Bee Colony (ABC) [56] have been introduced to tackle a variety of complex problems. These algorithms support some distinct advantages, (e.g., global search capability, no derivative dependency, robustness, supporting parallelization, etc. [7, 57]), which attract the attention of many researchers in optimization field.

Real-world large-scale optimization problems arise in many practical areas, hence, efficient algorithms are needed to solve these problems. The success of population-based algorithms has been achieved in solving a variety of complex problems in the engineering and science areas. However, their performance is degraded when solving high dimensional problems for the following two main reasons [23, 24]; first, when the number of the decision variable increases, the search space volume grows exponentially. Second, when the dimension of the problem increases, it may change the properties of the problem landscape toward a harder shape and conditions (e.g., increasing the number of modalities).

4.2 Differential Evolution

Differential Evolution algorithm (DE) is considered as an efficient algorithm for solving complex optimization problems. DE was proposed by Price and Storn in 1995 [58, 59]. DE is a powerful stochastic population-based evolutionary algorithm in which its efficiency has tested on several complex benchmark and real-word problems [10, 60, 61]. DE has three control parameters; NP (population size), F (mutation scale factor) and CR (crossover rate). The values of these parameters has a crucial impact on the obtained solution accuracy and the efficiency of the algorithm [61]. The DE algorithm starts with uniform randomly initialized population. Then, in every generation, it uses mutation and crossover operators to generate a new offspring solution. Next, the selection operator compares the parent and the offspring solution to select the surviving individual to the next generation [9, 61]. Finally, DE returns the best solution found so far [59, 61].The DE algorithm is explained in detail below:

1. Initialization In order to create a starting point for the evolutionary process, an initial population size (NP) is generated randomly for a D dimensional vector. The generation of DE is denoted by $t = 0, 1,..., t_{max}$. The ith vector of the population at the current generation is denoted by:

$$X_i^t = (x_{i,1}^t, x_{i,2}^t, \dots, x_{i,d}^t) \tag{4.1}$$

The initialization at ($t = 0$) must be in the specific range between minimum and maximum bounds: $X_{min} = x_{min,1}, x_{min,2}, \dots, x_{min,D}$ and $X_{max} = x_{max,1}$ $, x_{max,2}, \dots, x_{max,D}$. Thus, the initialization of jth component of the ith decision vector is

$$x_{i,j}^0 = x_{min,j} + rand_{i,j} \; . \; (x_{max,j} - x_{min,j}), \tag{4.2}$$

where $rand_{i,j}$ is a uniformly distributed random number between 0 and 1 [61].

2. Mutation

During mutation operation, DE creates a new candidate solution called a donor solution. The DE mutation scheme uses $DE/x/y/z$ notation to identify the mutation strategy, where x specifies the base vector, y is the number of difference vectors used, and z denotes the type of crossover approach. In the classical DE, $DE/rand/1$ mutation scheme was used [58, 59]. This scheme randomly selects three candidate solutions x_{r1}, x_{r2}, and x_{r3} from the current population to generate a new candidate trial solution as follows:

$$v_i^t = x_{r1}^t + F \,.\, (x_{r2}^t - x_{r3}^t) \tag{4.3}$$

Some of the most frequent mutation scheme used in literature are:

DE/best/1:

$$v_i^t = x_{best}^t + F \,.\, (x_{r1}^t - x_{r2}^t) \tag{4.4}$$

DE/current-to-best/1:

$$v_i^t = x_i^t + F \,.\, (x_{best}^t - x_i^t) + F \,.\, (x_{r1}^t - x_{r2}^t) \tag{4.5}$$

DE/best/2:

$$v_i^t = x_{best}^t + F \,.\, (x_{r1}^t - x_{r2}^t) + F \,.\, (x_{r3}^t - x_{r4}^t) \tag{4.6}$$

DE/rand/2:

$$v_i^t = x_{r1}^t + F \,.\, (x_{r2}^t - x_{r3}^t) + F \,.\, (x_{r4}^t - x_{r5}^t) \tag{4.7}$$

where the selected indices r_1^i, r_2^i, r_3^i, r_4^i, and r_5^i are mutually distinct randomly generated integers chosen from the range $[1, Np]$ and are all different from the index i. The scaling factor F is a positive control parameter for scaling the difference vectors between [0-2] [61].

3. Crossover

During the crossover operation, the component of the donor solution is replaced with its associated target vector x_i^t to diversify the population. The DE has two different crossover methods, namely, exponential and binomial [62]. The exponential crossover; first a random number is selected from the numbers $[1, D]$. This integer number is considered as a starting point in the target vector. Also, another integer L is selected from the interval $[1, D]$ according to the following pseudo-code:

$$u_{i,j}^{t-1} = \begin{cases} v_{i,j}^t & for\ j{=}\langle n \rangle_D, \langle n+1 \rangle_D, ..., \langle n+L-1 \rangle_D \\ x_{i,j}^t & for\ all\ other\ J \in [1, D], \end{cases} \tag{4.8}$$

where as the lowest value of L is one to guarantee that the trial vector u_i, G will vary from its corresponding target vector x_i, G by least one parameter. The angular brackets$\langle \rangle_D$ denote a modulo function with modulus D.

The binomial crossover for each of the d variables is defined as:

$$u_{i,j}^{t-1} = \begin{cases} v_{i,j}^t & if\ j{=}k\ or\ rand_{i,j} \leq Cr \\ x_{i,j}^t & otherwise, \end{cases} \tag{4.9}$$

where k is a random number in $[1, 2,..., d]$ and $rand_{i,j}$ is a uniform random number in $[0, 1]$. The $j{=}k$ guarantees that u_i^t is getting at least one element from v_i^t.

4. Selection

The selection operation compares two vectors, the target vector x_i^t and its corresponding trial vector u^t. The better vector, according to its minimum/maximum objective function value is chosen. For the minimization problem, the selection operator is defined as follows:

$$x_i^{t+1} = \begin{cases} u^t & if\ f(u^t) \leq f(x_i^t) \\ x_i^t & otherwise, \end{cases} \tag{4.10}$$

where $f(.)$ is the objective function to be minimized.

4.3 Discrete Differential Evolution (DDE)

Since the classical DE algorithm was designed to search in continuous search space, there are many different strategies which utilizes DE algorithm to tackle with discrete problems. Ho-Huu et al. proposed an adaptive elitist-based differential evolution for optimization of truss structures [63–65]. Their contributions were designed by using three steps to improve DE algorithm for solving discrete problems. First, in the mutation operation, they proposed an adaptive technique based on an absolute deviation of the objective function in order to keep the balance between local and global search capabilities. Second, in the selection operation, they used an elitist technique to improve the convergence rate of the algorithm. Third, a rounding technique was combined by using a fix function to deal with the discrete space.

Similarly, Zaheer and Pant [66–68] implemented DDE to solve integer programming problems. They introduced a DE-based algorithm using two main modifications: first, the initial population is generated integers number and, second, the mutation scheme is modified to handle discrete numbers by using a direct approach to improve the exploration ability for low dimensional integer problems as follows.

$$V_{t,i}^{G} = \begin{cases} \text{INT}\left[X_{t,i}^{G}\right] + 1 \text{ , if } x_{t,i}^{G} \geq 0 \\ \text{INT}\left[X_{t,i}^{G}\right] - 1 \text{, otherwise,} \end{cases}$$

where INT $[\cdot]$ operator is expressed as the integer part of the real number.

4.4 Monte-Carlo investigation about the center of three points

In this section, the probability of the center of three points to be inside the golden region is calculated by using Monte-Carlo simulation. As we can see from Figure 3.5a, the golden region was in a specific range between [0.2,0.8] for the interval [0,1] [3].

Let us assume we wanted to calculate the probability of the center of three points (c_3) to be inside the golden region (g) with the center point (c). Therefore, the probability was defined as follows:

$$P_g = P[d(c, g) > d(c, c_3)], \tag{4.11}$$

where P stands for probability function and d stands for the Euclidean distance.

Algorithm 1 implements the Monte-Carlo simulation in order to calculate P_g for D-dimensional search space. Furthermore, it calculates the average distance of the center of three points to be inside the golden region. The simulation results are depicted in Figure 4.1

and Figure 4.2 for dimensions (1D to 100D). As we can see, the probability P_g increased gradually with the dimensions, especially, for the dimension higher than 10. Also, The probability converged to one for the dimensions higher than 10.

Algorithm 1 :**Calculating P_g (probability of the center of 3 points to be inside the golden region) and (the average distance to be inside the golden region) by the Monte-Carlo simulation.**

```
1: $D = 1000$ // The dimension of the problem.
2: $TRIALS = 10^7$;
3: $Sum_{Ds} = 0$;
4: $count = 0$;
5: **for** $i = 1$ to $D$ **do**
6:    $Sum_{Ds} = 0$;
7:    $count = 0$;
8:    **for** $j = 1$ to $TRIALS$ **do**
9:       Generate three random points a, b and c in the D dimensional space.
10:      $center_{abc} = (a + b + c)./3$; // Calculating the center of 3 individuals
11:      Calculate the Euclidean distance of the center of three points $c_3$ and the average
         distance from the golden region $g$ from the center point $c$ ($dc_3$ and $dg$ ).
12:      **if** $dg > dc_3$ **then**
13:         $count$=$count + 1$
14:      **end if**
15:      $sum_{Ds}$ = $sum_{Ds}$+$dc_3$ ; // Calculating the summation of the distance for each di-
         mension.
16:   **end for**
17:   $ave_D$=$sum_{Ds}/TRIALS$ // Calculating the average distance to be inside the golden
      region.
18:   $P_g = count/TRIALS$; // Calculating the probability to be inside the golden region.
19: **end for**
```

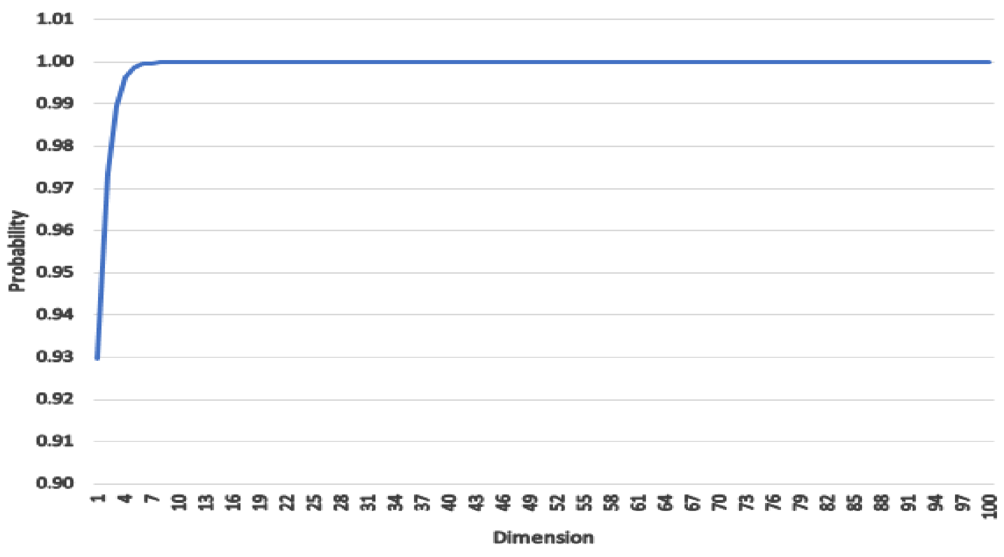


Figure 4.1: The graphs of Monte-Carlo simulations which present the probability of the center of three points to be inside the golden region [0.2,0.8] for dimension 100

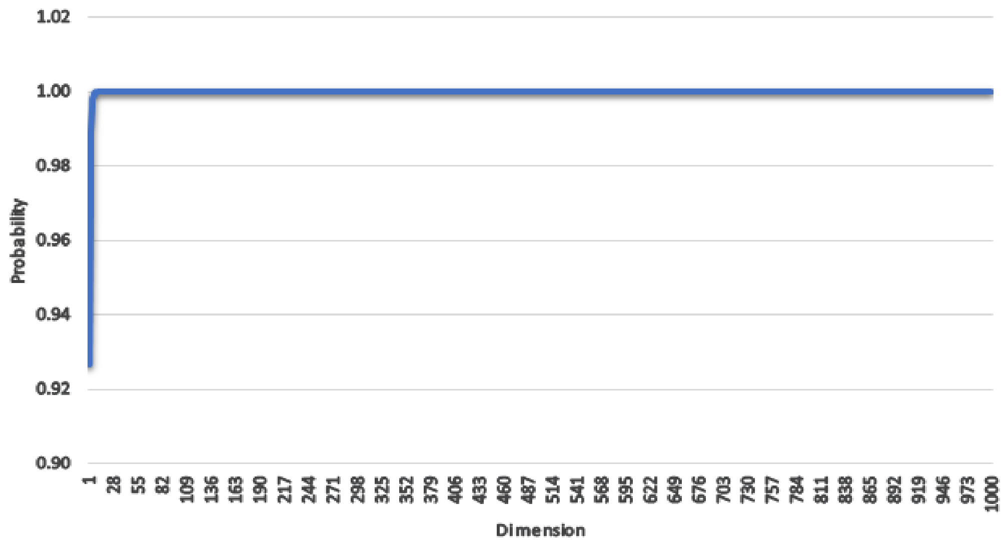


Figure 4.2: The graph of Monte-Carlo simulations which present the probability of the center of three points to be inside the golden region [0.2,0.8] for the higher dimension up to 1000

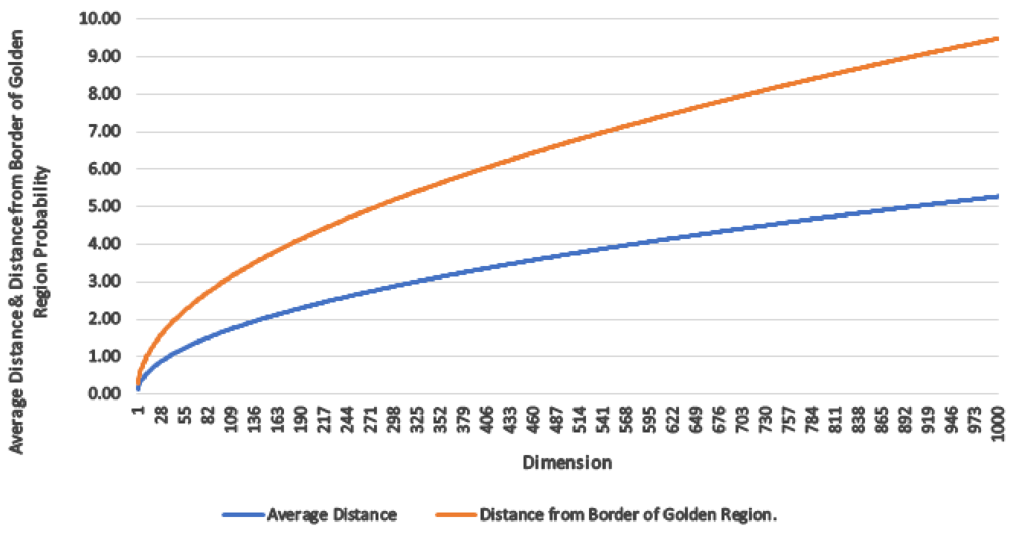


Figure 4.3: The graph illustrated the expected average distance for the center of three points to the center of search space and the distance from the border of golden region versus dimension of search space.

In addition, the simulation results for the average distance to be inside the golden region versus dimension is shown in Figure 4.3. As it can be seen, the expected average distance for the center of three individuals to be in the golden region increased sharply when the dimension is increased. However, the distance from the border of the golden region also rose dramatically when the dimension was increased. This means the chance of the center of three points to be in the golden region became almost 100% for higher dimensions. Therefore, this can improve the ability of the algorithm to explore the search space effectively.

Moreover, the probability of the center of three individual P_g to be inside the golden region is investigated when the optimal solution was in the worst-case scenario (the solution in the corner). The result is presented in Figure 4.4. As we can see, the probability increased gradually when the dimension is increased, and it converged to one from dimension 70D and higher. This means that even though the solution is in the worst-case scenario, the center of three individuals had approximately 100% likelihood to be closer to the solution.

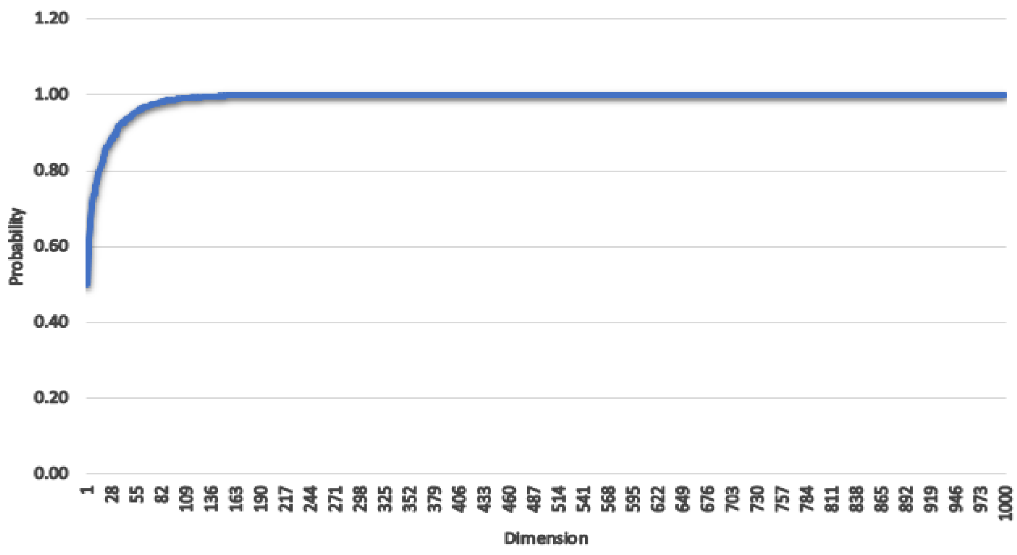


Figure 4.4: The probability of center of three points closeness to the optimum solution in the worst case scenario (solution in the corner) versus dimension of search space.

4.5 Case Study One: Center-based mutation DE strategies

In this section, a center-based mutation is proposed on the DE mutation schemes, called CDE. In the proposed algorithm, a new base vector is applied for each classical mutation scheme mentioned in section 4.2. The new base vector was generated by utilizing the center of three candidate solutions as a mean value of a normal distribution. The Algorithm 2 explains the center-based mutation scheme for DE algorithm. The proposed algorithm started with an initial population which had NP candidate solutions. A new base vector was generated by calculating the average of three selected randomly individuals x^t_{r1}, x^t_{r2}, x^t_{r3} from the current population and calculating the average value x_{center} (line 8) as following:

$$x^t_{center} = (\frac{x^t_{r1} + x^t_{r2} + x^t_{r3}}{3}) \tag{4.12}$$

This average value was utilized as a mean value (μ) of the normal distribution, which generates a new solution around the mean for each dimension j (line 11). Let us assume that the following matrix shows the current population for DE:

$$\text{Current population:=}\ [pop_{i,j}] = \begin{bmatrix} 1 & \dots & \dots & D \\ \vdots & \ddots & \ddots & \vdots \\ NP & \dots & \dots & \vdots \end{bmatrix}$$

The population size NP shows the number of rows and dimension D shows the number of the column in the matrix.

The reason behind selecting number (three) for the random vectors is to confirm that 90% of center-based points would be inside the golden region. However, by increasing the number of vectors, the center point gets closer to the center, which leads to reducing the population diversity.

The standard deviation σ was calculated as $\frac{max_j - min_j}{6}$. The maximum value (max_j) and the minimum value (min_j) of the $j - th$ column was chosen. The reason for using this σ was that it caused the generated numbers within the interval $[min_j, max_j]$ so that the probability of producing a random point within the range $[min_j, max_j]$ was 99.73% [5, 69]. In this case, in the current variable interval range, some numbers were produced using the border of each variable.

Therefore, the normal distribution generated a new point to be the base vector as follow (line 4.13):

$$x^t_{Ncenter} = N\left(x^t_{center}, \sigma\right) \tag{4.13}$$

The new mutant vector v_i for the new mutation schemes was obtained as follows: CDE/rand/1:

$$v_i^t = x_{Ncenter}^t + F\,.\,(x_{r4}^t - x_{r5}^t) \tag{4.14}$$

CDE/best/1:

$$v_i^t = x_{Ncenter-best}^t + F\,.\,(x_{r4}^t - x_{r5}^t) \tag{4.15}$$

CDE/current-to-best/1:

$$v_i^t = x_{Ncenter}^t + F\,.\,(x_{best}^t - x_i^t) + F\,.\,(x_{r4}^t - x_{r5}^t) \tag{4.16}$$

CDE/best/2:

$$v_i^t = x_{Ncenter-best}^t + F\,.\,(x_{r4}^t - x_{r5}^t) + F\,.\,(x_{r6}^t - x_{r7}^t) \tag{4.17}$$

CDE/rand/2:

$$v_i^t = x_{Ncenter}^t + F\,.\,(x_{r4}^t - x_{r5}^t) + F\,.\,(x_{r6}^t - x_{r7}^t) \tag{4.18}$$

The proposed center-based mutation schemes were utilized only over 10% of the beginning of generations (lines 7-13) in order to help the optimizer to have a better convergence toward the optimal solution. In fact, experiments primarily were conducted for the parameter using 5%, 10%, and 15% at the beginning of generations. The results show that utilizing 10% was the best. In the remaining portion of the generations, the classical DE mutation schemes were applied (For each center DE mutation scheme was applied with its counterpart classical DE mutation scheme) in order to keep the population diversified. For (DE/best/1) and (DE/best/2) schemes, two schemes were tested. First, the 10% of the beginning of generations (CDE/rand/1) was applied, and subsequently (DE/best/1) scheme for the remaining generations. Second, the 10% of the beginning of generations (CDE/best/1) was applied, and (DE/best/1) for the remaining ones. The same was implemented for (DE/best/2). In this scheme, which is called (CDE C3BEST), the base vector was generated by calculating the mean value of the first best, the second best, and the third best in order to promote the algorithm to explore effectively the search space which lead to more promising regions and optimal solutions. In fact, utilizing the gravity center as a mean value for the normal distribution gave more possibility for the generated solution to be around the center as can be seen in Figure 4.5.

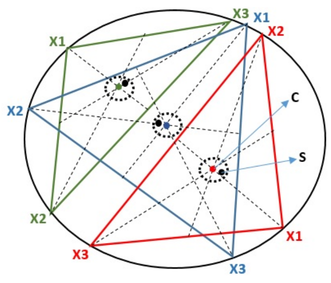


Figure 4.5: Illustration of the center (C) for three sample candidate solutions (x1, x2, and x3, for three sample cases) and the generated candidate solution (S) around the center by utilizing a normal distribution [9].

The main reason behind changing the base vector of the mutation operation to the center of the three points was that the base vector had a crucial impact on generating a more promising candidate solution. Figure 4.6 shows the selected individuals that lead the generated new vector moves toward a better solution.

Algorithm 2 : Algorithmic description of CDE Algorithm

```
1: //$NP$, $D$ and $MAX\_NFC$ are the population size, the problem dimension and the
   maximum number of function evaluations, respectively.
2: Generating the initial population of NP candidate solution randomly.
3: Set the generation counter $Gcounter = 1$;
4: Calculate the number of generations $Gen$=MAX_NFC/NP;
5: while $NFC < MAX\_NFC$ do
6:     for $i$ =1 to $NP$ do
7:         if $Gcounter <= (Gen \times 0.1)$ then
8:             Calculate the center of the three selected candidate solutions randomly $x_{Ncenter}$
               Eq. 4.12.
9:             for $j$ =1 to $D$ do
10:                Calculate the normal distribution of the center for each dimension $(j)$ $x_{Ncenter}$
                   Eq. 4.13.
11:            end for
12:            //Run mutation;
13:            Center-based mutation schemes Eq.(4.14, 4.15, 4.16, 4.17, 4.18).
14:        else
15:            Classical DE mutation schemes Eq.(5.4, 4.4, 4.5, 4.6, 4.7).
16:        end if
17:        Crossover.
18:        Selection.
19:    end for
20:    Increment the generation counter $Gcounter = Gcounter + 1$;
21: end while
```

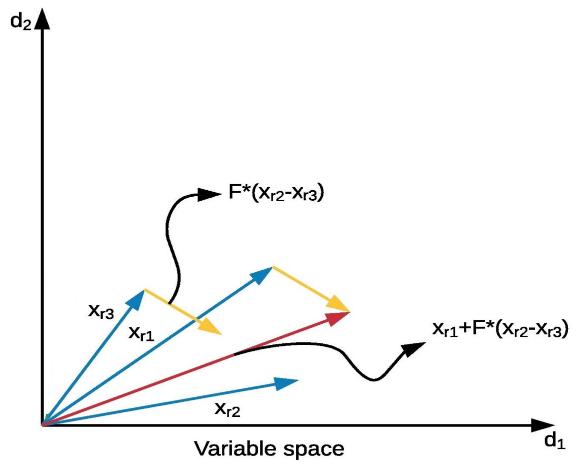


Figure 4.6: Illustration of the importance of the base vector for DE mutation operation [70].

In fact, since in high dimensional space the candidate solutions are in a thin shell near the surface, the exploring for the search space becomes more difficult. Thus, instead of selecting one candidate solution for the base vector as in the basic DE, I got the benefit from utilizing the mean of three candidate solutions, which preserved and increased the diversity of the population on the search space. Therefore, using the average of the three candidate solutions as the mean of the normal distribution was inspired to find the optimal solution. This is due to the central limit theorem, which states that when given a large sample size from a variant population, all of the samples will tend to approach a normal distribution pattern. Furthermore, having the base vector in the mutation will lead to promising regions that help to find better candidate solutions.

4.5.1 Experimental Results

The mean and the standard deviation of the obtained error values by classical DE and CDE algorithms with 1000 dimension for CEC 2013 LSGO benchmark functions mentioned in the section 3.2.2 are summarized in the Tables 4.1, 4.2, 4.3, 4.5, and 4.6. As it can be seen from Tables 4.5 and 4.6 most of CDE binomial schemes outperform classical DE schemes on all functions. While CDE scheme (CD/rand/2) performs worse than the classical DE scheme (DE/rand/2) classical DE scheme (DE/rand/2) on the most functions. It has only two functions winners out of 15 ($f2$ and $f15$), ten functions ties ($f1$, $f3 - f5$, $f7 - f11$, and $f14$) and three functions losses ($f6$ and $f12 - f13$). Besides, as it is revealed from the mentioned tables above, the standard deviation (STD) values of the proposed algorithm compared to classical DE for the winner schemes are much less than classical DE values. The latter shows higher robustness and consistency that occurs from center-based mutation schemes contribution.

In addition, some experiments are conducted on a set of high dimension discrete benchmark functions collected from [8]. Fifteen shifted scalable discrete benchmark functions were selected. These test functions were also shifted randomly to make sure there is no bias towards the center of the search space; i.e., the optimal solution is not in the center of the search space. The benchmark functions are given in Table A.1. Where $fmin$ denotes the function value of the global optimum.

Table 4.1 summarized a fifteen shifted discrete benchmark functions with two dimensions 500 and 1000. As it can be seen from the Table 4.1, for both dimensions 500 and 1000 CDDE/rand/1 outperforms DDE/rand/1 on 14 (f_1-f_{13},f_{15}) functions out of fifteen. Whereas CDDE/rand/1 has worse results than DDE/rand/1 for one (f_{14}) function. Furthermore, another contribution of CDDE/rand/1 and its enhancement to the (DE) algorithm can be seen by the standard deviations (STD) values of the proposed algorithm compared to

Table 4.1 Results of DDE/rand/1 and CDDE/rand/1 algorithms on the 15 discrete benchmark functions for dimensions 500 and 1000; the better results are highlighted in **bold-face**.

		D=500			D=1000		
Function		DDE/rand/1	CDDE/rand/1	Improved accuracy rate	DDE/rand/1	CDDE/rand/1	Improved accuracy rate
f_1	Mean	9.234e+09	**2.646e+08**	3.490e+01	1.066e+11	**5.678e+09**	1.877e+01
	Std	9.135e+08	3.174e+07		6.236e+09	6.096e+08	
f_2	Mean	8.725e+06	**2.314e+05**	3.771e+01	2.402e+07	**1.354e+06**	1.774e+01
	Std	9.332e+05	3.006e+04		1.672e+06	1.403e+05	
f_3	Mean	8.924e+04	**2.196e+03**	4.063e+01	2.462e+05	**1.353e+04**	1.820E+01
	Std	8.501e+03	3.102e+02		1.556e+04	1.045e+03	
f_4	Mean	5.665e+00	**3.406e+00**	1.663e+00	6.997e+00	**4.047e+00**	1.729e+00
	Std	1.267e-01	5.505e-02		8.418e-02	6.275e-02	
f_5	Mean	1.255e+02	**9.486e+01**	1.323e+00	1.075e+03	**7.366e+02**	1.460e+00
	Std	1.101e+01	7.997e+00		5.577e+01	5.022e+01	
f_6	Mean	8.215e+03	**5.417e+03**	1.517e+00	1.922e+04	**1.216e+04**	1.581e+00
	Std	1.798e+02	4.218e+01		2.691e+02	8.806e+01	
f_7	Mean	5.691e+06	**5.592e+06**	1.018e+00	4.566e+07	**4.366e+07**	1.046e+00
	Std	6.653e+04	1.519e+04		9.021e+05	1.683e+05	
f_8	Mean	5.051e+04	**4.986e+04**	1.013e+00	1.954e+05	**1.895e+05**	1.031e+00
	Std	8.050e+02	3.860e+02		3.902e+03	2.037e+03	
f_9	Mean	9.916e+11	**9.837e+11**	1.008e+00	2.580e+13	**2.501e+13**	1.032e+00
	Std	5.898e+09	7.653e+08		3.080e+11	4.074e+10	
f_{10}	Mean	7.067e+03	**4.352e+03**	1.624E+00	2.541e+04	**1.377e+04**	1.846e+00
	Std	3.673e+02	1.380e+02		1.768e+03	4.228e+02	
f_{11}	Mean	1.899e+05	**6.274e+04**	3.027e+00	1.175e+06	**3.288e+05**	3.573e+00
	Std	1.307e+04	4.772e+03		6.999e+04	1.725e+04	
f_{12}	Mean	5.399e+06	**5.289e+06**	1.021e+00	4.017e+07	**3.823e+07**	1.051e+00
	Std	6.831e+04	1.332e+04		1.010e+06	1.677e+05	
f_{13}	Mean	1.219e+04	**1.215e+04**	1.004e+00	9.268e+04	**9.164e+04**	1.011e+00
	Std	5.566e+01	4.536e+00		6.583e+02	9.740e+01	
f_{14}	Mean	**2.136e+01**	2.139e+01	9.989e-01	**2.140e+01**	2.142e+01	9.989e-01
	Std	2.751e-02	2.902e-02		2.647e-02	2.609e-02	
f_{15}	Mean	3.245e+06	**2.820e+06**	1.151e+00	6.738e+06	**5.792e+06**	1.163e+00
	Std	1.749e+05	1.056e+05		3.687e+05	2.236e+05	
w/t/l		-	14/0/1	**Ave.**8.640e+00	-	14/0/1	**Ave.**4.815e+00

DE for the both Tables 4.1. As it can be observed, the STD values of the proposed algorithm are much lower than DE which shows a higher consistency/ robustness of results by CDDE/rand/1, compared to DE. The higher consistency of CDDE/rand/1 express much more accurate results.

In order to illustrate exactly which center schemes performed very well, the improved accuracy rate (Irate) is calculated as follow:

$$I_{rate} = \frac{Error\ of\ Classical\ algorithm}{Error\ of\ Center\text{-}based\ algorithm} \tag{4.19}$$

Table 4.2 Results of DE and CDE algorithms for Exponential crossover on the CEC 2013 benchmark functions for dimension 1000; Symbols '$+$','$-$' and '$=$' denote the proposed algorithms are better than, worse than, or similar to the compared algorithm, respectively.

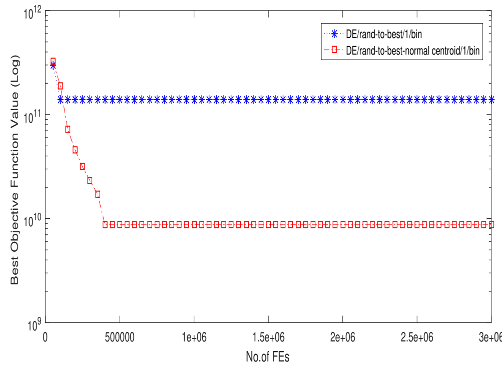
Functions		DE/rand/1	CDE		DE/best/1	CDE		CDE C3best		DE/rand-to-best/1	CDE	
f1	Mean	3.363e+11	**2.953e+11**	+	2.206e+11	2.719e+11	-	2.497e+11	-	2.222e+11	**2.022e+11**	+
	Std	1.796e+10	1.468e+10		4.175e+10	4.145e+10		3.957e+10		2.811e+10	2.099e+10	
f2	Mean	1.012e+05	**8.683e+04**	+	7.400e+04	8.398e+04	-	8.102e+04	-	6.355e+04	**6.029e+04**	+
	Std	4.793e+03	3.185e+03		7.544e+03	9.029e+03		6.148e+03		7.704e+03	5.487e+03	
f3	Mean	2.151e+01	**2.145e+01**	+	2.141e+01	2.145e+01	-	2.144e+01	-	2.133e+01	2.133e+01	=
	Std	1.139e-02	1.568e-02		2.679e-02	2.436e-02		2.163e-02		4.545e-02	4.859e-02	
f4	Mean	3.232e+13	3.426e+13	=	1.922e+13	2.120e+13	=	2.244e+13	-	2.122e+13	2.032e+13	=
	Std	1.007e+13	1.072e+13		8.477e+12	8.843e+12		8.091e+12		8.674e+12	7.474e+12	
f5	Mean	6.975e+07	**6.130e+07**	+	4.895e+07	5.474e+07	-	5.035e+07	=	4.679e+07	4.567e+07	=
	Std	6.187e+06	6.194e+06		7.901e+06	9.128e+06		9.263e+06		5.974e+06	7.488e+06	
f6	Mean	1.063e+06	**1.061e+06**	+	1.055e+06	1.058e+06	-	1.057e+06	-	1.055e+06	1.057e+06	-
	Std	2.334e+03	2.517e+03		7.053e+03	3.824e+03		4.806e+03		5.560e+03	5.265e+03	
f7	Mean	1.038e+16	**6.212e+15**	+	2.769e+15	4.486e+15	-	3.490e+15	=	3.198e+15	2.719e+15	=
	Std	7.607e+15	4.689e+15		2.959e+15	5.148e+15		4.337e+15		2.609e+15	2.787e+15	
f8	Mean	1.798e+18	1.806e+18	=	1.153e+18	1.201e+18	=	1.305e+18	=	1.145e+18	1.278e+18	=
	Std	6.327e+17	5.729e+17		4.017e+17	4.627e+17		5.530e+17		4.449e+17	4.287e+17	
f9	Mean	6.026e+09	**5.431e+09**	+	4.064e+09	4.249e+09	=	4.047e+09	=	4.103e+09	3.988e+09	=
	Std	7.186e+08	6.307e+08		8.687e+08	9.431e+08		6.563e+08		6.794e+08	9.541e+08	
f10	Mean	9.557e+07	9.550e+07	=	9.429e+07	9.480e+07	=	9.504e+07	=	9.519e+07	9.511e+07	=
	Std	5.567e+05	5.020e+05		2.382e+06	1.789e+06		9.282e+05		7.681e+05	8.417e+05	
f11	Mean	5.653e+17	3.967e+17	=	2.250e+17	2.559e+17	-	2.190e+17	=	1.341e+17	1.458e+17	=
	Std	4.176e+17	3.201e+17		4.397e+17	1.909e+17		3.365e+17		1.968e+17	1.271e+17	
f12	Mean	6.451e+12	**5.255e+12**	+	4.885e+12	5.248e+12	-	5.030e+12	=	3.202e+12	**3.014e+12**	+
	Std	4.273e+11	2.437e+11		3.820e+11	4.002e+11		3.262e+11		4.524e+11	4.702e+11	
f13	Mean	9.182e+17	**5.116e+17**	+	2.068e+17	3.601e+17	-	2.407e+17	=	2.056e+17	2.253e+17	=
	Std	7.447e+17	3.340e+17		2.104e+17	3.204e+17		2.193e+17		2.130e+17	2.148e+17	
f14	Mean	8.944e+17	7.942e+17	=	3.198e+17	4.977e+17	=	3.984e+17	=	2.942e+17	2.599e+17	=
	Std	5.464e+17	5.220e+17		3.298e+17	5.339e+17		3.525e+17		2.914e+17	2.117e+17	
f15	Mean	4.636e+17	**1.840e+17**	+	4.524e+16	1.329e+17	-	9.716e+16	-	3.593e+16	3.784e+16	=
	Std	2.074e+17	9.024e+16		3.129e+16	1.009e+17		8.438e+16		5.259e+16	5.383e+16	
	w/t/l		10\5\0			0\5\10		0\9\6			3\11\1	

As we can see, the percentage confirms that our proposed algorithm achieved better results than basic DE for all functions for CEC 2013 LSGO, also it obtained better results for 14 out of 15 functions for the shifted discrete benchmark functions.

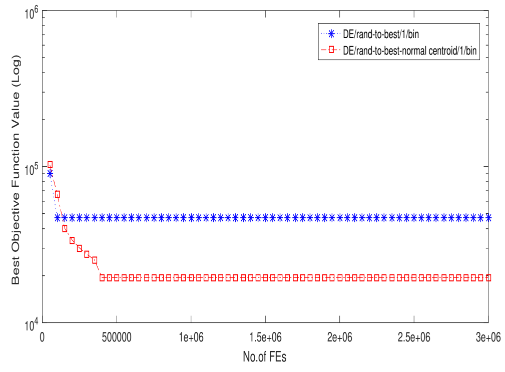
Tables 4.4 and 4.7 show the improvement accuracy rate for the mutation schemes. Table.4.7 shows the $Irate$ of the mutation schemes for the binomial crossover that achieved better than classical DE, which are (CDE/3best/1/bin) on average 2.057e+06 times and (CDE/bestr/1/bin) on average 3.187e+06 times.

Table 4.3 Results of DE and CDE algorithms for Exponential crossover on the CEC 2013 benchmark functions for dimension 1000; Symbols '+','−' and '=' denote the proposed algorithms are better than, worse than, or similar to the compared algorithm, respectively.

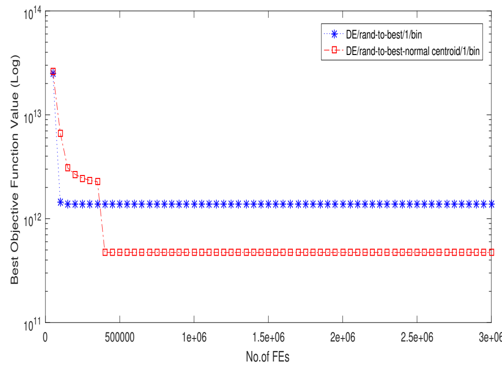
Functions		DE/best/2	CDE		CDE C 3best		DE/rand/2	CDE	
f1	Mean	3.097e+11	3.215e+11	-	3.123e+11	=	3.477e+11	**3.343e+11**	+
	Std	1.718e+10	2.241e+10		2.172e+10		1.728e+10	1.489e+10	
f2	Mean	9.500e+04	1.008e+05	-	9.840e+04	-	1.094e+05	**1.011e+05**	+
	Std	5.233e+03	4.622e+03		4.693e+03		3.044e+03	4.180e+03	
f3	Mean	2.149e+01	2.150e+01	-	2.149e+01	=	2.152e+01	**2.150e+01**	+
	Std	1.789e-02	1.487e-02		1.603e-02		1.050e-02	1.265e-02	
f4	Mean	2.949e+13	2.814e+13	=	2.881e+13	=	3.560e+13	3.354e+13	=
	Std	1.157e+13	1.037e+13		9.310e+12		1.223e+13	1.152e+13	
f5	Mean	6.255e+07	6.560e+07	-	6.300e+07	=	6.958e+07	**6.712e+07**	+
	Std	6.331e+06	6.646e+06		6.645e+06		6.121e+06	5.769e+06	
f6	Mean	1.061e+06	1.061e+06	=	1.061e+06	=	1.063e+06	**1.061e+06**	+
	Std	3.376e+03	2.642e+03		2.496e+03		1.893e+03	2.503e+03	
f7	Mean	7.872e+15	8.187e+15	=	7.054e+15	=	1.255e+16	1.090e+16	=
	Std	7.226e+15	6.337e+15		6.240e+15		9.974e+15	8.573e+15	
f8	Mean	1.575e+18	1.561e+18	=	1.709e+18	=	2.044e+18	1.957e+18	=
	Std	5.329e+17	4.701e+17		5.542e+17		5.281e+17	6.611e+17	
f9	Mean	5.083e+09	5.184e+09	=	5.116e+09	=	5.855e+09	5.829e+09	=
	Std	7.454e+08	7.723e+08		7.021e+08		7.128e+08	7.019e+08	
f10	Mean	9.499e+07	9.541e+07	=	9.532e+07	=	9.542e+07	9.547e+07	=
	Std	1.187e+06	6.653e+05		6.263e+05		6.769e+05	5.630e+05	
f11	Mean	4.598e+17	5.170e+17	=	3.704e+17	=	6.358e+17	5.446e+17	=
	Std	4.106e+17	3.866e+17		2.924e+17		4.375e+17	3.542e+17	
f12	Mean	6.489e+12	6.643e+12	-	6.517e+12	=	7.299e+12	**6.634e+12**	+
	Std	1.902e+11	2.315e+11		2.451e+11		2.349e+11	2.114e+11	
f13	Mean	5.877e+17	6.459e+17	=	5.618e+17	=	9.571e+17	8.729e+17	=
	Std	4.682e+17	5.067e+17		4.322e+17		6.126e+17	7.141e+17	
f14	Mean	7.876e+17	7.019e+17	=	6.694e+17	=	1.251e+18	1.116e+18	=
	Std	5.198e+17	4.603e+17		3.446e+17		5.848e+17	5.714e+17	
f15	Mean	3.397e+17	4.593e+17	-	3.154e+17	=	6.173e+17	**4.449e+17**	+
	Std	2.060e+17	2.161e+17		1.702e+17		3.047e+17	2.207e+17	
	w/t/l		0\9\6		0\14\1			7\8\0	



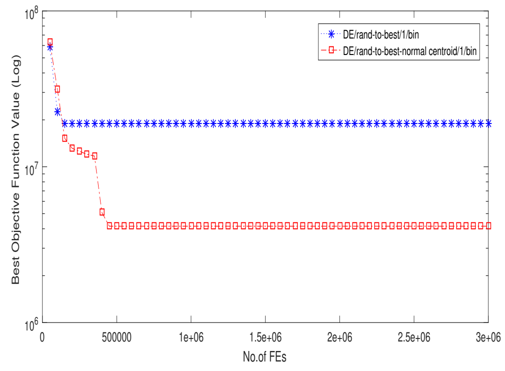
(a) f_1



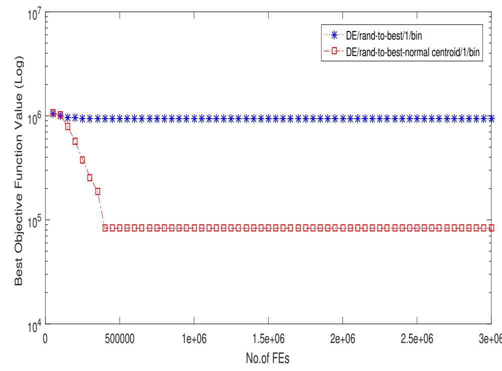
(b) f_2



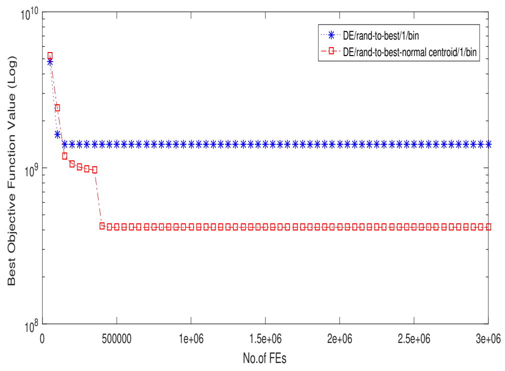
(c) f_4



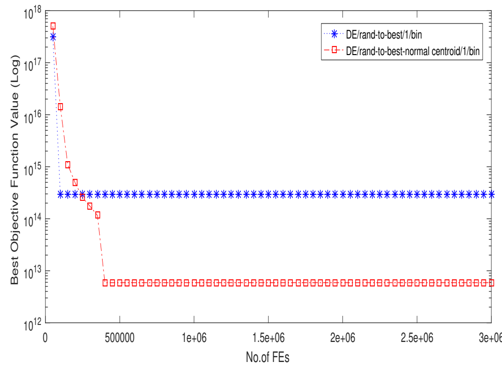
(d) f_5



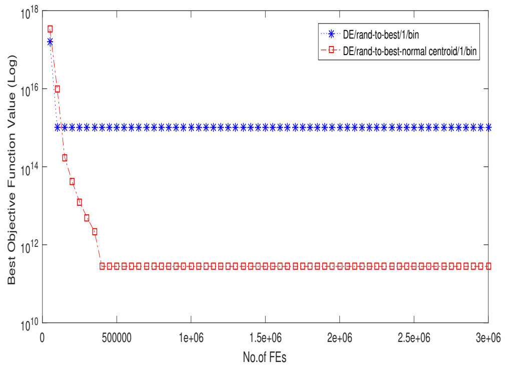
(e) f_6



(f) f_9

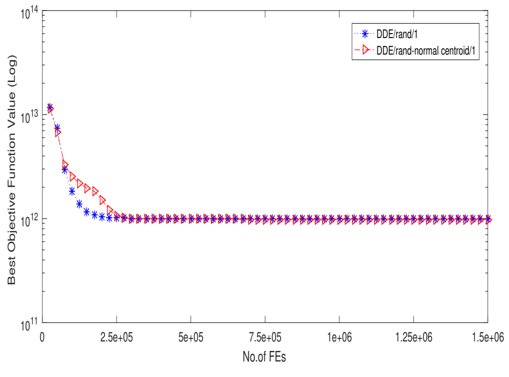


(g) f_{14}

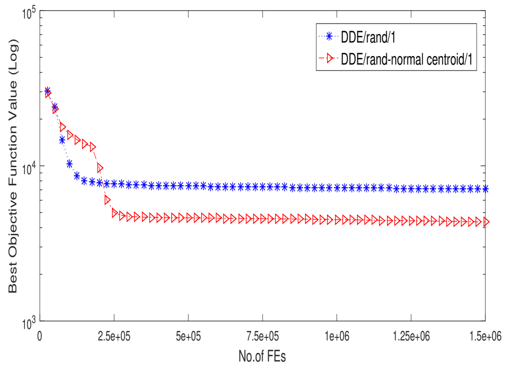


(h) f_{15}

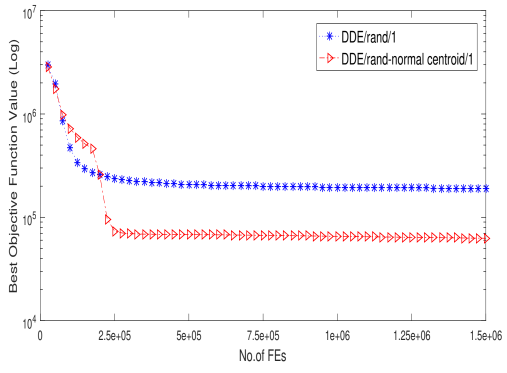
Figure 4.7: Convergence plots of some functions for CEC 2013 benchmark problems set with D=1000. The results were averaged over 51 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations (FEs).



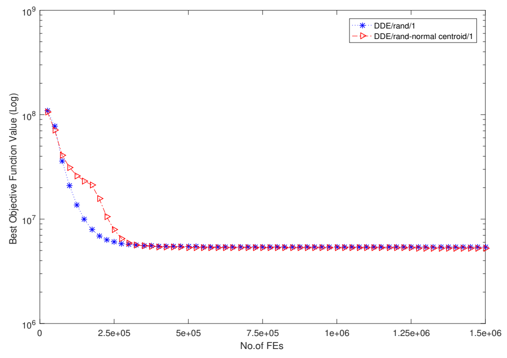
(a) f_9



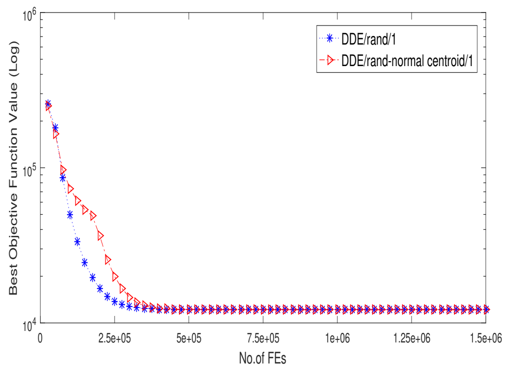
(b) f_{10}



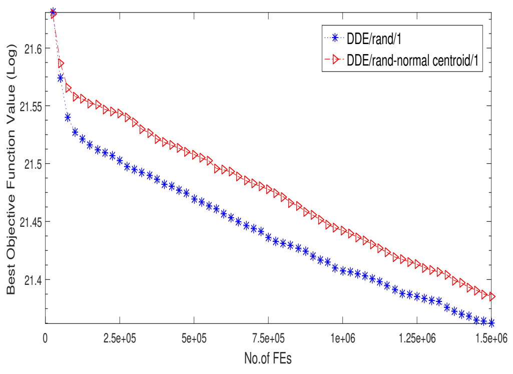
(c) f_{11}



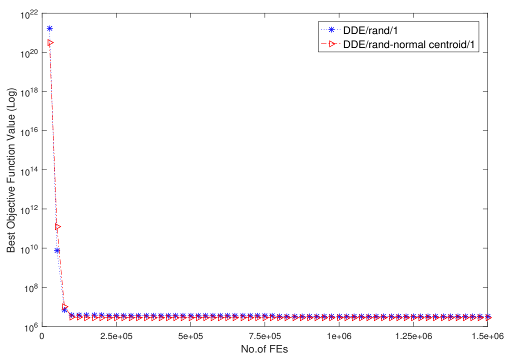
(d) f_{12}



(e) f_{13}



(f) f_{14}



(g) f_{15}

Figure 4.8: Convergence plots of f_9, f_10,f_{11},f_{12},f_{13},f_{14} and f_{15} functions for discrete benchmark problems set with D=500. The results were averaged over 51 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations (FEs).

Table 4.4 Improved accuracy rate for exponential crossover schemes

Functions	DE/rand/1	DE/best/1	CDE 10% 3best	DE/rand-to-best/1	DE/best/2	CDE 10% 3best	DE/rand/2vs CDE
f1	1.139e+00	8.112e-01	8.833e-01	1.099e+00	9.634e-01	9.915e-01	1.040e+00
f2	1.166e+00	8.812e-01	9.134e-01	1.054e+00	9.428e-01	9.655e-01	1.083e+00
f3	1.003e+00	9.980e-01	9.983e-01	1.000e+00	9.995e-01	9.998e-01	1.001e+00
f4	9.432e-01	9.066e-01	8.566e-01	1.044e+00	1.048e+00	1.024e+00	1.062e+00
f5	1.138e+00	8.941e-01	9.721e-01	1.025e+00	9.534e-01	9.928e-01	1.037e+00
f6	1.002e+00	9.971e-01	9.984e-01	9.984e-01	9.999e-01	1.000e+00	1.001e+00
f7	1.671e+00	6.172e-01	7.933e-01	1.176e+00	9.615e-01	1.116e+00	1.151e+00
f8	9.955e-01	9.599e-01	8.839e-01	8.961e-01	1.009e+00	9.213e-01	1.044e+00
f9	1.109e+00	9.564e-01	1.004e+00	1.029e+00	9.804e-01	9.936e-01	1.004e+00
f10	1.001e+00	9.946e-01	9.921e-01	1.001e+00	9.956e-01	9.966e-01	9.995e-01
f11	1.425e+00	8.791e-01	1.027e+00	9.198e-01	8.895e-01	1.241e+00	1.167e+00
f12	1.228e+00	9.307e-01	9.712e-01	1.062e+00	9.769e-01	9.958e-01	1.100e+00
f13	1.795e+00	5.744e-01	8.594e-01	9.126e-01	9.100e-01	1.046e+00	1.096e+00
f14	1.126e+00	6.425e-01	8.027e-01	1.132e+00	1.122e+00	1.177e+00	1.121e+00
f15	2.520e+00	3.403e-01	4.656e-01	9.494e-01	7.395e-01	1.077e+00	1.388e+00
Avg.	**1.284e+00**	8.255e-01	8.948e-01	1.020e+00	9.661e-01	1.036e+00	**1.086e+00**

The behaviors of both algorithms are depicted in Figure 4.7. According to these figures, CDE performs better on the most functions for CEC 2013 LSGO benchmark functions. As it can be observed, the proposed algorithm accelerates the convergence speed for the most functions during the exploration part which assisted the algorithm to reach the optimal solution.

4.6 Case Study Two: Center-based SHADE

In this section, first, a background review of SHADE algorithm is explained. Second, a center-based scheme for the SHADE algorithm is proposed. Finally, the experimental results are analyzed.

Table 4.5 Results of DE and CDE algorithms for binomial crossover on the CEC 2013 benchmark functions for dimension 1000; Symbols '$+$','$-$' and '$=$' denote the proposed algorithms are better than, worse than, or similar to the compared algorithm, respectively.

Functions		DE/rand/1	CDE		DE/best/1	CDE		CDE C3BEST		DE/rand-to-best/1	CDE	
f_1	Mean	1.923e+07	**7.345e+04**	$+$	1.497e+11	**3.546e+07**	$+$	**2.993e+07**	$+$	1.402e+11	**8.895e+09**	$+$
	Std	4.531e+07	5.306e+04		1.328e+10	4.375e+06		2.200e+07		9.305e+09	6.885e+08	
f_2	Mean	1.920e+04	**3.315e+03**	$+$	5.565e+04	**1.177e+04**	$+$	**1.033e+04**	$+$	4.691e+04	**1.948e+04**	$+$
	Std	9.599e+02	1.241e+02		2.617e+03	2.231e+02		6.460e+02		1.689e+03	5.341e+02	
f_3	Mean	1.888e+01	**9.691e+00**	$+$	2.106e+01	**1.398e+01**	$+$	**1.715e+01**	$+$	2.089e+01	**1.985e+01**	$+$
	Std	7.052e-02	3.603e-01		3.263e-02	2.238e-01		6.293e-01		3.200e-02	7.799e-02	
f_4	Mean	3.707e+10	**2.287e+10**	$+$	1.555e+12	**3.815e+11**	$+$	**1.456e+11**	$+$	1.384e+12	**4.733e+11**	$+$
	Std	1.572e+10	8.276e+09		4.418e+11	3.300e+10		9.781e+10		3.480e+11	6.065e+10	
f_5	Mean	2.207e+06	**1.048e+06**	$+$	2.197e+07	**5.194e+06**	$+$	**5.941e+06**	$+$	1.897e+07	**4.147e+06**	$+$
	Std	4.606e+05	1.897e+05		2.639e+06	5.959e+05		6.380e+05		1.918e+06	4.662e+05	
f_6	Mean	1.047e+05	**1.529e+01**	$+$	9.786e+05	**2.024e+01**	$+$	**3.528e+03**	$+$	9.374e+05	**8.321e+04**	$+$
	Std	3.853e+04	3.963e-01		1.369e+04	1.880e-01		1.428e+04		1.554e+04	9.262e+03	
f_7	Mean	1.125e+08	**8.820e+07**	$+$	3.819e+12	**2.522e+09**	$+$	**1.602e+09**	$+$	2.959e+12	**1.437e+10**	$+$
	Std	4.584e+07	2.707e+07		2.070e+12	5.718e+08		4.589e+08		1.426e+12	5.033e+09	
f_8	Mean	3.047e+14	**2.335e+14**	$+$	1.995e+16	**1.038e+15**	$+$	**1.103e+15**	$+$	1.466e+16	**1.286e+15**	$+$
	Std	1.289e+14	1.204e+14		1.191e+16	6.376e+14		7.771e+14		1.301e+16	8.111e+14	
f_9	Mean	2.069e+08	**1.002e+08**	$+$	1.546e+09	**4.190e+08**	$+$	**4.952e+08**	$+$	1.408e+09	**4.065e+08**	$+$
	Std	2.984e+07	1.226e+07		2.039e+08	3.457e+07		4.977e+07		1.277e+08	3.475e+07	
f_{10}	Mean	2.810e+03	**2.255e+02**	$+$	6.223e+07	**3.400e+02**	$+$	**4.317e+02**	$+$	5.498e+07	**5.891e+03**	$+$
	Std	6.446e+03	1.641e+01		8.399e+06	1.325e+01		1.064e+02		6.994e+06	6.154e+02	
f_{11}	Mean	1.509e+11	**4.983e+10**	$+$	3.734e+14	**5.383e+11**	$+$	**1.188e+11**	$+$	2.916e+14	**6.351e+12**	$+$
	Std	8.136e+10	2.472e+10		1.641e+14	2.150e+11		6.025e+10		1.198e+14	2.041e+12	
f_{12}	Mean	2.296e+09	**1.465e+07**	$+$	3.128e+12	**1.048e+10**	$+$	**4.829e+09**	$+$	2.144e+12	**4.569e+11**	$+$
	Std	2.583e+09	2.836e+07		1.732e+11	1.419e+09		2.822e+09		1.324e+11	2.680e+10	
f_{13}	Mean	7.290e+09	**6.118e+09**	$+$	2.802e+14	**1.365e+11**	$+$	**2.099e+10**	$+$	2.044e+14	**1.235e+12**	$+$
	Std	1.741e+09	1.171e+09		2.216e+14	3.138e+10		6.264e+09		9.796e+13	4.474e+11	
f_{14}	Mean	2.027e+11	**1.048e+11**	$+$	3.141e+14	**1.519e+12**	$+$	**2.243e+11**	$+$	2.906e+14	**5.918e+12**	$+$
	Std	7.332e+10	3.695e+10		1.773e+14	2.757e+11		8.883e+10		1.216e+14	1.128e+12	
f_{15}	Mean	2.462e+10	**5.525e+06**	$+$	3.002e+15	**6.311e+07**	$+$	**9.782e+07**	$+$	9.930e+14	**2.877e+11**	$+$
	Std	6.381e+10	1.157e+06		1.424e+15	1.252e+07		5.169e+07		2.941e+14	9.618e+10	
	w/t/l		15\0\0			15\0\0		15\0\0			15\0\0	

Table 4.6 Results of DE and CDE algorithms for binomial crossover on the CEC 2013 benchmark functions for dimension 1000; Symbols '$+$','$-$' and '$=$' denote the proposed algorithms are better than, worse than, or similar to the compared algorithm, respectively.

Functions		DE/best/2	CDE		CDE C3BEST		DE/rand/2	CDE	
f_1	Mean	3.634E+07	3.065E+07	=	2.475E+07	=	1.699E+08	1.856E+08	=
	Std	8.040E+07	4.407E+07		2.913E+07		1.146E+08	8.546E+07	
f_2	Mean	2.973E+04	**2.109E+04**	+	**1.679E+04**	+	1.128E+04	**1.050E+04**	+
	Std	1.301E+03	7.748E+02		6.403E+02		5.217E+02	4.863E+02	
f_3	Mean	1.962E+01	**1.935E+01**	+	**1.900E+01**	+	1.764E+01	1.756E+01	=
	Std	5.271E-01	4.414E-01		6.498E-02		3.578E-01	4.165E-01	
f_4	Mean	5.160E+10	**3.483E+10**	+	**3.524E+10**	+	2.848E+10	2.993E+10	=
	Std	2.284E+10	1.482E+10		1.329E+10		1.292E+10	1.047E+10	
f_5	Mean	6.506E+06	**2.460E+06**	+	**1.988E+06**	+	9.460E+06	9.456E+06	=
	Std	1.154E+06	3.657E+05		3.185E+05		3.598E+05	3.043E+05	
f_6	Mean	3.399E+05	**2.533E+05**	+	**1.440E+05**	+	2.217E+01	2.251E+01	-
	Std	6.351E+04	3.392E+04		1.454E+04		5.368E-01	5.434E-01	
f_7	Mean	1.533E+08	1.553E+08	=	1.439E+08	=	2.411E+09	2.721E+09	=
	Std	6.856E+07	4.598E+07		7.380E+07		6.530E+08	8.446E+08	
f_8	Mean	3.275E+14	**2.529E+14**	+	**2.613E+14**	+	9.419E+13	9.240E+13	=
	Std	1.286E+14	1.031E+14		1.047E+14		4.336E+13	4.870E+13	
f_9	Mean	4.941E+08	**2.531E+08**	+	**2.220E+08**	+	7.777E+08	7.744E+08	=
	Std	6.835E+07	2.898E+07		2.311E+07		2.995E+07	2.944E+07	
f_{10}	Mean	2.629E+06	**1.452E+04**	+	**6.249E+03**	+	3.504E+02	3.638E+02	=
	Std	3.252E+06	1.699E+04		1.486E+03		8.954E+01	8.430E+01	
f_{11}	Mean	7.295E+10	6.995E+10	=	7.951E+10	=	1.526E+11	1.625E+11	=
	Std	4.077E+10	4.352E+10		6.731E+10		5.987E+10	7.814E+10	
f_{12}	Mean	1.531E+09	1.318E+09	=	8.617E+08	=	2.654E+10	3.055E+10	-
	Std	2.896E+09	1.840E+09		1.138E+09		6.002E+09	5.499E+09	
f_{13}	Mean	6.360E+09	6.036E+09	=	5.923E+09	=	1.811E+10	2.269E+10	-
	Std	1.861E+09	1.924E+09		1.662E+09		4.340E+09	7.513E+09	
f_{14}	Mean	1.088E+11	1.060E+11	=	1.079E+11	=	2.467E+11	2.792E+11	=
	Std	4.340E+10	3.381E+10		4.746E+10		7.720E+10	9.879E+10	
f_{15}	Mean	4.233E+12	**1.244E+11**	+	**8.867E+09**	+	7.933E+08	**4.159E+08**	+
	Std	2.812E+12	8.132E+10		1.123E+10		4.369E+08	4.447E+08	
	w/t/l		9\6\0		9\6\0			2\10\3	

Table 4.7 Improved accuracy rate for binomial crossover schemes

Functions	DE/rand/1	DE/best/1	CDE 3best	DE/rand-to-best/1	DE/best/2	CDE 3best	DE/rand/2 vs CDE
f_1	2.617e+02	4.224e+03	5.004e+03	1.576e+01	1.186e+00	1.468e+00	9.154e-01
f_2	5.793e+00	4.729e+00	5.385e+00	2.408e+00	1.410e+00	1.771e+00	1.075e+00
f_3	1.948e+00	1.506e+00	1.228e+00	1.052e+00	1.014e+00	1.032e+00	1.005e+00
f_4	1.621e+00	4.077e+00	1.068e+01	2.923e+00	1.482e+00	1.464e+00	9.516e-01
f_5	2.107e+00	4.230e+00	3.698e+00	4.576e+00	2.645e+00	3.273e+00	1.000e+00
f_6	6.848e+03	4.836e+04	2.774e+02	1.127e+01	1.342e+00	2.361e+00	9.849e-01
f_7	1.276e+00	1.514e+03	2.383e+03	2.059e+02	9.870e-01	1.065e+00	8.862e-01
f_8	1.305e+00	1.922e+01	1.809e+01	1.140e+01	1.295e+00	1.253e+00	1.019e+00
f_9	2.065e+00	3.689e+00	3.121e+00	3.464e+00	1.953e+00	2.226e+00	1.004e+00
f_{10}	1.246e+01	1.830e+05	1.441e+05	9.333e+03	1.810e+02	4.206e+02	9.633e-01
f_{11}	3.029e+00	6.936e+02	3.143e+03	4.591e+01	1.043e+00	9.175e-01	9.391e-01
f_{12}	1.567e+02	2.984e+02	6.478e+02	4.694e+00	1.162e+00	1.777e+00	8.688e-01
f_{13}	1.192e+00	2.053e+03	1.335e+04	1.656e+02	1.054e+00	1.074e+00	7.981e-01
f_{14}	1.934e+00	2.068e+02	1.400e+03	4.910e+01	1.027e+00	1.008e+00	8.833e-01
f_{15}	4.455e+03	4.757e+07	3.069e+07	3.452e+03	3.404e+01	4.774e+02	1.907e+00
Avg.	7.838e+02	**3.187e+06**	**2.057e+06**	8.872e+02	1.551e+01	6.125e+01	1.013e+00

Success-History Based Parameter Adaptation for Differential Evolution (SHADE): SHADE [71] was proposed as an enhancement over a previous DE version, which is JADE [72]. SHADE uses the same current-to-pbest/1 mutation strategy used by JADE as well as the adoption of an external archive but proposes a new method for generating the CR and F parameters.

In SHADE, the authors proposed introducing a historical memory with H entries to save the control parameters CR and F. Initially, all memory contents are initialized to 0.5 (M_{CR}, M_F). In each generation G, when updating an individual x_i, the parameters CR_i and F_i are generated by selecting a random entry r_i in memory and then applying the following equations:

$$CR_i = rand_{ni}(M_{CR,r_i}, 0.1), \tag{4.20}$$

$$F_i = rand_{ci}(M_{F,r_i}, 0.1), \tag{4.21}$$

where $rand_n$ and $rand_c$ generate random numbers following the normal and cauchy mutations, respectively. If the values of CR_i and F_i are generated outside the range (0,1), they are adjusted/regenerated following the same approach in JADE.

Values for CR_i and F_i used by successfully updated individuals in each generation $G+1$ are stored in S_{CR} and S_F. At the end of the generation, memory contents are updated as follows:

$$M_{CR,k,G+1} = \begin{cases} mean_{WA}(S_{CR}) & if\ S_{CR} \neq \phi \\ M_{CR,k,G} & otherwise \end{cases} \tag{4.22}$$

$$M_{F,k,G+1} = \begin{cases} mean_{WL}(S_F) & if\ S_F \neq \phi \\ M_{F,k,G} & otherwise, \end{cases} \tag{4.23}$$

where k is the number of the element being updated in memory, which is initially set to 1, and $mean_{WA}$ and $mean_{WL}$ stand for the weighted arithmetic and weighted Lehmar means. If no individuals are successfully updated in the current generation $S_{CR} = S_F = \phi$, memory contents are not modified.

The authors also introduced an adaptive strategy to adjust the parameter p, which is used to control the greediness of the current-to-pbest/1 mutation strategy. In SHADE, the parameter is updated as follows:

$$p_i = rand(p_{min}, 0.2), \tag{4.24}$$

where $p_{min} = 2/NP$ to guarantee to select at least 2 individuals including the pbest one and 0.2 is the maximum value previously suggested in JADE.

Experiments run on the CEC 2013 benchmarks have shown that SHADE outperforms other state-of-the-art DE variants including JADE and CoDE [73].

SHADE algorithm utilized $current - to - pbest/1$ mutation strategy which is a variant of the $current - to - best/1$ strategy. It directs the generation of the selected vectors towards the best member of the population as follows:

$$v_i^t = x_i^t + F_i \,.\, (x_{pbest}^p - x_i^t) + F_i \,.\, (x_{r1}^t - x_{r2}^t), \tag{4.25}$$

where x_{pbest}^P is a randomly selected vector from the top $NP \times p$ (p in the range $[0, 1]$) members in each generation. x_i^t and x_{r1}^t are selected from the current population P. While x_{r2}^t is randomly chosen from the union of the current population and the archive A, $P \cup A$. The mutation factor F_i is used by individual x_i. The control parameter p is adjusting the greediness of the $current - to - pbest/1$ strategy in order to balance the exploration and the exploitation of the search space [72].

The proposed CSHADE algorithm: In the proposed algorithm, a novel center-based SHADE is introduced, called CSHADE. In this scheme, a new base vector is generated by utilizing the average of three randomly selected candidate solutions as a mean value of a normal distribution. Algorithm 3 summarized CSHADE algorithm. The algorithm begins with an initial population that has NP candidate solutions. In the new mutation operation, the normal distribution is utilized to generate a new base vector for a new mutation scheme as follows. First, the new mutation operation selects three randomly candidate solutions

x^t_{r1}, x^t_{r2}, x^t_{r3} from the current population. The average of these three vectors x_{center} (line 16) is calculated as mentioned in equation 4.12:

The average value of x_{center} is considered to be the mean (μ) of the normal distribution, which generates a new solution around the mean of each dimension. Therefore, the normal distribution generates a new point to be the base vector as mentioned in equation 4.13 (line 19). Thus, the new mutant vector v_i is obtained by using the following equation (line 22):

$$v_i = x_{Ncenter} + F_i \, . \, (x^p_{pbest} - x^t_i) + F_i \, . \, (x^t_{r1} - x^t_{r2}) \tag{4.26}$$

The proposed mutation scheme is utilized only over 10% of the beginning of generations (lines 15-22) to enhance the exploration of the search space and allow finding out the promising regions. For the rest of the generations, SHADE mutation scheme ($current\text{-}to\text{-}pbest/1$) is applied.

4.6.1 Experimental Results

In order to investigate the performance of the proposed algorithm versus SHADE, a comprehensive experiment was conducted. The proposed CSHADE algorithm was applied and tested on CEC 2010 LSGO [74] and CEC 2013 LSGO [17] benchmark functions with dimension 1000. The CEC 2010 consists of 20 functions (f_1-f_{20}) in five different groups of function types: Functions (f_1-f_3) are fully separable, functions (f_4-f_8) have one separable group of 950 variables and one group of 50 non-separable variables, functions (f_9-f_{13}) have one separable group of 500 variables and ten groups of 50 non-separable variables each, functions (f_{14}-f_{18}) have multiple non-separable groups, functions (f_{19}-f_{20}) are fully

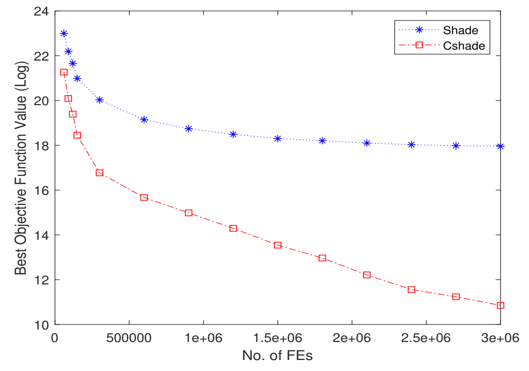
Algorithm 3 : Algorithmic description of CSHADE Algorithm

```
1: //$NP$, $D$ and $MAX\_NFC$ are the population size, the problem dimension and the
   maximum number of function evaluations, respectively.
2: Generate the initial population of $NP$ randomly.
3: Set the counter $k = 1$, and the generation counter $Gcounter = 1$;
4: $M_{CR} = M_F$ =0.5;
5: Archive $A = \phi$, $p_{min} = 2/NP$ ;
6: Calculate the number of generations $Gen$=$MAX\_NFC/NP$;
7: while $NFC < MAX\_NFC$ do
8:     $S_{CR} = \phi$, $S_F = \phi$;
9:     for $i = 1$ to $NP$ do
10:        $r_i$ = Select from $[1, H]$ randomly; //$H$ is a historical memory index.
11:        $CR_i$ = rand$n_i(M_{CR}, r_i, 0.1)$;
12:        $F_i$ = rand$c_i(M_F, r_i, 0.1)$;
13:        $p_i$ = rand$[p_{min}, 0.2]$;
14:        //Run mutation;
15:        if $Gcounter <= (Gen \times 0.1)$ then
16:            Calculate the center of the three selected candidate solutions randomly $x_{Ncenter}$
               Eq. 4.12
17:            for $j = 1$ to $D$ do
18:                Calculate the normal distribution of the center for each dimension $(j)$ $x_{Ncenter}$
                   Eq. 4.13.
19:            end for
20:            $v_i = x_{Ncenter} + F_i . (x^p_{pbest} - x^t_i) + F_i . (x^t_{r1} - x^t_{r2})$
21:        else
22:            $v^t_i = x^t_i + F_i . (x^p_{pbest} - x^t_i) + F_i . (x^t_{r1} - x^t_{r2})$
23:        end if
24:    end for
25:    for $i = 1$ to $NP$ do
26:        if $f(v^t_i) \leq f(xi; t)$ then
27:            $x^{t+1}_i = v^t_i$;
28:        else
29:            $x^{t+1}_i = x^t_i$;
30:        end if
31:        if $f(v^t_i) < f(xi; t)$ then
32:            $x^t_i \longrightarrow A$;
33:            $CR^t_i \longrightarrow S_{CR}$, $F^t_i \longrightarrow S_F$ ;
34:        end if
35:    end for
36:    Remove solutions from $A$ randomly so that $|A| \leq NP$;
37:    if $S_{CR} \neq \phi$, and $S_F \neq \phi$ then
38:        Update $M_{CR,k}$,$M_{F,k}$ based on $S_{CR}$, $S_F$; // Eq.(4.22, 4.23).
39:        $k = k + 1$;
40:        if $k > H$ then
41:            $k = 1$;
42:        end if
43:    end if
44:    Increment the generation counter $Gcounter = Gcounter + 1$;
45: end while
```

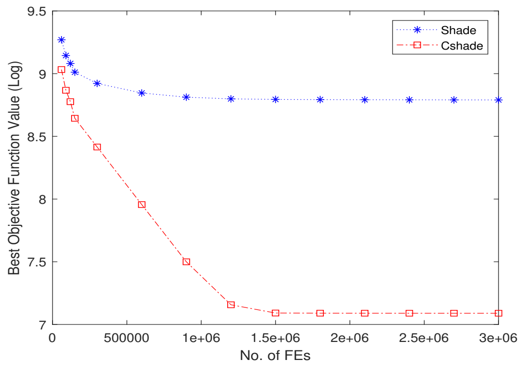
non-separable. Furthermore, some experiments are tested on CEC 2013 LSGO benchmark functions.

The results of SHADE and CSHADE on CEC 2010 LSGO are summarized in Table 4.8. As it can be observed from this table, CSHADE performs significantly better than SHADE on 15 (f_1-f_2,f_5-f_6,f_9-f_{10},f_{11}-f_{18} and f_{20}), whereas, CSHADE has worse results than SHADE for five functions (f_3-f_4, f_7-f_8 and f_{19}).

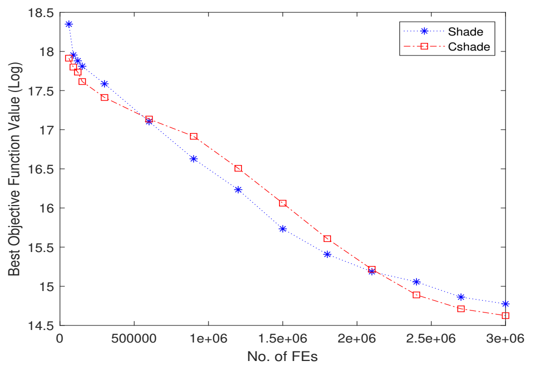
Table 4.9 summarized the results of CEC 2013 LSGO benchmark functions. As it can be seen, CSHADE outperforms SHADE on 10 (f_1-f_2, f_5-f_6 and f_9-f_{14}) functions out of fifteen functions. While CSHADE performs worse than SHADE on five functions (f_3-f_4, f_7-f_8 and f_{15}). In addition, the standard deviation of the most winner functions for CSHADE is lower than SHADE for both benchmark functions which indicates that the proposed algorithm shows more accurate and robust results compared to SHADE algorithm. Moreover, the improved accuracy rate column of the Tables 4.8 and 4.9 shows the reflection of the relative enhancement that obtained from the proposed algorithm. The improved accuracy rate is calculated as mentioned in equation 4.19. The improvement rate column for the Table 4.8 for CEC 2010 LSGO benchmark functions shows in average the enhancement rate of CSHADE is better than SHADE algorithm 6.71e+03 times. Moreover, the enhancement rate average for CEC 2013 LSGO benchmark functions in Table 4.9, indicates that CSHADE performs better than SHADE 1.402e+02 times. This rate confirms that utilizing center-based mutation which creates a new solution around the center helps to maintain the diversity at the exploration phase. Therefore, it has a huge impact on exploring promising regions to obtain the optimal solution.



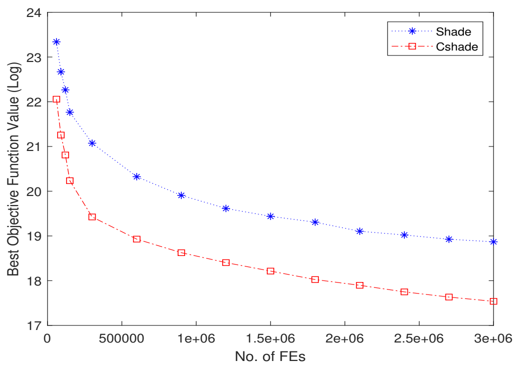
(a) f_1



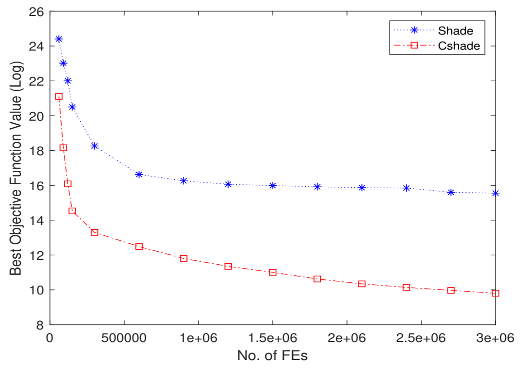
(b) f_2



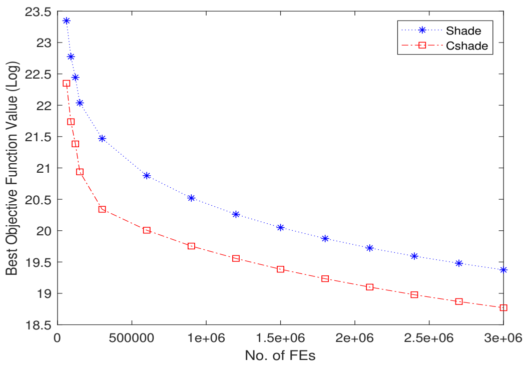
(c) f_8



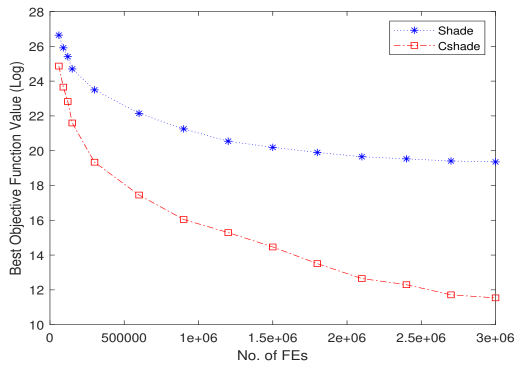
(d) f_9



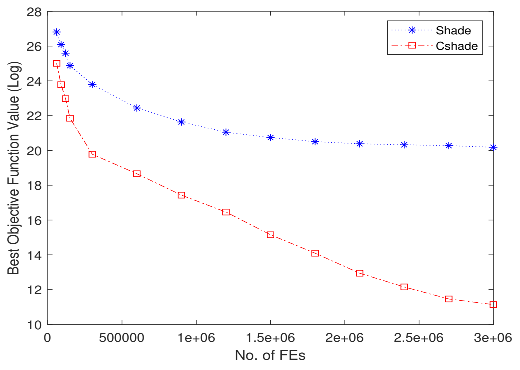
(e) f_{13}



(f) f_{14}



(g) f_{18}



(h) f_{20}

Figure 4.9: Convergence plots of f_1,f_2,f_8,f_9,f_{13},f_{14},f_{18} and f_{20} functions for CEC 2010 benchmark problems set with D=1000. The results were averaged over 31 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations (FEs).

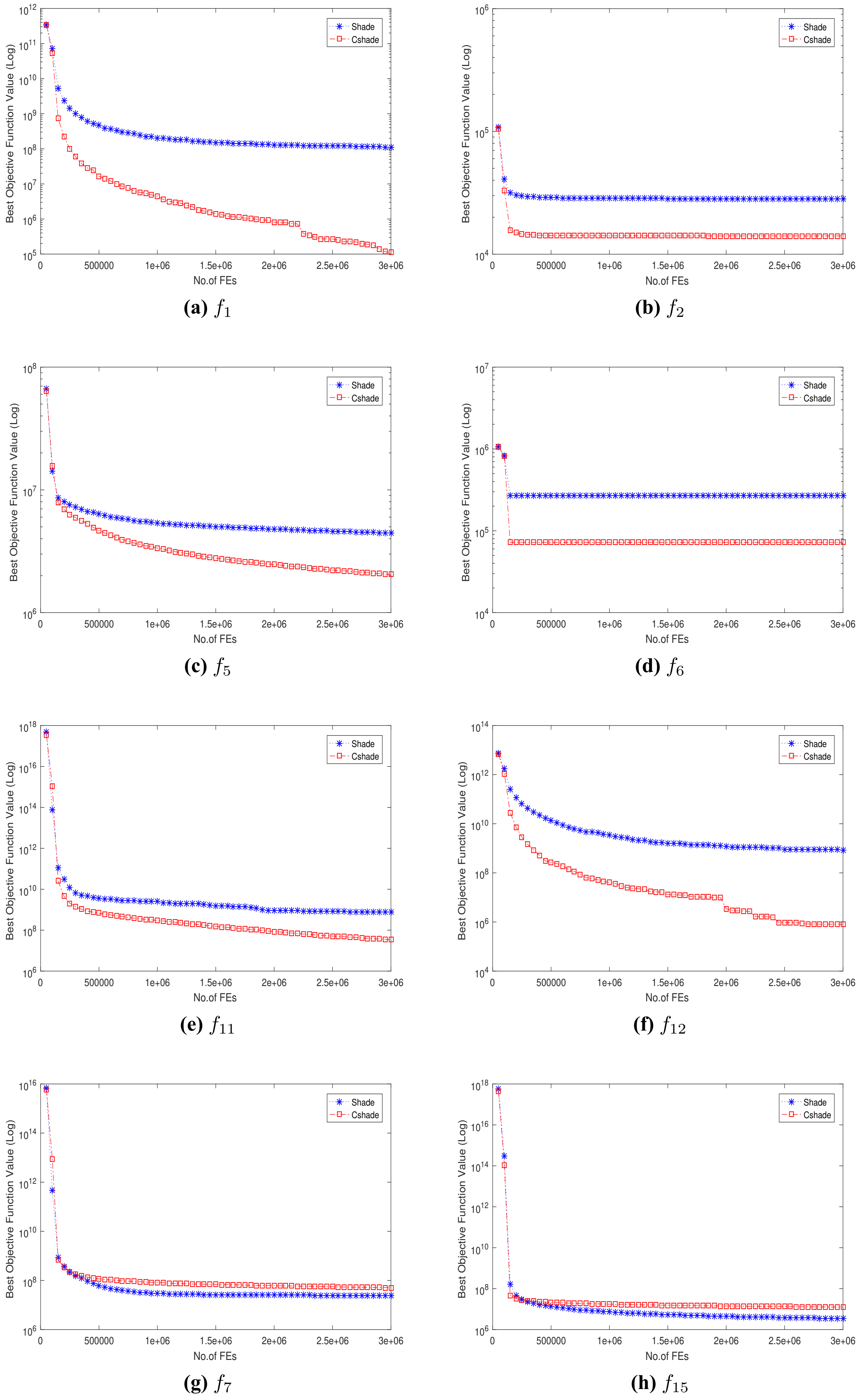


Figure 4.10: Convergence plots of f_1,f_2,f_5,f_6,f_{11},f_{12},f_7 and f_{15} functions for CEC 2013 benchmark problems set with D=1000. The results were averaged over 31 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations (FEs).

Table 4.8 Results of SHADE and CSHADE algorithms on the CEC 2010 benchmark functions for dimension 1000; the best results are highlighted in **bold-face**.

Function		SHADE	CSHADE	Improved accuracy rate
f_1	Mean	8.54e+07	**1.93e+05**	4.39e+02
	Std	7.48e+07	3.98e+05	
f_2	Mean	6.82e+03	**3.19e+03**	2.14e+00
	Std	1.11e+03	2.30e+03	
f_3	Mean	**5.67e-01**	1.67e+01	3.39e-02
	Std	3.15e+00	3.22e-01	
f_4	Mean	**1.64e+11**	2.03e+13	8.07e-03
	Std	6.14e+10	1.93e+13	
f_5	Mean	6.47e+07	**5.16e+07**	1.25e+00
	Std	1.09e+07	2.08e+07	
f_6	Mean	2.53e+06	**1.97e+01**	1.29e+05
	Std	4.89e+05	2.76e-02	
f_7	Mean	**1.15e+06**	9.22e+09	1.25e-04
	Std	2.18e+05	1.42e+10	
f_8	Mean	**5.72e+06**	2.50e+07	2.29e-01
	Std	8.00e+06	4.64e+07	
f_9	Mean	1.65e+08	**4.16e+07**	3.96e+00
	Std	5.72e+07	4.72e+06	
f_{10}	Mean	7.41e+03	**3.75e+03**	1.98e+00
	Std	1.84e+02	1.23e+03	
f_{11}	Mean	1.90e+02	**1.45e+02**	1.31e+00
	Std	2.41e+00	1.10e+01	
f_{12}	Mean	7.76e+04	**5.61e+04**	1.38e+00
	Std	2.55e+04	1.47e+05	
f_{13}	Mean	2.44e+07	**2.53e+04**	9.65e+02
	Std	6.41e+07	1.84e+04	
f_{14}	Mean	2.61e+08	**1.42e+08**	1.83e+00
	Std	3.05e+07	1.21e+07	
f_{15}	Mean	7.41e+03	**5.43e+03**	1.37e+00
	Std	1.80e+02	2.02e+02	
f_{16}	Mean	3.85e+02	**3.15e+02**	1.22e+00
	Std	8.92e-01	8.99e+00	
f_{17}	Mean	2.07e+05	**1.11e+05**	1.86e+00
	Std	4.65e+04	2.33e+05	
f_{18}	Mean	4.13e+08	**3.07e+05**	1.35e+03
	Std	4.48e+08	6.07e+05	
f_{19}	Mean	**5.99e+05**	5.23e+06	1.15e-01
	Std	9.95e+04	1.84e+06	
f_{20}	Mean	8.02e+08	**2.73e+05**	2.94e+03
	Std	6.76e+08	8.32e+05	
w/t/l		-	**15/0/5**	**Ave.**6.71e+03

Figures 4.9-4.10 depict the performance of CSHADE algorithm versus SHADE algorithm. They clearly show that the proposed algorithm (CSHADE) has achieved significant improvement in the most functions for CEC 2010 LSGO as well as CEC 2013 LSGO. As can be seen, CSHADE has better performance in terms of exploration of the search space,

Table 4.9 Results of SHADE and CSHADE algorithms on the CEC 2013 benchmark functions for dimension 1000; the best results are highlighted in **bold-face**.

Function		SHADE	CSHADE	Improved accuracy rate
f_1	Mean	1.126e+08	**1.112e+05**	1.012e+03
	Std	1.012e+08	1.240e+05	
f_2	Mean	2.835e+04	**1.409e+04**	2.012e+00
	Std	1.656e+03	9.217e+02	
f_3	Mean	**5.821e-01**	1.676e+01	3.472e-02
	Std	3.200e+00	6.521e-01	
f_4	Mean	**5.555e+09**	1.181e+11	4.704e-02
	Std	3.136e+09	1.258e+11	
f_5	Mean	4.456e+06	**2.045e+06**	2.179e+00
	Std	7.618e+05	3.440e+05	
f_6	Mean	2.684e+05	**7.222e+04**	3.717e+00
	Std	4.316e+04	3.009e+04	
f_7	Mean	**2.422e+07**	5.064e+07	4.783e-01
	Std	2.402e+07	1.977e+08	
f_8	Mean	**1.356e+12**	5.800e+15	2.338e-04
	Std	3.370e+12	4.120e+15	
f_9	Mean	4.363e+08	**2.093e+08**	2.085e+00
	Std	4.859e+07	2.185e+07	
f_{10}	Mean	3.363e+06	**1.023e+06**	3.289e+00
	Std	3.038e+06	4.344e+05	
f_{11}	Mean	7.820e+08	**3.422e+07**	2.285e+01
	Std	1.419e+09	1.734e+07	
f_{12}	Mean	8.481e+08	**8.073e+05**	1.050e+03
	Std	1.016e+09	2.332e+06	
f_{13}	Mean	3.416e+08	**1.561e+08**	2.188e+00
	Std	3.272e+08	6.800e+08	
f_{14}	Mean	2.761e+09	**1.557e+09**	1.774e+00
	Std	4.615e+09	8.425e+09	
f_{15}	Mean	**3.477e+06**	1.248e+07	2.786e-01
	Std	1.089e+06	5.475e+06	
	w/t/l	-	**10/0/5**	**Ave.**1.402e+02

which leads to finding a more accurate solution on overall results.

In summary, the influence of center-based mutation on the SHADE algorithm for solving large scale optimization problems was studied. Since center-based mutation has a powerful impact on enhancing the classical DE algorithm, the center-based mutation effect was investigated to improve the SHADE algorithm. In center-based mutation SHADE, the base vector of SHADE was replaced by using the mean of three randomly selected candidate solutions as a mean value for the normal distribution. This generates a new solution around the center, which increases the opportunity to explore more promising regions. Because the center point is crucial in a high dimensional search space with a distinct property (the closest point to the unknown solution), CSHADE could obtain more accurate solutions on large scale benchmark functions. All extensive experiments were performed on CEC 2010 and 2013 LSGO benchmark functions for dimension 1000. The experimental results confirmed that the CSHADE algorithm performs significantly better than the SHADE algorithm.

4.7 Case Study Three: Dynamic Center-based Mutation Strategy for DE

Inspired from [9], this section proposes five different dynamic center-based DE mutation schemes for solving large-scale optimization problems, called Dynamic Center-based Mutation strategy for DE (DCDE). As mentioned in the previous case (CDE), center-based mutation scheme is applied at the beginning of the optimization process (only 10%) of generations to generate new offspring. Thereafter, the classical DE mutation ($DE/rand/1$) is applied for the remaining portion of the optimization process (90%). The main reason to investigate was to see in which portion the center had a strong impact during the generations and when (during exploration or fine-tuning) in order to enhance the performance of the classical DE algorithm. The initialization process of DCDE is the same as classical DE, however, in each generation, DCDE applies dynamic center-based mutation to improve search abilities during the exploration and exploitation stages of DE.

The pseudo-code of the proposed mutation scheme for DE is presented in Algorithm 3. The algorithm starts with a random initial population which has NP candidate solutions. The maximum number of fitness function evaluations (MAX_NFC) is usually set as a terminal condition of the while loop (line 4). For the mutation stage, new solutions are generated using five different dynamic center-based mutation strategies as explained in the next subsections. In center-based mutation, three candidate solution are selected randomly to compute their center (line 8). Then, the normal distribution is utilized to generate a new solution around the center (line 10). For the portion of the offspring generation, the center vector is then utilized as a base vector for the center-based mutation scheme (line 12). For the remaining portion, the classical DE mutation operator is used to generate new offspring (line 13).

Scheme 1 (DCDES1):

In this scheme, at the beginning of the DCDE algorithm's generations, the center-based mutation scheme utilizes 100% of the population. Then, as the generation increases, the number of center-based individuals are gradually decreased (linearly) to zero (see Figure 4.11a). The number of individuals that contribute to the DE mutation and center-based mutation is calculated as follows:

$$DEpop = round(\frac{iter}{MAX\_NFC} \times NP) \tag{4.27}$$

$$DEcpop = NP - DEpop \tag{4.28}$$

Scheme 2 (DCDES2):

In this scheme, at the beginning of the DCDE algorithm's generations, the DE mutation scheme utilizes 100% of the population (zero individuals contributed to the center-based mutation scheme). Then, as the generation increases, the number of individuals that contributed to the center-based mutation is linearly increased to 100% (see Fig. 4.11b). The number of individuals that contribute to the center-based mutation and DE mutation is calculated as follows:

$$DEcpop = round(\frac{iter}{MAX\_NFC} \times NP) \tag{4.29}$$

$$DEpop = NP - DEcpop \tag{4.30}$$

Scheme 3 (DCDES3):

In this scheme, at the beginning of the DCDE algorithm's generations, the center-based mutation scheme utilizes 50% of the population. Then, as the generation increases, the number of center-based individuals is linearly decreased to zero (see Figure 4.11c). The number of individuals that contribute to the DE mutation and center-based mutation is calculated as follows:

$$DEpop = round(\frac{iter}{MAX\_NFC} \times \frac{NP}{2}) + \frac{NP}{2} \tag{4.31}$$

$$DEcpop = NP - DEpop \tag{4.32}$$

Scheme 4 (DCDES4):

In this scheme, at the beginning of the DCDE algorithm's generations, the DE mutation scheme utilizes 50% of the population, and 50% of the remaining population for center-based mutation). Then, as the generation increases, the number of individuals that contribute to the center-based mutation is linearly increased to 100% (see Figure 4.11d). The number of individuals that contribute to the center-based mutation and DE mutation is calculated as follows:

$$DEcpop = round(\frac{iter}{MAX\_NFC} \times \frac{NP}{2}) + \frac{NP}{2} \tag{4.33}$$

$$DEpop = NP - DEcpop \tag{4.34}$$

Scheme 5 (DCDES5):

In this scheme, at the beginning of the DCDE algorithm's generations, the center-based mutation scheme utilizes 100% of the population. Then, as the generation increases, the number of individuals that contribute to the center-based mutation is linearly decreased to 50% (see Figure 4.11c). The number of individuals that contribute to the DE mutation and center-based mutation is calculated as follows:

$$DEpop = round(\frac{iter}{MAX\_NFC} \times \frac{NP}{2}) \tag{4.35}$$

$$DEcpop = NP - DEpop, \tag{4.36}$$

in which $DEpop$ is the number of population that contributes to classical DE mutation and $DEcpop$ is the number of population that contributes to center-based DE mutation.

4.7.1 Experimental Results

The proposed algorithm DCDE was evaluated on CEC 2013 LSGO benchmark functions with the dimension of 1000 [17]. Five series of experiments have been conducted for the comparison of the proposed schemes against the classical DE. The main difference between

Algorithm 4 : **Algorithmic description of DCDE Algorithm**

```
1: //$NP$, $D$ and $MAX\_NFC$ are the population size, the problem dimension and the
   maximum number of function evaluations, respectively.
2: Generating the initial population of NP candidate solution randomly.
3: Set the generation counter $Gcounter = 1$;
4: **while** $NFC < MAX\_NFC$ **do**
5:    Calculate the DE and DEc population index based on Eq.4.35, 4.36 respectively;
6:    **for** $i = 1$ to $NP$ **do**
7:       //Run mutation;
         $/ * CenterBasedDEMutation * /$
8:       Calculate the center of the three selected candidate solutions randomly from the
         population that related to the center portion $x_{Ncenter}$ Eq. 4.12.
9:       **for** $j = 1$ to $D$ **do**
10:         Calculate the normal distribution of the center for each dimension $(j)$ $x_{Ncenter}$
            Eq. 4.13.
11:      **end for**
12:      Perform CDE mutation for the center population portion according to Eq. 4.14.
         $/ * ClassicalDEMutation * /$
13:      Perform DE mutation for the remaining portion of the population according to Eq.
         5.4.
14:      Crossover.
15:      Selection.
16:   **end for**
17:   Increment the generation counter $Gcounter = Gcounter + 1$;
18: **end while**
```

these five experiments is the involved numbers of individuals during the search for the optimal solution. As can be seen from Table 4.10 that the proposed schemes for DCDE algorithm outperform classical DE for all schemes. The best schemes among the five schemes are DCDE Scheme 1 and DCDE Scheme 3. For DCDE scheme 1, DCDE outperforms classical DE on 14 (f_1-f_4,f_6-f_{15}) functions out of fifteen. They have a comparable result for one function (f_5). While, DCDE scheme 3, DCDE performs better than classical DE for all functions (f_1-f_{15}). In addition, as can be observed, the standard deviations (STD) values of both schemes are lower than classical DE which means that the results of DCDE are more accurate and consistent compared to classical DE. Moreover, Table 4.11 shows the improved accuracy rate for the best schemes that confirm the enhancement of classical DE. The improved accuracy rate (I_{rate}) is calculated as mentioned in the equation 4.19.

As we can see, form Table 4.11 that the improvement rate of the DCDE scheme 1 achieved better than classical DE, on average 1.997e+05 times. Also, the improvement rate of the DCDE scheme 3 outperformed classical DE, on average 1.881e+03 times. In overall, scheme 1 is the best of all as a result of using center-based mutation for most of the exploration stage. Furthermore, these results confirm that utilizing center-based mutation which creates a new solution around the center helps to maintain the diversity at the exploration phase, and thus it has a huge impact on exploring the promising region to obtain the optimal value. On the other hand, as we can see, from the other schemes (2 and 4) center-based mutation could not improve that much, because it contributed more to the exploitation phase. Therefore, generating a new solution in this phase can not distract the algorithm in the fine-tuning the solution.

In addition, we compared the best scheme of the proposed method (DCDE scheme 1)

Table 4.10 Results of DE/rand/1 and the proposed algorithms on the CEC 2013 LSGO benchmark functions for dimension 1000. Symbols '$+$','$-$' and '$=$' denote the proposed algorithms are better than, worse than, or similar to the compared algorithm, respectively.

Functions		DE/rand/1	DCDE Scheme1		DCDE Scheme2		DCDE Scheme3		DCDE Scheme4		DCDE Scheme5		CDE Vs. DCDE Scheme1	
f_1	Mean	1.923e+07	**8.137e+00**	+	**1.914e+06**	+	**9.547e+02**	+	**3.896e+00**	+	**3.710e-01**	+	**7.345e+04**	+
	Std	4.531e+07	2.021e+01		1.312e+07		1.803e+03		9.671e+00		1.629e+00		5.306e+04	
f_2	Mean	1.920e+04	**2.253e+03**	+	1.924e+04	=	**7.626e+03**	+	**7.417e+03**	+	**1.941e+03**	+	**3.315e+03**	+
	Std	9.599e+02	9.071e+01		8.237e+02		4.463e+02		4.348e+02		8.946e+01		1.241e+02	
f_3	Mean	1.888e+01	**3.235e+00**	+	1.890e+01	=	**1.434e+01**	+	**1.358e+01**	+	**2.112e+00**	+	**9.691e+00**	+
	Std	7.052e-02	3.064e-01		7.236e-02		5.359e-01		6.500e-01		2.061e-01		3.603e-01	
f_4	Mean	3.707e+10	**1.714e+10**	+	**2.164e+10**	+	**2.248e+10**	+	**1.539e+10**	+	**1.673e+10**	+	**2.287e+10**	+
	Std	1.572e+10	6.690e+09		8.453e+09		1.010e+10		5.657e+09		6.585e+09		8.276e+09	
f_5	Mean	2.207e+06	4.927e+06	=	2.188e+06	=	**1.576e+06**	+	8.367e+06	-	8.555e+06	-	1.048e+06	-
	Std	4.606e+05	3.538e+06		3.740e+05		1.378e+06		2.693e+05		2.587e+05		1.897e+05	
f_6	Mean	1.047e+05	**3.294e+00**	+	1.029e+05	=	**1.906e+01**	+	**1.710e+01**	+	**1.793e+00**	+	**1.529e+01**	+
	Std	3.853e+04	3.659e-01		3.305e+04		5.815e-01		1.117e+00		3.302e-01		3.963e-01	
f_7	Mean	1.125e+08	**7.288e+07**	+	1.912e+08	-	**7.583e+07**	+	3.775e+08	-	4.282e+08	-	**8.820e+07**	+
	Std	4.584e+07	1.881e+07		1.141e+08		6.319e+07		1.466e+08		1.781e+08		2.707e+07	
f_8	Mean	3.047e+14	**1.877e+14**	+	**1.797e+14**	+	**1.691e+14**	+	**2.398e+14**	+	3.472e+14	=	2.335e+14	=
	Std	1.289e+14	7.722e+13		9.294e+13		7.923e+13		2.415e+14		2.625e+14		1.204e+14	
f_9	Mean	2.069e+08	**9.305e+07**	+	2.090e+08	=	**1.090e+08**	+	5.368e+08	-	6.278e+08	-	**1.002e+08**	+
	Std	2.984e+07	2.039e+07		3.040e+07		2.701e+07		1.666e+08		1.046e+08		1.226e+07	
f_{10}	Mean	2.810e+03	**1.186e+01**	+	**1.454e+03**	+	**2.971e+02**	+	**1.089e+02**	+	**2.826e+00**	+	**2.255e+02**	+
	Std	6.446e+03	6.892e+00		1.233e+03		6.862e+01		5.555e+01		2.508e+00		1.641e+01	
f_{11}	Mean	1.509e+11	**1.023e+10**	+	1.560e+11	=	**3.857e+10**	+	**5.280e+10**	+	**1.095e+10**	+	**4.983e+10**	+
	Std	8.136e+10	1.395e+10		9.804e+10		2.483e+10		2.560e+10		7.360e+09		2.472e+10	
f_{12}	Mean	2.296e+09	**3.831e+03**	+	**4.428e+06**	+	**3.120e+07**	+	**3.314e+03**	+	**3.006e+03**	+	**1.465e+07**	+
	Std	2.583e+09	5.667e+02		2.435e+07		1.221e+08		4.337e+02		2.436e+02		2.836e+07	
f_{13}	Mean	7.290e+09	**3.808e+09**	+	7.370e+09	=	**4.286e+09**	+	**6.295e+09**	+	**6.319e+09**	+	**6.118e+09**	+
	Std	1.741e+09	9.588e+08		2.055e+09		1.185e+09		1.599e+09		1.865e+09		1.171e+09	
f_{14}	Mean	2.027e+11	**3.493e+10**	+	**1.725e+11**	+	**6.369e+10**	+	**7.573e+10**	+	**2.994e+10**	+	**1.048e+11**	+
	Std	7.332e+10	1.780e+10		6.065e+10		2.234e+10		2.278e+10		1.217e+10		3.695e+10	
f_{15}	Mean	2.462e+10	**4.874e+07**	+	1.598e+10	=	**9.929e+06**	+	**8.277e+07**	+	**1.313e+08**	+	5.525e+06	-
	Std	6.381e+10	2.407e+07		3.154e+10		3.022e+06		3.768e+07		1.606e+07		1.157e+06	
	w/t/l		***14/1/0***		*6/8/1*		***15/0/0***		*12/0/3*		*11/1/3*			***12/1/2***

Table 4.11 Improvement accuracy rate for the best schemes (scheme1 and Scheme3)

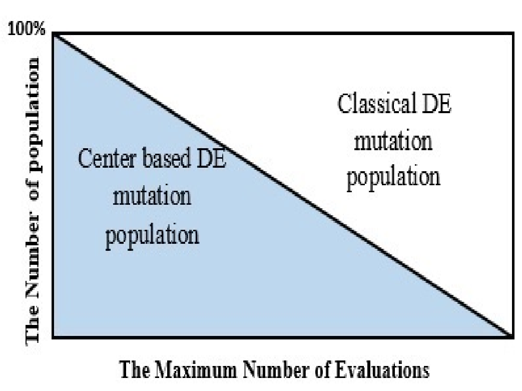
Function	DCDE Scheme1	DCDE Scheme3
f_1	2.363e+06	2.014e+04
f_2	8.522e+00	2.518e+00
f_3	5.836e+00	1.316e+00
f_4	2.163e+00	1.649e+00
f_5	4.480e-01	1.400e+00
f_6	3.179e+04	5.493e+03
f_7	1.544e+00	1.484e+00
f_8	1.623e+00	1.802e+00
f_9	2.223e+00	1.898e+00
f_{10}	2.368e+02	9.457e+00
f_{11}	1.475e+01	3.913e+00
f_{12}	5.993e+05	7.360e+01
f_{13}	1.914e+00	1.701e+00
f_{14}	5.805e+00	3.183e+00
f_{15}	5.051e+02	2.479e+03
Avg.	***1.997e+05***	*1.881e+03*

with CDE algorithm as seen in the last column of the Table 4.10. DCDE scheme 1 outperforms CDE on 12 (f_1-f_4, f_6-f_7, f_9-f_{14}) functions out of fifteen. They achieve the same result for one function (f_8), whereas DCDE scheme 1 has worse results than CDE for two (f_5, f_{15}) functions. Figures 5.1 depicts the performance of the DCDE scheme 1 algorithm versus the classical DE algorithm. It clearly shows that the proposed algorithm (DCDE scheme 1) has a significant improvement in most functions for CEC 2013 LSGO. As can be seen, DCDE scheme 1 has better performance than the classical DE on overall functions due to the enhancement of the exploration that leads to finding more accurate solutions[12].
Discussion:
The dynamic center-based DE mutation schemes (DCDE) for solving LSGO problems have achieved significant performance over classical DE and center DE algorithms. The detailed analysis for the benefit of using several schemes is to know how center-based could contribute during each generation of the optimization process. The utilized schemes show that at the beginning of the generation, the diversity of the population is required in the exploration stage. Using 50% or 100% of the population contributed in the center mutation has a successful impact on accelerating the convergence of the algorithm as Scheme 1 and Scheme 3 confirmed. On the other hand, utilizing the 50% or 100% of the population in center-based mutation schemes during the exploitation stage is challenging because generating a new solution from the mutation operation does not help the optimizer to improve the candidate solution. In the last generation, the algorithm needs fine-tuning with the current candidate solutions to find global optima instead of injecting a new solution.

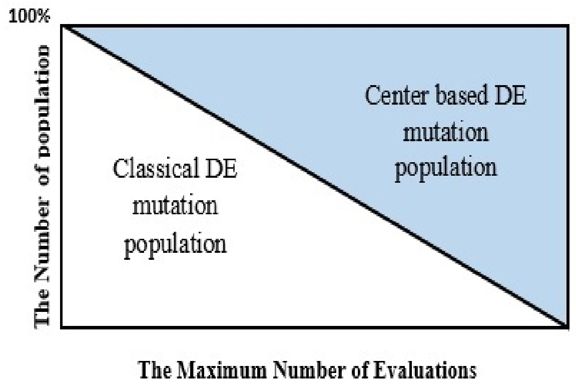
In summary, five different dynamic center-based DE mutation schemes (DCDE) were introduced for solving LSGO problems. In each generation, the proposed dynamic center-based mutation strategies linearly divided the population into two different groups in order to investigate the effects of center-based mutation during the exploration and exploitation phases. Then, the first population group utilized center-based mutation scheme and the remaining population employed the classical DE mutation. Our proposed algorithm was compared to the classical DE and evaluated on CEC 2013 LSGO benchmark functions for dimension 1000. The experimental results confirmed that the proposed algorithms exhibit significantly better solutions compared to classical DE.

4.8 Summary

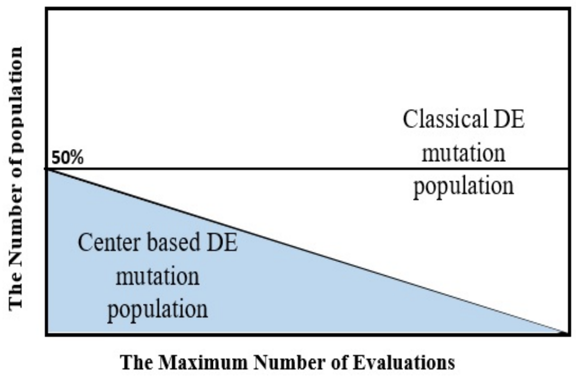
This chapter introduced the DE algorithm and the center-based DE algorithm provided in detail for three different experiments to improve the classical DE algorithm and SHADE algorithm for large-scale optimization problems. In fact, a center-based concept gives a higher chance to find promising regions that promote finding the global optimal. Therefore, in most cases, the proposed algorithms could obtain more accurate solutions on high dimensional benchmark functions. The proposed algorithms were compared to the classical algorithms and evaluated on CEC 2013 LSGO with dimension 1000. The experimental results confirmed that the proposed center-based algorithms could significantly improve the performance of the classical algorithms. Almost in all test problems, the proposed algorithm outperformed the classical algorithms in terms of solution accuracy. In the next chapter, center-based GDE3 for solving large-scale multi-objective large-scale optimization will be introduced.


100%
The Number of population
Center based DE mutation population
Classical DE mutation population
The Maximum Number of Evaluations

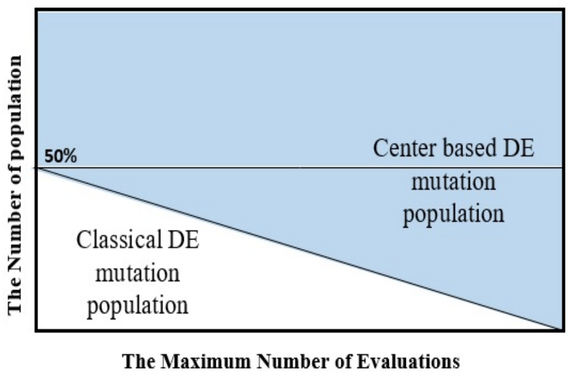
(a) $Scheme1$


100%
The Number of population
Classical DE mutation population
Center based DE mutation population
The Maximum Number of Evaluations

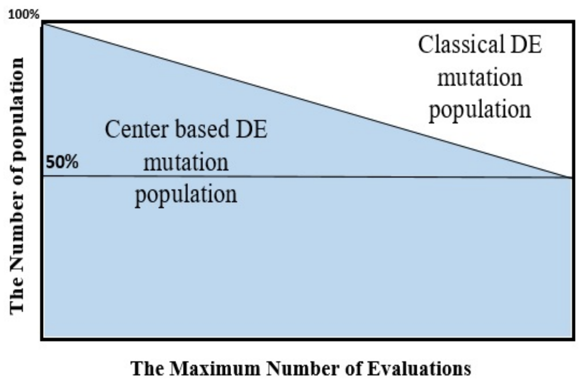
(b) $Scheme2$


The Number of population
50%
Classical DE mutation population
Center based DE mutation population
The Maximum Number of Evaluations

(c) $Scheme3$

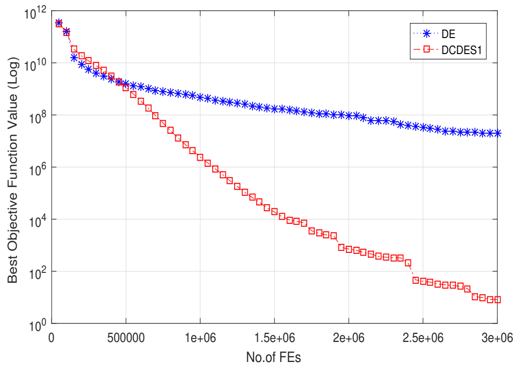

The Number of population
50%
Center based DE mutation population
Classical DE mutation population
The Maximum Number of Evaluations

(d) $Scheme4$

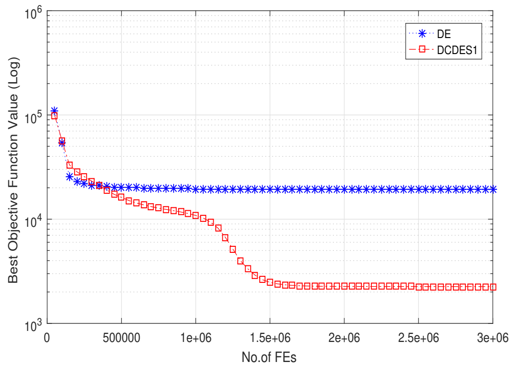

100%
The Number of population
50%
Classical DE mutation population
Center based DE mutation population
The Maximum Number of Evaluations

(e) $Scheme5$

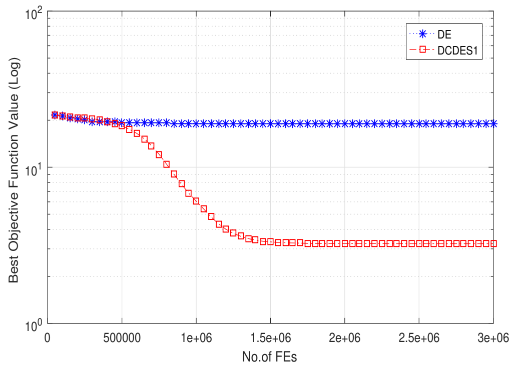
Figure 4.11: Illustration of the divided population of the proposed schemes. The vertical axis is the population number and the horizontal axis is the number of function evaluations (FEs) [12].



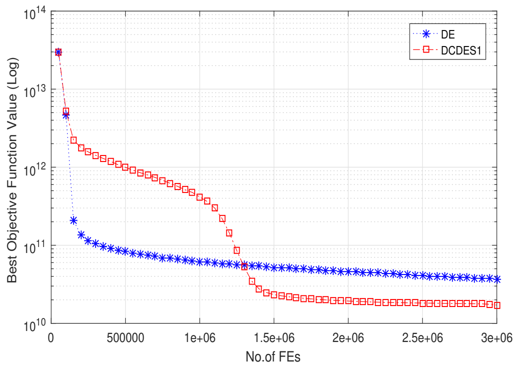
(a) f_1



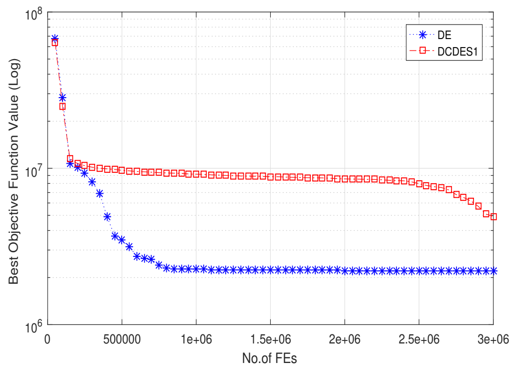
(b) f_2



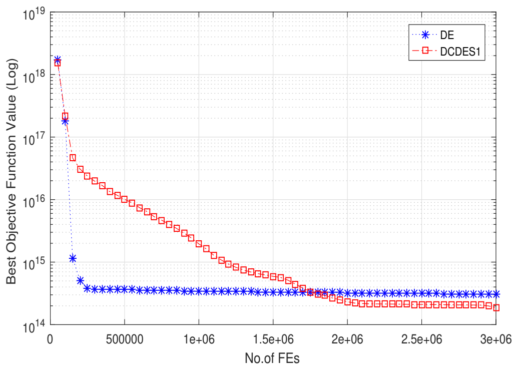
(c) f_3



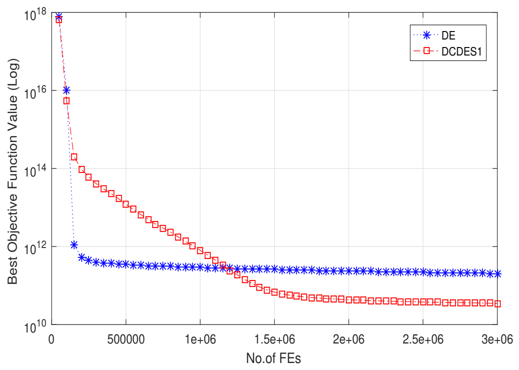
(d) f_4



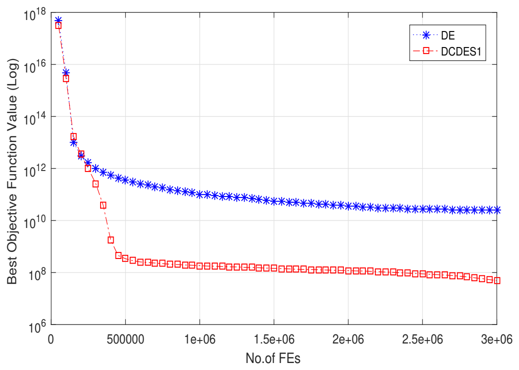
(e) f_5



(f) f_8



(g) f_{14}



(h) f_{15}

Figure 4.12: Convergence plots of f_1, f_2,f_3,f_4,f_5,f_8,f_{14} and f_{15} functions for CEC 2013 benchmark problems set with D=1000. The results were averaged over 51 runs. The vertical axis is the function value and the horizontal axis is the number of function evaluations (FEs) [12].

Chapter 5

CGDE3: An Efficient Center-based Algorithm for Solving Large-scale Multi-objective Optimization Problems

5.1 Introduction

In this chapter, a novel center-based mutation is introduced for the GDE3 algorithm (CGDE3) to solve large scale multi-objective optimization problems efficiently. To the best of our knowledge, it is the first time that a center-based sampling scheme is proposed to enhance a multi-objective algorithm. The CGDE3 algorithm is evaluated on CEC 2017 benchmark problems with dimensions 100, 500, and 1000. Experimental results confirm the superiority of the proposed algorithm over all three large-scale dimensions.

5.2 Background Review

Many real-world optimization problems have more than one conflicting objectives to be optimized. The definition of optimality is not as simple as the single-objective optimization. Therefore, it is necessary to make a trade-off between objective values. There are some well-known concepts to compare two candidate solutions in the multi-objective problem space. Since this case utilizes non-dominated sorting and crowding distance [75] to order candidate solutions for the discussed multi-objective algorithms, these metrics are defined in detail in this section.

A multi-objective optimization problem is defined as follows:

$$\begin{aligned} &Min\ F(\boldsymbol{x}) = [f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), ..., f_M(\boldsymbol{x})] \\ &s.t. \quad L_i \leq \boldsymbol{x}_i \leq U_i, i = 1, 2, ..., d \end{aligned} \tag{5.1}$$

Subject to following equality and inequality constraints:

$$g_i(\boldsymbol{x}) \leq 0 \quad j = 1, 2, ..., J$$

$$h_k(\boldsymbol{x}) = 0 \quad k = 1, 2, ..., K,$$

where M is the number of objectives, d is the number of decision variables (dimension) of solution vector, $\boldsymbol{x}_i$ is in interval $[L_i, U_i]$ (box-constraints). f_i represents the objective function which should be minimized.

If $\boldsymbol{x} = (x_1, x_2, ..., x_d)$ and $\acute{\boldsymbol{x}} = (\acute{x}_1, \acute{x}_2, ..., \acute{x}_d)$ are two vectors in the problem search space, $\boldsymbol{x}$ dominates $\acute{\boldsymbol{x}}$ ($\boldsymbol{x} \prec \acute{\boldsymbol{x}}$) if and only if:

$$\begin{aligned} &\forall i \in \{1, 2, ..., M\}, f_i(\boldsymbol{x}) \leq f_i(\acute{\boldsymbol{x}}) \wedge \\ &\exists j \in \{1, 2, ..., M\} : f_j(\boldsymbol{x}) < f_j(\acute{\boldsymbol{x}}) \end{aligned} \tag{5.2}$$

It defines optimality for solutions in objective space. Candidate solution $\boldsymbol{x}$ is better than $\acute{\boldsymbol{x}}$ if it is not better than $\acute{\boldsymbol{x}}$ in any of the objectives, and it has at least a worse value in one of the objectives. All solutions that are not dominated using any other solutions in the population are called non-dominated solutions and create the first Pareto front set.

Non-dominated sorting is an algorithm to rank obtained solutions to different levels in the processing of multi-objective optimization. All non-dominated solutions are in the first rank. The second rank is then made of solutions that are non-dominated by removing the first rank from the population. This process is repeated until all solutions are ranked using this iterative process. Crowding distance is another metric which is necessary to compare solutions along with non-dominating sorting. It is a measure to compute the diversity of obtained solutions by calculating the distance between adjacent solutions. Initially, the set of solutions in the same rank are sorted according to each objective function value in ascending order. In order to calculate crowding distance, the difference between neighboring objective values of each solution is computed as the following [76].

$$CD_i = \sum_{j=1}^{M} |f_j(i+1) - f_j(i-1)| \tag{5.3}$$

For all objectives the boundary solutions (with lowest and highest objective values) are assigned infinite crowding distance values for each. This guarantees that boundary solutions are always selected during the selection process. Then the sum of individual distance values corresponding to each objective is considered among the overall crowding distance. A larger value of crowding distance for a vector in population shows high diversity around that vector.

5.2.1 Third Generalized Differential Evolution (GDE3)

DE is an evolutionary algorithm originally for solving continuous optimization problems that improve the initial population using crossover and mutation operations. The creation of a new population generation is done by a mutation and a crossover operator. The mutation operator for a gene, j, is defined as follows:

$$v_{j,i} = x_{j,i_1} + F.(x_{j,i_2} - x_{j,i_3}) \tag{5.4}$$

Applying this operator generates a new D dimensional vector, v_i, using three randomly selected individuals, x_{j,i_1}, x_{j,i_2}, and x_{j,i_3} from the current population. Parameter F, mutation factor, scales the difference between two vectors. The crossover operator changes some or all of the genes of the parent solution based on the Crossover rate (C_r). Resembling other population-based algorithms, the single objective version of DE starts with a uniform randomly generated population. The next-generation is created using the previously mentioned mutation and crossover operations; then, the best individual (between parent and new individual) is selected based on their objective values, called a greedy selection. It iterates until meeting a stopping criterion such as a predefined number of generations.

There are also several variants of DE algorithms for multi-objective optimization. The first version of Generalized Differential Evolution (GDE) [77] changed the DE selection mechanism for producing the next generation. The idea in the selection was based on constraint-domination. The new vector is selected if it dominates the old vector. GDE2 [78], the next version of the multi-objective DE algorithm, added the crowding distance measure to its selection scheme. If both vectors are non-dominating, the vector with a higher crowding distance is selected.

The third version of GDE (GDE3) [70, 79] extends the DE algorithm of multi-objective optimization problems with M objectives and K constraints. DE operators are applied using three randomly selected vectors to produce an offspring per parent in each generation. The selection strategy is similar to the GDE2 except in two parts: 1. Applying constraints during the selection process; 2. When none of the solutions can dominate the other. Selection rules in GDE3 are as follows: when old and new vectors are infeasible solutions, the new vector is selected, which dominates the old vector over constraint violation space; otherwise, the old vector is selected. In the case that one of them is a feasible vector, the feasible vector is selected. If both vectors are feasible, the dominating vector is selected for the next generation. In the non-dominating case, both vectors are selected. Therefore, the size of the population generated may be larger than the population of the previous generation. If this is the case, it is then decreased to the original size; the election strategy for this step is similar to the NSGA-II algorithm [76]; it sorts individuals in the population-based on the non-dominated sorting algorithm and crowding distance measure. Similar to other population-based multi-objective algorithms, the selected individuals are passed to the next generation to continue the optimization processing. The common point about all of these versions is utilizing randomly selected individuals to produce a new vector using DE's main mutation operator, which is modified in the proposed algorithm in this chapter.

5.3 GDE3 with Center-based Mutation Scheme: The First Proposed Center-based Multi-objective Algorithm

In the GDE3 algorithm, the $DE/rand/1$ mutation strategy is utilized where three randomly candidate solutions x_{r1}, x_{r2}, and x_{r3} are selected from the current population to generate a new trial vector with the equation $v_i = x_{r1} + F.(x_{r2} - x_{r3})$.

A novel Center-based mutation for Third the Generalized Differential Evolution is introduced in the proposed algorithm, which is called CGDE3. In this scheme, a new base vector is generated using the average of three randomly selected candidate solutions as a mean value of a normal distribution. Algorithm 5 summarized the CGDE3 algorithm. The algorithm begins with an initial population that has NP candidate solutions. In the new mutation operation, five candidate solutions x_{r1}, x_{r2}, x_{r3}, x_{r4}, and x_{r5} are randomly selected from the current population to generate a new trial solution. The normal distribution is then utilized to generate a new base vector for the new mutation scheme. First, the average of three randomly selected candidate solutions x_{r1}, x_{r2}, and x_{r3} are selected to calculate x_{center} (line 8) as mentioned in the Eq.4.12:

The average value x_{center} is considered the mean (μ) of the normal distribution, which generates a new solution around the mean of each dimension.

The standard deviation σ is calculated, as mentioned in the previous Chapter's primary proposed method.

The normal distribution generates a new point as the base vector as mentioned in Eq.4.13 (line 10).

The new mutant vector v_i is obtained by using the $CDE/rand/1$ equation 4.14 (line 13).

The proposed mutation scheme is utilized over only 10% of the beginning of generations (lines 7- 13) to enhance the exploration of the search space and target the promising regions. For the rest of the generations, the GDE3 mutation scheme ($DE/rand/1$) is applied (line 15). After generating a new candidate solution using the proposed mutation, other components of the GDE3 algorithm remain untouched.

5.4 Experimental Results

5.4.1 Benchmark Functions

In order to investigate the performance of the proposed algorithm versus GDE3, comprehensive experiments have been conducted. The proposed CGDE3 algorithm was applied

Algorithm 5 : **CGDE3 Algorithm** (NP, MAX_NFC)

```
1: //$NP$, $D$ and $MAX\_NFC$ are the population size, the problem dimension and the maximum number of function evaluations, respectively.
2: Generating the initial population of NP candidate solution randomly.
3: Set the generation counter $Gcounter = 1$;
4: Calculate the number of generations $Gen$=MAX_NFC/NP;
5: while NFC < MAX_NFC do
6:     for $i = 1$ to $NP$ do
7:         if $Gcounter <= (Gen \times 0.1)$ then
8:             Calculate the center of the three selected candidate solutions randomly $x_{Ncenter}$ Eq. 4.12.
9:             for $j$ =1 to $D$ do
10:                Calculate the normal distribution of the center for each dimension $(j)$ $x_{Ncenter}$ Eq. 4.13.
11:            end for
12:            //Run mutation;
13:            Center-based mutation scheme Eq.(4.14).
14:        else
15:            Classical DE mutation scheme Eq.(5.4).
16:        end if
17:        Crossover.
18:        Selection: Non-dominated sorting and Crowing distance Pruning.
19:    end for
20:    Increment the generation counter $Gcounter = Gcounter + 1$;
21: end while
```

Table 5.1 Main properties of the utilized benchmark functions [80].

Problem	Properties
MaF1	Linear
MaF2	Concave
MaF3	Convex, multimodal
MaF4	Concave, multimodal
MaF5	Convex, biased
MaF6	Concave, degenerate
MaF7	Mixed, disconnected, multimodal
MaF10	Mixed, biased
MaF11	Convex, disconnected, nonseparable
MaF12	Concave, nonseparable, biased, deceptive
MaF13	Concave, unimodal, nonseparable, degenerate
MaF14	Linear, partially separable, large scale
MaF15	Convex, partially separable, large scale

and tested on MaF benchmark functions designed to assess MOEAs in the CEC 2017 multi-objective competition with dimensions 100, 500, and 1000. In these experiments, thirteen benchmark functions were selected $MaF1$-$MaF7$, $MaF10$-$MaF15$, functions $MaF8$ and $MaF9$ have been excluded because they are not defined for bi-objective case [80] . The following settings have been utilized: The number of objectives: 2, the population size: 100, the maximum number of fitness evaluations (Max_FEs): 10,000D, the mutation factor (F), and crossover rate (CR) were set to 0.5 and 1, respectively. Two algorithms were run for 31 independent times. The main properties of functions are explained in Table 5.1. GDE3 implementation of MATLAB MOEA platform (PlatEMO) [81] was used, and the mutation operation was modified to center-based mutation, as explained in Section 5.3.

5.4.2 Numerical Results and Discussion

In order to evaluate the performance of the proposed algorithm, the inverse generational distance (IGD) metric [82–84] is computed as the following:

Table 5.2 IGD results for GDE3 and the proposed algorithm (CGDE3) on the CEC 2017 benchmark problems dimensions 100, 500, and 1000. Symbols ”+”, ”-” and ”=” denote CGDE3 algorithm performs better than, worse than, or similar to the compared algorithm (GDE3), respectively. ”w/t/l” means that CGDE3 wins in w functions, ties in t functions, and loses in l functions, compared to GDE3 algorithm.

		D=100			D=500			D=1000		
Functions		GDE3	CGDE3		GDE3	CGDE3		GDE3	CGDE3	
$MaF1$	Min	1.152e-01	5.135e-03		8.131e-01	1.140e-02		1.732e+00	3.407e-02	
	Median	1.444e-01	5.602e-03		9.979e-01	1.336e-02		2.115e+00	4.881e-02	
	Max	2.159e-01	6.326e-03		1.344e+00	1.629e-02		2.927e+00	6.848e-02	
	Mean	1.538e-01	**5.639e-03**	+	1.029e+00	**1.352e-02**	+	2.161e+00	**4.920e-02**	+
$MaF2$	Min	1.752e-02	2.871e-03		9.988e-02	5.821e-03		2.221e-01	1.466e-02	
	Median	2.748e-02	3.235e-03		1.465e-01	6.631e-03		2.847e-01	1.727e-02	
	Max	3.645e-02	3.512e-03		1.813e-01	7.385e-03		3.828e-01	2.231e-02	
	Mean	2.748e-02	**3.201e-03**	+	1.444e-01	**6.686e-03**	+	2.866e-01	**1.751e-02**	+
$MaF3$	Min	4.703e-03	3.925e-03		4.771e-01	3.979e-03		4.771e-01	3.978e-03	
	Median	3.717e+03	5.097e+06		2.011e+06	1.101e+08		9.581e+06	6.174e+08	
	Max	1.253e+06	5.812e+06		1.448e+08	1.556e+08		5.495e+08	6.229e+08	
	Mean	1.189e+05	3.691e+06	-	1.314e+07	8.102e+07	=	3.885e+07	4.172e+08	-
$MaF4$	Min	1.458e-02	1.309e-02		9.350e+01	1.358e-02		4.358e+02	5.227e+02	
	Median	9.120e+01	2.484e+03		1.191e+03	1.361e+04		3.644e+03	2.778e+04	
	Max	1.035e+03	2.723e+03		6.910e+03	1.392e+04		2.247e+04	2.800e+04	
	Mean	2.475e+02	1.778e+03	-	2.026e+03	1.066e+04	-	5.440e+03	2.091e+04	-
$MaF5$	Min	4.321e-01	6.281e-02		2.450e+00	1.339e+00		5.781e+00	4.213e+00	
	Median	6.253e-01	8.803e-02		3.432e+00	1.574e+00		6.609e+00	5.493e+00	
	Max	7.896e-01	1.477e-01		4.861e+00	1.909e+00		8.578e+00	6.596e+00	
	Mean	6.169e-01	**9.380e-02**	+	3.588e+00	**1.589e+00**	+	6.787e+00	**5.487e+00**	+
$MaF6$	Min	1.264e+01	1.084e-02		1.153e+02	2.442e-02		2.284e+02	1.121e-01	
	Median	2.232e+01	1.584e-02		1.481e+02	2.938e-02		3.016e+02	1.667e-01	
	Max	2.965e+01	2.889e-02		2.012e+02	4.000e-02		3.885e+02	4.317e-01	
	Mean	2.226e+01	**1.626e-02**	+	1.487e+02	**2.988e-02**	+	3.059e+02	**1.781e-01**	+
$MaF7$	Min	4.440e-03	4.405e-03		9.480e-01	4.383e-03		1.839e+00	7.465e-02	
	Median	1.384e-01	4.473e-03		1.343e+00	4.461e-03		2.282e+00	1.667e-01	
	Max	5.332e-01	4.542e-03		1.841e+00	3.162e-02		2.839e+00	2.976e-01	
	Mean	1.882e-01	**4.475e-03**	+	1.367e+00	**7.357e-03**	+	2.279e+00	**1.772e-01**	+
$MaF10$	Min	1.177e+00	1.177e+00		1.267e+00	1.255e+00		1.269e+00	1.265e+00	
	Median	1.240e+00	1.230e+00		1.272e+00	1.268e+00		1.272e+00	1.269e+00	
	Max	1.283e+00	1.283e+00		1.281e+00	1.281e+00		1.273e+00	1.272e+00	
	Mean	1.239e+00	1.232e+00	=	1.272e+00	**1.267e+00**	+	1.272e+00	**1.269e+00**	+
$MaF11$	Min	1.971e-01	9.151e-02		2.442e-01	1.360e-01		2.478e-01	1.465e-01	
	Median	2.194e-01	1.050e-01		2.562e-01	1.410e-01		2.594e-01	1.538e-01	
	Max	2.423e-01	1.226e-01		2.660e-01	1.504e-01		2.782e-01	1.602e-01	
	Mean	2.203e-01	**1.060e-01**	+	2.552e-01	**1.419e-01**	+	2.608e-01	**1.534e-01**	+
$MaF12$	Min	2.948e-02	3.156e-02		1.445e-02	5.904e-02		1.372e-02	7.995e-02	
	Median	2.995e-02	3.863e-02		1.512e-02	6.380e-02		1.271e-01	8.319e-02	
	Max	3.110e-02	4.174e-02		1.404e-01	6.957e-02		1.474e-01	8.688e-02	
	Mean	3.000e-02	3.792e-02	-	4.154e-02	6.392e-02	-	1.131e-01	**8.328e-02**	+
$MaF13$	Min	3.912e-01	3.683e-01		4.888e-01	3.646e-01		5.072e-01	3.947e-01	
	Median	4.501e-01	4.412e-01		5.134e-01	4.323e-01		5.229e-01	4.252e-01	
	Mean	4.453e-01	4.403e-01	=	5.145e-01	**4.270e-01**	+	5.227e-01	**4.256e-01**	+
$MaF14$	Min	5.705e+00	1.538e+00		2.048e+01	1.580e+00		2.492e+01	1.557e+01	
	Median	1.093e+01	1.562e+00		2.391e+01	9.321e+00		2.621e+01	1.875e+01	
	Max	1.518e+01	6.273e+00		2.592e+01	1.353e+01		2.688e+01	2.030e+01	
	Mean	1.094e+01	**3.002e+00**	+	2.396e+01	**8.214e+00**	+	2.620e+01	**1.861e+01**	+
$MaF15$	Min	5.035e-02	2.346e-02		1.299e+00	8.531e-02		1.481e+00	1.179e-01	
	Median	5.098e-01	2.872e-02		1.527e+00	1.065e-01		1.629e+00	1.397e-01	
	Max	7.598e-01	4.449e-02		1.777e+00	1.247e-01		1.842e+00	1.734e-01	
	Mean	5.006e-01	**3.018e-02**	+	1.520e+00	**1.055e-01**	+	1.638e+00	**1.438e-01**	+
	w/t/l		**8/2/3**			**10/1/2**			**11/0/2**	

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i}}{n} \tag{5.5}$$

The IGD measures the convergence and the diversity of the obtained Pareto-optimal solutions simultaneously. The IGD metric measures the distances between each solution, composing the Pareto-optimal front and the obtained solution. Where d_i is the Euclidean distance between each point of the Pareto-optimal front and the nearest member of the obtained solution, and n is the number of solutions in the Pareto-optimal front.

The max, median, min, and mean of both algorithms are reported. Moreover, a Wilcoxon rank-sum test with a significance level of 95% was performed to have a statistical comparison for the best values achieved by the algorithms. The symbols ”+”, ”=”, and ”-” represent that performance of the CGDE3 is better than, similar to, or worse than the compared algorithm (GDE3), respectively. The best results are highlighted in **bold-face**. ”w/t/l” in the last row in the Table 5.2 means that CGDE3 wins in w functions, ties in t functions, and loses in l functions compared to the GDE3 algorithm.

As can be seen in Table 5.2, for D=100, CGDE3 outperforms GDE3 for eight functions ($MaF1$-$MaF2$, $MaF5$-$MaF7$, $MaF11$ and $MaF14$-$MaF15$). While, CGDE3 performs worse than GDE3 on three functions ($MaF3$-$MaF4$ and $MaF12$). They achieved the same result for two functions ($MaF10$ and $MaF13$). For D=500, it indicates that CGDE3 has a better result than GDE3 on 10 functions ($MaF1$-$MaF2$, $MaF5$-$MaF7$, $MaF10$-$MaF11$ and $MaF13$-$MaF15$). Whereas, CGDE3 has worse results than GDE3 on two functions ($MaF4$ and $MaF12$). They have the same result for one function ($MaF3$). For D=1000, it shows that CGDE3 performs better than GDE3 on 11 functions ($MaF1$-$MaF2$,

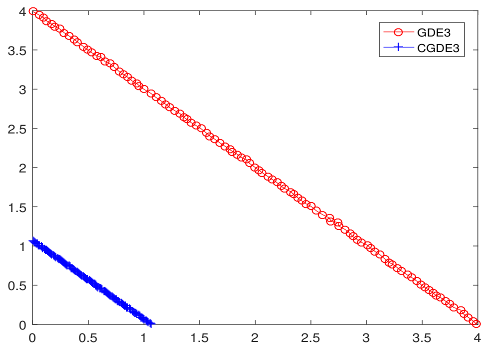
$MaF5$-$MaF7$, and $MaF10$-$MaF15$). Whereas, CGDE3 performs worse than GDE3 on two functions ($MaF3$ and $MaF4$). In fact, the proposed algorithm shows a superiority performance on all dimensions D=100, D=500 and D=1000.

Furthermore, the improved accuracy rate (I_{rate}) is computed and summarized in Table 5.3 as follows:

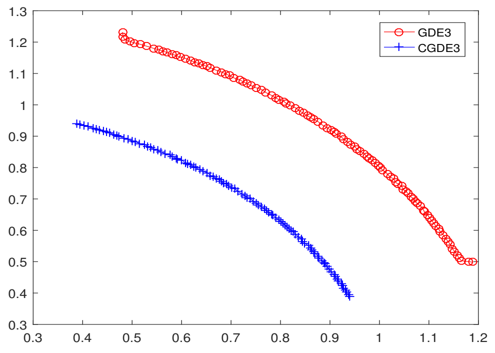
$$I_{rate} = \frac{Error\ of\ GDE3 - Error\ of\ CGDE3}{Error\ of\ GDE3} \times 100 \tag{5.6}$$

The improved accuracy rate reflects the relative improvement obtained from the proposed algorithm (CGDE3). The positive value shows that the CGDE3 algorithm outperforms GDE3. While the negative value indicates that the GDE3 algorithm is better than the CGDE3 algorithm. As seen from Table 5.3, on all dimensions, CGDE3 achieves better average improvement percentage than GDE3 with 41.32%, 39.39%, and 34.22% for dimensions 100, 500, 1000, respectively.

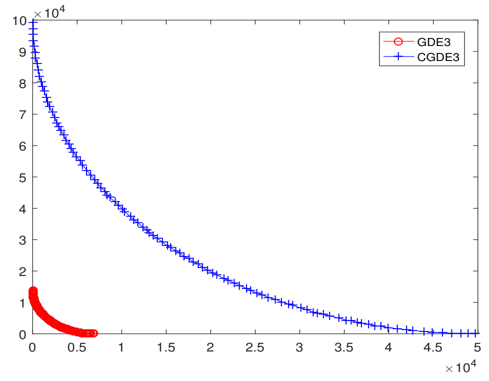
The obtained Pareto fronts by GDE3 and CGDE3 for some sample functions with dimension 1000 are given in Figure 5.1. The plots are based on the median IGD of 31 runs. As it is presented, the Pareto front for functions ($MaF1$, $MaF2$, $MaF5$, $MaF7$, $MaF11$, and $MaF12$) of the CGDE3 algorithm has better solutions compared with GDE3. As the IGD measure confirmed before, in these functions, Pareto fronts of CGDE3 are closer to the optimal Pareto front, so the solutions of CGDE3 could dominate the obtained solutions by GDE3. In $MaF10$, both algorithms have an almost similar performance so that their obtained Pareto fronts are overlapped. But for $MaF4$, GDE3 outperforms CGDE3.



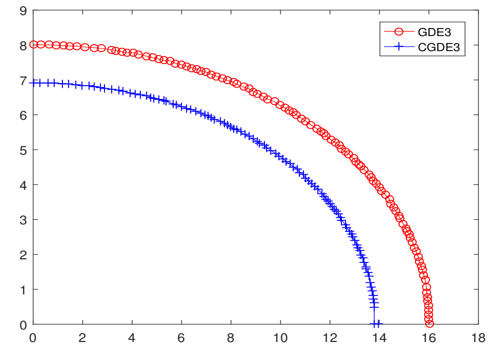
(a) $MaF1$



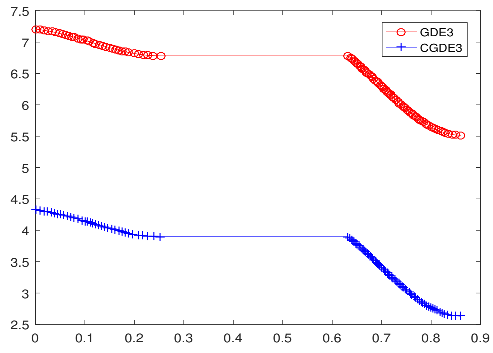
(b) $MaF2$



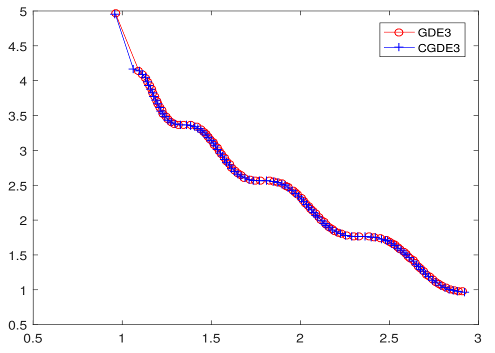
(c) $MaF4$



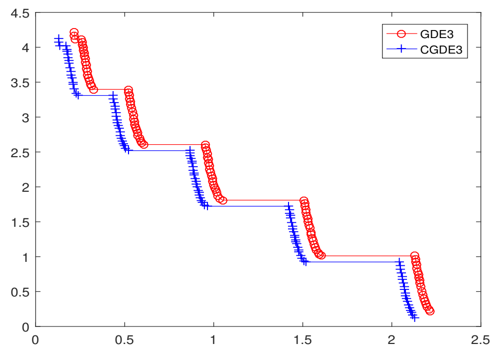
(d) $MaF5$



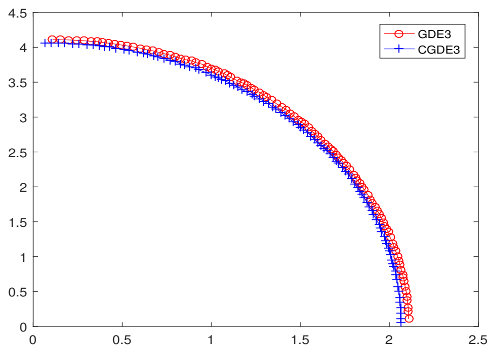
(e) $MaF7$



(f) $MaF10$



(g) $MaF11$



(h) $MaF12$

Figure 5.1: Obtained Pareto front of some sample functions for CEC 2017 benchmark problems for D=1000 using both algorithms GDE3 and CGDE3. The results are based on the median IGD of 31 runs.

Table 5.3 Improvement accuracy rate for dimensions 100, 500, 1000.

Functions	D=100	D=500	D=1000
$MaF1$	96.33%	98.69%	97.72%
$MaF2$	88.35%	95.37%	93.89%
$MaF3$	-96.78%	-83.79%	-90.69%
$MaF4$	-86.08%	-81.00%	-73.98%
$MaF5$	84.80%	55.70%	19.16%
$MaF6$	99.93%	99.98%	99.94%
$MaF7$	97.62%	99.46%	92.23%
$MaF10$	54.24%	42.49%	0.27%
$MaF11$	51.89%	44.39%	41.17%
$MaF12$	-20.87%	-35.01%	26.35%
$MaF13$	1.13%	17.00%	18.56%
$MaF14$	72.57%	65.72%	29.00%
$MaF15$	93.97%	93.06%	91.22%
Average	**41.32%**	**39.39%**	**34.22%**

5.5 Summary

This chapter studied the effectiveness of center-based mutation on the GDE3 algorithm for solving multi-objective optimization problems. Since center-based mutation has demonstrated a crucial impact on solving single-objective large-scale optimization problems, the influence of center-based mutation on the GDE3 algorithm was investigated to enhance its performance to solve bi-objective problems. In center-based mutation utilized in the GDE3 algorithm (CGDE3), five random candidate solutions were contributed to generate a new trial vector. The base vector of the GDE3 mutation scheme ($DE/rand/1$) is replaced by using the mean of three randomly selected candidate solutions as a mean value for the normal distribution generating a new solution around the center, which increases the opportunity to explore more promising regions in large-scale search spaces. Therefore, CGDE3 could obtain more accurate solutions for multi-objective optimization problems.

A comprehensive set of experiments were conducted on CEC 2017 benchmarks with dimensions 100, 500, and 1000. The experimental results confirmed that CGDE3 performs significantly better than GDE3. In the next chapter, the conclusion and future direction will be provided.

Chapter 6

Conclusion and Future Direction

This thesis has introduced comprehensive simulation-based investigations for center-based concepts and the idea of including a center-based concept for metaheuristic optimization algorithms to help reach an optimal solution, particularly for large-scale problems. First, the investigation for the closeness of the center to the unknown solution was introduced using two different approaches: the Monte-Carlo and Random Search algorithms. Second, center-based sampling at the operational level (mutation operation) for single solution algorithms in three different cases was proposed. Third, center-based sampling at the operational level for multi-objective optimization algorithms was proposed. The conclusion and future work of this thesis is described in the next two sections.

6.1 Conclusion

This thesis introduced an extended investigation in relation to center-based sampling using Monte-Carlo simulation for three different distances: Euclidean distance, Manhattan distance, and Cosine dissimilarity. The center-based concept has shown to be promising, especially for solving high denominational problems. According to the fact that the performance of the classical population-based algorithms is degraded when solving large-scale problems, in this thesis, the main contributions proposed were various techniques to improve the classical algorithms to solve large-scale optimization problems using center-based sampling effectively. Three different cases were proposed to improve several algorithms, such as the Random Search and Adaptive Random Search Algorithms, the Differential Evolution algorithm, the SHADE algorithm, and the GDE3 for multi-objective optimization algorithm. A center-based concept gives a higher chance of finding promising regions that promote identifying the global optimal. Therefore, most of the proposed algorithm cases could obtain more accurate solutions on high dimensional benchmark functions. The proposed algorithms were compared to the classical algorithms and evaluated on 15 selected discrete benchmark functions, CEC 2010, and 2013 LSGO with dimension 1000, CEC 2017 multi-objective competition with dimensions 100, 500, and 1000. Experimental results confirmed that the proposed center-based algorithms could significantly improve the performance of classical algorithms. In almost all test problems, the proposed algorithm outperformed the classical algorithms in terms of solution accuracy. These results indicate that utilizing a center-based concept at the operation level for optimizing algorithms is promising, especially in the exploration stage of the optimization process.

6.2 Future Work

Even though this thesis proposed several schemes using a center-based concept for solving large-scale problems, there are still many existing areas for continued improvement of meta-heuristic optimization algorithms. I seek to extend the idea of utilizing center-based sampling for optimization algorithms in several manners. Some recommendations are proposed to further using the center-based concept to improve the efficacy of the meta-heuristic algorithms:

- It is interesting to investigate the optimal value for the number of vectors to generate the center point. However, it is problem-dependent. It can be defined as a new control parameter.
- It is interesting to investigate the property of stagnation for center-based DE since the classical DE algorithm has difficulties in generating successful solutions when stagnation happens, in a scenario where the population is converged to a fixed point.

- Extend the use of center-based sampling ideas for other state-of-the-art multi-objective optimization algorithms. For example, implement center-based schemes at the operational level to enhance the classical multi-objective algorithms, such as the Multi-objective Differential Evolution (MODE) algorithm.
- It could be interesting to utilize a center-based concept for other population-based algorithms. For instance, one option could be embedding center-based sampling at the operation level of the Cooperative Co-evolution algorithms (CC).
- Another possibility could be designing a center-based scheme for the other operations, such as crossover and selection for the population-based algorithms.

References

[1] S. Tsutsui and A. Ghosh. “A study on the effect of multi-parent recombination in real coded genetic algorithms”. In: *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. IEEE. 1998, pp. 828–833.

[2] L. J. Eshelman. “The CHC adaptive search algorithm: How to have safe search when engaging”. In: *Foundations of Genetic Algorithms 1991 (FOGA 1)* 1 (2014), p. 265.

[3] S. Rahnamayan and G. G. Wang. “Center-based sampling for population-based algorithms”. In: *2009 IEEE Congress on Evolutionary Computation*. IEEE. 2009, pp. 933–938.

[4] A. Esmailzadeh and S. Rahnamayan. “Center-point-based simulated annealing”. In: *Electrical & Computer Engineering (CCECE), 2012 25th IEEE Canadian Conference on*. IEEE. 2012, pp. 1–4.

[5] S. Mahdavi, S. Rahnamayan, and K. Deb. “Center-based initialization of cooperative co-evolutionary algorithm for large-scale optimization”. In: *2016 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2016, pp. 3557–3565.

[6] A. Esmailzadeh and S. Rahnamayan. “Enhanced differential evolution using center-based sampling”. In: *2011 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2011, pp. 2641–2648.

[7] H.-Y. Fan and J. Lampinen. “A trigonometric mutation operation to differential evolution”. In: *Journal of global optimization* 27.1 (2003), pp. 105–129.

[8] H. Li and L. Zhang. “A discrete hybrid differential evolution algorithm for solving integer programming problems”. In: *Engineering Optimization* 46.9 (2014), pp. 1238–1268.

[9] H. Hiba, S. Mahdavi, and S. Rahnamayan. “Differential evolution with center-based mutation for large-scale optimization”. In: *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*. IEEE. 2017, pp. 1–8.

[10] H. Hiba, S. Mahdavi, and S. Rahnamayan. “Differential evolution with self-adaptive mutation scaling factor”. In: *Computational Intelligence (SSCI), 2017 IEEE Symposium Series on*. IEEE. 2017, pp. 1–8.

[11] H. Hiba, M. El-Abd, and S. Rahnamayan. “Improving SHADE with Center-based Mutation for Large-scale Optimization”. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2019, pp. 1533–1540.

[12] H. Hiba, A. Ibrahim, and S. Rahnamayan. “Large-scale Optimization Using Center-based Differential Evolution with Dynamic Mutation Scheme”. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2019, pp. 3189–3196.

[13] H. Hiba, A. A. Bidgoli, A. Ibrahim, and S. Rahnamayan. “CGDE3: An Efficient Center-based Algorithm for Solving Large-scale Multi-objective Optimization Problems”. In: *2019 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2019, pp. 350–358.

[14] Z. Yang, K. Tang, and X. Yao. “Multilevel cooperative coevolution for large scale optimization”. In: *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*. IEEE. 2008, pp. 1663–1670.

[15] S. Mahdavi, M. E. Shiri, and S. Rahnamayan. “Metaheuristics in large-scale global continues optimization: A survey”. In: *Information Sciences* 295 (2015), pp. 407–428.

[16] K. Tang, X. Yáo, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang. “Benchmark functions for the CEC 2010 special session and competition on large scale global optimization”. In: *Nature Inspired Computation and Applications Laboratory, USTC, China* 24 (2007).

[17] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, and H. China. “Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization”. In: *gene* 7.33 (2013), p. 8.

[18] D. Chen, X. Cao, F. Wen, and J. Sun. “Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 3025–3032.

[19] G. E. Hinton. “Training products of experts by minimizing contrastive divergence”. In: *Neural computation* 14.8 (2002), pp. 1771–1800.

[20] M. A. Potter and K. A. De Jong. “A cooperative coevolutionary approach to function optimization”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 1994, pp. 249–257.

[21] M. A. Potter. “The design and analysis of a computational model of cooperative coevolution”. PhD thesis. George Mason University Fairfax, Virginia, 1997.

[22] Y. Tenne and C.-K. Goh. *Computational intelligence in expensive optimization problems*. Vol. 2. Springer Science & Business Media, 2010.

[23] M. N. Omidvar, X. Li, and K. Tang. “Designing benchmark problems for large-scale continuous optimization”. In: *Information Sciences* 316 (2015), pp. 419–436.

[24] S. Mahdavi, M. E. Shiri, and S. Rahnamayan. “Cooperative co-evolution with a new decomposition method for large-scale optimization”. In: *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2014, pp. 1285–1292.

[25] S. Mahdavi, S. Rahnamayan, and M. E. Shiri. “Incremental cooperative coevolution for large-scale global optimization”. In: *Soft Computing* (2016), pp. 1–20.

[26] S. Mahdavi, S. Rahnamayan, and M. E. Shiri. “Multilevel framework for large-scale global optimization”. In: *Soft Computing* 21.14 (2017), pp. 4111–4140.

[27] S. Mahdavi, S. Rahnamayan, and M. E. Shiri. “Cooperative co-evolution with sensitivity analysis-based budget assignment strategy for large-scale global optimization”. In: *Applied Intelligence* (2017), pp. 1–26.

[28] R. Clarke, H. W. Ressom, A. Wang, J. Xuan, M. C. Liu, E. A. Gehan, and Y. Wang. "The properties of high-dimensional data spaces: implications for exploring gene and protein expression data". In: *Nature reviews cancer* 8.1 (2008), p. 37.

[29] K. Beyer, J. Goldstein, R. Ramakrishnan, and U. Shaft. "When is "nearest neighbor" meaningful?" In: *International conference on database theory*. Springer. 1999, pp. 217–235.

[30] P. Rigollet and J.-C. Hütter. "High dimensional statistics". In: *Lecture notes for course 18S997* (2015).

[31] *High Dimensional Space*. https://www.cs.cmu.edu/~venkatg/teaching/CStheory-infoage/chap1-high-dim-space.pdf. Accessed: 2019-11-15.

[32] S. Rahnamayan and G. G. Wang. "Toward effective initialization for large-scale search spaces". In: *Trans Syst* 8.3 (2009), pp. 355–367.

[33] G. Liu, C. Xiong, and Z. Guo. "Enhanced differential evolution using random-based sampling and neighborhood mutation". In: *Soft Computing* 19.8 (2015), pp. 2173–2192.

[34] S. Rahnamayan, G. G. Wang, and M. Ventresca. "An intuitive distance-based explanation of opposition-based sampling". In: *Applied Soft Computing* 12.9 (2012), pp. 2828–2839.

[35] V. Plagianakos, D. Tasoulis, and M. Vrahatis. "A review of major application areas of differential evolution". In: *Advances in differential evolution*. Springer, 2008, pp. 197–238.

[36] Y. Xu, L. Wang, and L. Li. "An effective hybrid algorithm based on simplex search and differential evolution for global optimization". In: *Emerging Intelligent Computing Technology and Applications. With Aspects of Artificial Intelligence* (2009), pp. 341–350.

[37] R. A. Khanum and M. A. Jan. "Centroid-based initialized jade for global optimization". In: *Computer Science and Electronic Engineering Conference (CEEC), 2011 3rd*. IEEE. 2011, pp. 115–120.

[38] M. Ali, M. Pant, and A. Nagar. "Two new approach incorporating centroid based mutation operators for Differential Evolution". In: *World Journal of Modelling and Simulation* 7.1 (2011), pp. 16–28.

[39] K. Chen and C. Wang. "Artificial bee colony algorithm improved by centroid strategy". In: ().

[40] H. Salehinejad and S. Rahnamayan. "Effects of centralized population initialization in differential evolution". In: *Computational Intelligence (SSCI), 2016 IEEE Symposium Series on*. IEEE. 2016, pp. 1–8.

[41] R. L. Anderson. "Recent advances in finding best operating conditions". In: *Journal of the American Statistical Association* 48.264 (1953), pp. 789–798.

[42] F. J. Solis and R. J.-B. Wets. "Minimization by random search techniques". In: *Mathematics of operations research* 6.1 (1981), pp. 19–30.

[43] L. Rastrigin. "The convergence of the random search method in the extremal control of a many parameter system". In: *Automaton & Remote Control* 24 (1963), pp. 1337–1342.

[44] D. C. Karnopp. “Random search techniques for optimization problems”. In: *Automatica* 1.2-3 (1963), pp. 111–121.

[45] J. Brownlee. *Clever algorithms: nature-inspired programming recipes*. Jason Brownlee, 2011.

[46] M. Alvo and L. Philip. *A parametric approach to nonparametric statistics*. Springer, 2018.

[47] P Gray, W Hart, L Painton, C Phillips, M Trahan, and J Wagner. *A survey of global optimization methods, Sandia National Laboratories*. 1997.

[48] M. G. Omran, A. Salman, and A. P. Engelbrecht. “Self-adaptive differential evolution”. In: *International Conference on Computational and Information Science*. Springer. 2005, pp. 192–199.

[49] D. E. Goldberg and H John. “Holland. Genetic algorithms and machine learning”. In: *Machine learning* 3.2-3 (1988), pp. 95–99.

[50] D. Corne, M. Dorigo, F. Glover, D. Dasgupta, P. Moscato, R. Poli, and K. V. Price. *New ideas in optimization*. McGraw-Hill Ltd., UK, 1999.

[51] S. Rahnamayan and G. G. Wang. “Investigating in scalability of opposition-based differential evolution”. In: *WSEAS Trans Comput* 7 (2008), pp. 1792–1804.

[52] J. Kennedy and R. E. P. S. Optimization. “Particle Swarm Optimization”. In: *Conf. on Neural Networks*. Vol. 4. 1995.

[53] R. C. Eberhart, Y. Shi, and J. Kennedy. *Swarm intelligence*. Elsevier, 2001.

[54] M. Dorigo, V. Maniezzo, and A. Colorni. “Ant system: optimization by a colony of cooperating agents”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 26.1 (1996), pp. 29–41.

[55] M. Dorigo and G. Di Caro. “Ant colony optimization: a new meta-heuristic”. In: *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*. Vol. 2. IEEE. 1999, pp. 1470–1477.

[56] D. Karaboga. *An idea based on honey bee swarm for numerical optimization*. Tech. rep. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[57] E.-G. Talbi. *Metaheuristics: from design to implementation*. Vol. 74. John Wiley & Sons, 2009.

[58] R. Storn and K. Price. “Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces”. In: *Journal of global optimization* 11.4 (1997), pp. 341–359.

[59] R. Storn and K. Price. “Differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces”. In: *Berkeley, CA: International Computer Science Institute* (1995).

[60] S. Rahnamayan, H. R. Tizhoosh, and M. M. Salama. “Opposition-based differential evolution”. In: *IEEE Transactions on Evolutionary computation* 12.1 (2008), pp. 64–79.

[61] S. Das and P. N. Suganthan. “Differential evolution: A survey of the state-of-the-art”. In: *IEEE Transactions on Evolutionary Computation* 15.1 (2011), pp. 4–31.

[62] S. Sardar, S. Maity, S. Das, and P. N. Suganthan. “Constrained real parameter optimization with a gradient repair based differential evolution algorithm”. In: *2011 IEEE Symposium on Differential Evolution (SDE)*. IEEE. 2011, pp. 1–8.

[63] V Ho-Huu, T Nguyen-Thoi, T Vo-Duy, and T Nguyen-Trang. “An adaptive elitist differential evolution for optimization of truss structures with discrete design variables”. In: *Computers & Structures* 165 (2016), pp. 59–75.

[64] D. T. Do, S. Lee, and J. Lee. “A modified differential evolution algorithm for tensegrity structures”. In: *Composite Structures* 158 (2016), pp. 11–19.

[65] V Ho-Huu, T Nguyen-Thoi, T Truong-Khac, L Le-Anh, and T Vo-Duy. “An improved differential evolution based on roulette wheel selection for shape and size optimization of truss structures with frequency constraints”. In: *Neural Computing and Applications* (2016), pp. 1–19.

[66] H. Zaheer and M. Pant. “A differential evolution approach for solving integer programming problems”. In: *Proceedings of Fourth International Conference on Soft Computing for Problem Solving*. Springer. 2015, pp. 413–424.

[67] P. Chaturvedi, P. Kumar, et al. “Population Segmentation-Based Variant of Differential Evolution Algorithm”. In: *Proceedings of Fifth International Conference on Soft Computing for Problem Solving*. Springer. 2016, pp. 401–410.

[68] H. Zaheer, M. Pant, O. Monakhov, and E. Monakhova. “A simple and efficient cooperative approach for solving multi-modal problems”. In: *Electrical, Electronics, and Optimization Techniques (ICEEOT), International Conference on*. IEEE. 2016, pp. 4001–4006.

[69] Z. Ma and G. A. Vandenbosch. “Impact of random number generators on the performance of particle swarm optimization in antenna design”. In: *2012 6th European Conference on Antennas and Propagation (EUCAP)*. IEEE. 2012, pp. 925–929.

[70] A. A. Bidgoli, S. Mahdavi, S. Rahnamayan, and H. Ebrahimpour-Komleh. “GDE4: The Generalized Differential Evolution with Ordered Mutation”. In: *International Conference on Evolutionary Multi-Criterion Optimization*. Springer. 2019, pp. 101–113.

[71] R. Tanabe and A. Fukunaga. “Success-History Based Parameter Adaptation for Differential Evolution”. In: *2013 IEEE Congress on Evolutionary Computation (CEC)*. IEEE. 2013, pp. 71–78.

[72] J. Zhang and A. C. Sanderson. “JADE: Adaptive differential evolution with optional external archive”. In: *IEEE Transactions on Evolutionary Computation* 13.5 (2009), pp. 945–958.

[73] W. Y., C. Z., and Z. Q. “Differential evolution with composite trial vector generation strategies and control parameters”. In: *IEEE Transactions on Evolutionary Computation* 15.1 (2011), pp. 55–66.

[74] K. Tang, X. Yáo, P. N. Suganthan, C. MacNish, Y.-P. Chen, C.-M. Chen, and Z. Yang. “Benchmark functions for the CEC’2010 special session and competition on large scale global optimization”. In: *Nature Inspired Computation and Applications Laboratory, USTC, China* 24 (2007).

[75] H. Seada and K. Deb. “Non-dominated Sorting Based Multi/Many-Objective Optimization: Two Decades of Research and Application”. In: *Multi-Objective Optimization*. Springer, 2018, pp. 1–24.

[76] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE transactions on evolutionary computation* 6.2 (2002), pp. 182–197.

[77] J. Lampinen. “DE’s selection rule for multiobjective optimization”. In: *Lappeenranta University of Technology, Department of Information Technology, Tech. Rep* (2001), pp. 03–04.

[78] S. Kukkonen and J. Lampinen. “An extension of generalized differential evolution for multi-objective optimization with constraints”. In: *International Conference on Parallel Problem Solving from Nature*. Springer. 2004, pp. 752–761.

[79] S. Kukkonen and J. Lampinen. “GDE3: The third evolution step of generalized differential evolution”. In: *Evolutionary Computation, 2005. The 2005 IEEE Congress on*. Vol. 1. IEEE. 2005, pp. 443–450.

[80] R. Cheng, M. Li, Y. Tian, X. Zhang, S. Yang, Y. Jin, and X. Yao. “A benchmark test suite for evolutionary many-objective optimization”. In: *Complex & Intelligent Systems* 3.1 (2017), pp. 67–81.

[81] Y. Tian, R. Cheng, X. Zhang, and Y. Jin. “PlatEMO: A MATLAB platform for evolutionary multi-objective optimization [educational forum]”. In: *IEEE Computational Intelligence Magazine* 12.4 (2017), pp. 73–87.

[82] N. Hansen and A. Ostermeier. “Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation”. In: *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. IEEE. 1996, pp. 312–317.

[83] A. W. Iorio and X. Li. “Solving rotated multi-objective optimization problems using differential evolution”. In: *Australasian Joint Conference on Artificial Intelligence*. Springer. 2004, pp. 861–872.

[84] K. Deb and H. Jain. “An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints.” In: *IEEE Trans. Evolutionary Computation* 18.4 (2014), pp. 577–601.

Appendix A

Discrete Benchmark Functions

Table A.1 Discrete Benchmark Functions, selected scalable ones from [8]

Shifted Benchmark Function	Variables range	$fmin$
$f_1(x) = (x_1-1)^2 + (x_n-1)^2 + n\sum_{i=1}^{n-1}(n-i)(x_i^2 - x_{i+1})^2$	[-5, 5]	0
$f_2(x) = \sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2 + (1-x_i)^2]$	[-5, 5]	0
$f_3(x) = \sum_{i=1}^{n} x_i^4 + \Bigg(\sum_{i=1}^{n} x_i\Bigg)^2$	[-5, 5]	0
$f_4(x) = -20\exp\Bigg(-0.02\sqrt{n^{-1}\sum_{i=1}^{n} x_i^2}\Bigg) - \exp\Bigg(n^{-1}\sum_{i=1}^{n} cos(2_i) + 20 + e\Bigg)$	[-30,30]	0
$f_5(x) = 1 + \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\Bigg(\frac{x_i}{\sqrt{i}}\Bigg)$	[-600, 600]	0
$f_6(x) = 10n + \sum_{i=1}^{n}\big[x_i^2 - 10\cos(2\pi x_i)\big]$	[-5, 5]	0
$f_7(x) = \sum_{i=1}^{n}(x_i - i)^2$	[-n, n]	0
$f_8(x) = \sum_{i=1}^{n} \mid x_i - 2i \mid$	[-2n, 2n]	0
$f_9(x) = \sum_{i=1}^{n} i^2(x_i - i)^2$	[-n, n]	0
$f_{10}(x) = \mid x_1 \mid + \mid x_2 \mid + ... + \mid x_n \mid$	[-100, 100]	0
$f_{11}(x) = x_1^2 + x_2^2 + ... + x_n^2$	[-100, 100]	0
$f_{12}(x) = 10n + \sum_{i=1}^{n}\big\{(x_i - i)^2 - 10\cos\big[2\pi\big(x_i - i\big)\big]\big\}$	[-n, n]	0
$f_{13}(x) = 1 + \sum_{i=1}^{n}\frac{(x_i-3i)^2}{400} - \prod_{i=1}^{n}\cos\Bigg(\frac{x_i-3i}{\sqrt{3i}}\Bigg)$	[-3n, 3n]	0
$f_{14}(x) = -20\exp\Bigg\{-0.02\sqrt{n^{-1}\sum_{i=1}^{n}\big[x_i - 4(n-i+1)^2\big]}\Bigg\} - \exp\Bigg\{n^{-1}\sum_{i=1}^{n}\cos\big[x_i - 4(n-i+1)\big] + 20 + e\Bigg\}$	[0, 4n]	0
$f_{15}(x) = \sum_{i=1}^{n} x_i^2 + \Bigg(\sum_{i=1}^{n}\frac{i}{2}x_i\Bigg)^6$	[-100, 100]	0

Appendix B

CEC-2010 Benchmark Functions

- Dimension: $D = 1000$

  Group size: $m = 50$

  $x = (x_1, x_2, \ldots, x_D)$: The candidate solution

  $o = (o_1, o_2, \ldots, o_D)$: The (shifted) global optimum

  $z = x - o, z = (z_1, z_2, \ldots, z_D)$: The shifted candidate solution

  P: A random permutation of $1, 2, \ldots, D$

$$F_{elliptic}(x) = \textstyle\sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} {x_i}^2$$

$$F_{rosenbrock} = \textstyle\sum_{i=1}^{D-1} [100({x_i}^2 - x_{i+1})^2 + (x_i - 1)^2]$$

$$F_{rastrigin} = \textstyle\sum_{i=1}^{D} [{x_i}^2 - 10\cos(2\pi x_i) + 10]$$

$$F_{ackley} = -20exp(-0.2\sqrt{\tfrac{1}{D}\textstyle\sum_{i=1}^{D} {x_i}^2}) - exp(\tfrac{1}{D})\textstyle\sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + c$$

$$F_{schwefel} = \textstyle\sum_{i=1}^{D} (\sum_{j=1}^{i} x_i)^2$$

$$F_1(x) = F_{elliptic}(z)$$

$$F_2(x) = F_{rastrigin}(z)$$

$$F_3(x) = F_{ackley}(z)$$

$$F_4(x) = F_{rot\_elliptic}[z(P_1 : P_m)] * 10^6 + F_{elliptic}[z(P_{m+1} : P_D)]$$

$$F_5(x) = F_{rot\_rastrigin}[z(P_1 : P_m)] * 10^6 + F_{rastrigin}[z(P_{m+1} : P_D)]$$

$$F_6(x) = F_{rot\_ackley}[z(P_1 : P_m)] * 10^6 + F_{ackley}[z(P_{m+1} : P_D)]$$

$$F_7(x) = F_{schwefel}[z(P_1 : P_m)] * 10^6 + F_{sphere}[z(P_{m+1} : P_D)]$$

$$F_8(x) = F_{rosenbrock}[z(P_1 : P_m)] * 10^6 + F_{sphere}[z(P_{m+1} : P_D)]$$

$$F_9(x) = \sum_{k=1}^{\frac{D}{2m}} F_{rot\_elliptic}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{rot_elliptic}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{10}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{rot\_rastrigin}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{rastrigin}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{11}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{rot\_ackley}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{ackley}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{12}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{schwefel}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{sphere}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{13}(x) = \sum_{k=1}^{\frac{D}{2m}} F_{rot\_rosenbrock}[z(P_{(k-1)*m+1} : P_{k*m})] + F_{sphere}[z(P_{\frac{D}{2}+1} : P_D)]$$

$$F_{14}(x) = \sum_{k=1}^{\frac{D}{m}} F_{rot\_elliptic}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{15}(x) = \sum_{k=1}^{\frac{D}{m}} F_{rot\_rastrigin}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{16}(x) = \sum_{k=1}^{\frac{D}{m}} F_{rot\_ackley}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{17}(x) = \sum_{k=1}^{\frac{D}{m}} F_{schwefel}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{18}(x) = \sum_{k=1}^{\frac{D}{m}} F_{rosenbrock}[z(P_{(k-1)*m+1} : P_{k*m})]$$

$$F_{19}(x) = F_{schwefel}(z)$$

$$F_{20}(x) = F_{rosenbrock}(z)$$

Appendix C

CEC-2013 Benchmark Functions

- Dimension: $D = 1000$

  Group size: $m = 50$

  $S = 50, 25, 25, 100, 50, 25, 25, 700$

  $S1 = \{50, 50, 25, 25, 100, 100, 25, 25, 50, 25, 100,$

  $25, 100, 50, 25, 25, 25, 100, 50, 25\}$

  x^{opt} : The optimum decision vector for which the value of the objective function is minimum. This is also used as a shift vector to change the location of the global optimum.

  $x = (x_1, x_2, \ldots, x_D)$: the candidate solution and D-dimensional row vector

  P: A random permutation of $1, 2, \ldots, D$

  T_{osz}: A transformation function to create smooth local irregularities.

  T_{asy}: A transformation function to break the symmetry of the symmetric functions.

  λ : A D-dimensional diagonal matrix with the diagonal elements is used to create ill-conditioning.

  R : An orthogonal rotation matrix which is used to rotate the fitness landscape randomly around various axes

  m : The overlap size between subcomponents

  $y = x - x^{opt}$

  $y_i = y(P_{[C_{i-1}+1]} : P_{[C_i]}) i \in 1, \ldots, |S|,$

  $y_{i1} = y(P_{[C_{i-1}-(i-1)m+1]} : P_{[C_i-(i-1)m]}) i \in 1, \ldots, |S|,$

  $y_{i2} = y(P_{[C_{i-1}-(i-1)m+1]} : P_{[C_i-(i-1)m]}) - x_i^{opt} i \in 1, \ldots, |S|,$

  $z_i = T_{osz}(R_i y_i), i \in 1, \ldots, |S| - 1$

  $z_{i1} = T_{asy}^{0.2} T_{osz}(R_i y_{i1}), i \in 1, \ldots, |S| - 1$

  $z_{i2} = T_{asy}^{0.2} T_{osz}(R_i y_{i2}), i \in 1, \ldots, |S| - 1$

$z_{|S|} = T_{osz}(y_{|S|})$

$R_i : a|S_i| \times |S_i|$

$F1_{elliptic}(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} {x_i}^2$

$F2_{rastrigin}(x) = \sum_{i=1}^{D} [{x_i}^2 - 10\cos(2\pi x_i) + 10]$

$F3_{ackley}(x) = -20exp(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} {x_i}^2}) - exp(\frac{1}{D})\sum_{i=1}^{D} \cos(2\pi x_i)) + 20 + c$

$F_4(x) = \sum_{k=1}^{|s|-1} w_i f_{elliptic}(z_i) + f_{elliptic}(z_{|s|})$

$F_5(x) = \sum_{k=1}^{|s|-1} w_i f_{rastrigin}(z_i) + f_{rastrigin}(z_{|s|})$

$F_6(x) = \sum_{k=1}^{|s|-1} w_i f_{ackley}(z_i) + f_{ackley}(z_{|s|})$

$F_7(x) = \sum_{k=1}^{|s|-1} w_i f_{schwefel}(z_i) + f_{schwefel}(z_{|s|})$

$F_8(x) = \sum_{k=1}^{|s1|} w_i f_{elliptic}(z_i)$

$F_9(x) = \sum_{k=1}^{|s1|} w_i f_{rastrigin}(z_i)$

$F_{10}(x) = \sum_{k=1}^{|s1|} w_i f_{ackley}(z_i)$

$F_{11}(x) = \sum_{k=1}^{|s1|} w_i f_{schwefel}(z_i)$

$$F_{12_{rosenbrockh}} = \sum_{i=1}^{D-1}[100({x_i}^2 - x_{i+1})^2 + (x_i - 1)^2]$$

$$F_{13}(x) = \sum_{k=1}^{|s1|} w_i f_{schwefel}(z_{i1})$$

$$F_{14}(x) = \sum_{k=1}^{|s1|} w_i f_{schwefel}(z_{i2})$$

$$F_{15}(x) = \sum_{k=1}^{D} (\sum_{j=1}^{i} x_i)^2$$