Enhanced Diversity based Differential Evolution Algorithm

MS Thesis

By:

Akhtar Abbas

1119-FBAS/MSCS/F21

Supervised By:

Dr. Qamar Abbas

Department of Computer Science

International Islamic University, Islamabad

2025

Enhanced Diversity based Differential Evolution Algorithm

By:

Akhtar Abbas

Registration No. 1119-FBAS/MSCS/F21

A thesis

Submitted in partial fulfilment of the requirements for the award of the Degree of

Master of Science (MS)

In Computer Science

Department of Computer Science

Faculty of Computing and Information Technology

International Islamic University Islamabad

2025

Dedicated to my father (Late) and my mother who make me able to introduce myself, and my wife and daughter who always gave me confidence in every circumstances, and all my teachers who always helped me out to became me successful.

Declaration

I hereby declare that this Thesis **“Enhanced Diversity based Differential Evolution Algorithm”** neither as a whole nor a part has been copied out from any resource. It is further declared that I have written this thesis entirely on the basis of my personal efforts, made under the proficient guidance of my thesis supervisor, **Dr. Qamar Abbas.**

If any part of this research thesis proved to be copied or found to be a research of some other individual, I shall standby any consequences.

No portion of this research work presented in this thesis report has been submitted in support of any other degree or qualification, of this or any other University or Institute of learning.

Akhtar Abbas

1119-FBAS/MSCS/F21

Abstract:

Differential evolution (DE) is an efficient and powerful population-based stochastic search technique for solving numerical optimization problems over continuous space, which has been widely applied in many scientific and engineering fields, especially in solving real world problems. Enhanced Diversity based Differential Evolution Algorithm focuses to incorporate diversity by in the population of DE algorithm. The diversity is incorporated by forming a q% pool of top performing individuals from the population and then choosing three best individuals unlike one individual as used in the base paper. The three randomly chosen individuals from a q% pool of best performing individuals will be helpful to enhance the convergence speed and maintain the diversity in the population that will be helpful to improve the solution quality in terms of average fitness values and enhance the convergence speed by escaping from local optima. A comprehensive test suit of 11 benchmark functions is used to test the convergence power of proposed enhancement. Research result shows significance improvement in convergence speed of proposed enhancement.

Table of Contents

Chapter 1: Introduction

1.1 Chapter Summary

In this chapter introduction of this thesis is discussed. Then the problem statement is also discussed in this chapter that is considered in this research work. The research objectives are also discussed in the chapter as well as research question are also stated.

1.2 Introduction:

Differential Evolution Algorithm was introduced in 1995 by Price and Storm. It was most simple algorithm and was giving best optimization solution that’s why many researchers attracted by this algorithm [1]. DE is a very powerful and easily understandable technique and use limited number of parameters which make DE more effective and advantage able [2]. In Evolutionary Algorithm it’s difficult to find the computations and take more efforts to find the results and comparatively Differential Evolution Algorithm provide fast and strong searching capability causing quality solutions of given problems.DE has only three control parameters which are NP number of population famously population size, F is scaling factor and CR is crossover rate. DE use different mutation strategies which gives different results some are better searching in local optima and some provides best results in Global convergence [3]. That’s why it’s easy to apply on many optimization problems as well as on soft computation problems. It’s using mutation, crossover and selection operation that makes DE more effective than other EA’s, these operations also guide EAs to provide high performance.

The mutant vector is created by using mutation operation. It uses the amplified difference vector to a third vector to create a mutant vector. It is helpful to explore the search space by randomly selecting individuals. It controls the deviation of new vectors from the existing solutions and help the algorithm to avoid from local optima [4]. The trial vector is created during the crossover operation of differential evolution algorithm. It basically includes the components from the target vector and donor vector by using the crossover rate parameters by random comparison. The operator is helpful to incorporate diversity in the population members of differential evolution algorithm [5]. The selection operator works on the idea of survival of the fittest. It uses greedy approach and search for the best performing individual among the trial and parent vector by comparing their fitness values. This operator helps to keep better performing individuals in the next generation of the population that helps to improve the convergence speed of DE algorithm [6].

Differential evolution algorithm has number of real life applications such as DE algorithm is used for relevant feature selection for various machine learning applications [7]. DE algorithm is also applied to image processing applications like alignment of different times images by reducing the variation of overlapping pixel [8]. DE can succefully determine maximum coverage in the farm field to place optimal sensor in the field of wireless sensor network optimization [9]. DE algorithm is also applied in the domain of bioinformatics for key genes for cancer classification [10]. DE algorithm has been successfully applied in financial application [11], cryptanalysis [12], manufacturing process optimization [13], Wind Farm Layout Optimization [14], image segmentation [15], Water Distribution Network [16], Renewable Energy Systems [17], Robotics Path Planning [18], Cloud job scheduling [19], Drone and UAV [20] [21], underwater vehicles path planning [22] etc.

1.3 Problem Statement:

The Modified Differential Evolution Algorithm encounters difficulties in effectively navigating complex problem landscapes due to its tendency to get trapped in local optima. This issue arises primarily from a lack of search space exploration from the algorithm's mutation strategy, which leads to reduced population diversity. Consequently, the algorithm becomes prone to becoming stuck in local optima/valleys, preventing its ability to find optimal solutions for complex problems.

1.4 Objective

- To present a new mutation strategy in the Modified Differential Evolution Algorithm to enhance its exploration capabilities and increase population diversity.
- To incorporate diversity in the proposed mutation and assess its performance by considering standard benchmark optimization problems.

1.5 Research Questions

- RQ1: What specific mutation strategies can be employed within the Modified Differential Evolution Algorithm to enhance exploration and promote population diversity, thus mitigating the occurrence of local optima traps in complex problem landscapes?

- RQ2: How proposed mutation strategy is helpful to incorporate diversity and enhance exploration capability in Differential Evolution Algorithm in achieving improved performance.

1.6 Thesis Organization

Chapter # 2 of this thesis discusses DE algorithm, its operators. Literature review of this thesis, pseodocode and flowchart are given chapter # 2 and 3 of this thesis. Chapter # 4 contains Benchmark functions, Chapter # 5 contains detail of proposed methodology, Chapter # 6 contains detail of parameter settings and simulation results. At the end conclusion of the thesis is given

Chapter # 2 Introduction to Differential Evolution Algorithm

2.1 Evolutionary Algorithms

As we know that Differential Evolution Algorithm are the type of Evolutionary Algorithms and before understanding the DE it is mandatory to initially understand the Evolutionary Algorithms. Hierarchical structure of Evolutionary Algorithm may be shown as under.

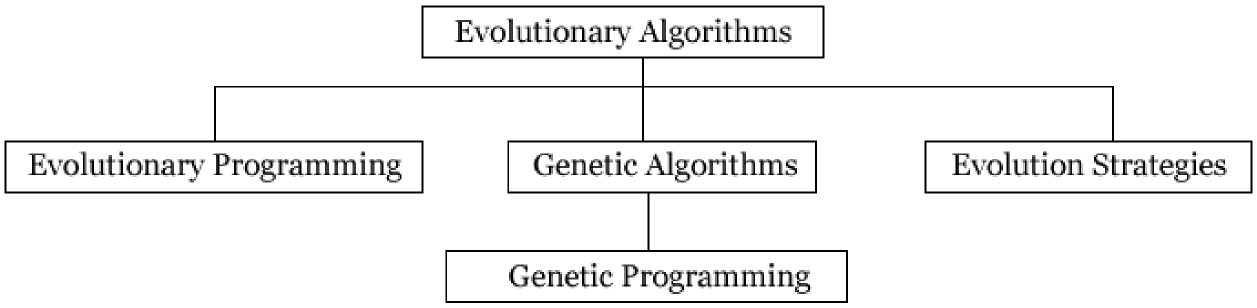


Figure 1. Hierarchical Structure of Evolutionary Algorithms

2.2 Working of Differential Evolution Algorithm

As we know that DE algorithm provides optimization solutions. It is best searching algorithm that provide best results in given population. DE algorithm is a population based algorithm that has three operators of mutation, crossover and selection. These operators are similar to genetic algorithms but working of these steps is different from genetic algorithm. This algorithm performs mutation operation on the selected parents and then perform crossover on child and parent vectors and then perform selection between child individual and the parent individual in a greedy manner.

Differential Evolution uses NP (no of population) D-dimensional parameter vectors and it is a parallel direct search method. When we get the resulting offspring we compare these offspring with parents and then we chose the better fitness holders from both parents and offspring and we get this fitness from mutation, crossover and selection steps, on the base of selection process we chose once from parents and from child and this selection is purely on the basis of best fitness. From given below pictures and equation working of DE is more clearly understandable.

As we already know that DE uses NP (no of population) D-dimensional parameter vectors by parallel direct search method. Firstly, proposed by **Rainer Storn** and **Kenneth Price**.

$X_{i,G}$, i = 1, 2, ….. , NP (Number of Population ; for the population of each generation G.

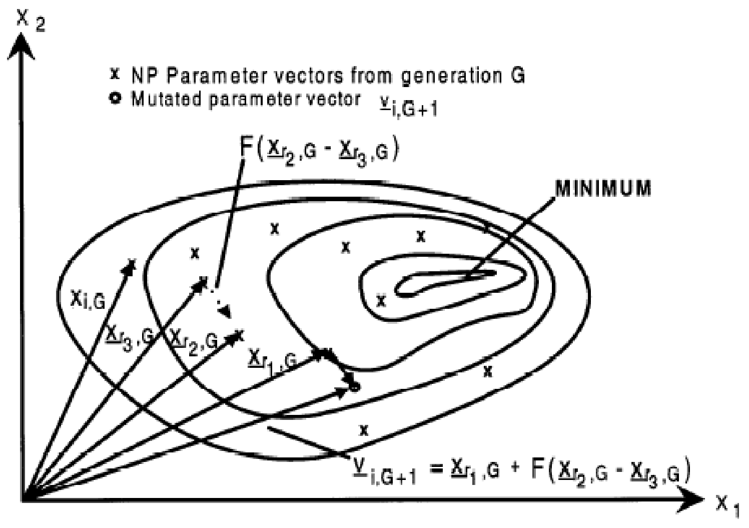
Differential Evolution works in three steps Mutation, Crossover and selection phase as explained by Rainer Storn and Kenneth Price, we will see each step in details as below.

2.2.1 Mutation Phase

To calculation of each target vector by $X_{i,G}$, i = 1, 2, ….. , NP, and mutation vector may be generated as under

$$V_{i,G+1} = X_{r1,G} + F \,.\, \left(X_{r2,G} - X_{r3,G}\right) \qquad (1)$$

By random indexes may be done as $r_1, r_2, r_3 \in \{1, 2, …., NP\}$, integer mutually different and F > 0. In mutation operation we need minimum of three parents to perform this operation, normally the selection of parents depends on the mutation strategy being used and for default mutation strategy it uses three random parents namely r_1, r_2 and r_3 are taken to perturb the current population member i, from the population having size NP. The amplified difference vector $\left(X_{r2,G} - X_{r3,G}\right)$uses the amplification factor F ϵ [0, 1] in order to generate the donor vector from this operation. The illustration of mutation strategy is given in the following diagram which will generate the donor vector the $V_{i,G+1}$.



x_2
x NP Parameter vectors from generation G
o Mutated parameter vector $\underline{v}_{i,G+1}$
$F(\underline{x}_{r_2,G} - \underline{x}_{r_3,G})$
MINIMUM
$x_{i,G}$
$\underline{x}_{r_3,G}$
$\underline{x}_{r_2,G}$
$\underline{x}_{r_1,G}$
$\underline{v}_{i,G+1} = \underline{x}_{r_1,G} + F(\underline{x}_{r_2,G} - \underline{x}_{r_3,G})$
x_1

Figure 2. Generic Mutation Process

Two-dimensional illustration which is responsible for the generation of $V_{i,G+1}$.

2.2.2 Crossover Phase:

Crossover is for the elimination of the disconcerted parameters. The trial vector is:

$$u_{i,G+1} = (u_{1i,G+1}, u_{2i,G+1}, \quad . . . , \quad u_{Di,G+1})$$

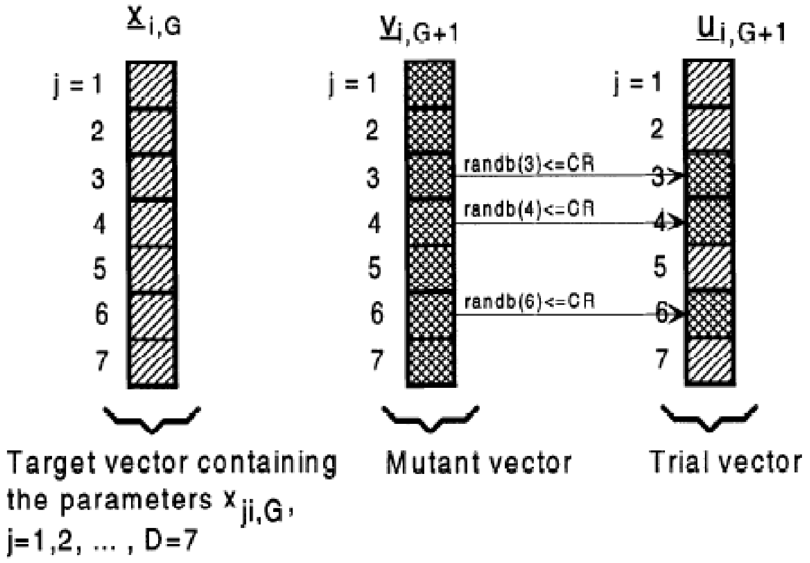


Figure 3. Generic Crossover Process

The above mentioned is the example of crossover process for D = 7 parameters is formed, where

$$U_{ji,G+1}tio = \begin{cases} v_{ji,G+1}, \ if \ (randb(j) \ \leq \ CR) \ or \ j \ = \ rnbr(i) \\ x_{ji,G} \quad , \ if \ (randb(j) > \ CR) \ or \ j \ \neq \ rnbr(i) \end{cases}$$

(2)

j = 1, 2, …, D.

The above equation is performing the crossover operation of differential evolution algorithm. It basically utilizes the crossover probability CR∈[0, 1] to generate the trial vector. Trial vector contains the components from the donor vector and the target vector. A random number is generated at run time and compared with the specified crossover rate to get components from the donor and the target vector. The index j is for the current dimension, index I is for the population individual, v is a donor vector and x is the target vector. For each individual of the population the crossover operations is performed after the mutation operation.

2.2.3 Selection Phase

The selection operator is another important operator of the differential evolution algorithm. It basically implements survival of the fittest in a greedy manner. The result of mutation operation is the donor vector, and the output of crossover operation is the trial vector. Selection operator selects the best performing individual between parent and the child and select the individual with best average fitness values. The trial vector is denoted by $u_{i,G+1}$, donor vector is $v_{i,G+1}$ and the target vector is $x_{i,G}$.

2.4 Commonly Used Mutation Strategies

The most commonly used mutation strategies used in DE algorithm are discussed in this section. The detail of these mutations strategies is discussed as follows:

$$V_i = X_{r1} + F \,.\, (X_{r2} - X_{r3}) \tag{3}$$

$$V_i = X_{best} + F \,.\, (X_{r1} - X_{r2}) \tag{4}$$

$$V_i = X_i + F \,.(X_{best} - X_i) + F \,.\, (X_{r1} - X_{r2}) \tag{5}$$

$$V_i = X_{r1} + F \,.(X_{r2} - X_{r3}) + F \,.(X_{r4} - X_{r5}) \tag{6}$$

$$V_i = X_{best} + F \,.\, (X_{r1} - X_{r2}) + F \,.\, (X_{r3} - X_{r4}) \tag{7}$$

2.5 Crossover strategies in DE Algorithm

The detail of crossover strategies is discussed as follows

2.5.1 Binomial crossover (or) uniform crossover:

Binomial crossover is a process of crossover in which taking offspring vector U from mutant vector V, and current vector of population X by using probability of $C_r$that described in Algorithm 1.

Algorithm 1: Binomial Crossover

```
1: $\boldsymbol{crossoveBin}(X, V)$
2: $k \leftarrow irand(\{1,2, \ldots\ldots, n\})$
3: for $j \in \{1,2, \ldots, n\}$do
4: if$rand(0,1) < C_r \boldsymbol{or}\ j = k$then
5: $U_j = V_j$
6: else
7: $U_j = X_j$
8: end if
9: end for
```

```
    10:  **return** $U$
```

2.5.2 Exponential crossover:

Exponential crossover is a crossover process in which choosing offspring one parameter randomly and copy it from trail vector or from mutant and offspring is may be completely different from the which comparison its generated. In exponential crossover with two-point crossover select randomly first point from $\{1,2,\ldots\ldots,n\}$ and second point is select by using L consecutive components from mutant that described in Algorithm 2.

Algorithm 2: Exponential Crossover

```
1:  $\boldsymbol{crossoveExp}(X,V)$
2:  $U \leftarrow X;\ k \leftarrow irand(\{1,2,\ldots\ldots,n\});\ j \leftarrow k;\ L \leftarrow 0$
3:  while $rand(0,1) > C_r \boldsymbol{or}\ L = n$ do
4:      $U_j = V_j;\ \ j \leftarrow \langle j+1\rangle_n\ ;\ L \leftarrow L+1$
5:  end while
6:  return $U$
```

2.6 Flow Chart of Differential Evolution Algorithm

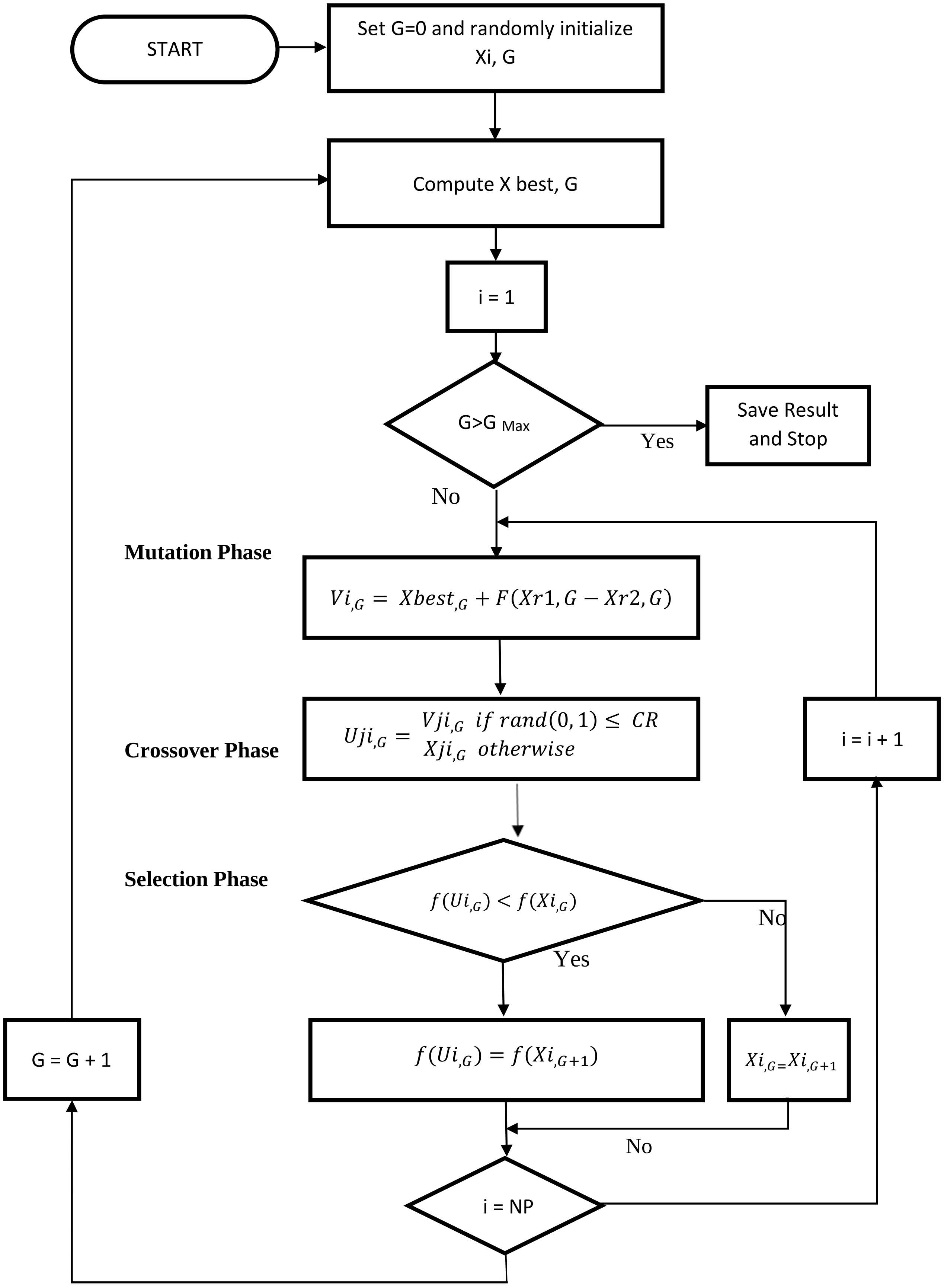

START
Set G=0 and randomly initialize Xi, G
Compute X best, G
i = 1
G>G Max
Yes
Save Result and Stop
No
Mutation Phase
$Vi_{,G} = Xbest_{,G} + F(Xr1, G - Xr2, G)$
Crossover Phase
$Uji_{,G} = \begin{matrix} Vji_{,G} \ if \ rand(0,1) \leq CR \\ Xji_{,G} \ otherwise \end{matrix}$
i = i + 1
Selection Phase
$f(Ui_{,G}) < f(Xi_{,G})$
No
Yes
G = G + 1
$f(Ui_{,G}) = f(Xi_{,G+1})$
$Xi_{,G=}Xi_{,G+1}$
No
i = NP

Figure 4. Flow Chart of Differential Evolution Algorithm

Chapter 3: Literature Review

3.1 Summary of chapter

This chapter presents detail of literature review that contains differential evolution algorithmic evolution, parametric evolution and hybridization evolution by various researchers. The summary of literature review is also presented in tabular form also. At the end of this chapter, research gap is discussed.

3.2 Literature review

Differential Evolution is very simple algorithm but many researchers purposed many techniques for better results for optimization and some researchers provide some techniques that work in collaboration with DE to provide better results. This study [23] introduces a modified version of the LSHADE-SPACMA algorithm and improves several key features. The key improvements are the generation and precise removal mechanism adopted that removes the low-performing members and replaces them with new solutions extracted from the high-performing individuals, improving local exploitation. This study also presents a new mutation strategy named rank-based selective pressure which helps maintain the population's diversity and directs the search to better solutions. Maintain only high-quality solutions for refining the process of high-quality external achievement mechanisms which reduces the randomness and improves convergence. These modifications aim to improve the efficiency and robustness of the algorithm. The modified mLSHADE-SPACMA algorithm was tested on benchmark datasets such as CEC2014 and CEC2017 to enhance its performance. This algorithm was also tested on a real-world point cloud registration dataset to evaluate its practical suitability. The test results of the CEC2014 and CEC2017 datasets prove that the mLSHADE-SPACMA algorithm beats the most competing algorithms. The modified version of the algorithm is tested on higher dimensions (30, 50, and 100) where its performance is slightly struggled on lower dimensions and outperformed on high-dimensional problems and can tackle complex problems. On the other hand, it showed better convergence and accuracy on point cloud registration and demonstrated its real-world effectiveness compared to previous methods. The modified version is a powerful tool for solving complex optimization problems.

This study [24] proposed an improved version of Differential Evolution (DE) to tackle parametric sensitivity, convergence speed, and high dimensional adaptability in optimization problems. A few major improvements in the proposed LSHADE-Code DE algorithm include a novel mutation strategy to balance the exploration and exploitation, also introduces a dual-stage self-adaptation technique to adjust the dynamic parameters. Furthermore, to preserve high-performing solutions and avoid premature convergence, it uses an upgraded population size reduction scheme. The target of the proposed algorithm is to enhance stability, accuracy, and convergence speed. LSHADE-Code algorithm was applied to benchmark functions like CEC 2011, CEC 2020, and CEC 2022, which cover various optimization problems. The output proved that the LSHADE-Code outperformed state-of-the-art algorithms like IMODE and AGSK in most cases even achieving 100% accurate results in many functions across various dimensions. But it positioned 3rd in CEC 2022, and it still proved strong performance, particularly when it achieved the global optima in F1, F3, F5, and F11 functions. The proposed algorithm effectively balances exploration and exploitation while achieving accurate solutions, stability, and convergence speed in high-dimensional optimization problems. However, specialized models perform slightly better in some cases.

The study [25] proposed the Two-Stage Differential Evolution (TDE) algorithm, which improved the estimation of parameters for Proton Exchange Membrane Fuel Cells (PEMFCs). Also introduced an innovative mutation strategy in which historical and inferior solutions are used to improve exploration and convergence speed. It also utilizes fitness-independent parameter control, in which parameters are adjusted based on solution diversity rather than fitness, which is aimed to maintain population diversity to gain improvement in accuracy and convergence speed. These enhancements make TDE more efficient and powerful. The introduced algorithm was tested on six commercial PEMFC stacks like BCS 500 W, Nedstack PS6, and Ballard Mark V, across 12 case studies. It minimizes the Sum of Squared Errors (SSE) between experimental and predicted cell voltages to optimize seven key parameters. TDE consistently succeeded in achieving the lowest SSE and faster runtime as compared to nine state-of-the-art DE algorithms. TDE reduced SSE by 41% in the BCS 500 W case and 98% faster than HARD-DE, also similar enhancements were found across all tested models. The key improvements of TDE are accuracy, stability, convergence speed, balance of exploration and exploitation, reduction in SSE, and 98% to 99% faster than existing DE algorithms.

This research work [1] introduces a Multi-Hybrid Differential Evolution (MHDE) algorithm which is designed to improve traditional Differential Evolution functions for structural optimization. It balances exploitation and exploration in a better way, reduces the adaptive population size, and uses some hybrid techniques like Grey Wolf Optimization (GWO) and Cuckoo Search (CS) to boost global and local search capabilities. These modifications help to enhance the convergence speed and improve efficiency. The key improvements of the proposed algorithm are performing efficiently on structural and engineering optimization problems, better convergence due to the hybrid approach, avoiding local optima, reduced computational costs, highly effective for real-world applications, industrial design and mechanical engineering, and large-scale computational intelligence problems optimization. MHDE was tested on IEEE CEC benchmark functions (2005, 2014, 2017) and some real-world engineering problems and frame structure design problems like pressure vessel design, rolling element bearing design, tension/compression spring design, cantilever beam design, 1-bay 8-story frame, 3-bay 15-story frame, 3-bay 24-story frame. It consistently performs better on start-of-the-art algorithms like JADE, SHADE, SHADE, CMA-ES, and other DE variants, and GWO-E. MHDE at 1st position in various CEC 2005 functions and outperformed in minimizing structural weights while maintaining integrity. Its statistical significance superiority is confirmed by the Wilcoxon rank-sum and Friedman rank tests.

This research work [26] proposed an Enhanced Differential Evolution (EDE) algorithm to improve the accuracy and efficiency of photovoltaic system parametric identification. The existing PV parameter identification techniques and metaheuristic approaches face some problems like lack of accuracy, suffering from computational inefficiencies, or being problem-dependent. To tackle these problems, they do a few modifications like stage-specific mutation strategies designed to maintain balance exploration and exploitation, on the search stages, it utilizes Adaptive mutation factors and crossover rates dynamically, minimizes Root Mean Square Error (RMSE) between experimental and estimated current values of PV models, and to improving accuracy, it adopted the approach to single, double, and triple diode PV models. The improved EDE is highly accurate, and stable, leading to faster convergence, better parametric estimation, improved search efficiency, and efficient for PW system modeling. The proposed DED was tested on two PV cells (RTC France, PVM-752-GaAs) and three PV modules (ND-R250A5, STM6 40/36, STP6 120/36), it achieved the lowest Root Mean Square Error (RMSE) value as compare to ABSO, PSO, and GA. It consistently gets the results like RTC France (RMSE: 7.730062e-4 for single-diode model), PVM-752-GaAs (RMSE: 1.30181e-4 for triple-diode model), ND-R250A5 PV module (RMSE: 7.697716e-3).

The article [27] introduces a Hybrid DE algorithm to tackle a problem known as Resource-Constrained Project Scheduling, focusing on flexible project structures when a subset of activities must be selected and scheduled, and cumulative resource constraints during the scheduling of consumption and production process. While existing algorithms struggle with scalability, optimality, and handling cumulative resources. It uses a group ordering strategy to solve the activity selection problem efficiently in polynomial time, also implements Forward-Backward Improvement (FBI) to refine schedules and avoid local optima, and proposes a penalty-based approach to handle infeasibility in cumulative resource constraints. The key improvements are construction project scheduling in which activities may be optional and resources have complex dependencies, manufacturing and production planning to optimize assembly lines with non-renewable materials, shipbuilding & transportation logistics, where tasks consume and produce different resources dynamically, and smart city infrastructure projects to ensuring efficient allocation of limited space and financial resources. The proposed HDE was tested on benchmark functions & datasets from the literature (Servranckx & Vanhoucke (2019)) and a Tabu Search (TS) algorithm from Servranckx & Vanhoucke (2019). And compared with the Ant Colony Optimization (ACO) algorithm. The key results achieved that the HDE performed better or equal in 94% of test cases, the lower bound on the optimality gap (BOG) was significantly reduced as compared to other heuristic algorithms, and HDE achieved optimal solutions in 98% of known optimal instances.

The research study [28] introduced a Multi-Strategy Probabilistic Discrete Differential Evolution (MSPDDE) Algorithm for optimizing the parameter rendering of the Cesium 3DTiles model. The introduced MSPDDE algorithm automates this optimization process by using multiple mutation strategies to avoid local optima and enhance convergence speed, manage discrete and continuous variables through the application of a probabilistic discretization approach, reduce rendering time while maintaining visualization quality by optimal configuration of Cesium 3DTiles rendering parameters. The proposed solution is based on a mutation strategy pool to balance exploration and exploitation, probabilistic discretization to handle discrete parameter ranges, and a rendering time fitness function to improve parameter configurations.

The introduced MSPDDE algorithm was applied to three real-world Cesium 3DTiles models Huxi Village (2.54 GB), Qishan Park (5.39 GB), Huludao City (54.8 GB). The performance was compared with default Cesium rendering parameters, and an existing tuning method known as Progressive Loading on Demand (PLOD). Key improvements of the proposed algorithm can reduce the time of rendering. Huxi Village was 28.84% faster, Qishan Park was 27.89% faster, and Huludao City was 13.32% faster. It reduces computational errors, achieves higher optimization accuracy compared to manual tuning, more robust parameter selection, avoids local minima, and eliminates manual parameter adjustments.

This study [29] introduces a Two-Stage Adaptive Differential Evolution Algorithm with Accompanying Populations (APDE) to improve global optimization performance, enhancing convergence speed while avoiding premature stagnation, a two-stage evolution mechanism adopted to balance exploration and exploitation, and proposes an accompanying population that helps to maintain diversity and guide the search. This proposed mechanism tackles problems like premature convergence, exploration and exploitation, and parametric adaptability. APDE algorithm focused on global exploration and emphasizing local exploitation, high quality solutions and diverse solution to accompanying population mechanism, and adapted dynamic mutation and crossover operators. The proposed solution was applied on benchmark functions from CEC2005 and CEC2017 and compared with nine state-of-the-art optimization algorithms like CoDE, JADE, SaDE, NSDE, BDE, and Standard DE variants. Each algorithm was tested 30 times independently, with a maximum function evaluation limit of 1000 × D. The performance of APDE was outperformed competing methods in 16 out of 23 CEC2005 functions and 13 out of 30 CEC2017 functions, improved convergence speed, more accurate results, and enhanced the handling of Multi-Modal functions that effectively escapes local optima and making it highly effective in complex landscapes.

This research work [30] introduces the Dual-Performance Multi-Subpopulation Adaptive Restart Differential Evolutionary (DPR-MGDE) Algorithm to deal with the high-dimensional optimization problems and limitations of traditional DE algorithms in complex. The key improvements are dual-performance metrics which categorize the fitness-based population and historical update frequency, multi-subpopulation division that splits the population into three groups promising individuals, medium individuals, and unpromising individuals where each group is used for targeted mutation strategies to refine solutions, employs balanced mutation for local exploitation, random mutation strategies to maintain diversity respectively, adaptive crossvariation strategy to adjust mutation and crossover rates dynamically to enhance the efficiency of local and global search, and utilizing a collision-based Gaussian wandering restart strategy to Prevents stagnation using detecting search stagnation and restarting members dynamically. The Proposed algorithm was tested on the CEC2017 benchmark functions, which include 30 optimization test functions covering unimodal functions, multimodal functions, and composite functions, and then compared with seven state-of-the-art DE variants like NPADE, IDE-EDA, FADE, MPEDE, ADE-DMRM, MMDE, DPMADE. Performance metrics are measured by mean error, standard deviation, and Wilcoxon rank-sum test. The introduced DPR-MGDE algorithm outperformed all seven DE variants in most benchmark functions, effectively escapes local optima and finds optimal solutions, utilizes intelligent subpopulation division for structured evolution, improved convergence speed and adaptive variation control enhance diversity while optimizing solutions.

The study [31] introduces a Multi-Population Differential Evolution Approach for Feature Selection with Mutual Information Ranking to reduce dimensionality. To enhance the global search and prevent premature convergence, it uses a multi-population strategy, Lens Imaging Opposition-Based Learning (LensOBL) to maintain the diversity of population and avoid local optima, and a hierarchical individual improvement approach to maintaining the balance of exploration and exploitation. The proposed solution aims to tackle four problems like High-dimensional data challenges where huge feature spaces increase cost in computation and affect accurate classification, redundant and unimportant feature selection, premature convergence in local optima, and unbalanced exploration and exploitation. The algorithm was applied 11 UCI datasets like Heart, Zoo, Parkinsons, Dermatology, Ionosphere, etc., and 9 ASU high-dimensional datasets like Madelon, Leukemia, Colon, Prostate_GE, Arcene, etc., and results were compared with benchmark algorithms like PSO, DE, MSPSO, COPSO, TLPSO, BASO, ECSA, ISSA, and PLTVACIW-PSO. Performance is measured based on classification accuracy, Number of selected features, computation cost, and optimal fitness value. The proposed MI-MPODE performs better state-of-the-art algorithms in majority of the datasets, feature reduction, improvement in better handling of high-dimensional data, improved population diversity and robust global search, and enhanced convergence speed on complex problems and datasets.

This study [32] proposed HSA-DELF, a Hybrid Harmony Search Algorithm (HSA) integrating Differential Evolution and Lévy Flight for engineering optimization problems. The paper focused on integrating DE and LF into HSA for improving exploration and exploitation, adapting the dynamic adjust search mechanism by multi-mutation strategies and high-performing members, enhancing convergence speed and avoiding local optima through pairwise iterative updates, population diversity, and avoiding local optima in stagnation. The proposed solution was tested on 23 classic benchmark functions like unimodal, multimodal, and fixed-dimension functions, and 12 CEC 2022. also tested across seven real-world constrained engineering optimization problems. The results were compared with 5 metaheuristic algorithms HSA, DE, CSA, GA, PSO, and 4 HSA variants (IHS, MHSA, IHSDE, IMGHSA). The key improvements that the algorithm achieved better results in 22 out of 23 benchmark functions, enhanced convergence speed by decreasing iterations needed to meet optimal solutions, better robustness, and improved performance in 6 out of 7 engineering problems by achieving 85.71% accuracy, minimized stagnation ratio in optimization runs by Lévy Flight significantly, and improved population diversity.

The study [33] presents a Q-learning-based Differential Evolution (QLDE) algorithm. The proposed solution was integrated with K-means clustering for customer segmentation in digital marketing. The introduced framework enhances the clustering performance of K-means. The algorithm improves customer segmentation by feature dimensionality reduction, avoiding redundancy in data, retaining related features for classification improvements, dynamic mutation factor to improve search performance and prevent stagnation, diverse cluster segmentation, and determining the optimal number of clusters by elbow method. The proposed QLDE algorithm was applied on customer segmentation dataset which is publicly available on Kaggle. The test was based on RFM (Recency, Frequency, Monetary) analysis, PCA for dimensionality reduction, clustering evaluation, and machine learning classifiers like ANN, KSVM, Decision Tree, AdaBoost to validate cluster assignments. The test results were highly accurate in segmentation and achieved over 95% accurate classifications across four machine learning classifiers, reduced dimensionality by 90%, improved optimization and convergence speed, and successful customer groups like high spenders, price-sensitive, balanced customers, and cautious customers.

This research paper [34] compares hybrid firefly PSO with six hybrid firefly DE. They applied these models on the problem of ride sharing, with the focus on allocating rides in such a way to optimize for the cost. and they authors have compared six FA-DE variations (FDE1 to FDE6) and an FA-PSO (FPSO) model. The FPSO model had the best optimization performance also it converged after fewer generations. With 1 driver and 4 passenger dataset the FPSO model optimized on average in 1.9 generations while FA-DE models converged on average in 3.1 generations. And for the complex datasets like 5 drivers and 11 passengers FA-PSO obtained 70.91 fitness in 692.9 generations, and FA-DE models achieved 65.47–69.45 fitness in 445.8–1022.6 generations, for datasets with 5 drivers and 12 passengers, FA-PSO reached a 41.715 fitness value in 187.3 generations, while FA-DE models achieved 40.76–41.24 fitness in up to 1999 generations and for datasets with 6 drivers and 12 passengers, FA-PSO obtained a 51.11 fitness value in 171.5 generations, outperforming FA-DE models, which achieved 48.50–50.43 fitness in up to 2702 generations. A new allocation method was also proposed, this method tries to enhance the number of acceptable shared rides. It also allows adjustments based on the minimal expectations of drivers and passengers. Two models DGPGP1 and DGPGP2 were proposed, and the results showed that one of these two methods consistently outperformed existing allocation approaches such as FF, LP, and GP.

This research paper [35] proposed an Adaptive Coordinate System for Constrained Differential Evolution (ACS-CDE). This method is focused to solve constrained optimization problems (COPs). Traditional Differential Evolution (DE) methods struggle with balancing objective optimization and constraint satisfaction. But this method tried to tackle this problem ACS-CDE by introducing two specialized Eigen coordinate systems, one for guiding the search towards feasible solutions and another for optimizing the objective function. These coordinate systems dynamically adjust based on feedback from the evolutionary process, improving exploration and convergence.

This method was then tested on few of the benchmark datasets, i.e. (IEEE CEC2010 and CEC2017) which contain challenging optimization problems. This method out performed the existing DE-based methods such as C2oDE, FROFI, CORCO, and DeCODE. In 30-dimensional CEC2010 test functions, their method ranked first in performance. While in 50D and 100D CEC2017 benchmarks, this method consistently achieved the best rankings and lowest error values. Statistical tests, including Wilcoxon and Friedman tests, confirmed its superiority. With 50D test functions from CEC2017, proposed method surpassed competitors in 19 out of 22 test cases, on the other hand methods like ITLBO, LSHADE44, and DeCODE underperformed. Similarly, on

100D test functions, suggest method led in 20 out of 21 cases, demonstrating its scalability for high-dimensional problems. To improve solution diversity and prevent convergence at local optima, it uses an adaptive selection process based on Upper Confidence Bound (UCB) to choose the best coordinate system dynamically.

This research paper [36] presents a self-adaptive differential evolutionary random vector functional link (SaDE-RVFL) classifier for Alzheimer's disease (AD) diagnosis, and for the testing purposes they used fused MRI and SWI images. To enhance the classification accuracy the authors integrated structural atrophy information from T1-weighted (T1-w) MRI with the iron accumulation data from the susceptibility-weighted imaging (SWI). To test the accuracy of the model they used the OASIS dataset, which contains MRI and SWI scans from 373 AD and 200 cognitively normal (CN) participants. The results from this model were compared against traditional machine learning classifiers like KNN, Naïve Bayes, Decision Trees, Random Forest, and SVM, as well as few of the advanced methods like KRR-RVFL and standard RVFL networks. Yet this model achieved the highest classification accuracy of 95.17%. When the model was tested on T1-weighted MRI only, it achieved 93.43% accuracy, and when tested on SWI only it had 88.73% accuracy. These results were improved with fused MRI-SWI images resulting in an accuracy of 95.17%, this result proved the effectiveness of combining both modalities, they also tested different activation functions, hidden neurons, and population sizes in the SaDE-RVFL model. Results showed that 80 hidden neurons provided optimal performance, and increasing the population size beyond 50 had no significant impact on accuracy. Another advantage of SaDE-RVFL is its adaptive differential evolutionary optimization, which fine-tunes the weights and biases of the RVFL network, preventing overfitting and slow convergence. The fused imaging approach also enables better visualization of AD progression, highlighting both brain atrophies and iron deposition areas in affected regions.

This research paper [37] presented Fractional-Order Difference-driven Differential Evolution (FODE) which is an algorithm for the optimization of wind farm layout. The reason for this novel method is that the traditional Genetic Algorithms (GAs) and Particle Swarm Optimization (PSO) fall behind with global exploration. This method integrates fractional-order difference mechanisms to balance global search and local exploitation, which significantly improved wind power generation efficiency. To test the efficiency of this algorithm it was then tested under 10 complex wind farm conditions, like layouts with 10, 20, 30, 50, and 80 turbines and different wind directions, its results were then compared against GA, PSO, and DE variants (AGA, SUGGA, LSE, AGPSO, CGPSO, CLSHADE, and LSHADE). In small-scale layouts (10 turbines), it achieved 100% wind energy conversion efficiency, other method also showed similar, and for medium-sized layouts (30 turbines), it obtained 95.92% efficiency, while GA achieved 94.22% efficiency and PSO achieved 94.96% efficiency, while in the large-scale scenarios like 80 turbines, it achieved 88.45% efficiency, but the obtained PSO only achieved 87.13% efficiency and DE variants got to 86.81% only. On the other hand, not only, it has higher efficiency but also it achieved this efficiency with less iterations, it required 30–40% fewer iterations to reach optimal solutions as compare to the standard DE

The authors of this paper [38] introduced a new training model which is optimized for physical education, this model is based on multi objection differential evolution technique, the existing model for physical training do not have good results due to many factors like physical health and effort by the student or weather conditions. When tested on various bench mark functions and compared with other models, their model achieved an optimal value of 0 as compared to PSO achieving the value of 0.032 and DE achieved the value of 0.015 on the sphere function i.e. (f1), and for the Rosenbrock function (f2) this model reached 1.02, while DE only reached value of 1.31 while PSO remained only 1.58. similarly, when tested on Rastrigin function (f3) it again got the best value of 1.42, whereas PSO and DE had 2.85 and 3.17, finally for the Ackley function (f5): this model's result was −19.98, while PSO and DE achieved −18.67 and −17.34, respectively.

When the model was applied to predict the grades, for 90+ marks the model correctly predicted 5 out of 8 students, for 80–90 marks range the model identified 7 out of 12 students correctly, for student with 70–80 marks The model performed well, by predicting 7 out of 8 students accurately, for the students obtaining 60–70 marks the model overestimated performance, predicting 11 students instead of 6, leading to a lower accuracy of 54.5% and finally for the students achieving below 60 marks (Fail): The model predicted failure for 10 students, while only 6 actually failed, making its accuracy 40.0%. So, the overall results are mixed, the model seem optimize good when tested on bench mark functions, but its performance is underwhelming when used to predict the actual grades of the students.

This research paper [39] has introduced yet another DE algorithm which is as Surrogate-Assisted Differential Evolution with Multiple Sampling Mechanisms (SADE-MSM), this algorithm is aimed to tackle the High-Dimensional Expensive Optimization Problems (HEOPs). Traditional Surrogate-Assisted Evolutionary Algorithms (SAEAs) have shortcoming when solving problems with 30 to 500 dimensions, they are not good at balancing exploration and exploitation. To solve that problem this suggested algorithm integrates three sampling mechanisms, firstly it uses Centroid Sampling (CS) to Enhances early exploration, then applies Global Pre-screening Sampling (GPS) to Balances global search and local refinement, and finally uses Local Search Sampling with Adaptive Optimal Region (LSS-AOR) to Improves exploitation and fine-tuning. When test on Ellipsoid with 100D, this algorithm achieved a result of 4.05×10^{-107}, outperforming SADE-FI which obtained the value of only (2.48×10^{-27}) and SHPSO that got to(1.61×10^{-64}), similarly for the Ackley with 50D, this algorithm quickly reached 0.0000, while SAHO and TSDDEO had errors of 5.05×10^{-5} and 2.63×10^{-1}, respectively, while on Rastrigin function with 100D, it obtained 0.0000, outperforming SACSO (4.81×10^{-11}) and ESA (8.20×10^{2}), On SRR (50D), SADE-MSM was slightly worse than SACSO, showing room for improvement in highly shifted problems, and finally for RHC with 30D, this algorithm maintained 9.10×10^{2}, the global optimum, proving its stability.

This research paper [40] has proposed a new multi-model fusion (MMF) approach for news recommendation system. It also uses text similarity (TS) calculations to enhance news sentiment classification which increases the recommendation accuracy. Other existing news recommendation systems tend to have poor classification accuracy and lack of personalization, that leads to inefficient recommendations. this study integrates multiple sentiment classification models (KNN, SVM, Naïve Bayes) alongside an enhanced differential evolution algorithm for fusion. This method was then tested against traditional models such as KNN, SVM, and Naïve Bayes (NB). Results for the Classification Accuracy (CAR) of this achieved 93% while the KNN model reached 87%, SVM attained 84% and the NB had the accuracy of 89%. The precision of their model reached 95%, outperforming SVM by 10%, while KNN and NB achieved 94% and 89%, respectively. In terms of recall, this model achieved 92%, which was 14% higher than KNN. The recall score for KNN was 78%, for SVM it was 83%, and the recall for the NB was 91%. The F1 Score for this model was 95%, surpassing SVM by 10%, while KNN and NB attained 94% and 89%, respectively. The system was then tested on 100 volunteers, they included 50 professionals and 50 general users, to measure user satisfaction. The news feed generated by their model had satisfaction scores of 97.1% (average) for general users and 95.2% (average) for professionals, which indicates that their system effectively met user expectations across different groups. Another crucial aspect is the speed of the recommendation system. This system demonstrated fast response times, with the news retrieval completing in under 500ms and personalized recommendations were generated in just under 300ms.

This research work [41] introduces a new method named as Quantum-Inspired Differential Evolution (QIDE), which is as name suggests inspired by quantum computing. This method can be used for automatic clustering of unlabelled datasets. Currently available clustering methods require prior knowledge of the number of clusters, which makes them inefficient for real-world datasets. But this method leverages quantum computing principles, like quantum gates (Pauli-X, CNOT, CCNOT) and quantum-inspired mutation, which results in improvement of search efficiency and convergence speed while preventing premature convergence. For testing this method several datasets were chosen, each with different number of cluster and size of feature set, some of which are Glass Identification Database, it consists of 214 instances, categorized into 6 clusters with 9 features. Another is Iris Plants Database, which contains 150 instances, that are grouped into 3 clusters with 4 features. Similarly, the Wine Recognition Data, it has 178 instances, separated into 3 clusters, with feature set size of 13. The mean fitness values across different datasets were calculated to gauge its performance, for the Glass Dataset: this suggested model achieved a fitness value of 0.2685, outperforming FQEA (0.2816) and significantly surpassing CDE (0.4990), and for the Iris Dataset their model achieved fitness value of 0.4741, which was slightly better than FQEA (0.4809) and much lower than CDE (0.7019). This algorithm also significantly reduced computational time, which proves its efficiency in handling large datasets, when compare the other models, with the Glass Dataset, it completed clustering in 9.49 seconds, as compared to CDE, which took 22.25 seconds, and for the Iris Dataset, it processed the dataset in just 2.82 seconds, whereas CDE required 15.52 seconds, showing a notable speed improvement, and its effectiveness in high-dimensional datasets.

The research study [42] have introduced a novel technique, which they have named as Adaptive Parameter Strategy Differential Evolution (APSDE), it is designed to overcome parameter dependence and prevents local optima trapping, like in traditional Differential Evolution (DE).

This technique improves search efficiency and solution diversity by using three steps, Parameter Update Mechanism (PUM), which Reduces sensitivity to parameters, secondly Adaptive Proportion Adjustment Mechanism (APAM), which Balances exploration and exploitation, and lastly Random Restart Mechanism (RRM) to Prevents premature convergence. This model was then tested on 30 CEC2018 benchmark functions (unimodal, multimodal, hybrid, and composition functions) and compared against nine state-of-the-art metaheuristic algorithms, including LSHADE, HHO, MTDE, WDE, and AVOA. For the Unimodal Functions, this proposed model ranked first in 10D, 30D, and 100D problems, with a mean ranking of 1.25, proving strong exploitation ability. similarly for the Multimodal Functions, this model again ranked first in 50D and 100D problems, with a mean ranking of 1.88, showing superior global search performance, lastly for Hybrid & Composition Functions, the suggested model had an overall mean ranking of 1.90, outperforming competing methods. For real life testing this model was applied to Beyond Visual Range (BVR) air combat midcourse guidance simulation to enhance missile targeting and evasion manoeuvres. The MGMD system based on their optimized UCAV manoeuvres by balancing, Missile positioning constraints, Radar scanning constraints and Evasion strategies. The model's decision-making time was under 1s, meeting real-time air combat requirements. While in sideway midcourse guidance simulations, this model achieved a 90% success rate, outperforming HHO, AVOA, and MTDE. And in adaptive midcourse guidance confrontations, it had a winning rate of 54%, defeating MTDE (34%-win rate) in 50 simulated engagements.

This research paper [43] presents a Constrained Composite Differential Evolution (C2oDE) algorithm for solving Optimal Power Flow (OPF) problems. OPF is a highly complex and non-linear optimization challenge in power systems, where the goal is to optimize power generation, minimize costs, and reduce emissions while considering multiple physical and electrical constraints. Traditional evolutionary algorithms struggle with these types of constraint handling, which often requiring static penalty functions that make it difficult to balance feasible and infeasible solutions. To address these issues, this algorithm integrates two of the advanced constraint-handling techniques, Feasibility Rule (FR), to ensure that the solutions remain within feasible constraints, and an ε-Constraint Method (ECM) to dynamically adjust the constraint thresholds to enhance the search efficiency this model. The researcher then tested their model on IEEE 30, 57, and 118-bus standard power networks with 16 single and multi-objective optimization scenarios, to prove its effectiveness in handling large-scale OPF problems.

For the IEEE 30-Bus System with the constraints on Fuel Cost, Power Loss, Emissions, and Voltage Stability, this model achieved a minimum cost of $800.41/hour, outperforming AGSO model, which costed $801.75/hour although the BSA achieved $799.07 but with voltage violations. And for Power Loss Reduction: The best-performing variant of their model reduced power loss to 3.08 MW, significantly improving efficiency, as for the Emission Minimization, the proposed model achieved a minimum emission rate of 0.2048 t/h, better than SF-DE (0.36652 t/h), and lastly for Voltage Stability (L-index), the lowest recorded L-index was 0.1363, ensuring improved system stability.

This research work [44] introduces a Hybrid Differential Evolution (HDE) algorithm to solve the Resource-Constrained Project Scheduling Problem (RCPSP) with flexible project structures and resource consumption/production constraints (RCPSP-PS/CPR). Traditional RCPSP methods assume fixed activity sets and renewable-only resources, but real-world scheduling requires handling variable activity selection and non-renewable (cumulative) resources, and to address these challenges, they used Group Graph Selection (GGS) for feasible activity selection, and Differential Evolution for the optimization of the scheduling order then applied the Forward-Backward Improvement (FBI) to further refine those scheduling solutions.

This algorithm was then evaluated on three datasets:

- Servranckx & Vanhoucke (2019) dataset (3,207 instances without cumulative resources)
- Optimal dataset (900 instances with known optimal solutions)
- Non-Optimal dataset (2,000 instances for large-scale problems)

On the Servranckx & Vanhoucke Dataset, their model outperformed TS (Tabu Search) in 1,335 instances (41.6%) and tied in 1,858 cases (58%), for the Optimal Dataset (900 Instances, 10 Runs Each), this model achieved the optimal solution in 98.4% of runs (4,428 out of 4,500), surpassing ACO's 88.8% success rate, and it found at least one optimal solution in 446 out of 450 instances, while ACO succeeded in 432 cases. Finally for the Non-Optimal Dataset (2,000 Large-Scale Instances, 10 Runs Each), their model again outperformed ACO in 758 cases, while ACO was better in just 6 c ases, the average BOG (Bound Optimality Gap) for their model was 0.031% as compared to 0.275% for ACO.

This research work [45] introduces a modified version Differential Evolution algorithm designed to solve non-linear and non-convex engineering optimization problems, they have applied mutation strategies inspired by Particle Swarm Optimization (PSO) to enhance global search capability, they also modified crossover by using Dynamic crossover rate adaptation to improve convergence speed, and a new selection strategy to escape local minima and enhance information sharing. When the authors compared their algorithm against eight meta-heuristic optimization methods (PSO, DE, MABC, ETLBO, HBLPSO, NDE, EDDE, and CMPSOWV), for optimization accuracy, their model achieved the best results on 19 out of 22 functions and tied for the best in the remaining cases, while the stability and convergence Speed of their model was lowest across test cases, and they claimed that wilcoxon and Friedman tests confirmed that their model outperformed competing algorithms with p-values < 0.05.

Table.21 Summary of Relevant Paper Literature Review

S. No & Year	Title	Contribution	Weakness
[46] [2023]	A Modified Differential Evolution Algorithm Based on Improving A New Mutation Strategy and Self-Adaptation Crossover	Presented mutation strategy directed towards global best Individual	▪ Lack of diversity and exploration capability ▪ Focuses on exploitation that results local optima issue
[47] [2023]	Self-Adaptive Differential Evolution with Gauss Distribution for Optimal Mechanism Design	Introduced a self-adaptive amplification factor in the mutation operation	The amplification factor is not helpful to improve the diversity of the population.
[48] [2021]	An improved differential evolution algorithm and its application in optimization problem	Focuses on the best performing neighbor by considering 50% closest neighbors	Its more focus is on exploitation and lacks balance between exploration and exploitation.

3.3 Research Gap

The paper utilizes a mutation approach that relies on selecting only the best solution in the population to guide mutation. While this strategy may lead to faster convergence, it lacks diversity and increases the risk of getting trapped in local optima. This limitation becomes particularly problematic in complex or multimodal optimization landscapes, where reliance on a single elite individual can prevent the exploration of potentially better regions of the search space.

$$V_{i,G} = \boxed{X_{best,G}} + F(\boxed{X_{best,G}} - X_{i,G}) + F(\boxed{X_{best,G}} - X_{r,G}) \tag{3.1}$$

The reliance on a mutation strategy that uses only the best individual in the population introduces high selection pressure, which significantly reduces genetic diversity. This biased exploitation may lead the algorithm to converge prematurely to a local optimum, especially in complex or multimodal landscapes. Since the mutation lacks stochastic variability, it becomes less capable of exploring unexplored regions of the solution space. Moreover, the absence of mechanisms like crowding, random parent selection, or diversity-aware strategies limits the algorithm's ability to maintain population diversity over generations. As a result, the algorithm may miss better global solutions due to insufficient exploration.

Chapter #4 Proposed Enhanced Diversity based Differential Evolution Algorithm

4.1 Summary of chapter

This chapter presents the proposed enhanced diversity based mutation strategy in the DE algorithm. The flowchart and the pseudocode of the proposed algorithm are also discussed in the chapter.

4.2 Improved Differential Evolution (Base Paper)

A Self-Adaptive Differential Evolution with Gauss Distribution for Optimal Mechanism Design [46] is considered as base study in this research work. A novel approach to optimize mechanism design problems using a self-adaptive version of the Differential Evolution (DE) algorithm is used. The study addresses the challenge of effectively designing mechanisms for complex systems by proposing a method that incorporates Gauss distribution to adaptively adjust mutation rates during the evolutionary process. The presented research work introduced a self-adaptive DE algorithm enhanced with Gauss distribution. Unlike traditional DE algorithms with fixed mutation rates, the self-adaptive approach dynamically adjusts mutation rates based on the performance of individuals in the population. This adaptability enables the algorithm to strike a balance between exploration and exploitation, thereby improving convergence speed and solution quality.

To validate the effectiveness of the proposed method, the authors conduct experiments on a set of benchmark mechanism design problems. They compare the performance of the self-adaptive DE with Gauss distribution against traditional DE variants and other state-of-the-art optimization algorithms. The results demonstrate superior performance in terms of convergence speed and solution quality, showcasing the potential of the proposed approach for solving complex mechanism design problems efficiently. The pseudocode of improved differential evolution is presented as follows.

Algorithm 4.1: Pseudocode of Improved Differential Evolution Algorithm [46]

```
1. $Generate\ initial\ population\ with\ N\ random\ solutions\ and\ store\ the\ best\ in\ solution_{best}$
2. $Set\ G = 1$
3. $While\ G \le MaxItr\ do:$
4.     $set\ i = 1$
5.     $While\ i \le N\ do:$
6.         $V_{i,G} = X_{best,G} + F(X_{best,G} - X_{i,G}) + F(X_{best,G} - X_{r,G})$
7.         $If\ \ \mathrm{mod}(G,2) = 0\ do:$
8.             $Generate\ CR \in [0, 0.4]$
9.         $Else$
10.        $Generate\ CR \in [0.8, 1]$
11.        $Generate\ C_{i,G}\ using\ C_{i,G} = \begin{cases} Solution_{i,G,j}\ \ if\ \ rand() > CR \\ Mutated\ Solution_{i,G,j}\ otherwise \end{cases}$
12.        $Solution_{best} = \arg\min\left\{ f(Solution_{best}),\ f(V_{i,G}),\ f(C_{i,G}),\ f(X_{i,G} \right\}$
13.        $i = i + 1$
14.        $G = G + 1$
15.        $\mathrm{Re}turn\ Solution_{best}$
```

The base algorithm is starting from the initialization of the population then fitness of the initial population is calculated. After calculation of fitness, the best individual is selected among the population. Then the iterations of the algorithm start up to a specific number of times or until the stopping condition remains true. To generate mutant vector current vector, best vector and a random selected vector are used in the equation of the mutation. The amplification factor of the mutation operation was taken from the range [0, 0.5]. After the mutation operation the crossover rate parameters is generated in the range of [0, 0.4] for the even iterations and [0.8, 1] for the odd iterations. The chosen CR is then used in the crossover operation to generate a trial vector and after generation of the trial vector, selection of the individual is selected among the target vector and trial vector.

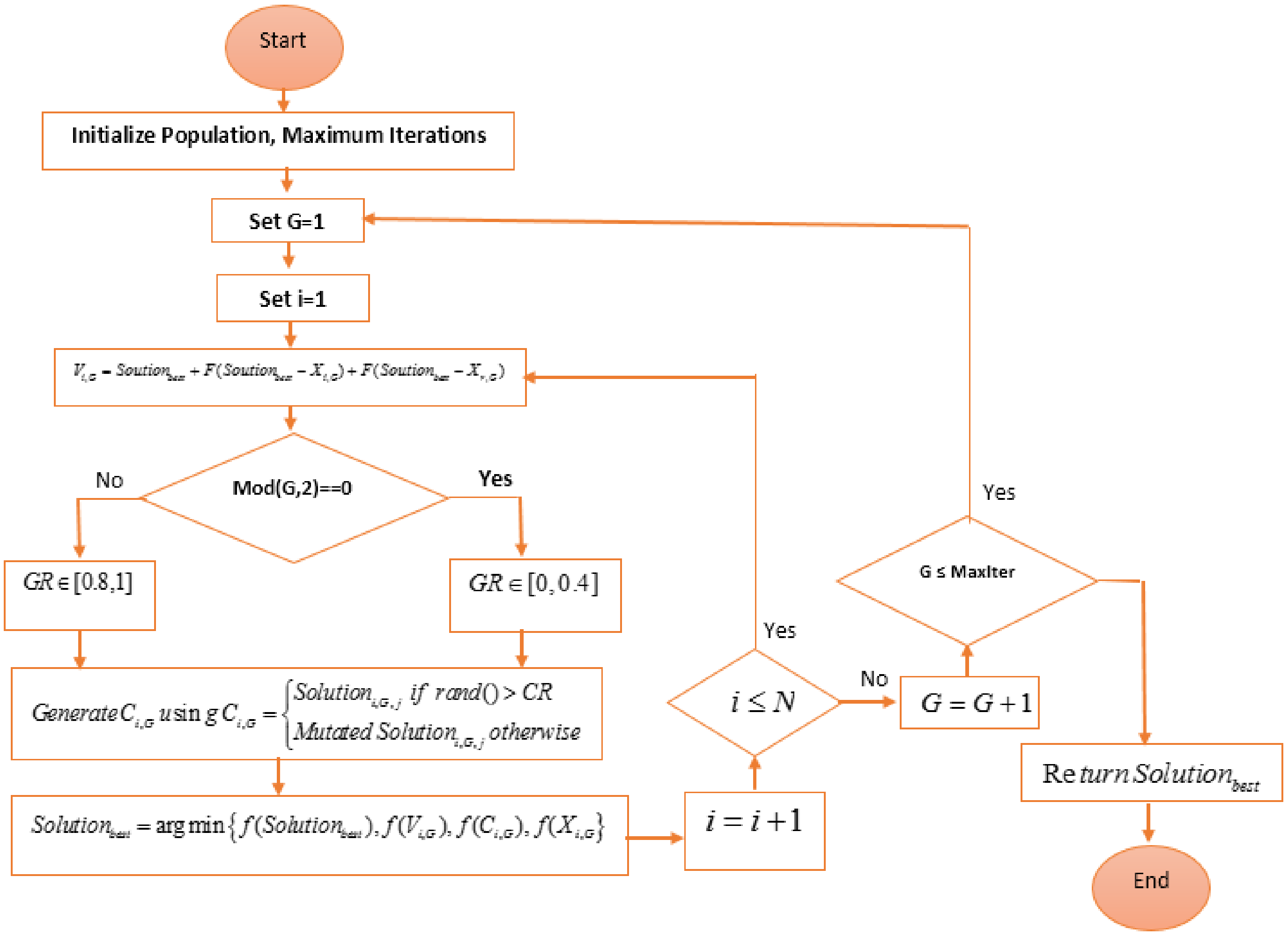

Start
Initialize Population, Maximum Iterations
Set G=1
Set i=1
$V_{i,G} = Soution_{best} + F(Soution_{best} - X_{i,G}) + F(Soution_{best} - X_{r,G})$
Mod(G,2)==0
No
Yes
$GR \in [0.8,1]$
$GR \in [0,0.4]$
$Generate\, C_{i,G}\, using\, C_{i,G} = \begin{cases} Solution_{i,G,j} & if\ rand() > CR \\ Mutated\ Solution_{i,G,j} & otherwise \end{cases}$
$Solution_{best} = \arg\min\{f(Solution_{best}), f(V_{i,G}), f(C_{i,G}), f(X_{i,G})\}$
$i = i + 1$
$i \leq N$
Yes
No
$G = G + 1$
$G \leq MaxIter$
Yes
$Return\ Solution_{best}$
End

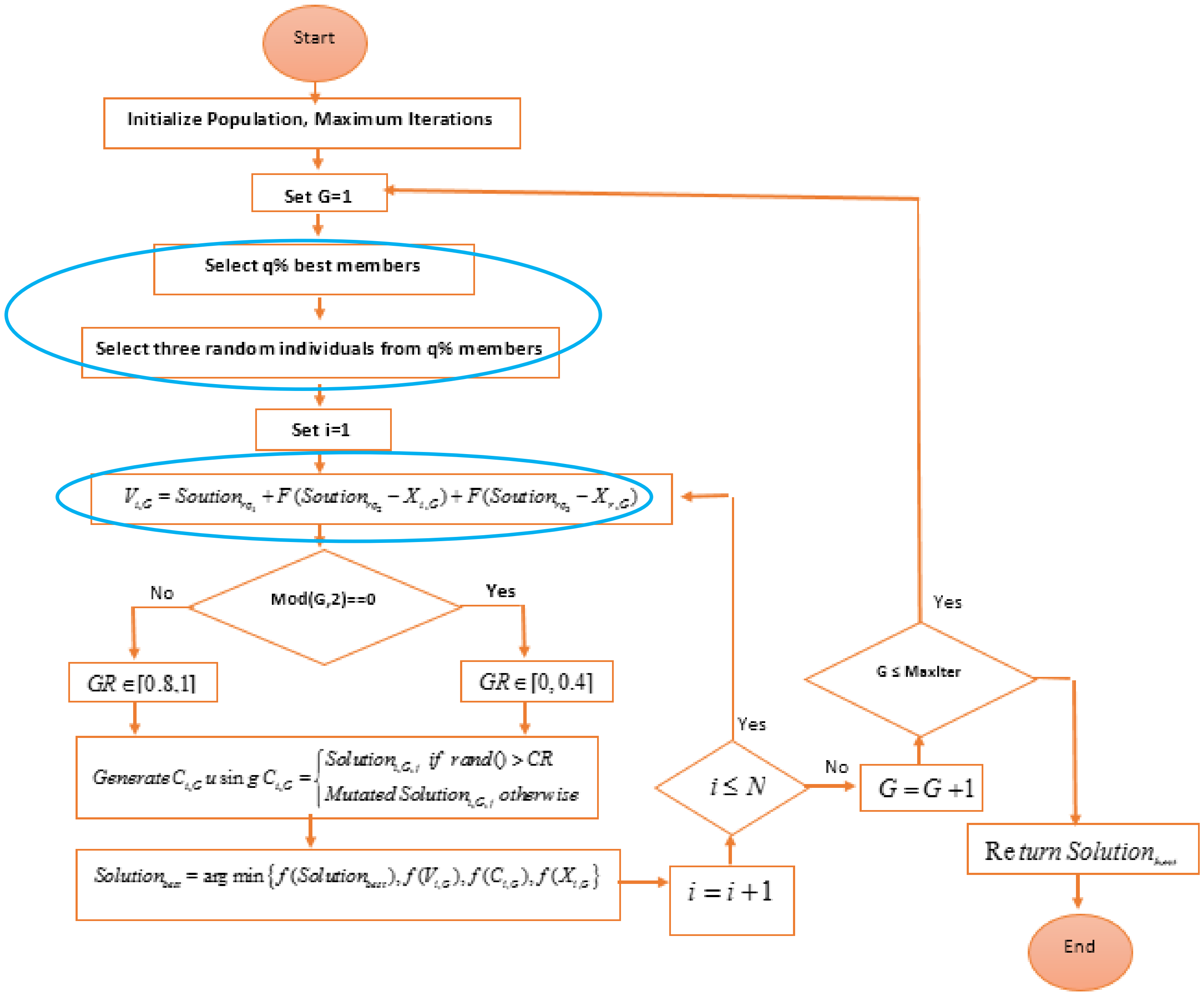
Figure 4.1. Flow Chart of Improved Differential Evolution Algorithm(Base algorithm)

4.3 Proposed Enhanced Diversity based Differential Evolution Algorithm

In Differential Evolution (DE), one approach to improving diversity and exploration involves selectively perturbing a certain percentage of individuals within the population. This strategy aims to improve the exploration in the search process, potentially helping the algorithm escape local optima and explore new regions of the search space. By selectively perturbing a percentage of individuals in the population, Differential Evolution can achieve improved diversity and exploration, leading to better convergence and solutions in optimization problems, especially in complex and multimodal search spaces. In the equation of mutation operation, instead of using one best vector, three best vectors, randomly selected from the top q% pool of members is uses. The three different best performing individuals are helpful to improve the diversity in the population and increase the exploration capability.

Algorithm 4.2: Proposed Enhanced diversity based Differential evolution algorithm

```
1. Generate initial population with N random solutions, r random individual and store the best in $solution_{best}$
2. Set $G = 1$
3. While $G \le MaxItr$ do:
4.     Select q% pool of best performing individuals
5.     Set $i = 1$
6.     While $i \le N$ do:
7.         Select three random individuals from the pool of q% members
8.         $V_{i,G} = Soution_{rq_1} + F(Soution_{rq_2} - X_{i,G}) + F(Soution_{rq_3} - X_{r,G})$
9.         If $\mathrm{mod}(G,2) = 0$ do:
10.                Generate $CR \in [0, 0.4]$
11.        Else
12.            Generate $CR \in [0.8, 1]$
13.        Generate $C_{i,G}$ using $C_{i,G} = \begin{cases} Solution_{i,G,j} & if\ rand() > CR \\ Mutated\ Solution_{i,G,j} & otherwise \end{cases}$
14.        $Solution_{best} = \arg\min\{ f(Solution_{best}), f(V_{i,G}), f(C_{i,G}), f(X_{i,G}) \}$
15.        $i = i + 1$
16.        $G = G + 1$
17.        Return $Solution_{best}$
```



Start
Initialize Population, Maximum Iterations
Set G=1
Select q% best members
Select three random individuals from q% members
Set i=1
$V_{i,G} = Soution_{rq_1} + F(Soution_{rq_2} - X_{i,G}) + F(Soution_{rq_3} - X_{r,G})$
Mod(G,2)==0
No
Yes
$GR \in [0.8, 1]$
$GR \in [0, 0.4]$
$Generate\ C_{i,G}\ using\ C_{i,G} = \begin{cases} Solution_{i,G,i} & if\ rand() > CR \\ Mutated\ Solution_{i,G,i} & otherwise \end{cases}$
$Solution_{best} = \arg\min\{f(Solution_{best}), f(V_{i,G}), f(C_{i,G}), f(X_{i,G})\}$
$i = i + 1$
$i \le N$
Yes
No
$G = G + 1$
G ≤ MaxIter
Yes
$Return\ Solution_{best}$
End

Figure 4.2 . Flow Chart of Proposed Enhanced Diversity based Differential Evolution Algorithm

The proposed algorithm starts in the same way like the base algorithm. In order to generate a mutant vector it is considering a q% (10%) pool of best individuals from the size of the population and then utilizing three random selected individuals to be used in the equation of mutation operation. The three different random best selected from top q% performing algorithm will be helpful to incorporate the diversity and enhance the convergence speed of the base algorithm. The crossover probability, mutation probability and selection operator were used in the same way as used in the base algorithm.

Chapter #5 Benchmark Functions

5.1 Summary of chapter

The detail of benchmark functions used in this research work is presented in this section. The sample illustration of each functions among the test suit of 11 benchmark functions is shown in the section along with the mathematical equations and search space ranges used in this research work.

5.2 Function Optimization

Function optimization is used to find best parameters used in solving maximization and minimization problems. Function optimization contains set of problems of varying nature that are helpful to solve uni-modal, multimodal, separable and non-separable problems that are helpful to solve various real world problem of the various domains like computer science, economics, engineering.

5.3 Test Suit of Function Optimization

In this research work we have used a test suit of 11 benchmark functions taken from the base paper of this study. The detail of these problems is given in this chapter.

5.3.1 Sphere Function (f1):

Equation: $f(x) = \sum_{i=0}^{n} x_i^2$

Search Space: $-5.12 \le x_i \le 5.12$

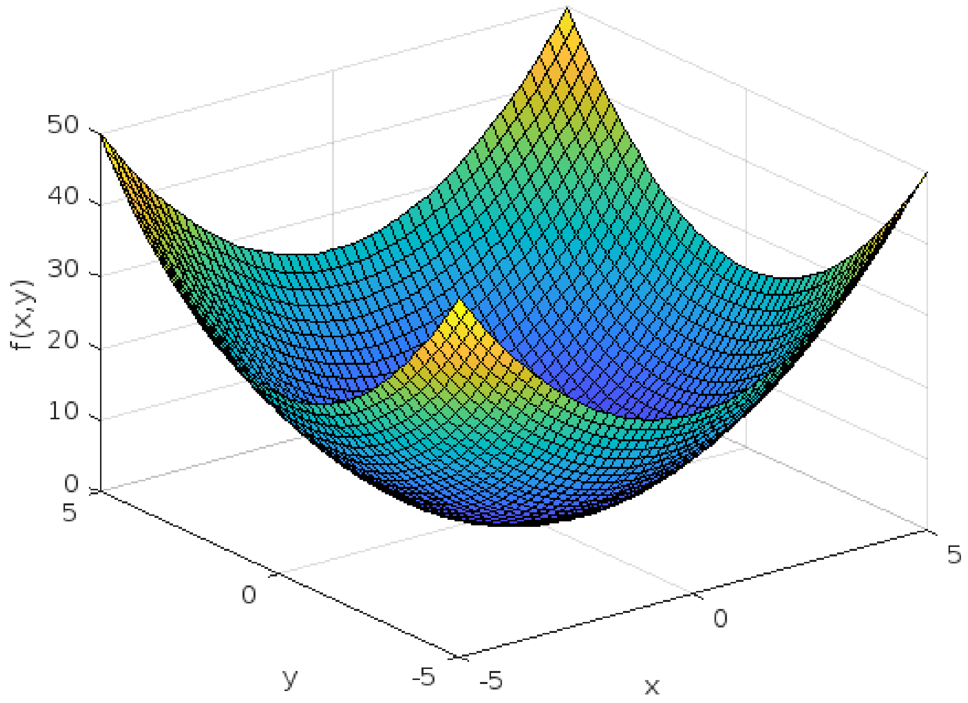


Figure 5.1: 3D illustration of Sphere Function(f1)

5.3.2. Rastrigin Function (f2)

Equation: $f(x) = 10n + \sum_{i=1}^{n} (x_i^2 - 10\cos(2\pi x_i))$

Search Space: -5.12 $\le x_i \le$ 5.12

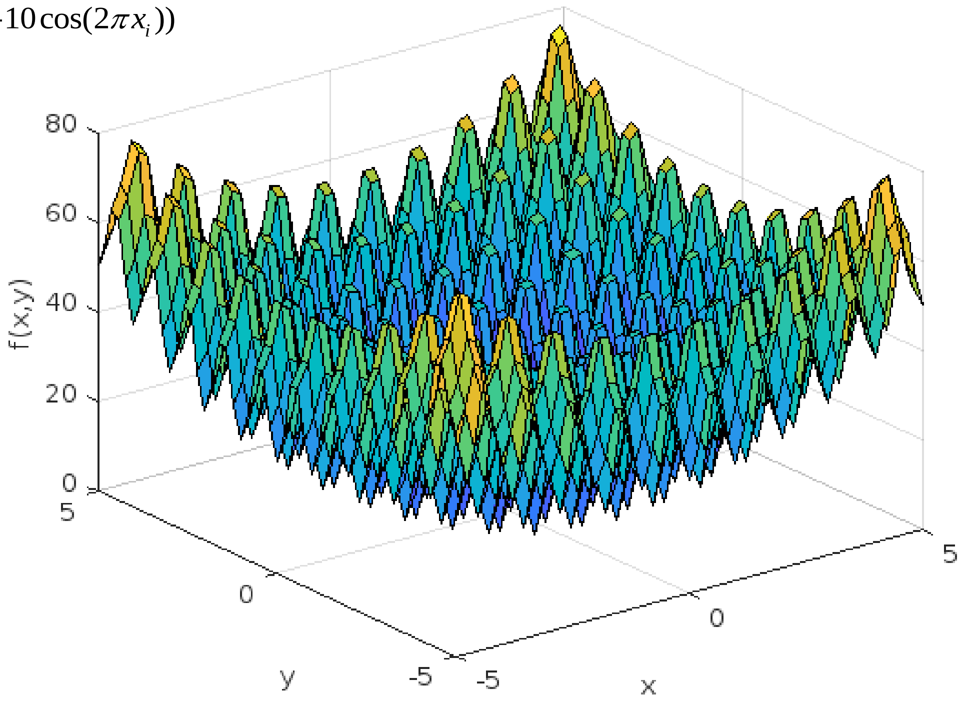


Figure 5.2: 3D illustration of Rastrigin Function(f2)

5.3.3. Ackley Function (f3)

Equation: $$f(x) = -20\exp\left(-0.2\sqrt{\frac{\sum_{i=1}^{n} x_i^2}{n}}\right) - \exp\left(\frac{\sum_{i=i}^{n} Cos(2\pi x_i}{n}\right) + 20 + e$$

Search Space: -32$\le x_i \le$32

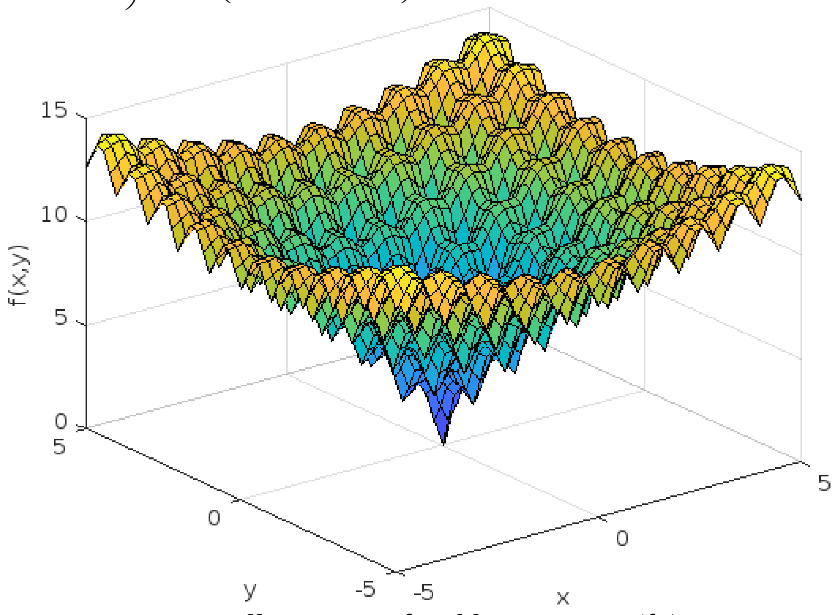


Figure 5.3: 3D illustration of Ackley Function(f3)

5.3.4. Rosenbrock Function (f4)

Equation: $$f(x) = \sum_{i=1}^{n-1}[100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2]$$

Search Space: $-30 \le x_i \le 30$

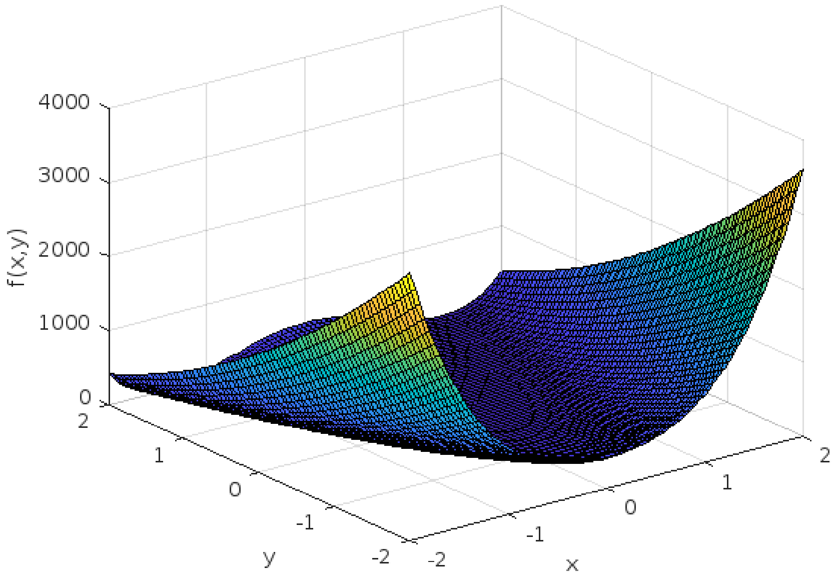


Figure 5.4: 3D illustration of Rosenbrock Function(f4)

5.3.5. Beale Function (f5)

$$f(x) = (1.5 - x1 + x_1x_2)^2\left(\left(2.25 - x_1 + x_1x_2^2\right)^2 + \left(2.625 - x1 + x_1x_2^3\right)^2\right)$$

Search Space: $-4.5 \le x_i \le 4.5$

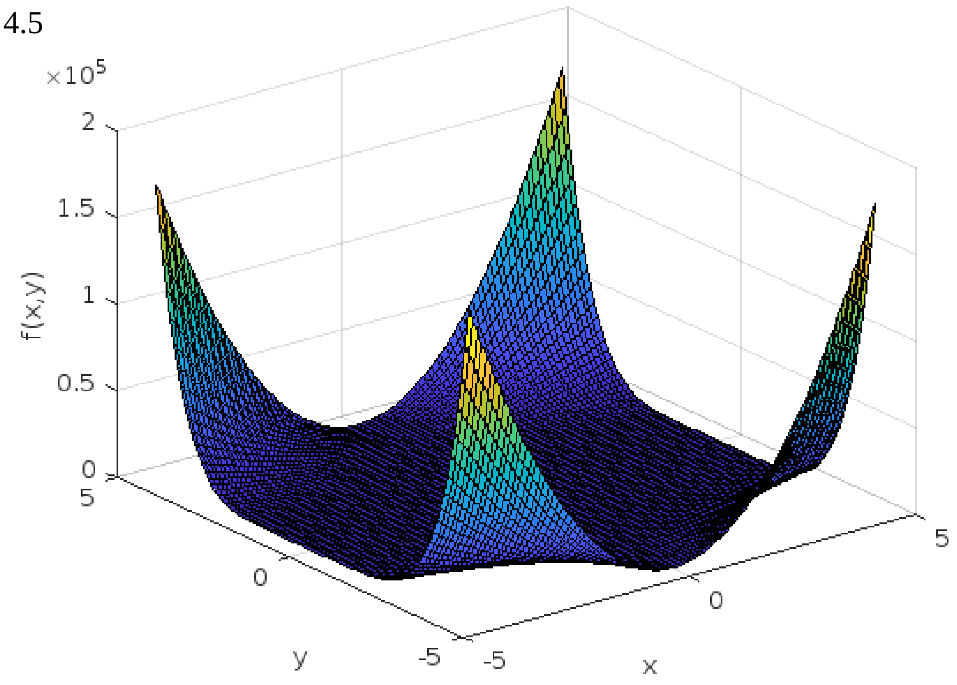


Figure 5.5: 3D illustration of Beale Function(f5)

5.3.6. Goldstein-Price Function(f6)

$$f(x) = \left(1 + (x_1 + x_2 + 1)^2 \left(19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2\right)\right)$$
$$* \left(30 + (2x_2 - 3x_2)^2 \left(18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2\right)\right)$$

Search Space: $-100 \le x_i \le 100$

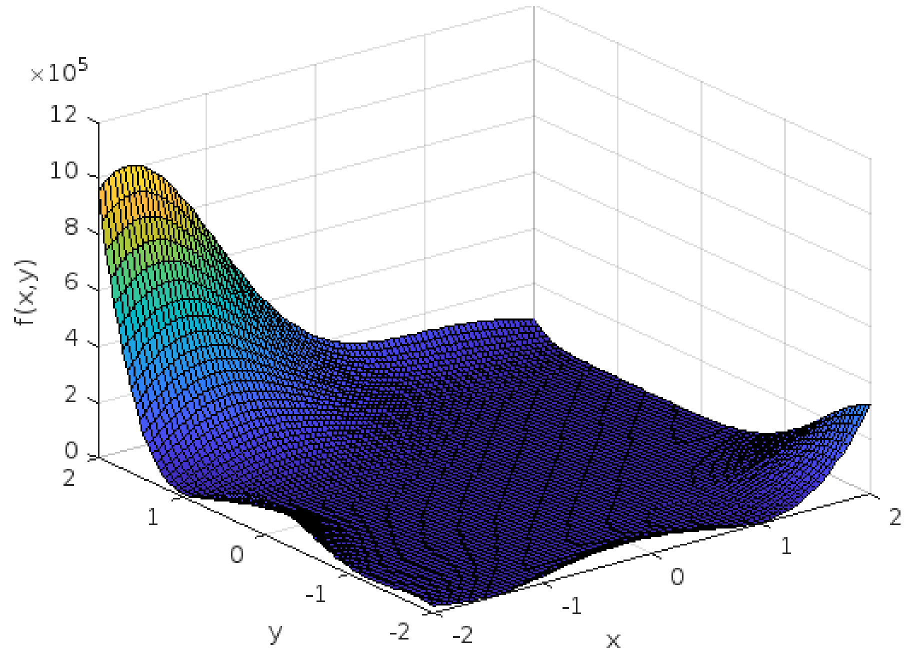


Figure 5.6: 3D illustration of Goldstein_Price Function(f6)

5.3.7. Bohachevsky Function (f7)

$$f(x) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7$$

Search Space: $-100 \le x_i \le 100$

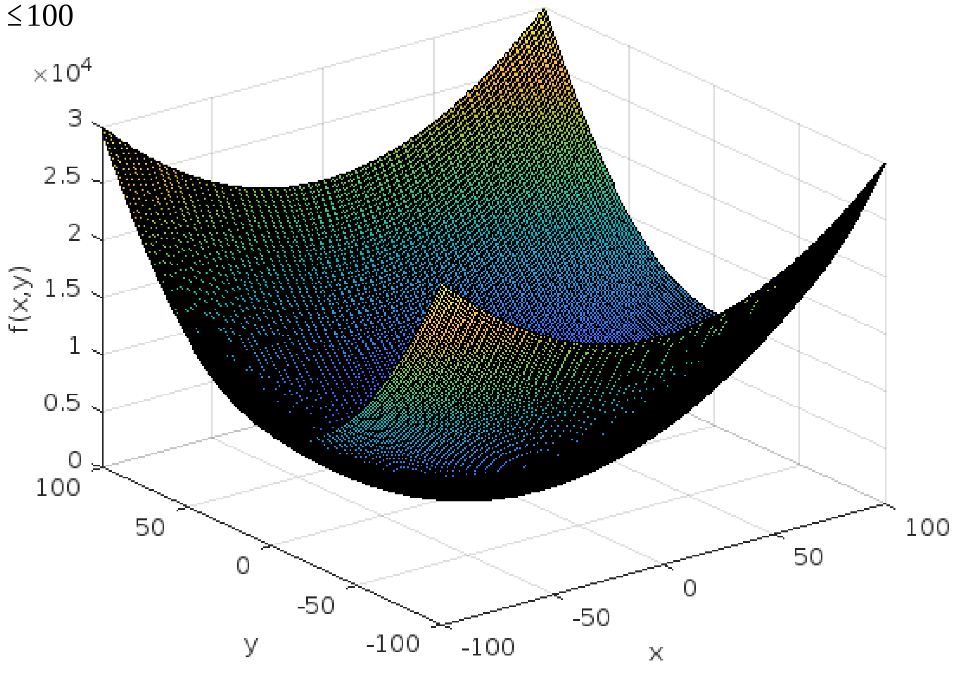


Figure 5.7: 3D illustration of Bohachevsky Function(f7)

5.3.8. Booth Function

$$f(x) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2$$

Search Space: $-10 \le x_i \le 10$

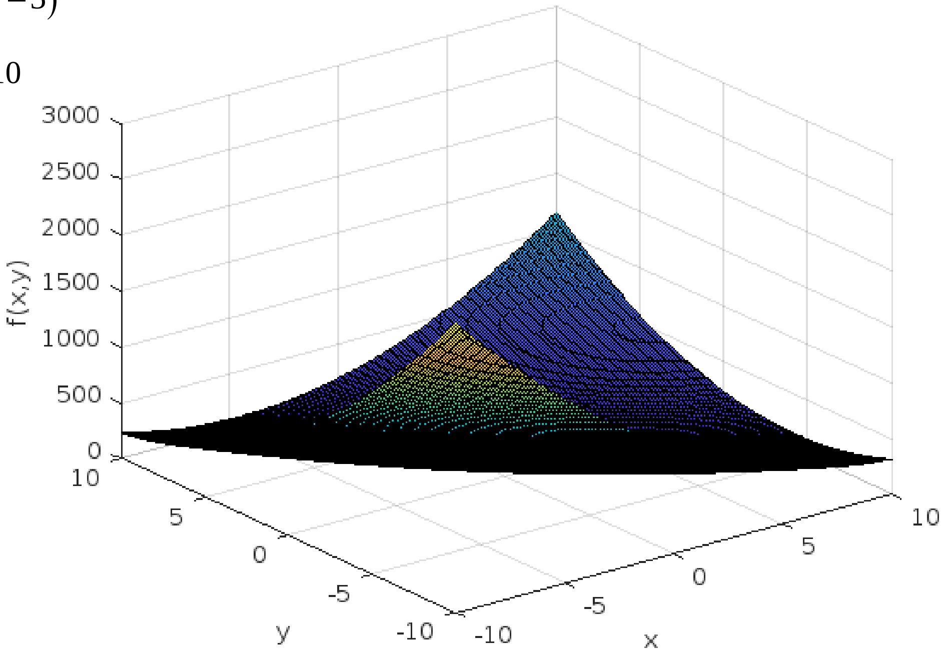


Figure 5.8: 3D illustration of Booth Function(f8)

5.3.9. Matyas Function

$$f(x) = 0.26\left(x_1^2 + x_2^2\right)^2 - 0.48x_1x_2$$

Search Space: $-10 \le x_i \le 10$

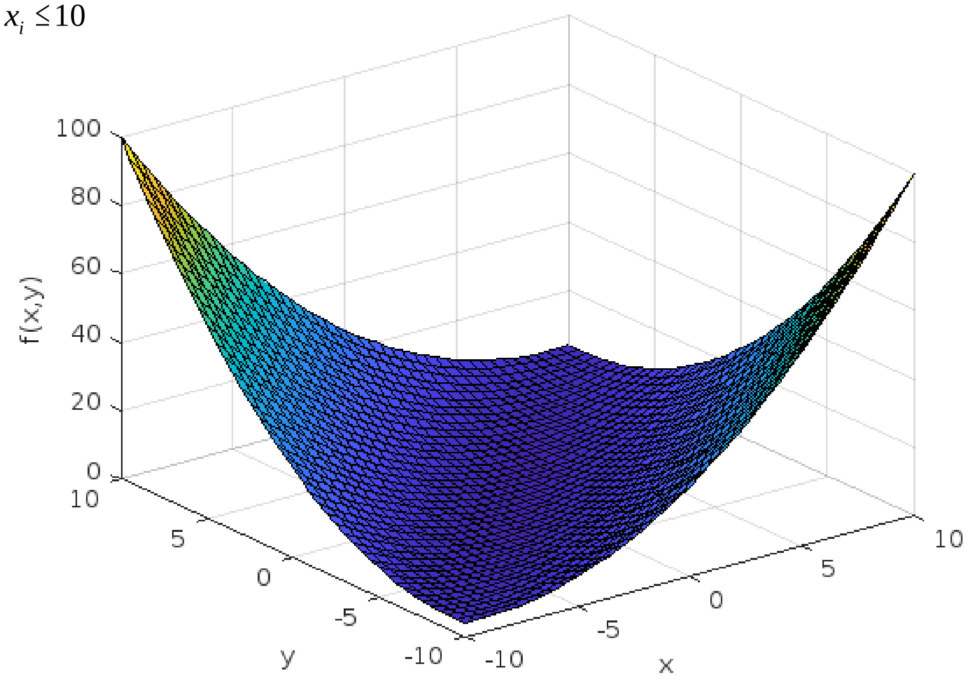


Figure 5.9: 3D illustration of Matyas Function(f9)

5.3.10. Zakharov Function

$$f(x) = \sum_{i=1}^{n} x_i^2 + \left(\sum_{i=i}^{n} 0.5 i x_i \right)^2 + \left(\sum_{i=i}^{n} 0.5 i x_i \right)^4$$

Search Space: $-5 \le x_i \le 10$

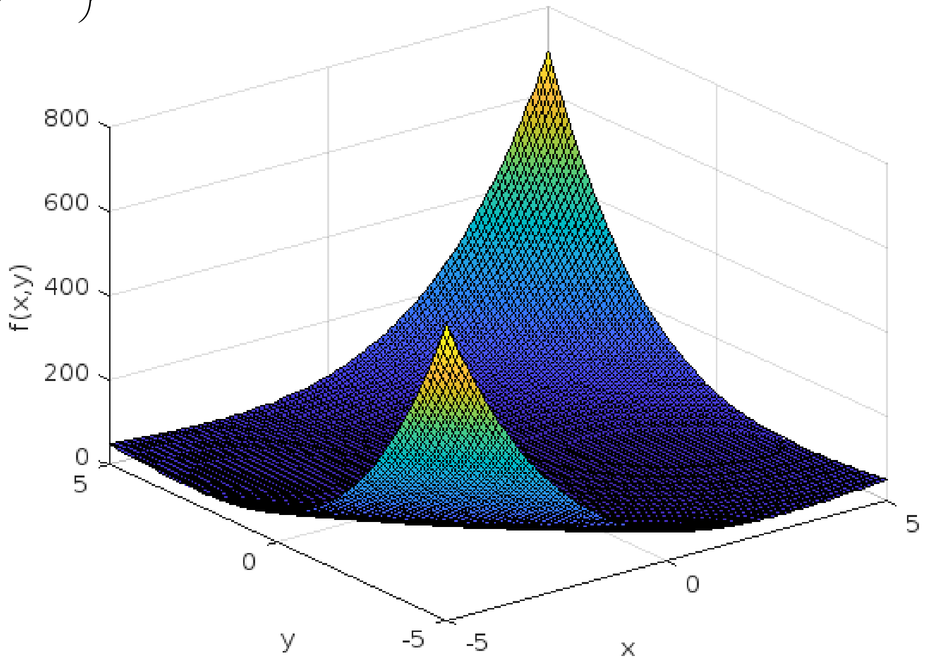


Figure 5.10: 3D illustration of Zakharov Function(f10)

5.3.11. Six-Hump Camel Function

$$f(x) = \left(4 - 2.1 x_1^2 + \frac{x_1^4}{3} \right) x_1^2 + x_1.x_2 + \left(-4 + 4 x_2^2 \right) x_2^2$$

Search Space: $-3 \le x_i \le 3$

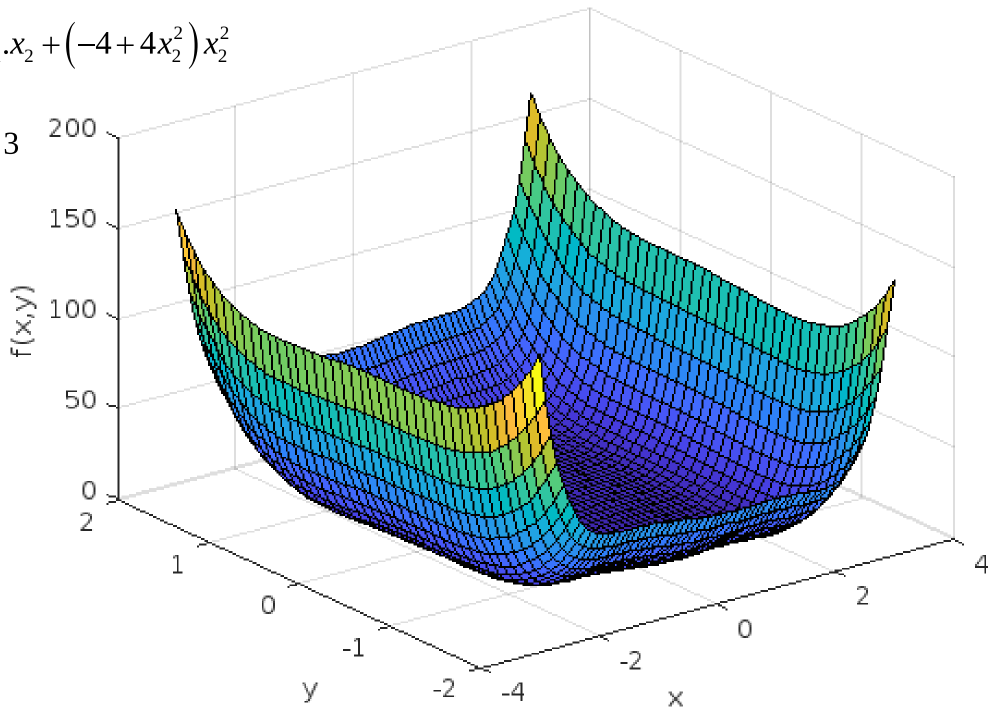


Figure 5.11: 3D illustration of Six_hump Function(f11)

Chapter # 6 Experimental results and discussion

6.1 Chapter Summary

In this chapter the experimental results of average fitness values of test suit of benchmark functions are presented in this chapter. The logarithmic convergence graphs of some cases of the test suit of benchmark functions is also discussed in this chapter.

6.2 Parameter Settings

We have used 11 different benchmark functions during the experimental results generation. The considered benchmark functions have different characteristics such as uni-modal and multimodal. To generate experimental results of base algorithm and the proposed algorithm, we have use same set of parameters to ensure unbiased performance comparison. The parameters used in this research work are given in the following table.

Name of parameter	**Value of parameter**
Crossover rate (CR)	[0, 0.4] for even iterations [0.8, 1.0] for odd iterations
Mutation rate (F)	[0.1, 0.5]
Dimensions	10, 20, 30, 50
Population Size	30, 60, 90, 150
Iterations	200, 500
Trials	30
Crossover Method	Binomial

6.3 Average Fitness Results

The performance of the base algorithm and the proposed enhanced algorithm was evaluated using a consistent set of 11 standard benchmark functions over 200 iterations in four separate cases, varying in dimensionality and population size. The benchmark functions are taken from the base paper that includes functions of varying nature to test the robustness and adaptability of both algorithms.

We have generated experimental results by using various settings of parameters to generate the results presented in this section by varying number of dimesons, population size and number of iterations.

Table 6.1: Average fitness results for the test suit of 11 benchmark functions using 200 iterations

Function	DIM=10D, NP=30		DIM=20D, NP=60		DIM=30D, NP=90		DIM=50D, NP=150	
	Base	**Proposed**	**Base**	**Proposed**	**Base**	**Proposed**	**Base**	**Proposed**
f_1	21.5372	**11.9851**	55.8432	**30.2971**	95.4248	**44.7553**	198.054	**120.246**
f_2	61.9206	**43.6777**	150.144	**109.798**	246.589	**203.345**	494.019	**391.613**
f_3	4.38829	**2.92439**	5.1369	**3.66969**	5.4643	**4.18651**	6.00539	**4.83312**
f_4	910.232	**904.295**	1932.25	**1915.95**	2957.71	**2931.27**	5021.99	**4967.66**
f_5	0.534585	**0.186867**	0.45358	**0.051398**	0.256263	**0.101609**	0.312692	**0.101609**
f_6	7569.87	**169.548**	1811.57	**11.7**	414.561	**38.0802**	342.711	**0**
f_7	298.245	**3.28335**	158.93	**0.031326**	80.3782	**0**	58.8717	**0**
f_8	5.68467	**0.419373**	2.52848	**8.22E-33**	2.19204	**0.0001**	0.686884	**0**
f_9	0.108156	**0.008679**	0.100239	**7.06E-34**	0.040089	**8.81E-79**	0.026456	**1.78E-76**
f_{10}	110.298	**36.226**	234.574	**100.14**	2455.22	**154.08**	920.398	**125.95**
f_{11}	0.001481	**0**	0.001519	**0**	0.000588	**0**	0.00038	**0**

Table 6.2: Average fitness results for the test suit of 11 benchmark functions using 500 iterations

Function	DIM=10D, NP=30		DIM=20D, NP=60		DIM=30D, NP=90		DIM=50D, NP=150	
	Base	**Proposed**	**Base**	**Proposed**	**Base**	**Proposed**	**Base**	**Proposed**
f_1	20.2069	**7.28908**	52.3899	**26.9067**	94.8492	**49.0526**	179	**113.317**
f_2	56.811	**37.513**	132.434	**108.532**	224.189	**180.599**	441.546	**382.301**
f_3	4.27169	**2.98053**	4.90753	**3.68551**	5.22045	**4.3172**	5.74865	**4.98152**
f_4	909.435	**904.604**	1928.81	**1916.1**	2956.43	**2932.66**	5014.77	**4971.31**
f_5	0.613206	**0.332132**	0.487519	**0.133932**	0.24655	**0.050805**	0.118172	**0**
f_6	18449.6	**482.956**	1755.99	**0.9**	370.353	**7.36667**	233.221	**0**
f_7	226.41	**0.20129**	204.55	**0.031326**	92.4103	**0.013764**	60.7872	**0**
f_8	4.44273	**0.000285**	1.9084	**2.79E-32**	1.0401	**0**	0.690798	**7.77E-05**
f_9	0.14968	**0.006159**	0.039416	**2.09E-26**	0.030971	**9.8E-10**	0.019036	**1.8E-119**
f_{10}	84.9402	**41.3747**	240.365	**121.925**	408.718	**163.572**	760.057	**74.3017**
f_{11}	0.011494	**0.00081**	0.000734	**0**	0.000208	**0**	8.4E-05	**0**

The performance of the base algorithm and the proposed enhanced algorithm was evaluated using a consistent set of 11 standard benchmark functions over 200 and 500 iterations in four separate cases, varying in dimensionality and population size. The benchmark functions are taken from the base paper that includes functions of varying nature to test the robustness and adaptability of both algorithms. It can be observed for low dimensional results of 10 dimensions and a population size of 30, both algorithms initially demonstrated similar performance. However, the proposed algorithm showed slightly faster convergence and achieved better final fitness values on most functions, indicating improved exploration and early exploitation capabilities for 200 as well as 500 iterations. After increasing a number of dimensions to 20 dimensions and a population size of 60, for 200 & 500 iterations the proposed algorithm performs well. The proposed DE algorithm consistently outperformed the base algorithm across nearly all functions. The number of dimensions then increased to 30 dimensions and a population size of 90 for 200 & 500 iterations, also confirms that the proposed algorithm has better performance as compared to base algorithm for the test suit of all functions. Furthermore the dimensions were increased to 50 and a population size of 150, the proposed DE algorithm demonstrated its scalability and robustness. It significantly outperformed the base algorithm on almost all functions, achieving better final results and faster convergence. The proposed modifications helped the algorithm adapt better to the high-dimensional search space and avoid premature convergence, which was observed in the base DE.

6.4 Convergence Graphs

This section presents convergence graphs for the proposed and base algorithms, illustrating their optimization performance over a test suite of benchmark functions. Each graph plots the number of iterations along the x-axis and the corresponding average fitness values along the y-axis. The experiments were conducted using a 10 & 50-dimensional search space, a population size of 30 and 150, and a total of 200 iterations. We have used benchmark functions of Sphere function, Rastrigin function, Ackley function, Rosenbrock function, Beale function, Goldstein_Price function, bohachevsky function, booth function, matyas function, zakharov function and six_hump function used is the base paper.

6.4.1 Convergence Graphs using 10D, 30NP and 200 iterations

In this section we have generated convergence graphs of 11 benchmark functions using 10 dimensions, 30 as size of population and 200 as number of training iterations.

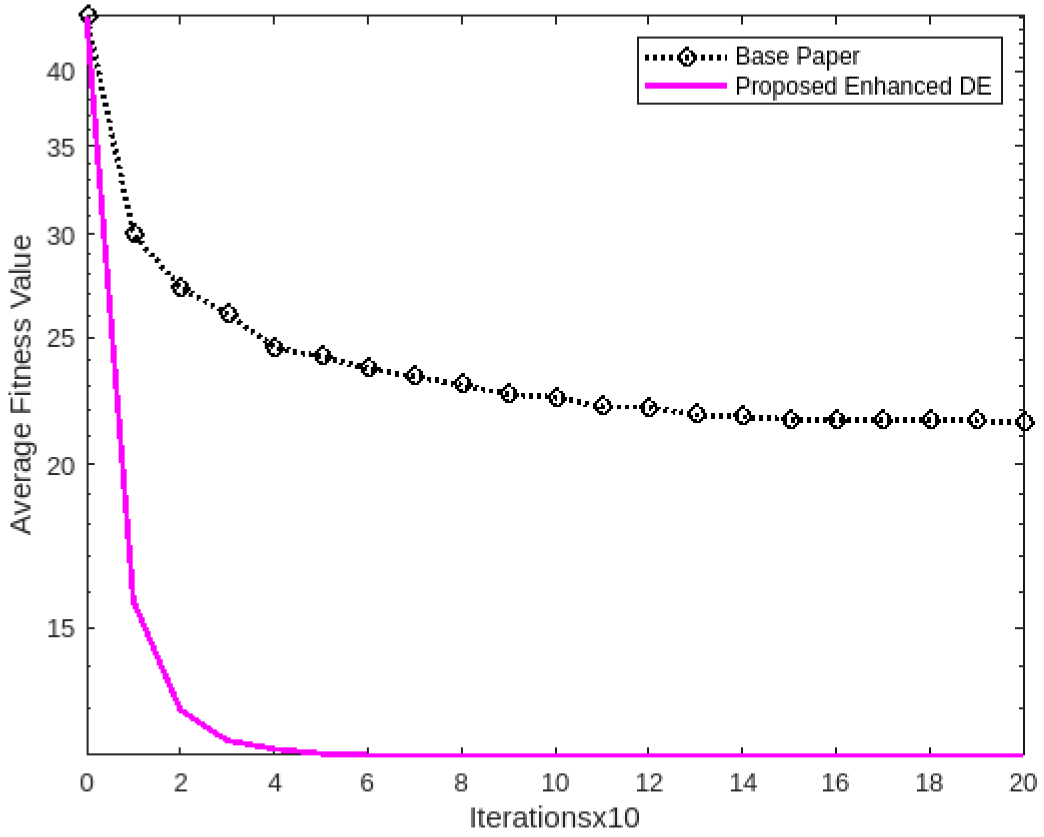


Figure 6.1: Sphere Function(f1) Convergence graph showing iterations along x-axis and average fitness values along y-axis

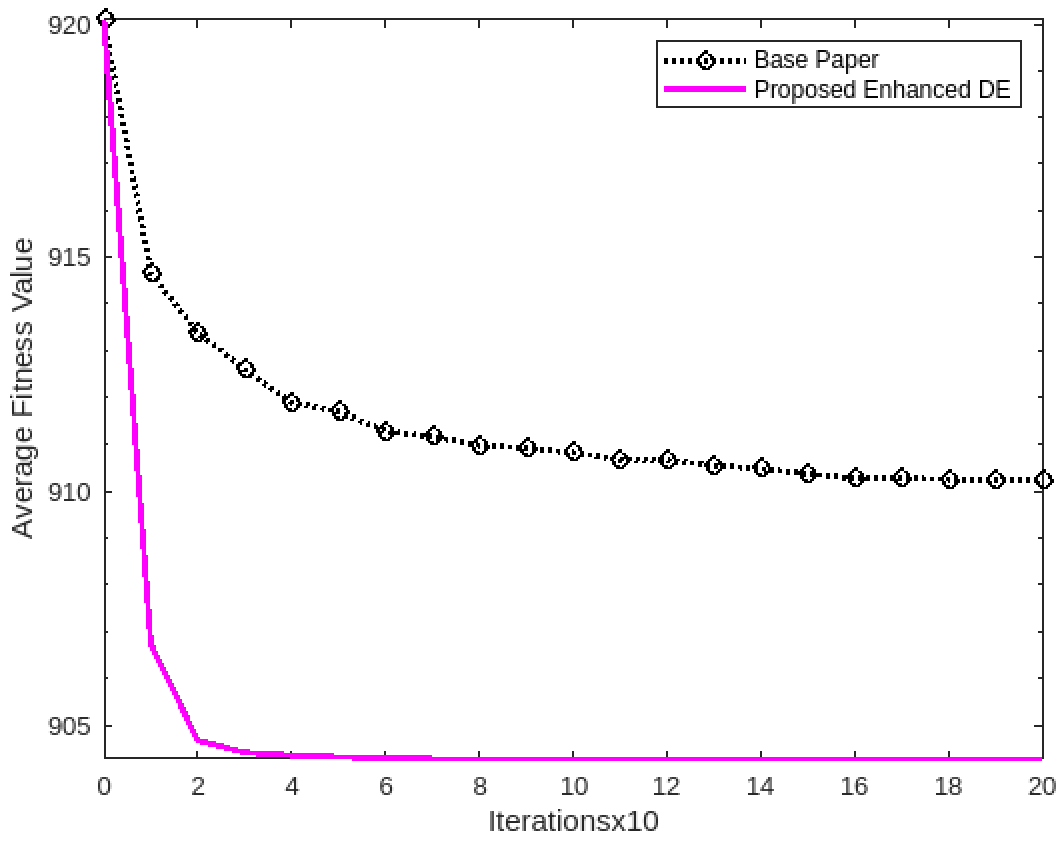


Figure 6.2: Rastrigin Function(f2) Convergence graph showing iterations along x-axis and average fitness values along y-axis

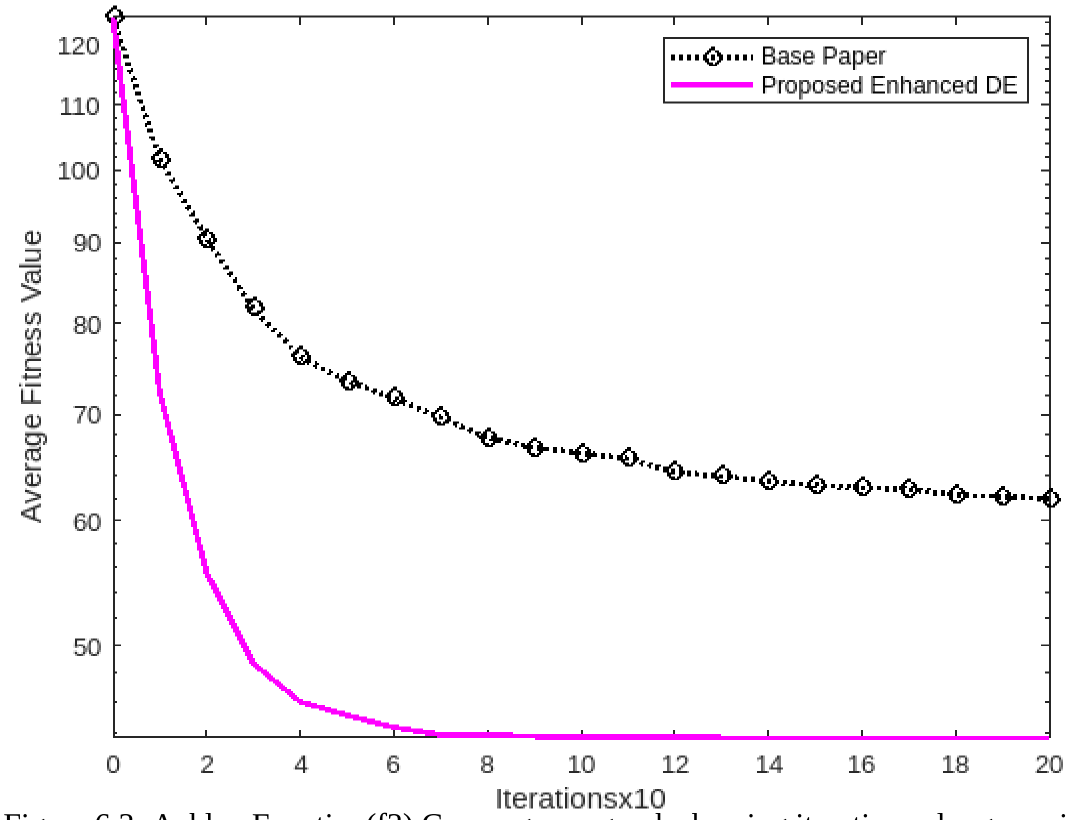


Figure 6.3: Ackley Function(f3) Convergence graph showing iterations along x-axis and average fitness values along y-axis

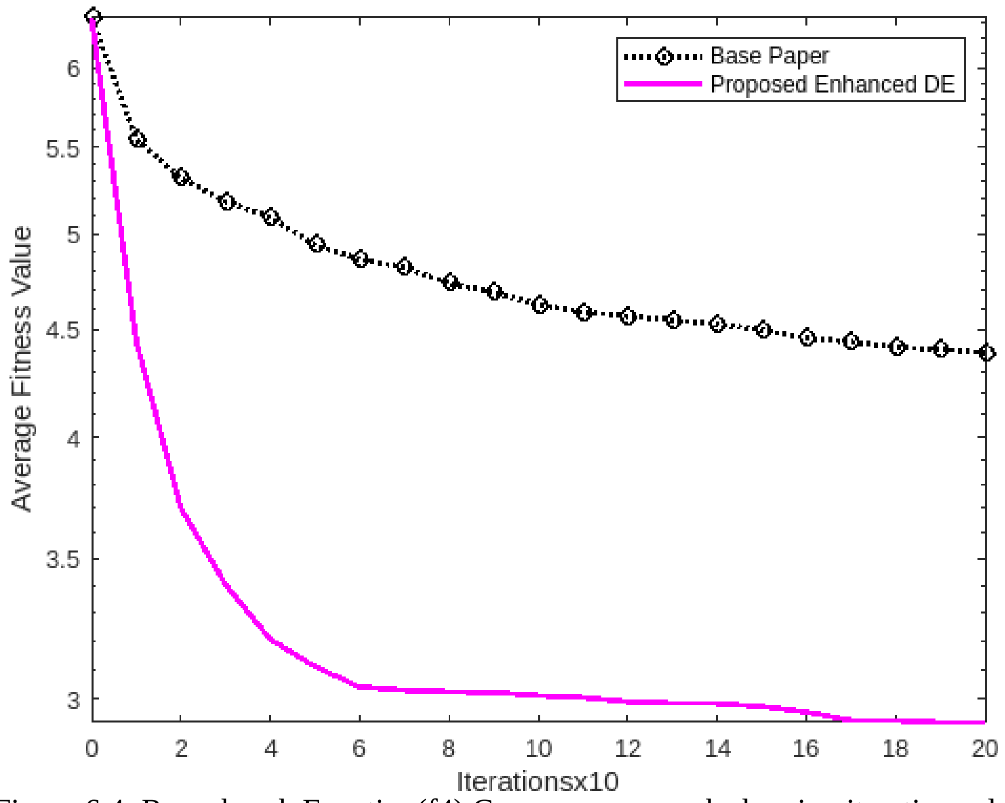


Figure 6.4: Rosenbrock Function(f4) Convergence graph showing iterations along x-axis and average fitness values along y-axis

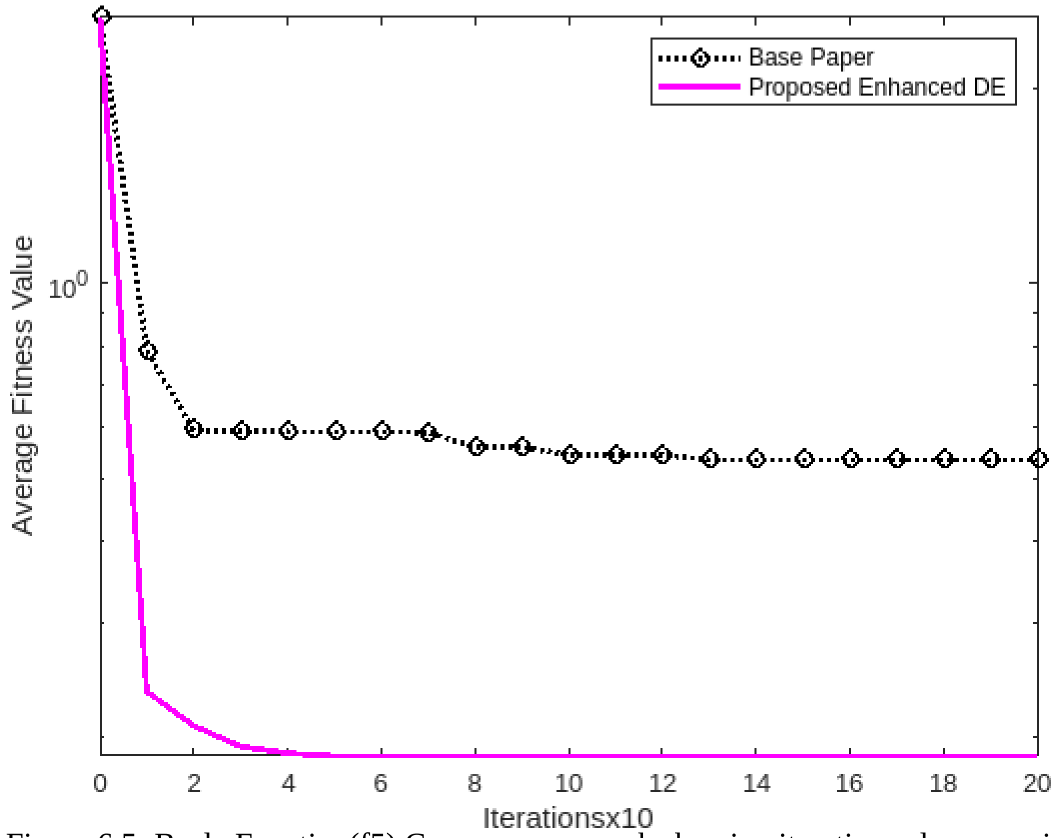


Figure 6.5: Beale Function(f5) Convergence graph showing iterations along x-axis and average fitness values along y-axis

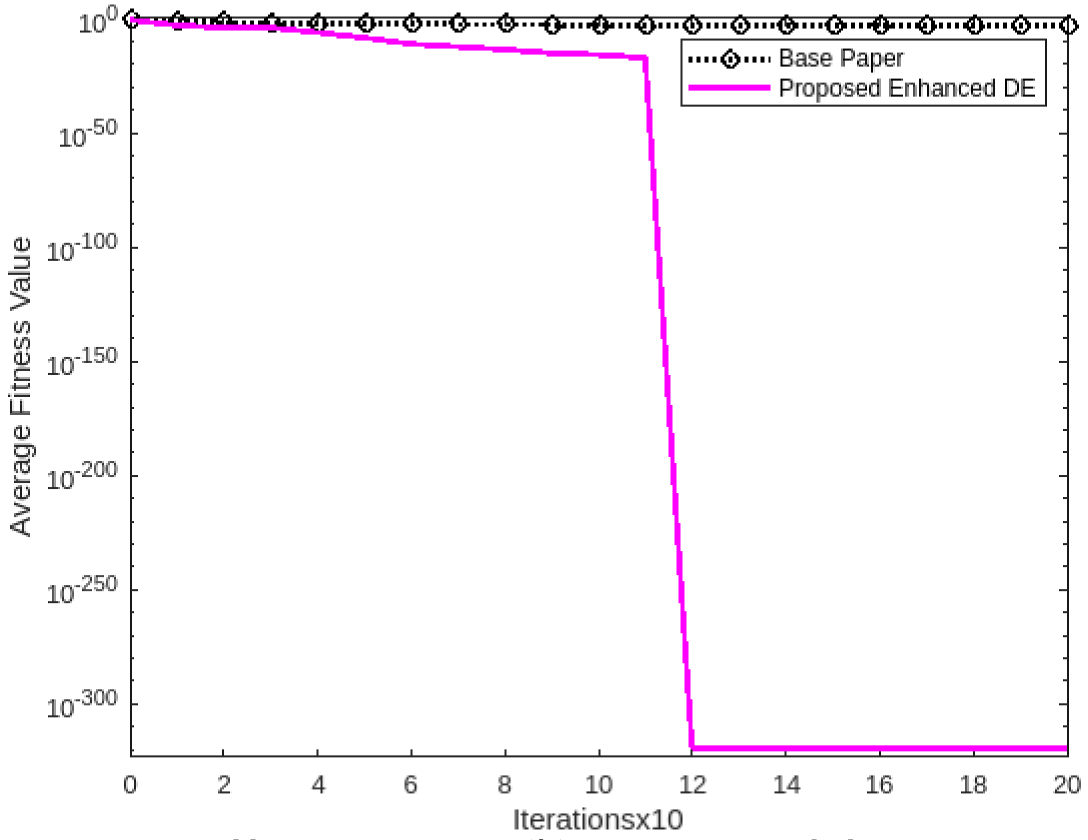


Figure 6.6: Goldstein_Price Function(f6) Convergence graph showing iterations along x-axis and average fitness values along y-axis

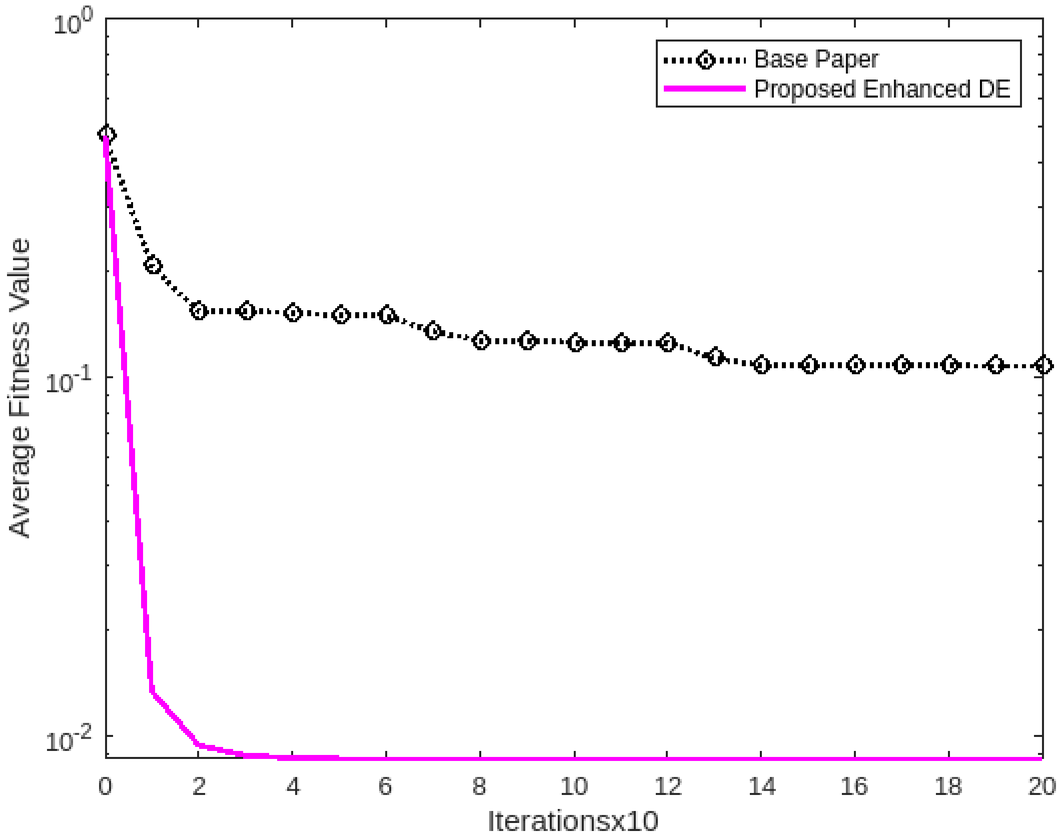


Figure 6.7: bohachevsky Function(f7) Convergence graph showing iterations along x-axis and average fitness values along y-axis

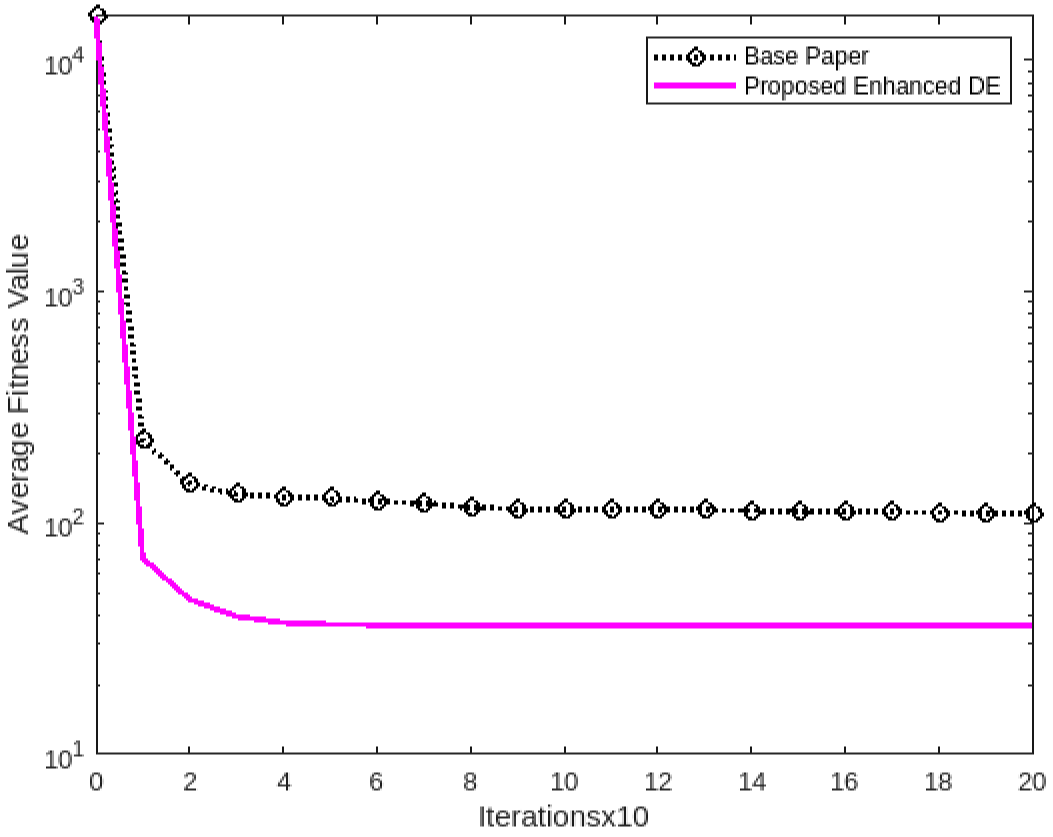


Figure 6.8: booth Function(f8) Convergence graph showing iterations along x-axis and average fitness values along y-axis

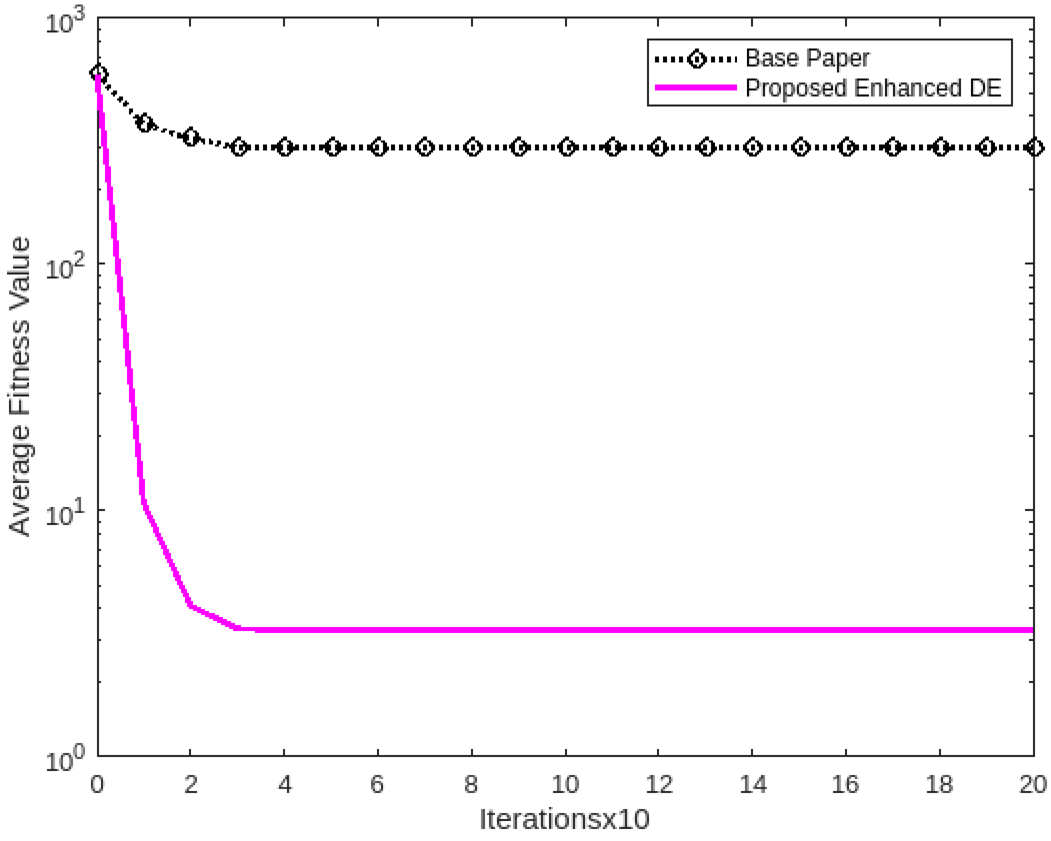


Figure 6.9: Matyas Function(f9) Convergence graph showing iterations along x-axis and average fitness values along y-axis

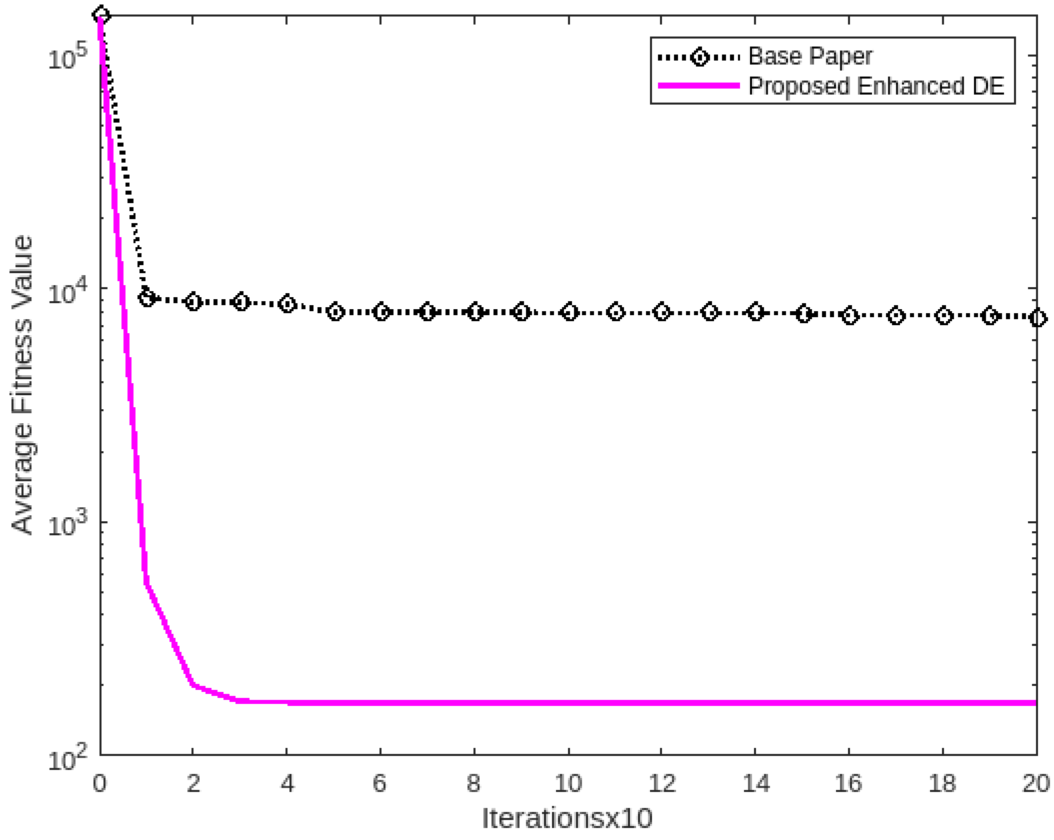


Figure 6.10: Zakharov Function(f10) Convergence graph showing iterations along x-axis and average fitness values along y-axis

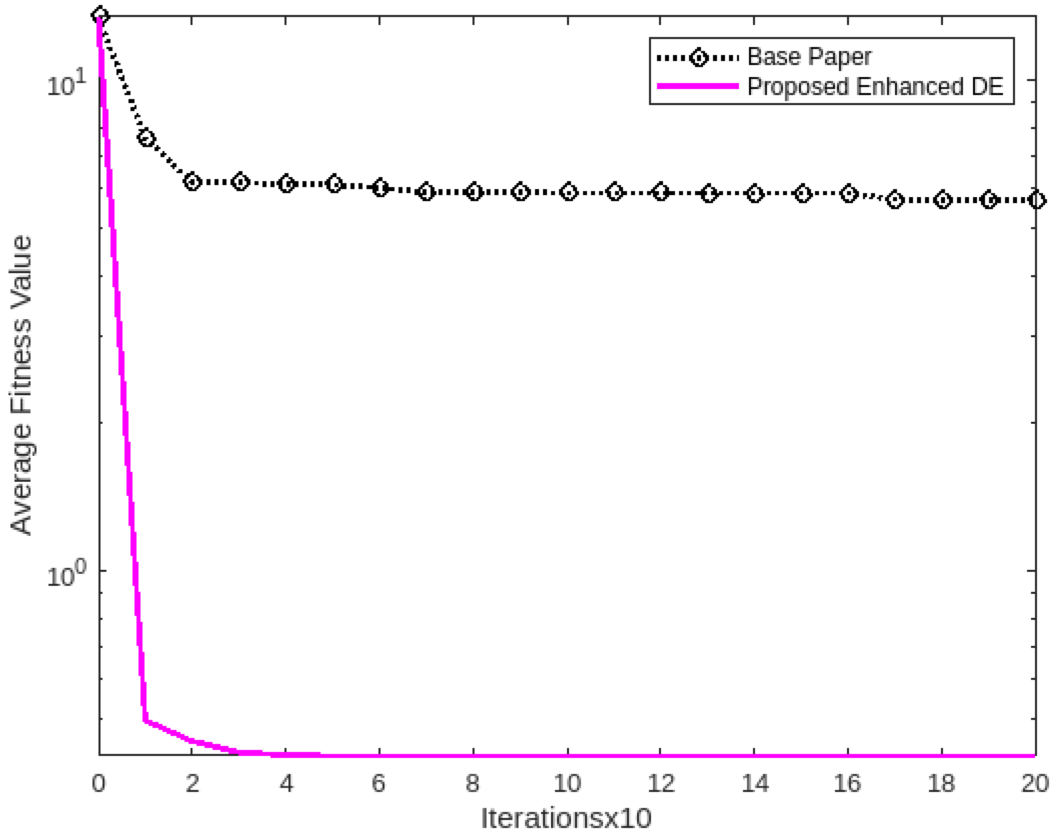


Figure 6.11: Six_hump Function(f11) Convergence graph showing iterations along x-axis and average fitness values along y-axis

5.4.2 Convergence Graphs using 50D, 150NP and 200 iterations

In this section we have generated convergence graphs of 11 benchmark functions using 50 dimensions, 150 as size of population and 200 as number of training iterations. Average fitness values are reported along the vertical axis of results and the iterations are shown along with the the x-axis of convergence graphs. To improve readability of experimental results, legends are also given to differentiate between the base algorithm and the proposed algorithm.

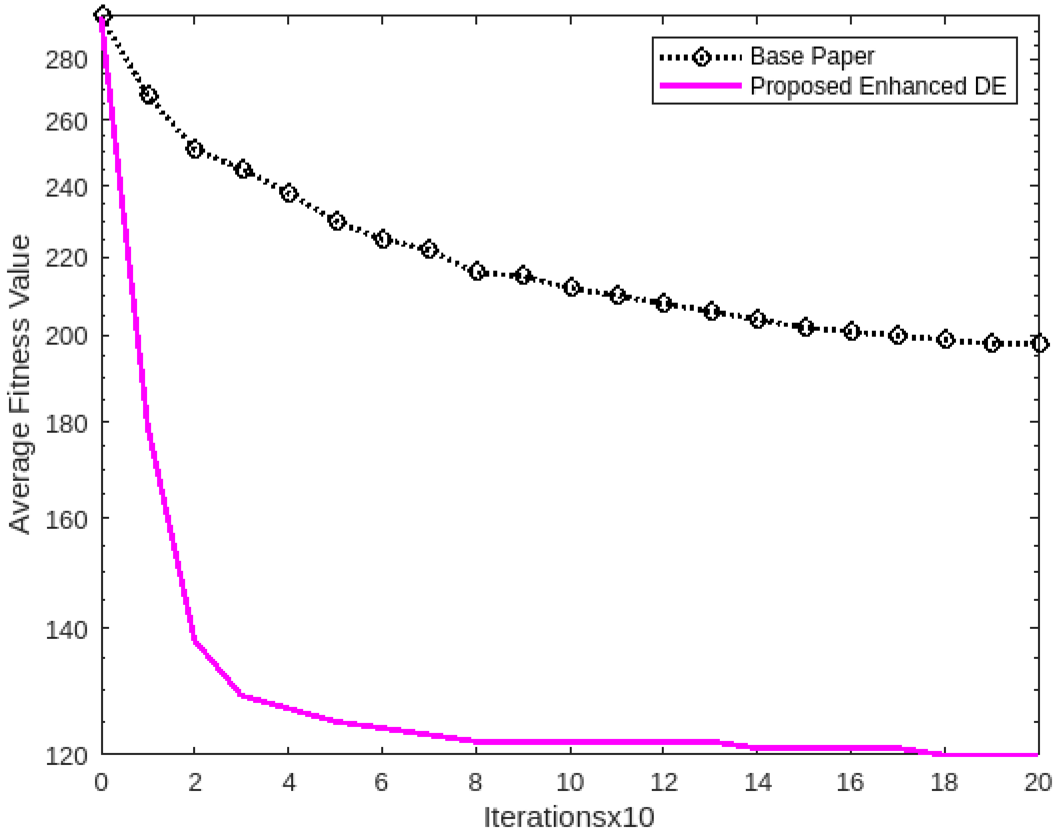


Figure 6.12: Sphere Function(f1) Convergence graph showing iterations along x-axis and average fitness values along y-axis

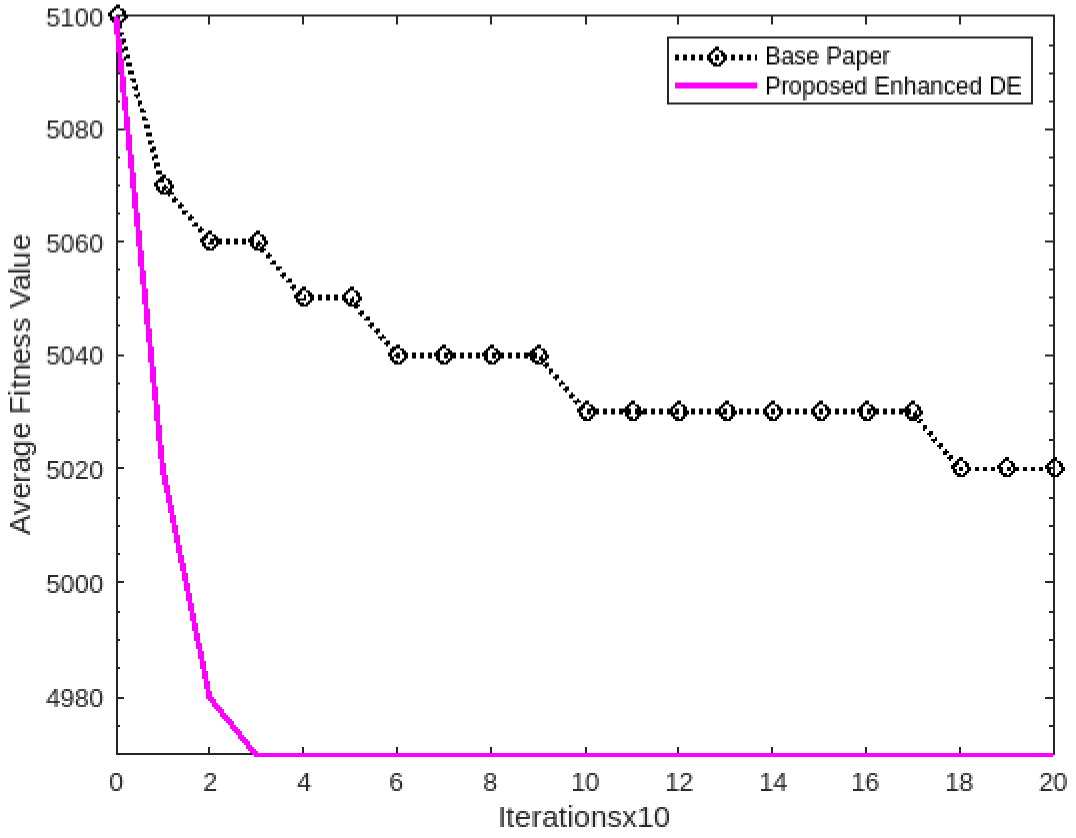


Figure 6.13: Rastrigin Function(f2) Convergence graph showing iterations along x-axis and average fitness values along y-axis

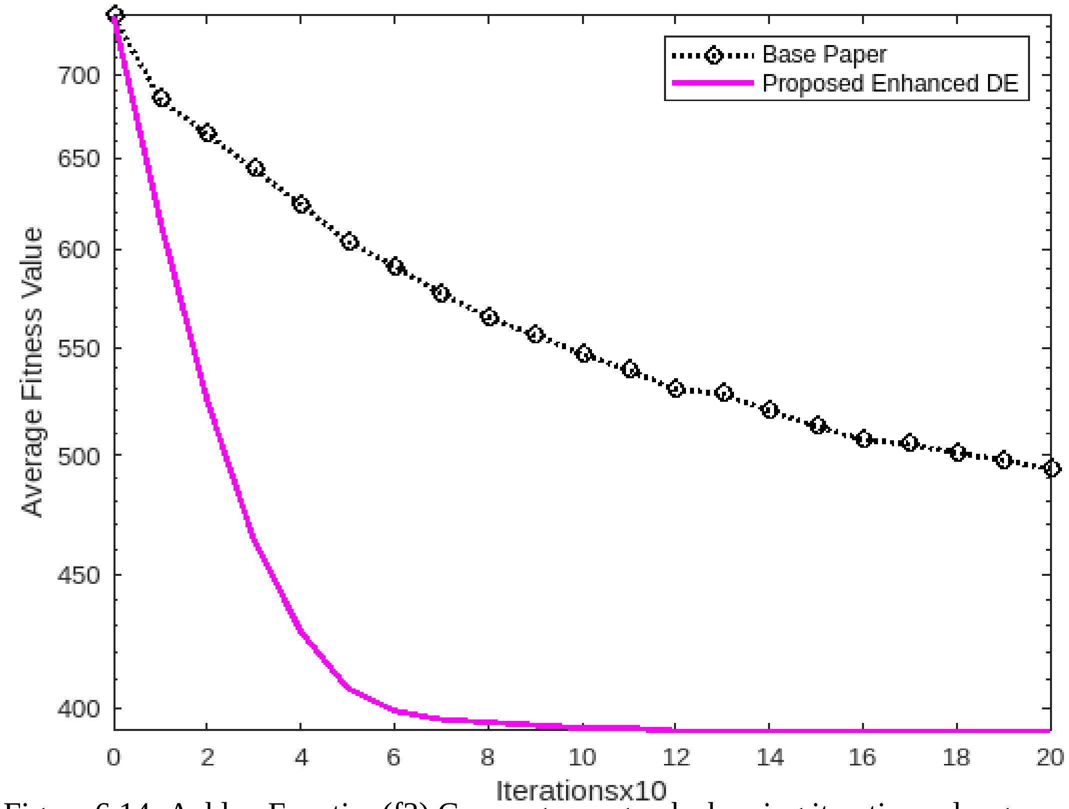


Figure 6.14: Ackley Function(f3) Convergence graph showing iterations along x-axis and average fitness values along y-axis

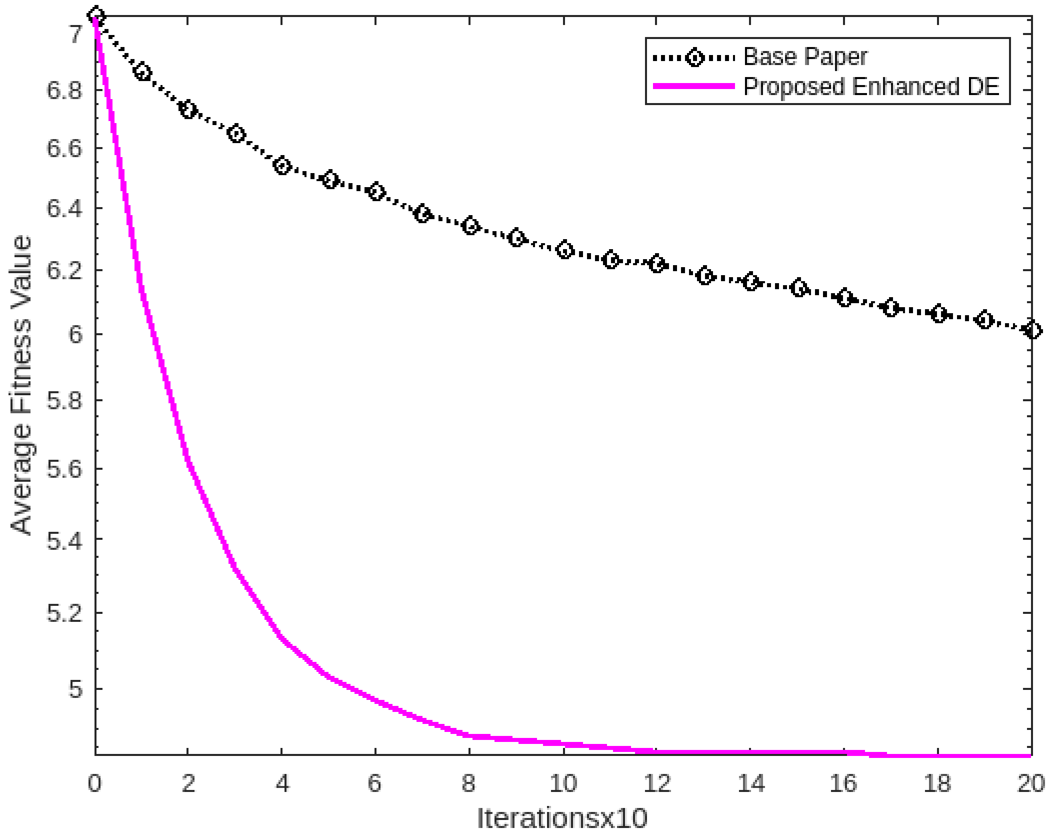


Figure 6.15: Rosenbrock Function(f4) Convergence graph showing iterations along x-axis and average fitness values along y-axis

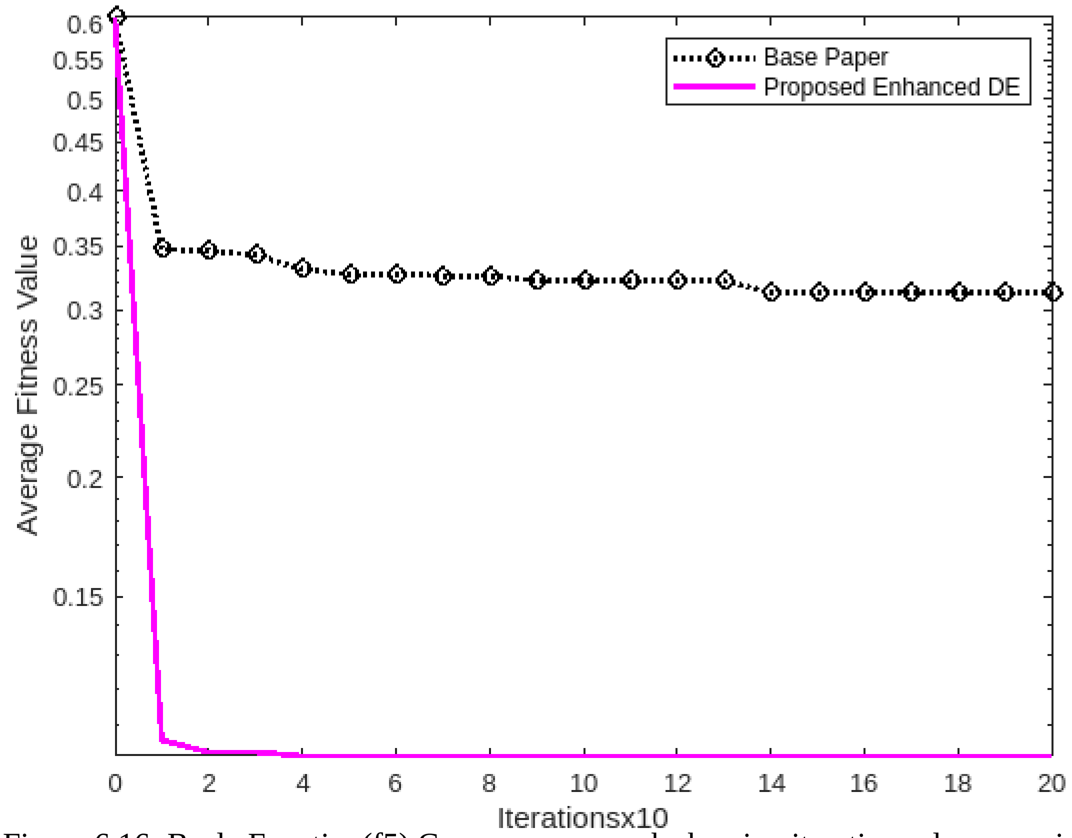


Figure 6.16: Beale Function(f5) Convergence graph showing iterations along x-axis and average fitness values along y-axis

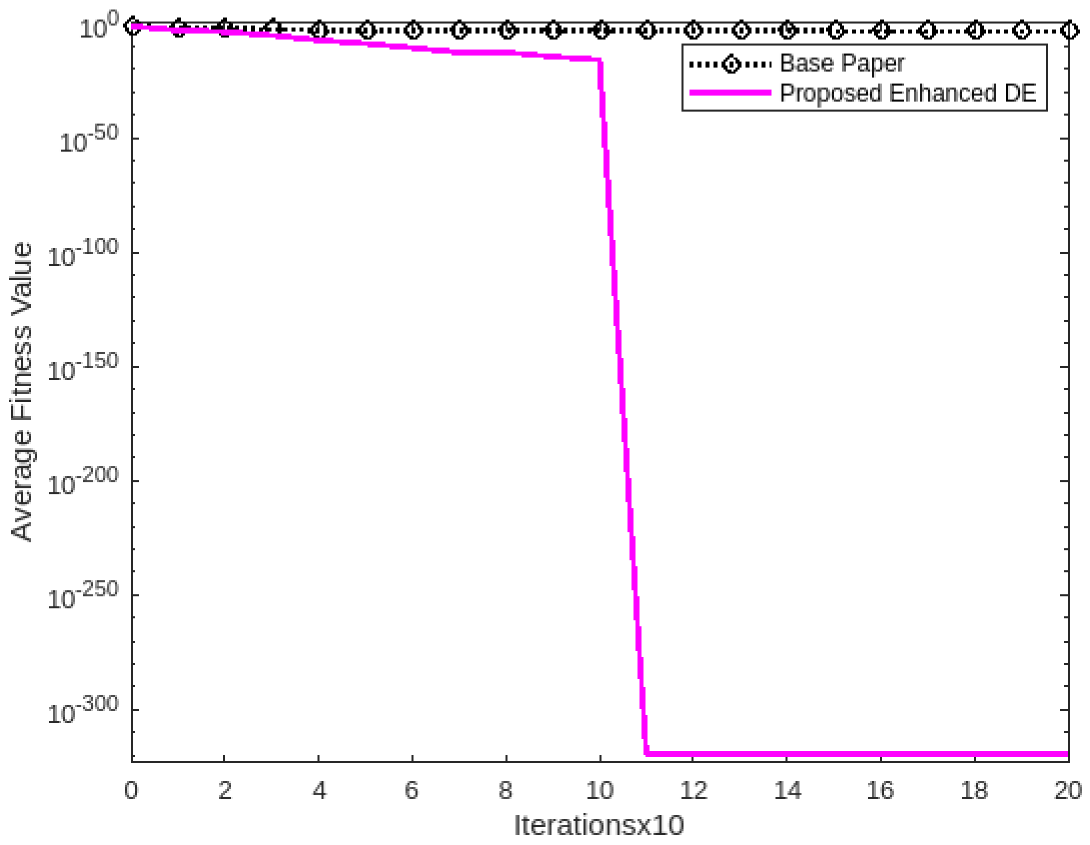


Figure 6.17: Goldstein_Price Function(f6) Convergence graph showing iterations along x-axis and average fitness values along y-axis

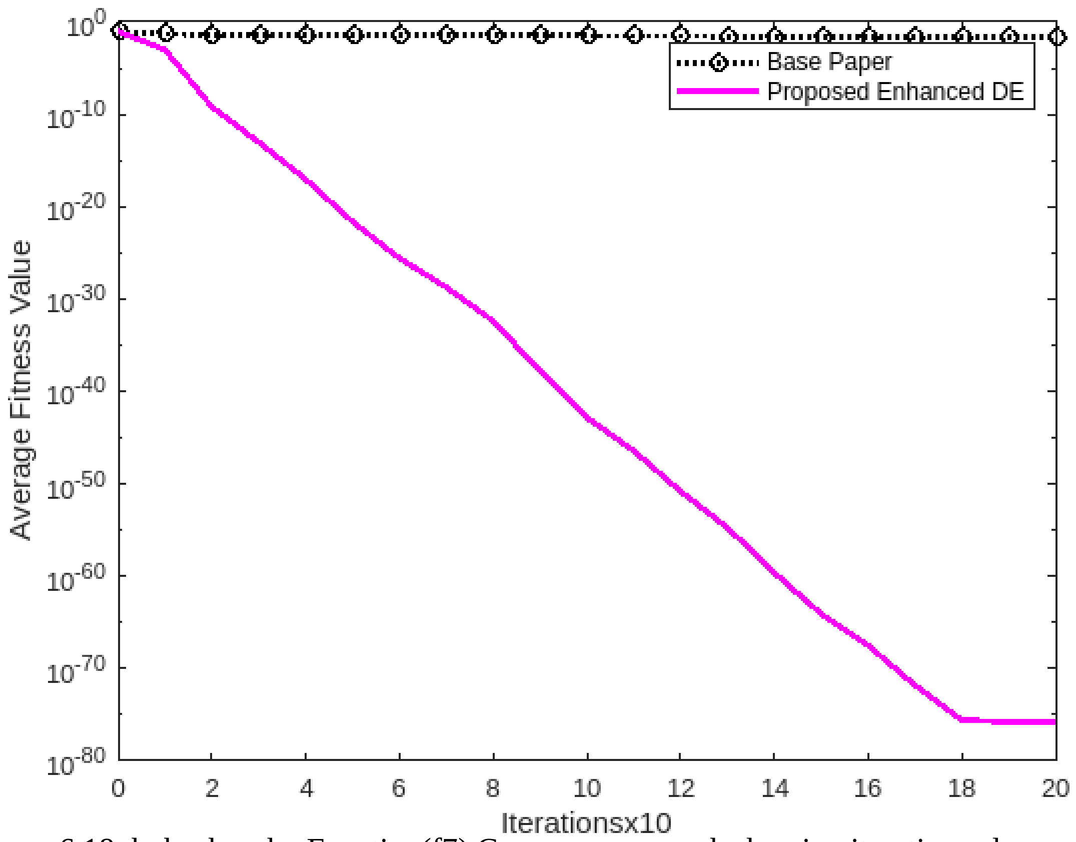


Figure 6.18: bohachevsky Function(f7) Convergence graph showing iterations along x-axis and average fitness values along y-axis

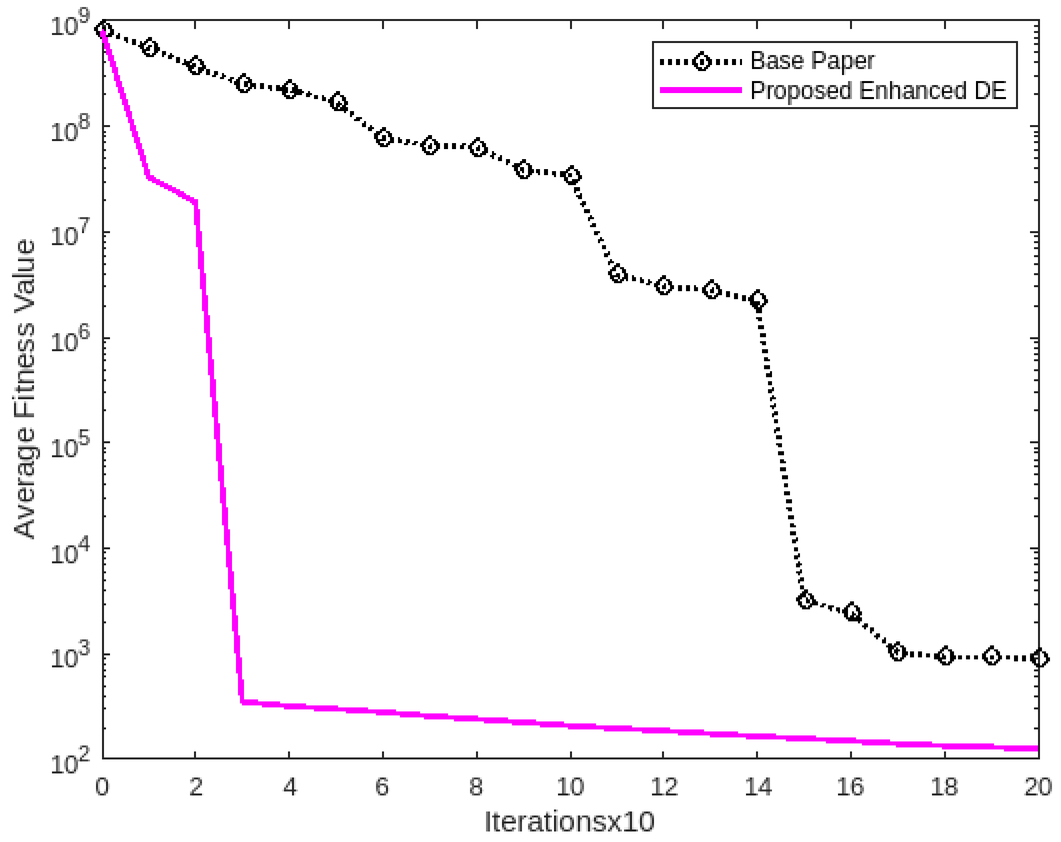


Figure 6.19: booth Function(f8) Convergence graph showing iterations along x-axis and average fitness values along y-axis

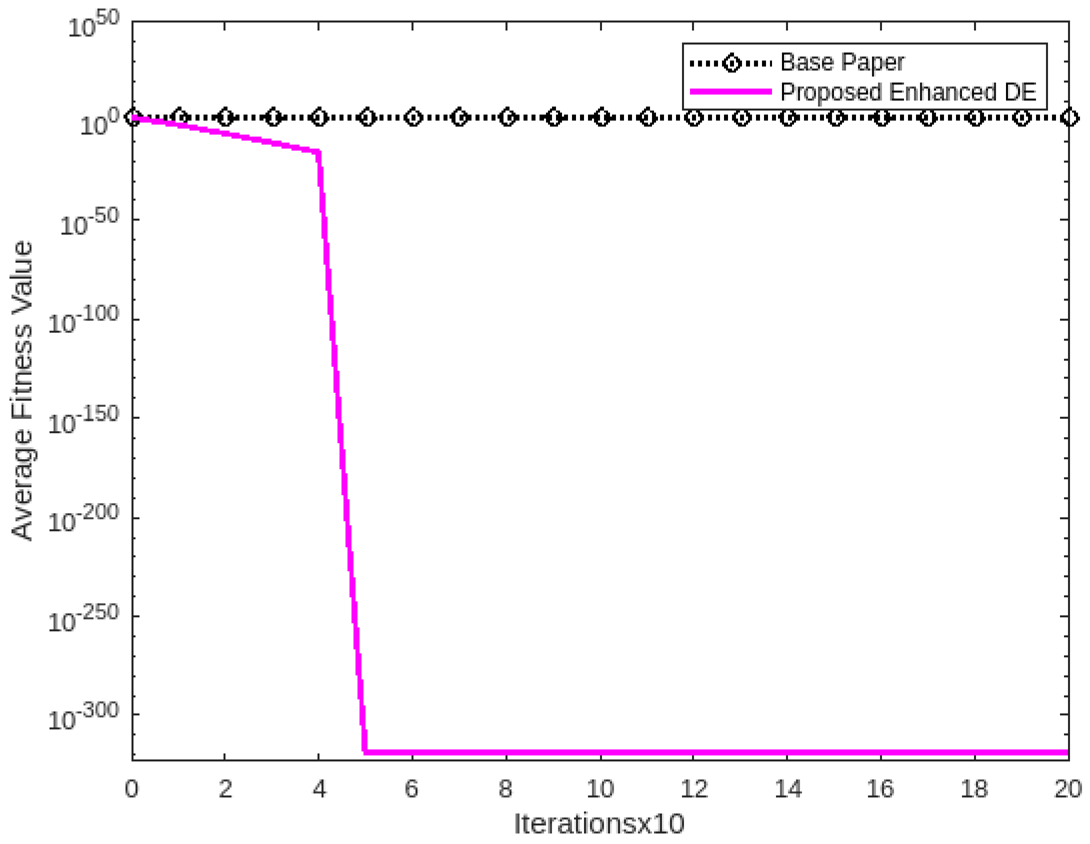


Figure 6.20: Matyas Function(f9) Convergence graph showing iterations along x-axis and average fitness values along y-axis

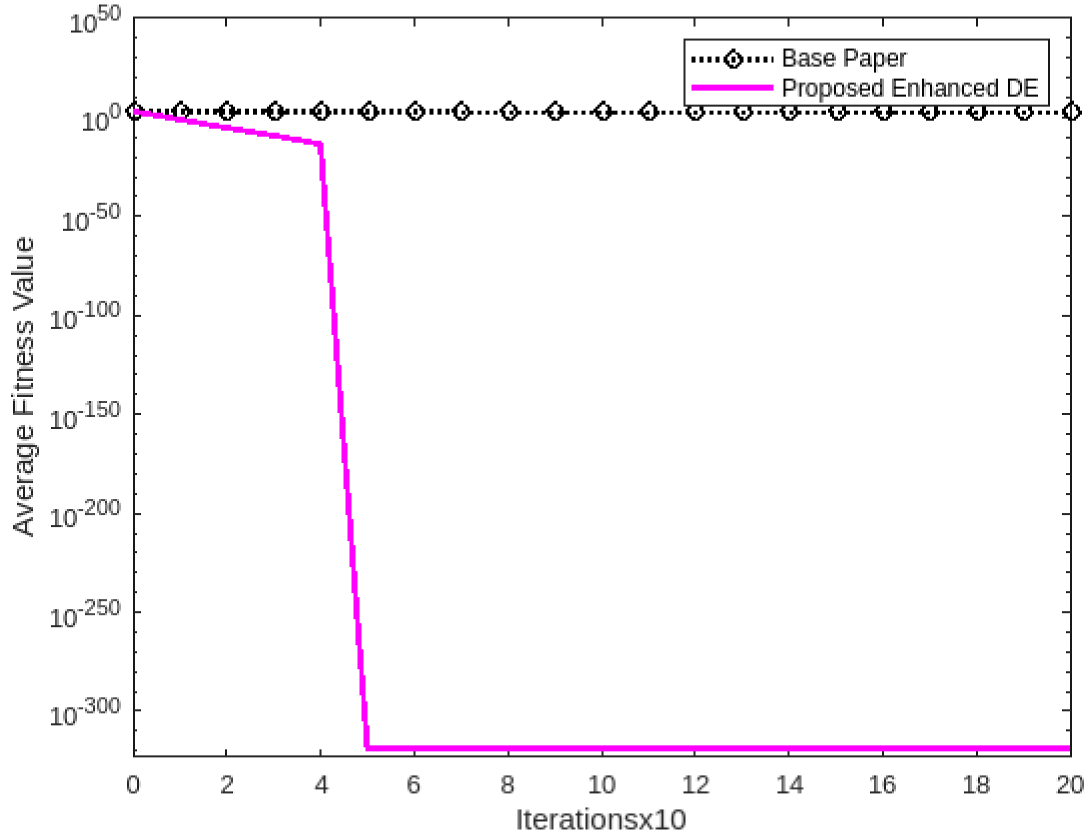


Figure 6.21: zakharov Function(f10) Convergence graph showing iterations along x-axis and average fitness values along y-axis

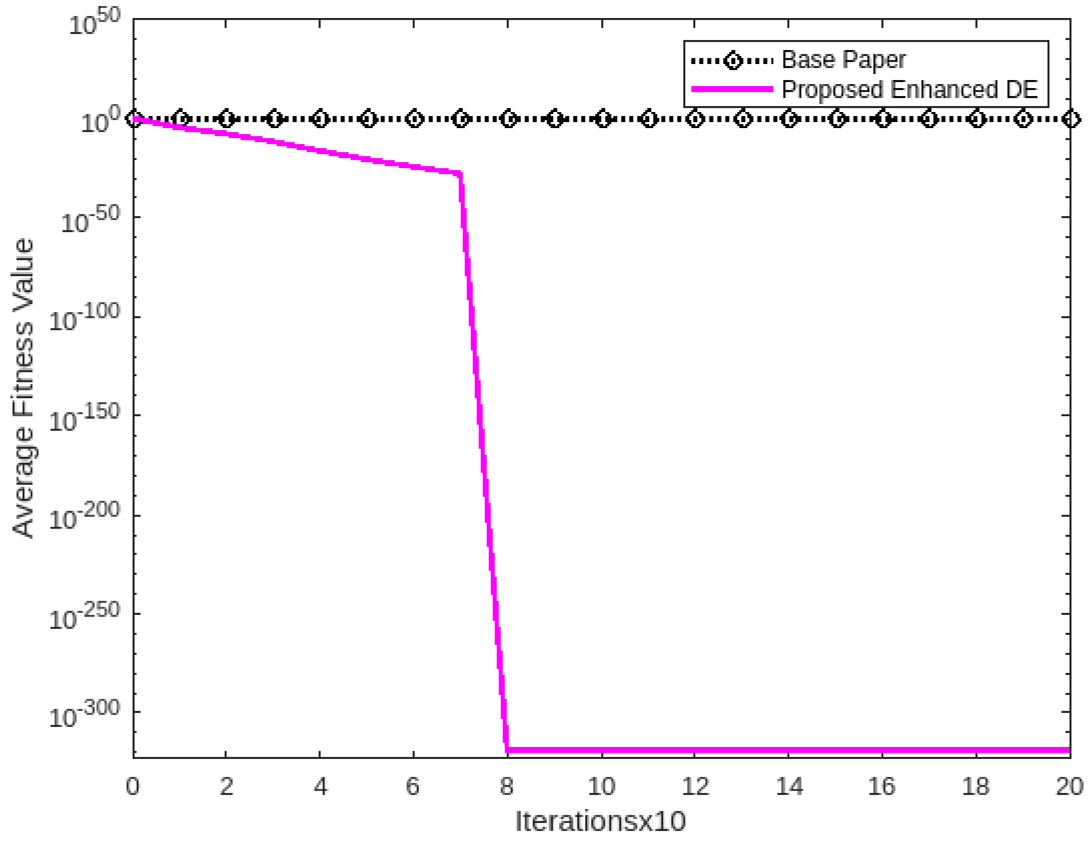


Figure 6.22: Six_hump Function(f11) Convergence graph showing iterations along x-axis and average fitness values along y-axis

The convergence analysis is based on experiments conducted using a 10 & 50-dimensional search space, with a population size of 30 & 150 and a maximum of 200 iterations. Both the proposed algorithm and the base algorithm were evaluated under these identical settings to ensure a fair comparison. The average fitness values were recorded at each iteration, and convergence graphs were plotted with the number of iterations on the x-axis and average fitness values on the y-axis. This setup provides a consistent and controlled environment to assess the convergence behavior and optimization capability of the proposed algorithm relative to the base method.

The proposed algorithm demonstrates superior convergence performance when compared to the base algorithm across a suite of benchmark functions. When plotted with iterations along the x-axis and average fitness values along the y-axis, the convergence graphs show that the proposed method consistently reaches lower fitness values in fewer iterations. This indicates that the algorithm is able to explore the search space more effectively in the early stages and transitions into exploitation more efficiently. A steep decline in average fitness is observed for the proposed algorithm, while the base algorithm shows a slower reduction. Additionally, the proposed algorithm stabilizes more quickly, indicating that it converges not only faster but also more reliably. Across multiple independent runs, the fitness values remain consistently better than those obtained by the base method, highlighting both the speed and robustness of convergence. These trends are evident across various types of benchmark functions, including those that are unimodal, multimodal, or deceptive in nature. Overall, the analysis confirms that the proposed approach offers enhanced optimization efficiency and is particularly suitable for problems where quick convergence to high-quality solutions is crucial.

Chapter # 7 Conclusion and Future Work

7.1 Conclusion

In this research work a novel mutation strategy utilizing top performing individual is introduced. The proposed mutation strategy considers the pool of top q% performing individuals from the population. After forming a pool of best performing individuals, three random individuals are selected from the pool unlike one best in the base algorithm. The three random selected best performing individuals will be helpful to incorporate diversity and results to escape from local optima issue. The selected individuals belong to top performing individuals that will enhance the convergence speed of the proposed algorithm. Experimental results were generated using a test suit of 11 benchmark functions. The experiments were averaged by performing 30 runs of the base algorithm and proposed algorithm. It can be observed from the average fitness results and convergence graphs that the proposed algorithm has better performance in all the cases. Research result shows significance improvement in convergence speed of proposed enhancement.

7.2 Future Work

The future work of this study is to use the memory to save the convergence track and make control parameters self-adaptive to increase the speed of the algorithm and to evaluate the performance of proposed algorithm using other real world applications.

References:

[1] R. Salgotra and A. H. Gandomi, "A novel multi-hybrid differential evolution algorithm for optimization of frame structures.," *Scientific Reports,* vol. 14, no. 1, p. 4877, 2024.

[2] X. Wang and X. Yu, "Differential Evolution Algorithm with Three Mutation Operators for Global Optimization.," *Mathematics,* vol. 12, no. 15, p. 2311, 2024.

[3] M. F. Ahmad and e. al., "Differential evolution: A recent review based on state-of-the-art works.," *Alexandria Engineering Journal,* vol. 61, no. 5, pp. 3831-3872, 2022.

[4] B. Chen and e. al., "Differential evolution algorithm with a complementary mutation strategy and data fusion-based parameter adaptation.," *Information Sciences,* vol. 668, p. 120522, 2024.

[5] E. Reyes-Davila and e. al., "Differential evolution: a survey on their operators and variants.," *Archives of Computational Methods in Engineering,* vol. 32, no. 1, pp. 83-112, 2025.

[6] Z. Zeng and e. al., "A new selection operator for differential evolution algorithm," *Knowledge-Based Systems,* vol. 226, p. 107150, 2021.

[7] P. Dhal and C. Azad, "A comprehensive survey on feature selection in the various fields of machine learning.," *Applied Intelligence,* vol. 52, no. 4, pp. 4543-4581, 2022.

[8] Y. Sun and e. al., "Differential evolution algorithm with population knowledge fusion strategy for image registration," *Complex & Intelligent Systems,* vol. 8, no. 2, pp. 835-850, 2022.

[9] S. S. Mohar, S. Goyal and R. Kaur, "Optimum deployment of sensor nodes in wireless sensor network using hybrid fruit fly optimization algorithm and bat optimization algorithm for 3D Environment.," *Peer-to-Peer Networking and Applications,* vol. 15, no. 6, pp. 2694-2718, 2022.

[10] R. Gikera and e. al., "K-hyperparameter tuning in high-dimensional genomics using joint optimization of deep differential evolutionary algorithm and unsupervised transfer learning from intelligent GenoUMAP embeddings.," *International Journal of Information Technology,* vol. 17, no. 3, pp. 1679-1701, 2025.

[11] D. K. Rakesh and P. K. Jana, "An improved differential evolution algorithm for quantifying fraudulent transactions.," *Pattern Recognition,* vol. 141, p. 109623.

[12] Y. Liu, M. Li and H. Fan, "Cryptanalysis of a color image encryption using minimax differential evolution-based 7D hyper-chaotic map.," *Multimedia Tools and Applications,* vol. 82, no. 28, pp. 44209-44225, 2023.

[13] S. Lee and e. al., "Extreme gradient boosting-inspired process optimization algorithm for manufacturing engineering applications," *Materials & Design,* vol. 226, p. 111625, 2023.

[14] X. Yu and Y. Lu, "Reinforcement learning-based multi-objective differential evolution for wind farm layout optimization.," *Energy,* vol. 284, p. 129300, 2023.

[15] S. Chakraborty and e. al., "Differential evolution and its applications in image processing problems: a comprehensive review.," *Archives of computational methods in engineering,* vol. 30, no. 2, pp. 985-1040, 2023.

[16] M. Pant and V. Snasel, "Design optimization of water distribution networks through a novel differential evolution.," *IEEE Access,* vol. 9, pp. 16133-16151, 2021.

[17] U. Guvenc and e. al., "Fitness–Distance Balance based adaptive guided differential evolution algorithm for security-constrained optimal power flow problem incorporating renewable energy sources.," *Applied Soft Computing,* vol. 108, p. 107421, 2021.

[18] J. Muñoz and e. al., "Geometrically constrained path planning for robotic grasping with Differential Evolution and Fast Marching Square.," *Robotica,* vol. 41, no. 2, pp. 414-432, 2023.

[19] R. Nithiavathy, S. Janakiraman and P. M. Deva, "Adaptive guided differential evolution-based slime mould algorithm-based efficient multi-objective task scheduling for cloud computing environments.," *Transactions on Emerging Telecommunications Technologies,* vol. 24, no. 1, p. e4902, 2024.

[20] M. F. Pairan, S. Shamsudin and M. F. Yaakub, "Autotuning PID Controllers for Quadplane Hybrid UAV using Differential Evolution Algorithm.," *Journal of Aeronautics, Astronautics and Aviation,* vol. 15, no. 1S, pp. 341-356.

[21] L. Xie and e. al., "A novel adaptive parameter strategy differential evolution algorithm and its application in midcourse guidance maneuver decision-making," *Complex & Intelligent Systems,* vol. 10, no. 1, pp. 847-868, 2024.

[22] J. Fan and L. Qu, "Innovative differential evolution algorithm with double-layer coding for autonomous underwater vehicles path planning in complex environments.," *Ocean Engineering,* vol. 303, p. 117806, 2024.

[23] S. Fu, C. Ma, K. Li and e. al., "Modified LSHADE-SPACMA with new mutation strategy and external archive mechanism for numerical optimization and point cloud registration," *Artificial Intelligence Review,* vol. 58, no. 3, pp. 58-72, 2025.

[24] B. Chen, H. Ouyang, S. Li and W. Ding, "Dual-stage self-adaptive differential evolution with complementary and," *Dual-stage self-adaptive differential evolution with complementary and,* vol. 93, p. 101855, 2025.

[25] M. Aljaidi and e. al., "A two stage differential evolution algorithm for parameter estimation of proton exchange membrane fuel cell.," *Scientific Reports,* vol. 15, no. 1, p. 5354, 2025.

[26] S. M. Parida and e. al., "Optimal parameter identification of photovoltaic systems based on enhanced differential evolution optimization technique.," *Scientific Reports,* vol. 15, no. 1, p. 2124, 2025.

[27] V. D. Beek and e. al. , "Hybrid differential evolution algorithm for the resource constrained project scheduling problem with a flexible project structure and consumption and production of resources.," *European Journal of Operational Research,* vol. 313, pp. 91-111, 2024.

[28] D. Zhang, "Optimization of Rendering Parameters of Cesium 3DTiles Model Based on Differential Evolution Algorithm.," *Applied Sciences,* vol. 15, no. 2, p. 801, 2025.

[29] C. Min and e. al. , "A Two-Stage Adaptive Differential Evolution Algorithm with Accompanying Populations.," *Mathematics*, vol. 13, no. 3, p. 440, 2025.

[30] Y. Shen, Y. Xie and Q. Chen, "Dual-Performance Multi-Subpopulation Adaptive Restart Differential Evolutionary Algorithm.," *Symmetry,* vol. 17, no. 2, p. 223, 2025.

[31] F. Yu and e. al. , "Multi-population differential evolution approach for feature selection with mutual information ranking," *Expert Systems With Applications,* vol. 260, p. 125404, 2025.

[32] F. Qin and e. al. , "Hybrid Harmony Search Algorithm Integrating Differential Evolution and Lévy Flight for Engineering Optimization.," *IEEE Access,* vol. 13, pp. 13534-13572, 2025.

[33] G. Wang, "Customer segmentation in the digital marketing using a Q-learning based differential evolution algorithm integrated with K-means clustering," *PloS one,* vol. 20, no. 2, p. e0318519, 2025.

[34] S. H. Hsieh, "Comparison of a hybrid firefly–Particle swarm optimization algorithm with six hybrid firefly–Differential evolution algorithms and an effective cost-saving allocation method for ridesharing recommendation systems.," *Electronics,* vol. 13, no. 2, p. 324, 2024.

[35] S. A. E. M. Boualem, B. Meftah and F. Debbat, "An adaptive coordinate systems for constrained differential evolution.," *Cluster Computing,* vol. 28, no. 1, pp. 1-24, 2025.

[36] T. Goel and e. al. , "Alzheimer's disease diagnosis from MRI and SWI fused image using self adaptive differential evolutionary RVFL classifier.," *Information Fusion,* vol. 118, p. 102917, 2025.

[37] S. Tao, S. Liu, R. Zhoa and e. al. , "A State-of-the-Art Fractional Order-Driven Differential Evolution for Wind Farm Layout Optimization.," *Mathematics,* vol. 13, no. 2, p. 282, 2025.

[38] M. Wu, "Research and Application of Optimization of Physical Education Training Model Based on Multi-Objective Differential Evolutionary Algorithm.," *Systems and Soft Computing,* vol. 7, p. 200200, 2025.

[39] L. Yu and Z. Meng, "Surrogate-Assisted Differential Evolution with multiple sampling mechanisms for high-dimensional expensive problems.," *Information Sciences,* vol. 687, p. 121408, 2025.

[40] Q. Xi and P. Jiang, "Design of news sentiment classification and recommendation system based on multi-model fusion and text similarity.," *International Journal of Cognitive Computing in Engineering,* vol. 6, pp. 44-54, 2025.

[41] A. Dey, S. Bhattacharyya, S. Dey and e. al., "A quantum inspired differential evolution algorithm for automatic clustering of real life datasets.," *Multimedia tools and applications,* vol. 83, no. 3, pp. 8469-8498, 2024.

[42] L. Xie and e. al., "A novel adaptive parameter strategy differential evolution algorithm and its application in midcourse guidance maneuver decision-making.," *Complex & Intelligent Systems,* vol. 10, no. 1, pp. 847-868, 2024.

[43] A. Ali and e. al., "A novel solution to optimal power flow problems using composite differential evolution integrating effective constrained handling techniques.," *Scientific Reports,* vol. 14, no. 1, p. 6187, 2024.

[44] V. D. Beek and e. al. , "Hybrid differential evolution algorithm for the resource constrained project scheduling problem with a flexible project structure and consumption and production of resources.," *European Journal of Operational Research,* vol. 313, no. 1, pp. 92-111, 2024.

[45] P. Tiwari and e. al., "Developments and design of differential evolution algorithm for non-linear/non-convex engineering optimization.," *Archives of Computational Methods in Engineering,* vol. 31, no. 4, pp. 2227-2263, 2024.

[46] S. Fadhil, H. Zaher, N. Ragaa and E. Oun, "A modified differential evolution algorithm based on improving a new mutation strategy and self-adaptation crossover," *MethodsX,* p. 102276, 2023.

[47] V. T. Nguyen, V. M. Tran and N. T. Bui, "Self-Adaptive Differential Evolution with Gauss Distribution for Optimal Mechanism Design.," *Applied Sciences,* vol. 13, no. 10, p. 6284, 2023.

[48] W. Deng and e. al., "An improved differential evolution algorithm and its application in optimization problem.," *Soft Computing,* vol. 25, pp. 5277-5298, 2021.