

**REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES**

**A METAHEURISTIC OPTIMIZATION TECHNIQUE FOR
FEATURE SELECTION**



MUNEER MAAROOF HASAN

**MSc. THESIS
DEPARTMENT OF COMPUTER ENGINEERING
PROGRAM OF COMPUTER ENGINEERING**

**ADVISER
ASSIST. PROF. DR. OĐUZ ALTUN**

İSTANBUL, 2016

REPUBLIC OF TURKEY
YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

**A METAHEURISTIC OPTIMIZATION TECHNIQUE FOR
FEATURE SELECTION**

A thesis submitted by Muneer Maaroof HASAN in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE** is approved by the committee on 25.11.2016 in Department of Computer Engineering.

Thesis Adviser

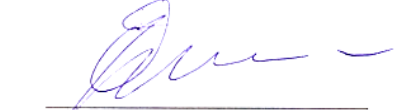
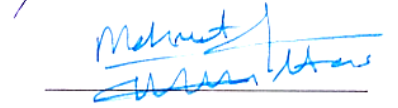
Assist. Prof. Dr. Oğuz ALTUN
Yıldız Technical University

Approved By the Examining Committee

Assist. Prof. Dr. Oğuz ALTUN
Yıldız Technical University

Assist. Prof. Dr. Mehmet AKTAŞ
Yıldız Technical University

Assist. Prof. Dr. Ali Haydar ÖZER
Marmara University

ACKNOWLEDGEMENTS

First of all, I would like to thank God for the blessing, which bestowed on us of health, science, a great family, and good friends in my life.

I would like to express my uttermost gratitude to my supervisor Assist. Prof. Dr. Oğuz ALTUN and Assist. Prof. Dr. Gökhan BILGIN for their motivation, enthusiasm, and guidance, which have been essential at all stages of my research.

My sincere gratitude goes to my entire family: parents Maaroo HASSAN and Zahra ALI; my sisters; and every single person in my family. The completion of this MSc. would not have been possible without their kind support and encouragement.

My deepest thanks go to my father for making me believe that there is no meaning to the “impossible” word.

I am grateful to my brother Moshtaq MAAROOF, who is the friend of my soul, who I would not be able to see the world without.

I would also like to thank all my friends, especially the ones I met in Turkey and lived with for their continuous support.

November, 2016

Muneer Maaroo HASSAN

TABLE OF CONTENTS

| | Page |
|--|------|
| LIST OF SYMBOLS | vi |
| LIST OF ABBREVIATIONS..... | vii |
| LIST OF FIGURES | viii |
| LIST OF TABLES..... | ix |
| ABSTRACT..... | x |
| ÖZET | xi |
| CHAPTER 1 | 1 |
| INTRODUCTION | 1 |
| 1.1 Literature Review..... | 1 |
| 1.2 Objective of the Thesis | 2 |
| 1.3 Hypothesis..... | 2 |
| 1.4 Motivation..... | 2 |
| 1.5 Knowledge Discovery..... | 3 |
| 1.6 Feature Selection..... | 3 |
| 1.7 Feature Selection Measure | 5 |
| 1.8 Differential Evolution as an Evolutionary Algorithm | 7 |
| 1.8.1 The Initialization Step..... | 8 |
| 1.8.2 The Mutation Operation of Differential Evolution..... | 9 |
| 1.8.3 The Crossover Operation of Differential Evolution | 9 |
| 1.9 k NEAREST NEIGHBOR (k-NN) | 10 |
| CHAPTER 2 | 11 |
| ENHANCED DIFFERENTIAL EVOLUTION ALGORITHM AND EVALUATIONS | 11 |
| 2.1 Background About Feature Selection | 11 |
| 2.2 The Enhanced Differential Evolution (EDE)..... | 12 |
| 2.3 Testing the Enhanced and the Standard Versions of Differential Evolution | 13 |
| 2.4 Enhanced Differential Evolution Feature Selector (EDEFS) | 15 |

| | |
|--|----|
| 2.4.1 Key Notions | 24 |
| 2.4.2 Initialization | 24 |
| 2.4.3 Work Flow | 25 |
| 2.4.4 Final Step and Showing Results | 28 |
| CHAPTER 3 | 31 |
| COMPARING EDEFS TO OTHER'S WORKS..... | 31 |
| 3.1 Results of High Dimensional Datasets | 31 |
| 3.2 Results of Small Scale Datasets | 34 |
| 3.3 Results of Comparing the Binary and the Real Differential Evolution FS ... | 36 |
| 3.4 Results of Comparing the EDEFS with Random Feature Selection..... | 37 |
| 3.5 Results of Comparing the EDEFS with Random Indexes DEFS | 39 |
| CHAPTER 4 | 41 |
| CONCLUSION AND FUTURE WORK | 41 |
| 4.1 Conclusion | 41 |
| 4.2 Future Works | 42 |
| REFERENCES | 43 |
| CURRICULUM VITAE..... | 47 |

LIST OF SYMBOLS

| | |
|-------------|--|
| F | Amplification weight factor |
| $T(j,i,G)$ | Tweaked individual after the crossover operation in Differential Evolution |
| $V(j,I,G)$ | The original Vector in Differential Evolution |
| $V_{r_i,G}$ | Vectors chosen randomly out of the Differential Evolution's population |
| $X(j,i,G)$ | The output vector of the Differential Evolution's mutation operation |
| $X_{i,G}$ | Tweaked individual after the Differential Evolution's mutation operation |

LIST OF ABBREVIATIONS

| | |
|----------|---|
| BDEFS | Binary Differential Evolution Feature Selection |
| CEC 2015 | Congress on Evolutionary Computation |
| CFS | Correlation-based Feature Selection |
| Cr | Crossover rate |
| DE | Differential Evolution |
| DED | Enhanced Differential Evolution |
| EDEFS | Enhanced Differential Evolution Feature Selector |
| f1...f15 | function1...function15 in the CEC 2015 benchmark |
| FES | Function Evaluation Times |
| FRFS | Fuzzy-rough Feature Selection |
| FS | Feature Selection |
| KD | Knowledge Discovery |
| k-NN | k Nearest Neighbor |
| PCFS | Probabilistic Consistency-based Feature Selection |
| PCR | Probability of Crossover Rate |
| POP | Population |
| RFS | Random Feature Selection |
| RIDEFS | Random Indexed Differential Evolution Feature Selection |

LIST OF FIGURES

| | | Page |
|-------------|---|------|
| Figure 1.1 | Knowledge Discovery process [16] | 4 |
| Figure 1.2 | Filter model [27] | 6 |
| Figure 1.3 | Wrapper model [27] | 6 |
| Figure 1.4 | Pseudocode of standard DE [3]..... | 8 |
| Figure 1.5 | k Nearest Neighbor classifier. (A) when k equal to 1; (B) when k equal to 4 [45]..... | 10 |
| Figure 2.1 | Pseudocode of the EDE algorithm, Bold lines represent the changes | 12 |
| Figure 2.2 | Function 1 results | 16 |
| Figure 2.3 | Function 2 results | 17 |
| Figure 2.4 | Function 3 results | 17 |
| Figure 2.5 | Function 4 results | 18 |
| Figure 2.6 | Function 5 results | 18 |
| Figure 2.7 | Function 6 results | 19 |
| Figure 2.8 | Function 7 results | 19 |
| Figure 2.9 | Function 8 results | 20 |
| Figure 2.10 | Function 9 results | 20 |
| Figure 2.11 | Function 10 results | 21 |
| Figure 2.12 | Function 11 results | 21 |
| Figure 2.13 | Function 12 results | 22 |
| Figure 2.14 | Function 13 results | 22 |
| Figure 2.15 | Function 14 results | 23 |
| Figure 2.16 | Function 15 results | 23 |
| Figure 2.17 | Example of the assigned variables to each individual in population | 24 |
| Figure 2.18 | Four individual's values after the initialization step..... | 27 |
| Figure 2.19 | Input and Output of the sorting operation..... | 27 |
| Figure 3.1 | Pseudocode of the Binary Feature Selection | 37 |
| Figure 3.2 | The pseudocode of the Random Feature Selection..... | 38 |
| Figure 3.3 | The pseudocode of the Random Indexes DEFS | 40 |

LIST OF TABLES

| | Page |
|------------|---|
| Table 2. 1 | Functions of the CEC2015..... 13 |
| Table 2. 2 | Results of comparing the stander DE algorithm and the Enhanced DE algorithm using 10 dimensions, 20 individuals, and 500 function evaluations on CEC 2015 test bed. Smaller values are better..... 14 |
| Table 2.3 | The output of EDEFS with Wine dataset by using k-NN.....29 |
| Table 2.4 | The indexes of each case in table 2.2.....30 |
| Table 3.1 | Details about the high-dimensional datasets.....32 |
| Table 3.2 | The result of implementing EDEFS on high dimensional data33 |
| Table 3.3 | Details about the small-scale datasets.....34 |
| Table 3.4 | Implementing EDEFS on datasets with 1-NN35 |
| Table 3.5 | The group of datasets to test BDEFS and EDEFS36 |
| Table 3.6 | The group of datasets to compare RFS and EDEFS38 |
| Table 3.7 | The group of datasets to compare RIDEFS and EDEFS39 |

ABSTRACT

A METAHEURISTIC OPTIMIZATION TECHNIQUE FOR FEATURE SELECTION

Muneer Maaroof HASAN

Department of Computer Engineering
MSc. Thesis

Adviser: Assistant Prof. Dr. Oğuz ALTUN

In Knowledge Discovery (KD) one of the problems we face is unrelated data. Unrelated data causes performance reduction when attempting to learn or draw insights from the original data. There are several ways to remove unrelated data, and one of them is Feature Selection (FS). FS aims to select features that represent the original data in a relevant and understandable way. There are many ways to select features. This thesis provides an approach to implementing a stochastic algorithm as a feature selector. Differential Evolution algorithm is one of the popular metaheuristic algorithms, so we will employ it as a feature selector. Results and comparisons show that our approach is successful according to a comparison with two recent works in this field.

Key words: Differential Evolution, Expensive Optimization, Feature Selection, CEC2015, Metaheuristic, Binary Optimization

YILDIZ TECHNICAL UNIVERSITY
GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

ÖZELLİK SEÇİMİ İÇİN SEZGİSEL OPTİMİZASYON

Muneer Maarroof HASAN

Bilgisayar Mühendisliği Bölümü

Yüksek Lisans Tezi

Danışman: Yrd. Doç. Dr. Oğuz ALTUN

Bilgi Keşfi sürecinde birçok problemle karşı karşıya gelinmektedir. Bu problemlerden biri biri ise, birbiriyle ilişkisi olmayan, bağlantısız verilerdir. Bu bağlantısız verileri ortadan kaldırmak için çeşitli yollar bulunmaktadır ve Özellik Seçimi onlardan biridir. Özellik Seçimi'nin amacı, orjinal veriyi en iyi şekilde yansıtacak olan özellikleri seçmektedir. Bu özellikleri seçmek için birçok kullanılabilir. Bu tezde bir stokastik algoritmayı özellik seçicisi olarak kullanıyoruz.

Diferansiyel Evrim Algoritması, yaygın sezgisel optimizasyon algoritmalarından biridir.. Bu yüzden Özellik Seçici olarak bu algoritma çalıştırılmıştır. Bu alanda son zamanlarda yapılan iki çalışma ile kıyaslandığında yaklaşımımızın başarılı olduğu görülmektedir.

Anahtar Kelimeler: Diferansiyel Evrim, Pahalı Optimizasyon, Özellik Seçimi, İkinci Geliştirilmiş Diferansiyel Evrim Özellik Seçimi. CEC 2015

INTRODUCTION

1.1 Literature Review

When mathematical conditions are met, there will be several applicable optimization methods to solve the given problem. For example, linear programming is efficient when the objective function is expressed as a decision variable [1]. In several cases, the objective function is nonlinear. Therefore, the non-linear programming will be applicable. In practice, usually and unfortunately many problems include more complications. As a result, these classical methods cannot be used [1]. Metaheuristic algorithms have been designed to solve these kinds of problems [2].

There are several problems that have no clear results or the output is unpredictable. These kinds of problems are explained with the term "you know it when you see it" [3]. That means the optimal solutions are hard to find from the first sight at the problem or in the first iteration of a solution algorithm.

Feature Selection is considered as a multi-objective problem [4] and also considered as one of the "you know it when you see it" problems [3]. We can conclude that metaheuristic algorithms are able to solve it successfully.

Many metaheuristic algorithms have been applied to solve the feature selection problem. The applied algorithms are mostly binary algorithms, for example Genetic algorithm [5]. Also, there are many researchers trying to make the Differential Evolution algorithm works on binary space just to employ it as a feature selector, for example Chen et al. [6]. Moreover, Particle Swarm algorithm also has a binary version that is employed on feature selection problem [7]. Finally, this thesis tries to represent the feature selection problem in real domain not in binary domain.

The recent works on a metaheuristic approach for feature selection can be categorized into two. The first category implements the given algorithm, for example Differential Evolution, on a high-dimensional dataset, e.g. Boldt et al. [8]. The second category cares about small scale datasets, [9] being an example. In both cases, the accuracy of learning is used to compare algorithms.

1.2 Objective of the Thesis

In this thesis, we will employ one of the metaheuristic algorithms, which is Differential Evolution (DE), to work as a feature selector. Feature selection treated as a binary optimization problem in many researches. This thesis presents a real approach for the feature selection problem. The presented approach is a multi-objective optimization, which is an enhanced version of the single objective optimization. The multi-objective optimization helps in selecting few features and retaining solid learning accuracy as well.

1.3 Hypothesis

The Second Chapter can be considered the core of this thesis. It explains the framework that we used for evaluating algorithms.

The explanation in Chapter 2 starts by expressing the enhancing step on the DE algorithm. In the next step, testing of the enhanced DE is explained. The main framework and the workflow of the presented feature selection approach are explained in this Chapter 2 as well.

There is a way to test any feature selection approach which is by implementing the given approach on a benchmark of datasets. Then, the accuracy of learning will decide whether the approach succeed or failed. For that reason, testing the new approach for feature selection takes place by using group of datasets is explained in Chapter 3.

1.4 Motivation

Feature selection is one of the interesting filed, which considered as a pre-step for learning. Many approaches for applying feature selection are available and one of them is stochastic approach. When talking about the stochastic approach, it means there is a metaheuristic algorithm working as a feature selector. Implementing a binary metaheuristic algorithm, as a feature selector, is the popular approach nowadays.

Consequently, implementing a stochastic feature selection approach requires an algorithm that works in binary search space efficiently. Almost all the new and the robust algorithms work in real search space. As a result, there should be a way to apply the real metaheuristic algorithms to work as a feature selector directly. This study provides an efficient way to apply a real metaheuristic algorithms to work as a feature selector. The results of testing and comparing with the recent work in this field show how far this approach succeed.

1.5 Knowledge Discovery

Processing data is one of the main issues that we face nowadays. The amount of data we have is huge. As a result, less information can be analyzed. Therefore, there are many requests for automated systems to help humans with extracting useful knowledge from data and making it understandable.

Knowledge discovery is also called information harvesting [10], data mining [11], pattern discovery [12] and knowledge extraction [5]. Knowledge discovery tries to make the data understandable to humans [13]. As the size of data grows rapidly, the old approach, manual approach, of knowledge discovery process became a time-consuming and expensive process [14] and Medical image analysis [15] is an example. Figure 1.1 illustrates the main processes of Knowledge Discovery.

High dimensional data sets are also a problem for automated systems. Search space and the computational complexity increase exponentially by the problem dimensionality. The Naïve assumption of “more features = more knowledge” leads to a problem known as the curse of dimensionality [17]. This kind of problem occurs when irrelevant or correlated data is collected when training data is built.

1.6 Feature Selection

Feature Selection (FS), or data-reduction in general, aims to find or select minimal feature subsets out of many features in data while also taking accuracy into account [18]. When we can remove the irrelevant and the redundant features [20], data classification and text processing [21, 15] can benefit.

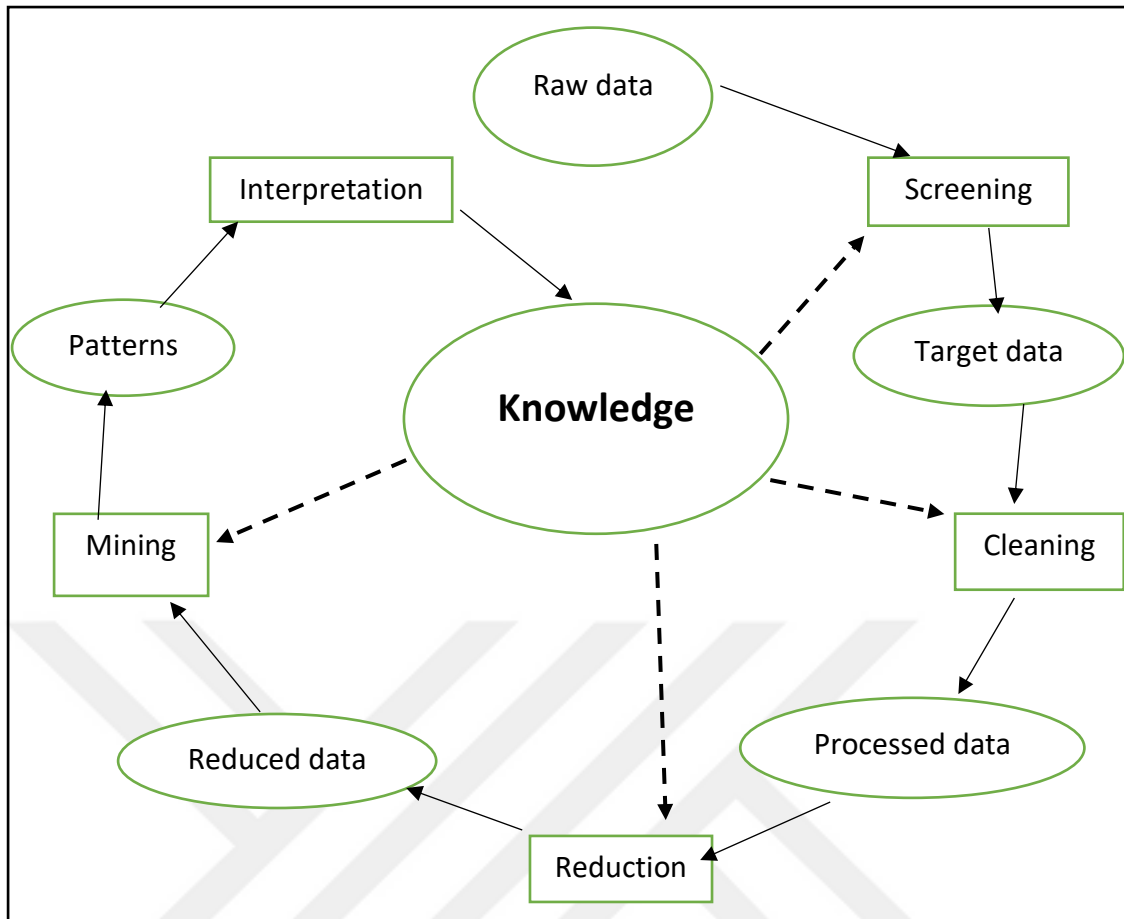


Figure 1.1 Knowledge Discovery process [16]

Feature Selection is a preparation step for data mining. We need to know the overall comparison for the following parameters to analysis and understand the importance of feature selection:

1. *Computing Time*: The result of the data - reduction process can reduce the time taken for data mining, or data learning algorithm; and
2. *Predictive/Descriptive Accuracy*: We expect by using only relevant features, the learning algorithm learn with higher accuracy;

Recent works, in developing the FS methods, focus on the whole feature subsets forming an alternative approach to the feature selection operation. Techniques such as fuzzy-rough FS (FRFS) [22], probabilistic consistency-based FS (PCFS) [23], and correlation-based FS (CFS) [24], also called population-based approach, are often collectively classified as the filter-based techniques. Typically, population-based approaches are independent of any learning algorithm. On the other hand, wrapper-based [25] and hybrid algorithms [26] are used with learning algorithm, which are employed in place of an evaluation metric as used in the filter-based approach.

Now, what is the output of a feature selection operation? The answer will be illustrated in the next four points:

1. less data which leads to learning faster;
2. higher accuracy that the model can generalize better from the data;
3. simple results that easier to understand and use; and
4. fewer features.

1.7 Feature Selection Measure

The concept of optimality for any given feature subset is multiple, which means we have to focus on two objectives. The first one is the quality: e.g. how well it encloses the information contained within the original full set of features. The second one is the size: less features are better. Due to these two objects, the feature selection process can be formulated a “multi-objective optimization” problem.

There are three approaches for modelling a feature selection operation: the first one is the Filter model, the second one is the Wrapper model, and the third one is the Embedded model [27]. The basic difference between these three models is how the algorithm evaluates the selected features. The selection of features in a Filter model is done without trying to optimize the quality of any learning algorithm directly. The Filter model is simply using an ad hoc function and implementing exhaustive search. The Wrapper model selects group of features and then evaluates the quality based on the learning algorithm [27]. This means a learning algorithm runs in each function evaluation and that is why Wrapper model considered as a time-consuming model. The Wrapper model is an example of so-called “expensive optimization”. The mechanisms of Filter and Wrapper models are given in Figures 1.2 and 1.3 respectively.

Finally, in the Embedded model, the advantages of both Filter and Wrapper models are gained. The feature search and the learning algorithm is incorporated into a single optimization problem formulation in Embedded model [27].

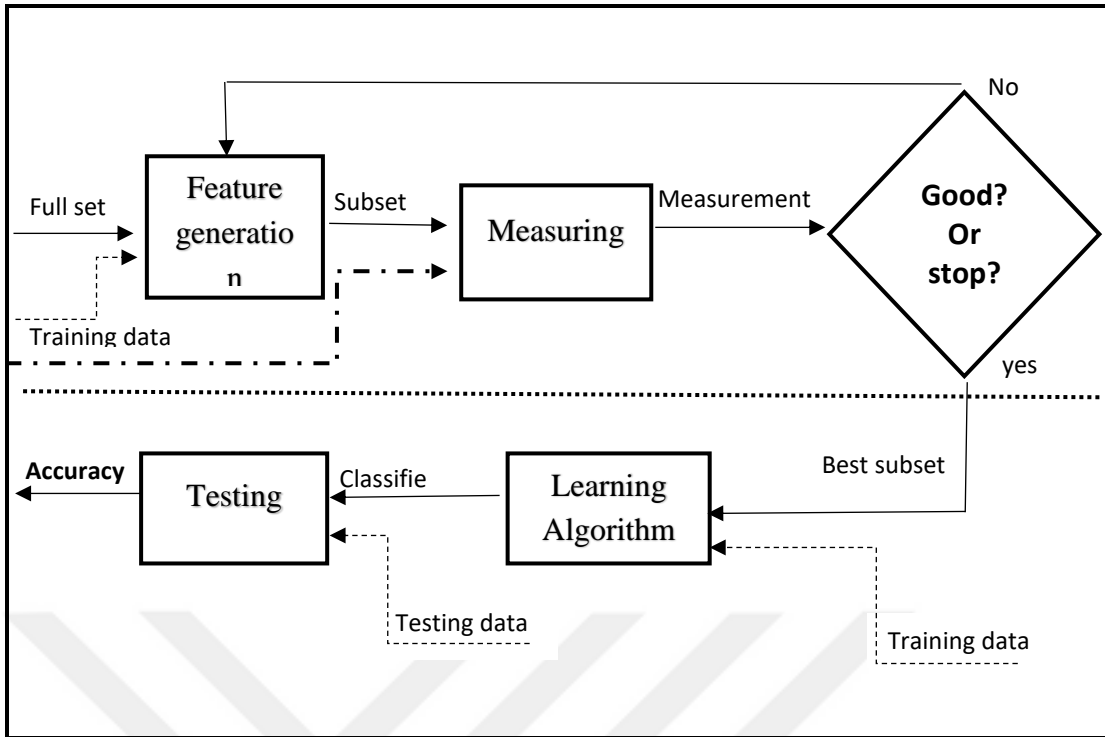


Figure 1.2 Filter model [27]

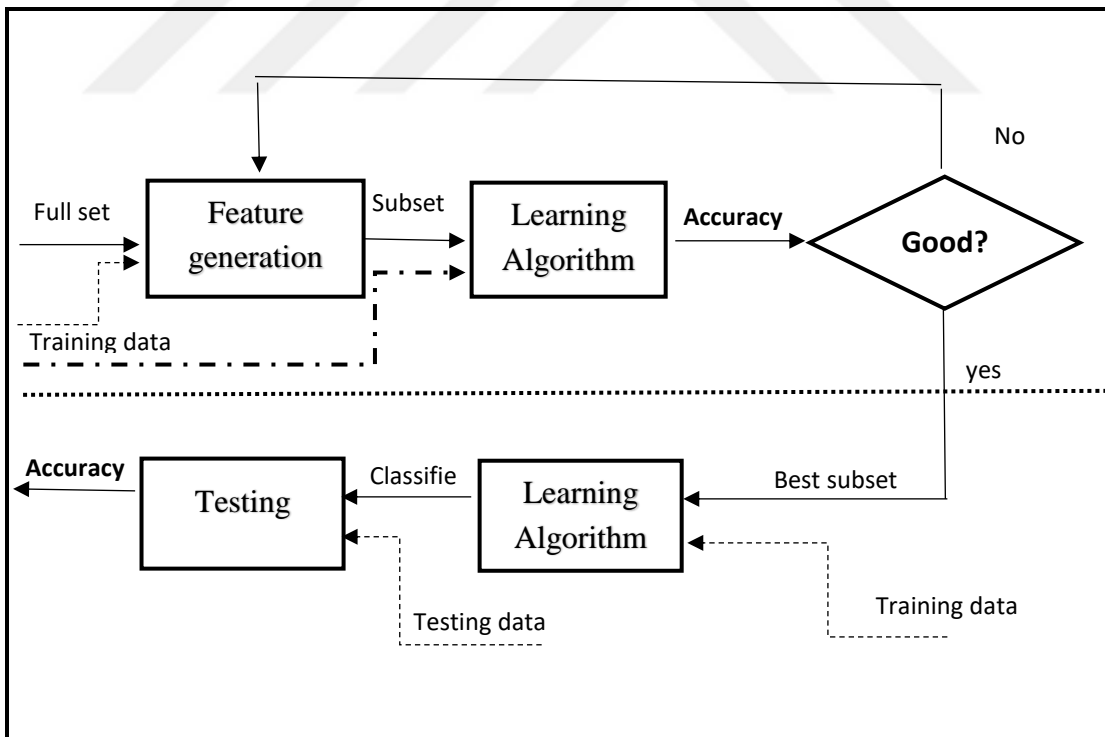


Figure 1.3 Wrapper model [27]

1.8 Differential Evolution as an Evolutionary Algorithm

Differential Evolution (DE) algorithm is a nature inspired optimization algorithm [3]. It is a population-based algorithm and it was developed for searching in continuous domains [28]. It has been implemented on many real-world problems like image processing [29], feature selection [4], and pattern recognition [30]. Figure 1.4 shows the pseudocode of the standard Differential Evolution algorithm.

We tested Differential Evolution, and its Enhanced version that we developed, on all the 15 test functions of the well-known test bed used in Congress on Evolutionary Computation 2015 (CEC2015) [31].

The performance of DE is controlled by two parameters, mutation factor (also called amplification weight factor) and crossover probability (also called Probability of Crossover Rate (PCR)) [32]. There are many types of research about how to set the parameters for DE but till now there is no distinct relation between search space and settings of DE's parameters [33].

Sometimes, during the executing, DE stops generating diverse solutions. In a way, it stops tweaking its population's individuals. This situation is called stagnation [34]. Usually, this stagnation problem happens when the algorithm is stuck in some local optima. Lou et al. [35] solves this problem by letting the parameters automatically change their value to get out of local optima.

The point of enhancing the DE algorithm is firstly to improve the performance of choosing the optimal individual. The second reason is to reduce the computational time needed when we use the DE algorithm on a Feature Selection problem. After we tested the new enhanced versions, we noticed that the enhanced version of DE is faster than the standard version. For that reason, we will employ the enhanced version for feature selection operation, and that will reduce the time of overall operation.

DIFFERENTIAL EVOLUTION ALGORITHM

P ← Is an empty desired population
popsiz ← Is the size of the population **P**
PCR = 0.7 ← Is the Crossover Rate
 α ← Is a random vector, commonly between 0.1 and 0.5
Best = 0 ← Is the best solution
FES = 1000 ← The Function Evaluation Times

- For $i = 1$ to **popsiz** do
 - P_i = new random individual
 - If $\text{Fitness}(P_i) > \text{Best}$
 - $\text{Best} = \text{Fitness}(P_i)$
- Repeat
- For **Iteration = 1** to **FES** do
 - For each individual $P_i \in P$ do
 - $A \leftarrow$ copy of a random individual
 - $B \leftarrow$ copy of a random individual
 - $C \leftarrow$ copy of a random individual
 - $T = A + \alpha (B - C)$
 - For $j = 1$ to $\text{Size}(P_i)$
 - $$X(j) = \begin{cases} T(j) & \text{if } \text{rand}[1, 0] \leq \text{PCR} \\ P_i(j) & \text{otherwise} \end{cases}$$
 - Repeat
 - If $\text{Fitness}(X) > \text{Fitness}(P_i)$ then
 - $P_i = X$ ← Replace the parent with the kid
 - If $\text{Fitness}(P_i) > \text{Fitness}(\text{Best})$ then
 - $\text{Best} = P_i$
 - Repeat
 - Until **Best** is the ideal solution or we ran out of time
 - Return **Best**

Figure 1.4 Pseudocode of standard DE [3]

1.8.1 The Initialization Step

The first step of DE is initializing the population. Every population-based stochastic algorithm starts with random numbers to fill out its population. Calculating the quality, or so-called fitness, of the candidate solutions takes place for one time in the initialization step.

Starting with good candidates improves the performance and saves computational resources [36]. However, it is almost impossible to know the best values for the initial population when trying to work with black-box optimization. Consequently, the techniques to generate uniform distributed random numbers are used to initialize the population of DE.

1.8.2 The Mutation Operation of Differential Evolution

In Mutation, DE randomly selects three candidates out of the population. Then uses these candidates to tweak the current solution, or the individual. There are different possible schemes for doing this, e.g. DE/rand/bin [40]. The Mutation operation is controlled by an amplification weight factor that lies between 1 and 0. The formula (1.1) shows the DE/rand/bin mutation scheme.

$$X_{i,G} = Vr1,G + F (Vr2,G - Vr3,G) \quad (1.1)$$

In equation (1.1), $X_{i,G}$ is the tweaked individual after the mutation operation, $Vr1,G$, $Vr2,G$, and $Vr3,G$ are vectors chosen randomly out of the population and F is an amplification weight factor.

1.8.3 The Crossover Operation of Differential Evolution

The crossover operation comes after the mutation operation. It takes the output of the mutation and the current individual as inputs parameters. The crossover operation is controlled by a parameter which is called Crossover Rate (Cr). As we see in formula (1.2), the crossover operation is simply a probability kind of operations. The new child is considered as a mixed vector. The values of the child are a mix of both the current individual and the output of the mutation operation values.

$$T(j,i,G) = \begin{cases} X(j,i,G) & \text{if } rand [1,0] \leq Cr \\ V(j,i,G) & \text{otherwise} \end{cases} \quad (1.2)$$

Where T is the tweaked individual after the crossover operation. X is the output of the mutation operation. V is the current individual (out of the population) used to generate a child by implementing the mutation and the crossover operation. And where rand [1,0] is a function that generates a random number between 0 and 1.

1.9 k NEAREST NEIGHBOR (k-NN)

k Nearest Neighbor or k-NN classifier determines the decision boundary locally [27]. If the k was equal to 1, the new sample will be assigned to its closest neighbor's class as shown in Figure 1.5 (A). Assume we have classes as in figure 1.5 (+ and -) in our training data, the new sample (?) would be labeled with the class of its closest neighbor. The (?) sample will be even + or - according to its distance from them. 1-NN classifier is not a very robust methodology [27]. When k is greater than 1, the new sample will be assigned to the majority class of its k closest neighbors. An example for $k = 4$ is given in Figure 1.5 (B). k-NN classifier for $k > 1$ is more robust, as results, larger k means the lowest noise in classification.

k-NN keeps all samples in the training set. When the new sample came to be classified then it will be compared with the values and labels that already learned from the training. This reason makes k-NN called as a memory-based learning.

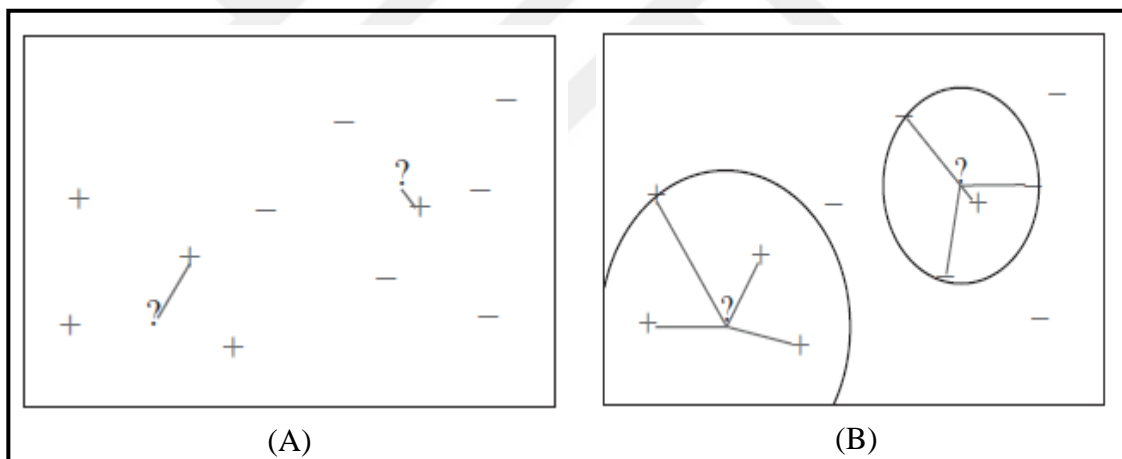


Figure 1.5 k Nearest Neighbor classifier. (A) when k equal to 1; (B) when k equal to 4 [45]

We used k-NN classifier as a fitness function, since we are trying to implement a Wrapper model for feature selection. The advantage of k-NN being fast allows us to increase the Function Evaluation Times (FES) for DE with remaining its fast work.

ENHANCED DIFFERENTIAL EVOLUTION ALGORITHM AND EVALUATIONS

In this chapter, we discuss Differential Evolution (DE) algorithm and our enhancements, our tests, and using enhanced versions of DE for feature selection. Moreover, the initialization and the optimization for the framework will be presented in details with an explanation about what is the new in our work. This chapter can be considered the core of this thesis. It explains the framework that we used for evaluating algorithms by starting with expressing the enhancing step on the DE algorithm. In the next step, testing of the enhanced DE is explained. The main framework and the workflow of the presented feature selection approach are explained in this chapter as well.

2.1 Background About Feature Selection

As the amount of available data increases, the need for efficient technologies for dimensionality reduction increased. Feature Selection (FS) methods achieve this by finding minimal or close-to-minimal feature subsets. If the algorithm can maintain the basic data semantics after feature selection process, the reduced data may be transparent to human scrutiny. General speaking, FS involves two computational processes: Feature subset evaluation process and Feature subset search process.

The studies in the literature [41] break down FS into smaller sub processes. Two of these sub processes are feature subset generation (as we will discuss in Section 2.5.2) and termination (as discussed in Section 2.5.4). Exceptional cases [42] exist where the two parts are indifferentiable or inseparable.

2.2 The Enhanced Differential Evolution (EDE)

The enhancing on DE has been made on the mutation process to gain a fast convergence. Since searching around the best solution leads to find the optimal solution, fast convergence is achievable by selecting the right individual in each iteration. In EDE, the best solution is tracked. If the best solution did not change for 10 times, the mutation scheme will be changed to DE/best/bin. And also, the crossover of the DE will be changed from the normal one to the Two Points Crossover. In that way, the new child will be a mix of some features (or genes) from the best solution and also some of its own features. As a result, the desired change in the individual will be done.

ENHANCED DIFFERENTIAL EVOLUTION ALGORITHM

```

P          ← Is an empty desired population
popsize   ← Is the size of the population P
α         ← Is a random vector, commonly between 0.1 and 0.5
Best = 0   ← Is the best solution
FES = 1000 ← The Function Evaluation Times
• For i = 1 to popsize do
  ➤ Pi = new random individual
  ➤ If Fitness(Pi) > Best
    ○ Best = Fitness(Pi)
• Repeat
• For Iteration = 1 to FES do
  • For j = 1 to popsize do
    ➤ A ← copy of a random individual
    ➤ B ← copy of a random individual
    ➤ C ← copy of a random individual
    ➤ If I > 11 && Best(Iteration-1) = Best(Iteration-11)
      ○ T = Best(Iteration-1) + α (B-C)
    ➤ Else
      ○ T = A + α (B-C)
    ➤ R1 = rand [1, Size(Pj)]
    ➤ R2 = rand [1, Size(Pj)]
    ➤ X = Pj
    ➤ X(from R1 to R2) = T(from R1 to R2)
    ➤ If Fitness(X) > Fitness(Pj) then
    ➤ Pj = X ← Replace the parent with the kid
    ➤ If Fitness(Pj) > Fitness(Best(i-1)) then
      ○ Best(i) = Pi
  • Repeat
• Until Best is the ideal solution or we ran out of time
• Return Best

```

Figure 2.1 Pseudocode of the EDE algorithm, Bold lines represent the changes

The Two Points Crossover means choosing two random points on the vectors. Then, exchange the features that lies inside the two point of the two vectors. Figure 2.1 shows the pseudocode of the EDE and shows how the two mutation operations take place in the main procedure.

2.3 Testing the Enhanced and the Standard Versions of Differential Evolution

The performance of the standard and the enhanced versions of DE are tested by using the fifteen computationally expensive single objective functions, which are presented in the Congress on Evolutionary Computation 2015 [31]. Two of the functions are unimodal, three of them are simple multimodal, other three of them are hybrid, and the rest are composition functions. Table 2.1 illustrates the fifteen function with their range of search and the global optima.

Table 2. 1 Functions of the CEC2015

| Group No. | No | Function name | Range | F_i^* |
|-----------------------------|----|--|------------|---------|
| Unimodal Functions | 1 | Rotated High Conditioned Elliptic Function | [-100,100] | 100 |
| | 2 | Rotated Cigar Function | [-100,100] | 200 |
| Simple Multimodal Functions | 3 | Shifted and Rotated Ackley's Function | [-100,100] | 300 |
| | 4 | Shifted and Rotated Rastrigin's Function | [-100,100] | 400 |
| | 5 | Shifted and Rotated Schwefel's Function | [-100,100] | 500 |
| Hybrid Functions | 6 | Hybrid Function 1 (N=3) | [-100,100] | 600 |
| | 7 | Hybrid Function 2 (N=4) | [-100,100] | 700 |
| | 8 | Hybrid Function 3(N=5) | [-100,100] | 800 |
| Composition Functions | 9 | Composition Function 1 (N=3) | [-100,100] | 900 |
| | 10 | Composition Function 2 (N=3) | [-100,100] | 1000 |
| | 11 | Composition Function 3 (N=5) | [-100,100] | 1100 |
| | 12 | Composition Function 4 (N=5) | [-100,100] | 1200 |
| | 13 | Composition Function 5 (N=5) | [-100,100] | 1300 |
| | 14 | Composition Function 6 (N=7) | [-100,100] | 1400 |
| | 15 | 15 Composition Function 7 (N=10) | [-100,100] | 1500 |

We tested our work by setting the solution dimensionality to 10D and allowed 500 function evaluations (FES), as we are simulating an expensive optimization (e.g. as if each function evaluation is the result of a classification experiment).

The results compare the best, standard deviation, time and median for the two algorithms as shown in Table 2.2. The experiment results are for 30 times with random

initialization. That means the best, standard deviation, and median values in Table 2.2 are an average of 30 iterations with random start points.

Table 2. 2 Results of comparing the stander DE algorithm and the Enhanced DE algorithm using 10 dimensions, 20 individuals, and 500 function evaluations on CEC 2015 test bed. Smaller values are better.

| Function No. | Algorithm Name | Mean | Standard Deviation | Median | Time in second |
|--------------|----------------|-----------------|--------------------|-----------------|----------------|
| F1 | DE | 18256110 | 52481237 | 452535.9 | 2.686 |
| | EDE | 12940039 | 48353826 | 251833.1 | 2.544 |
| F2 | DE | 13975.83 | 7670.185 | 12410.56 | 2.700 |
| | EDE | 10741.48 | 6147.09 | 9289.593 | 2.689 |
| F3 | DE | 302.0908 | 1.694143 | 301.8833 | 3.723 |
| | EDE | 301.6306 | 1.355487 | 301.4214 | 3.599 |
| F4 | DE | 481.1714 | 104.5238 | 424.9232 | 2.775 |
| | EDE | 449.0032 | 61.02273 | 409.8091 | 2.563 |
| F5 | DE | 501.5076 | 0.311221 | 501.4579 | 3.646 |
| | EDE | 501.417 | 0.259725 | 501.4118 | 3.202 |
| F6 | DE | 600.2274 | 0.215809 | 600.1958 | 2.692 |
| | EDE | 600.1949 | 0.090044 | 600.1667 | 2.670 |
| F7 | DE | 700.4289 | 0.574207 | 700.2947 | 2.626 |
| | EDE | 700.3191 | 0.139404 | 700.2862 | 2.596 |
| F8 | DE | 801.6431 | 0.672089 | 801.6779 | 2.829 |
| | EDE | 801.4711 | 0.61487 | 801.3187 | 2.613 |
| F9 | DE | 902.8359 | 0.397755 | 902.8248 | 2.800 |
| | EDE | 902.8456 | 0.38164 | 902.9181 | 2.574 |
| F10 | DE | 5170.448 | 3803.941 | 3845.374 | 2.768 |
| | EDE | 4469.002 | 2987.391 | 3909.215 | 2.761 |
| F11 | DE | 1102.974 | 1.126325 | 1103.076 | 2.883 |
| | EDE | 1102.561 | 0.893995 | 1102.228 | 2.958 |
| F12 | DE | 1240.118 | 11.90536 | 1238.472 | 2.819 |
| | EDE | 1238.064 | 14.23116 | 1232.238 | 2.845 |
| F13 | DE | 1526.978 | 13.06892 | 1522.065 | 3.044 |
| | EDE | 1522.927 | 11.67048 | 1518.95 | 2.796 |
| F14 | DE | 1585.507 | 29.45875 | 1600.66 | 3.140 |
| | EDE | 1568.094 | 37.17384 | 1600.231 | 2.674 |
| F15 | DE | 1868.769 | 36.50787 | 1873.231 | 4.149 |
| | EDE | 1857.414 | 40.12524 | 1861.926 | 3.962 |

As obvious in Table 2.2, since the problems are minimization problems, the algorithm with the lowest mean value is better than the others. No algorithm seems to be the best all the time. The results imply that in general 6.67% of the time, DE performs better than the EDE in terms of time and the optimization result. In addition, 93.3% of the time EDE performs better than the DE in terms of optimization result.

Table 2.2 shows that in Unimodal functions (f1 and f2), in Simple Multimodal Functions (f3, f4, and f5), and in Hybrid Functions (f6, f7, and f8), the EDE was better than the standard DE in both time and optimization results.

Finally, for composition functions (f9, f10, f11, f12, f13, f14 and f15), the time is nearly uniformly distributed between EDE and DE. Overall, it can be deduced that EDE performs better than the standard DE. EDE can reach the optimal and it requires less computation time most of the time.

2.4 Enhanced Differential Evolution Feature Selector (EDEFS)

Feature selection is a slow process so it needs a fast algorithm to reduce the executing time. By testing the performance of the two versions of DE algorithm, EDE showed good results with less computation time. Therefore, applying the EDE as a feature selector is more suitable than applying DE.

The EDEFS is a new FS approach based on EDE, a simple and powerful optimization technique. It guarantees a good accuracy of learning with high dimensional datasets. The flexibility of DEDEFS allows to use any fitness function such as CFS, PCFS, and FRFS. Also, it can use any given classifier accuracy or error rate as a fitness of the given subset feature. According to the enhancements in the EDE, EDEFS can escape from the local optima and can find multiple feature subsets quality.

For representing the data, each individual in the population represents an index vector. The index vector is a vector which its values represent the location of the features in the raw data. We only save the best individual and its fitness value in every iteration when we use the basic DE. In several cases, there will be two different feature subsets with the same accuracy and different number of features. Since we are searching for less number of features with high accuracy, we assume that high accuracy and less number of features is better than the other one. By holding the best accuracy for each individual, we will be able to choose the subset with few features and high accuracy.

For optimizing the population size of EDE, we tested the performance of EDEFS with a slight change in population size in each time. The testing started with 10 individuals...till 50 individuals. The results showed that only 20 individuals are enough to get acceptable accuracy.

For optimizing the FES for the EDEFS, when we were testing the performance of DE, and EDE, we have seen the results of EDE converge at the 50th FES for 7 functions. The Figures 2.2to2.16 shows the performance of EDE with 20 individuals and 500 FES on all the 15 functions in CEC2015.

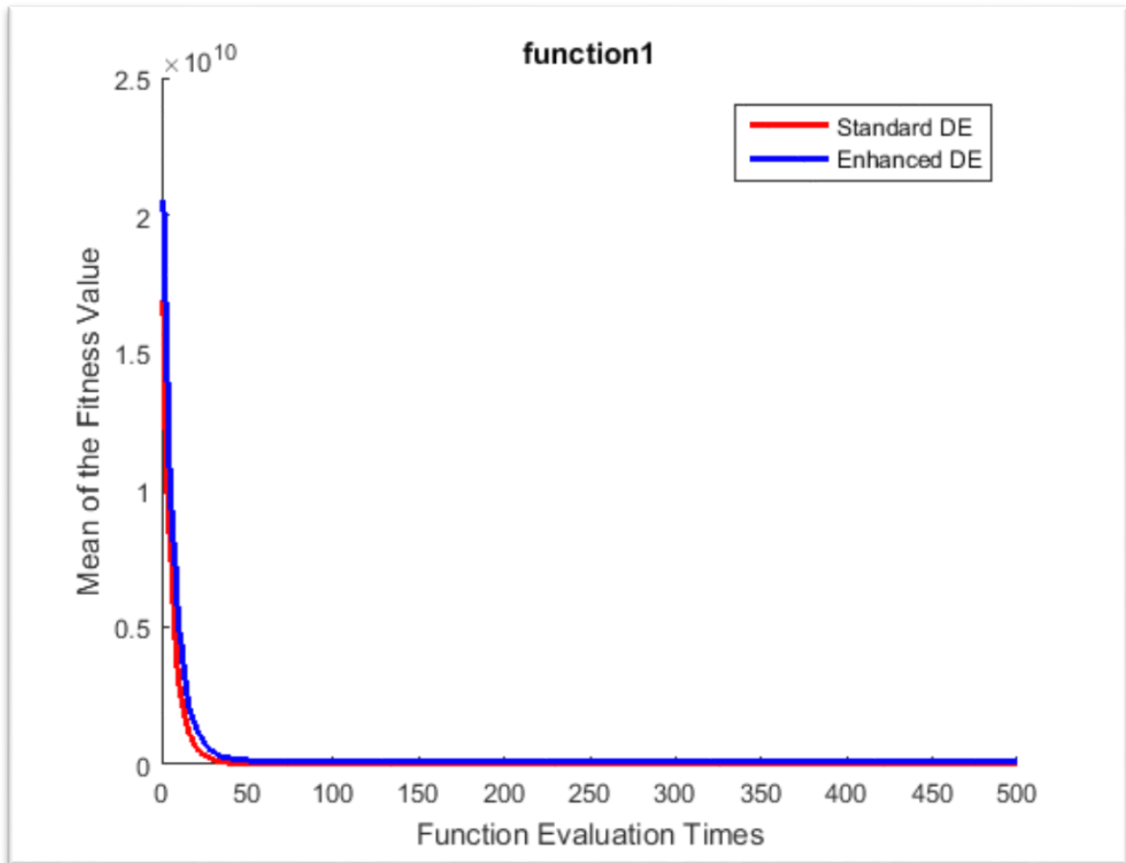


Figure 2.2 Results of Function 1

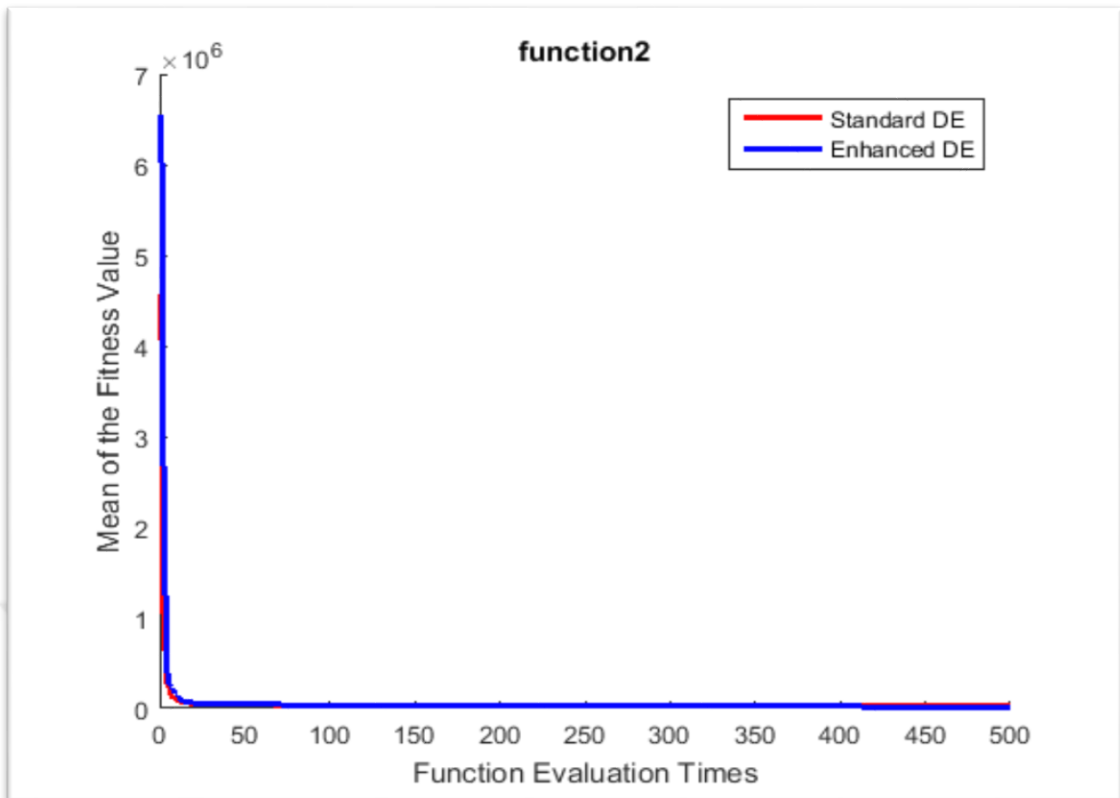


Figure 2.3 Results of Function 2

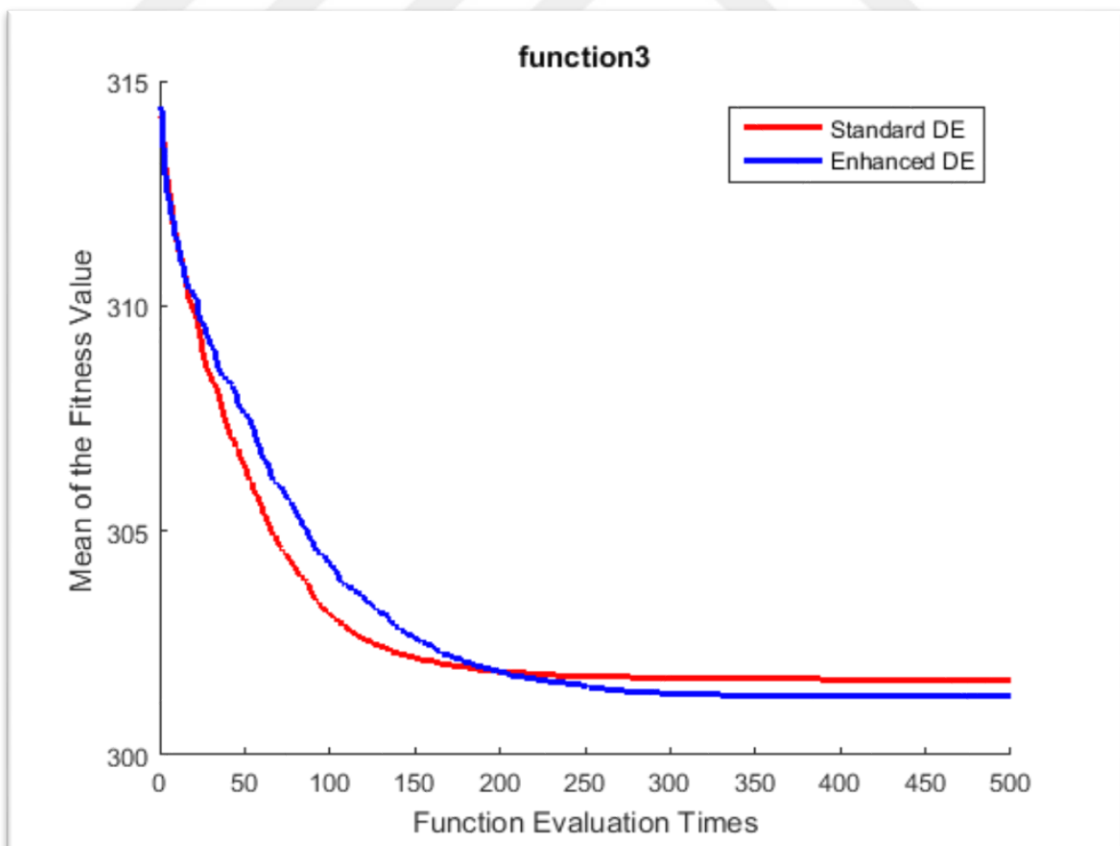


Figure 2.4 Results of Function 3

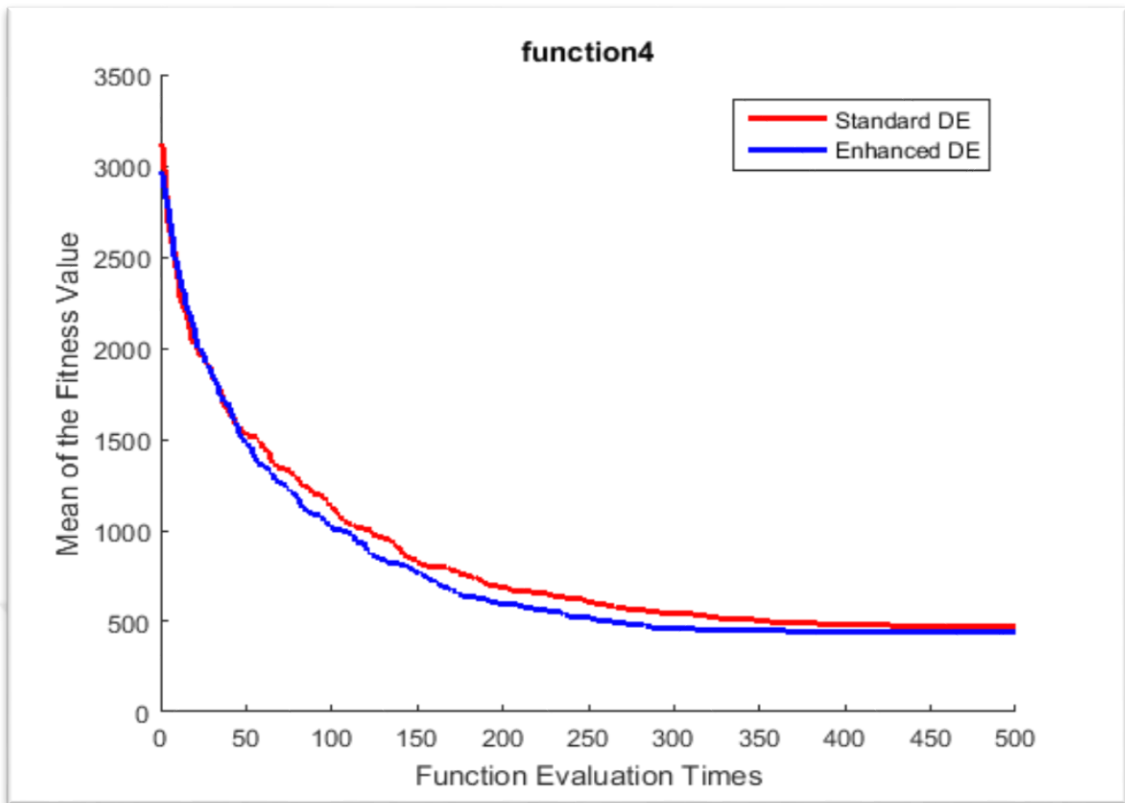


Figure 2.5 Results of Function 4

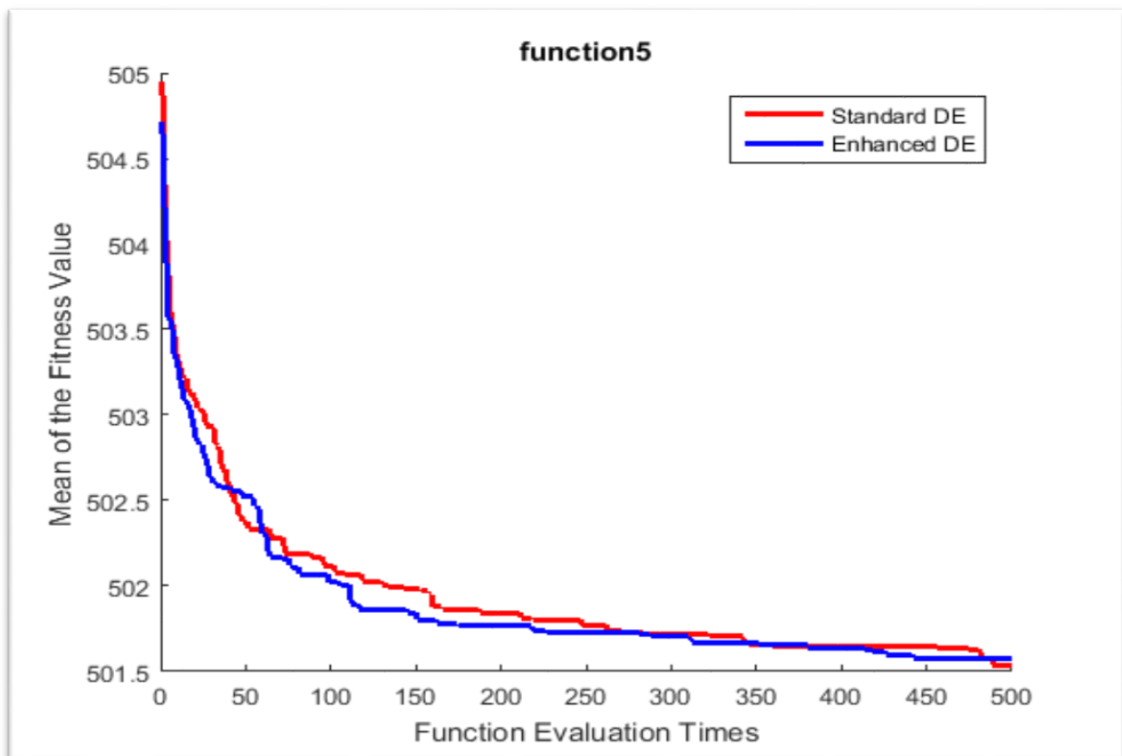


Figure 2.6 Results of Function 5

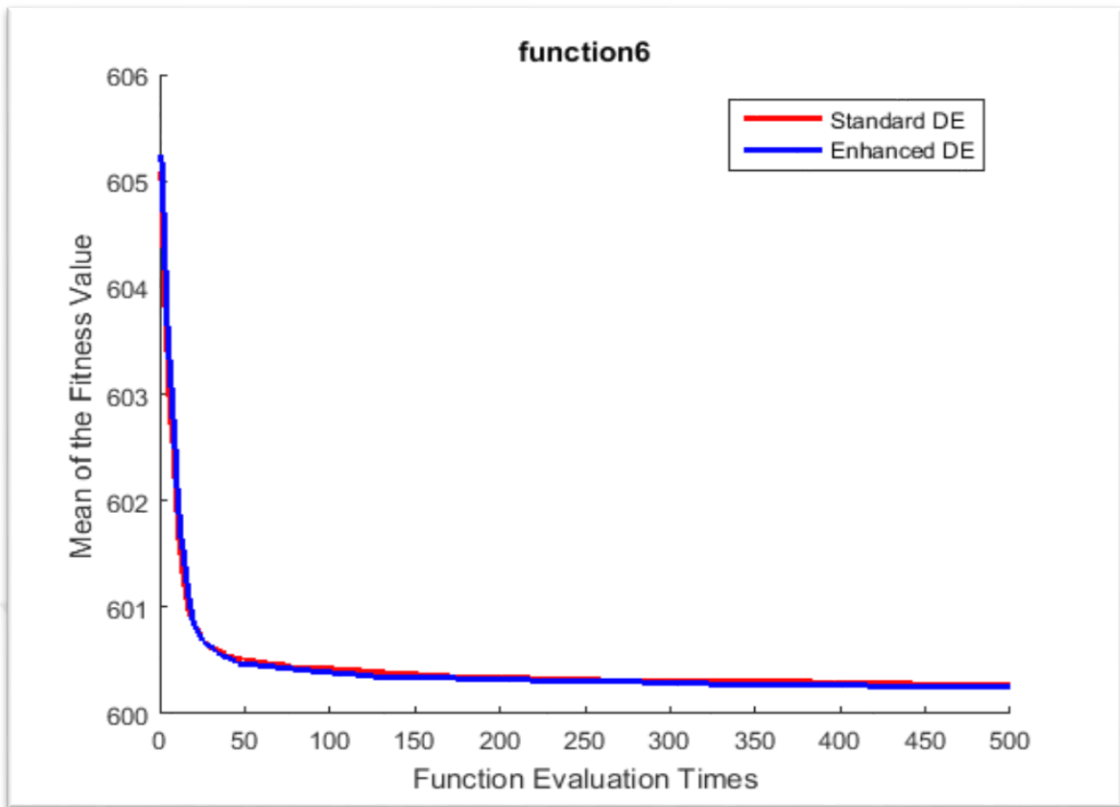


Figure 2.7 Results of Function 6

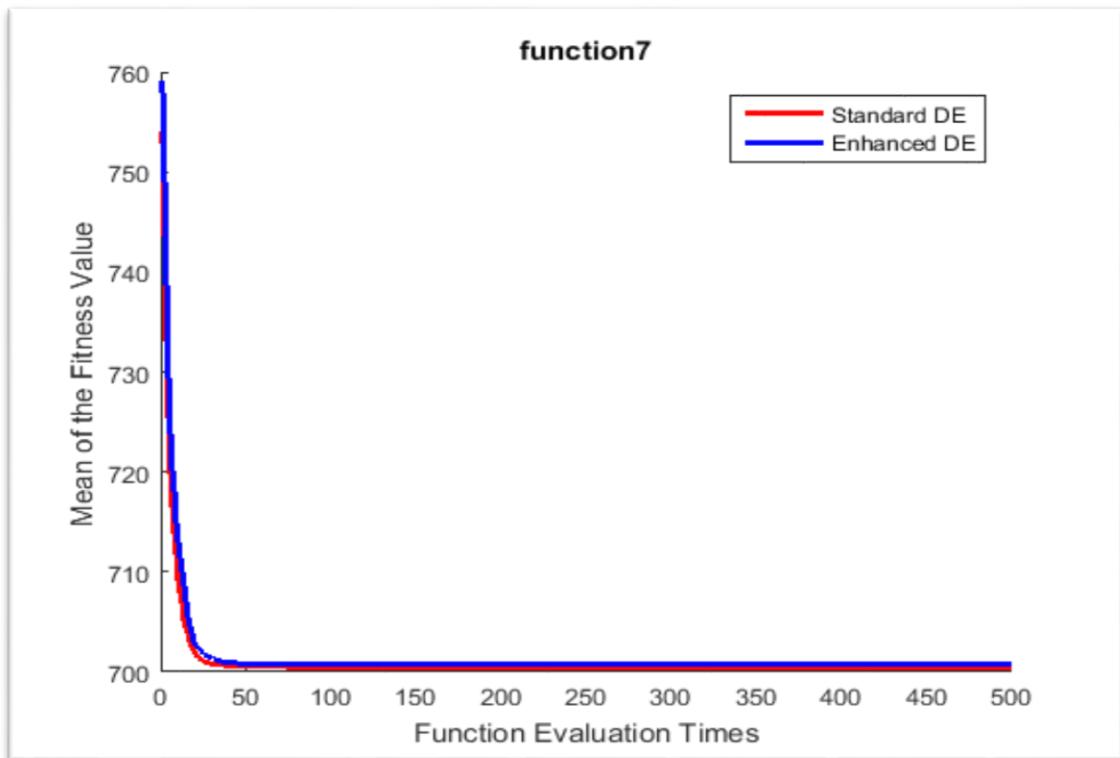


Figure 2.8 Results of Function 7

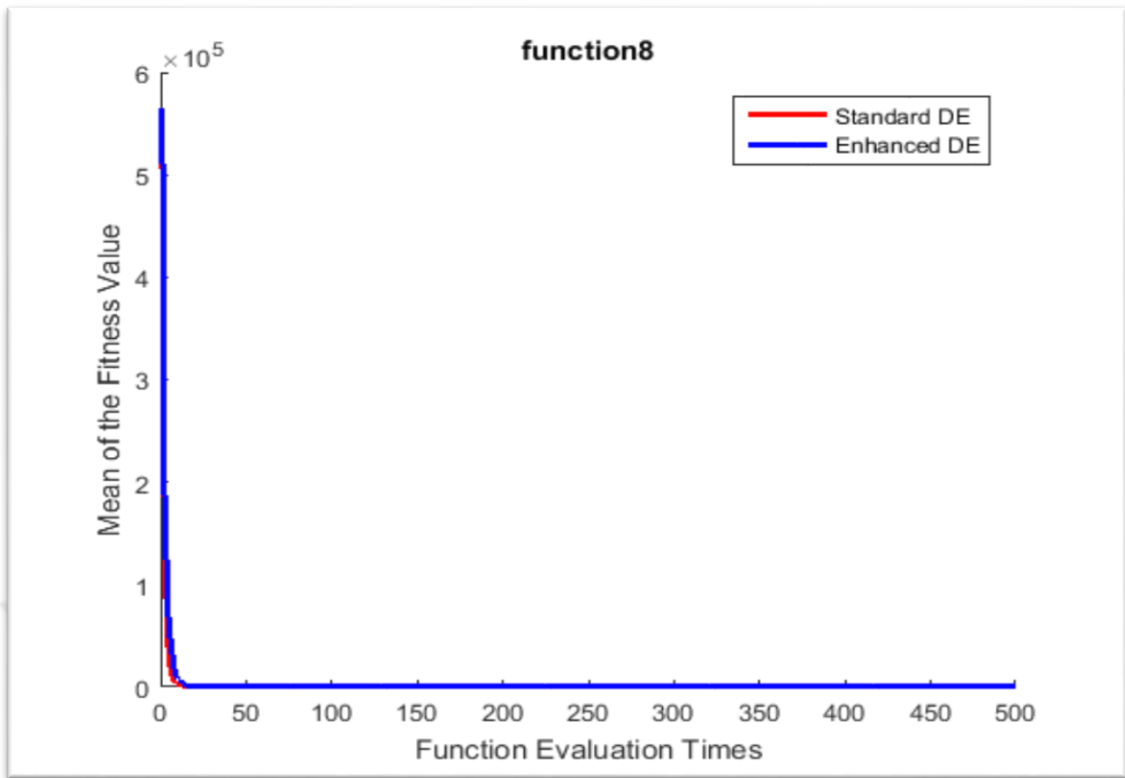


Figure 2.9 Results of Function 8

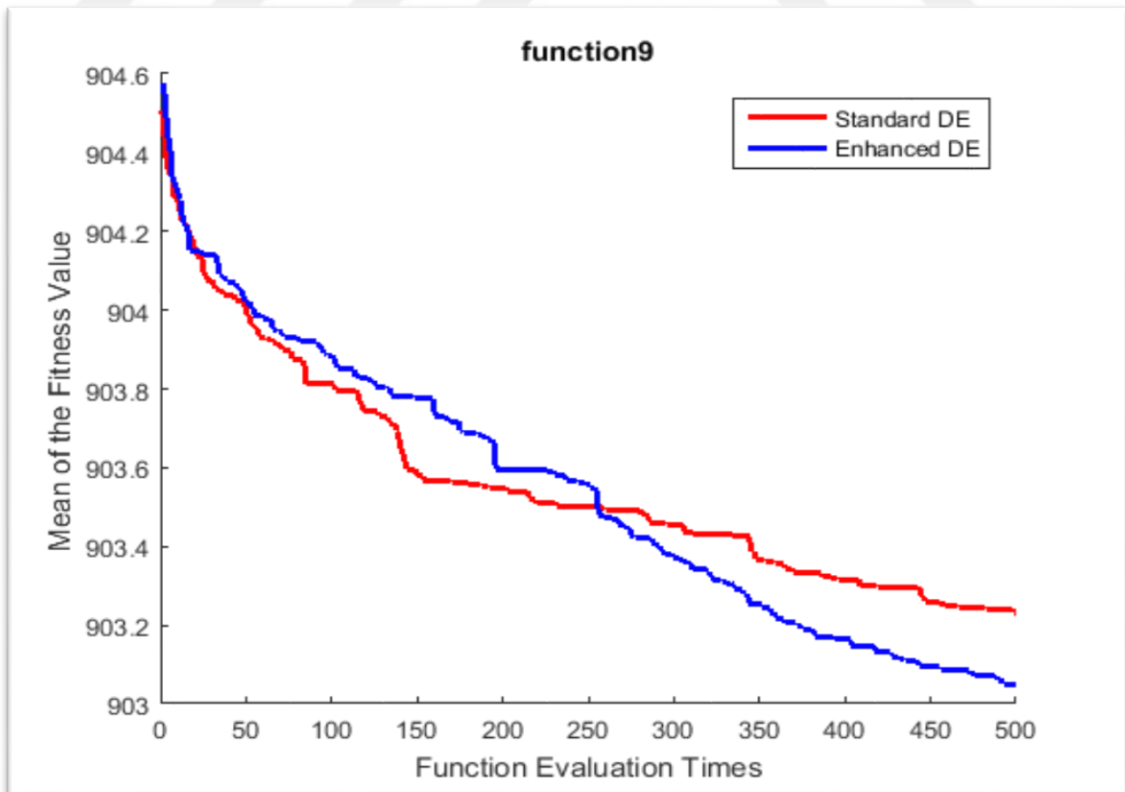


Figure 2.10 Results of Function 9

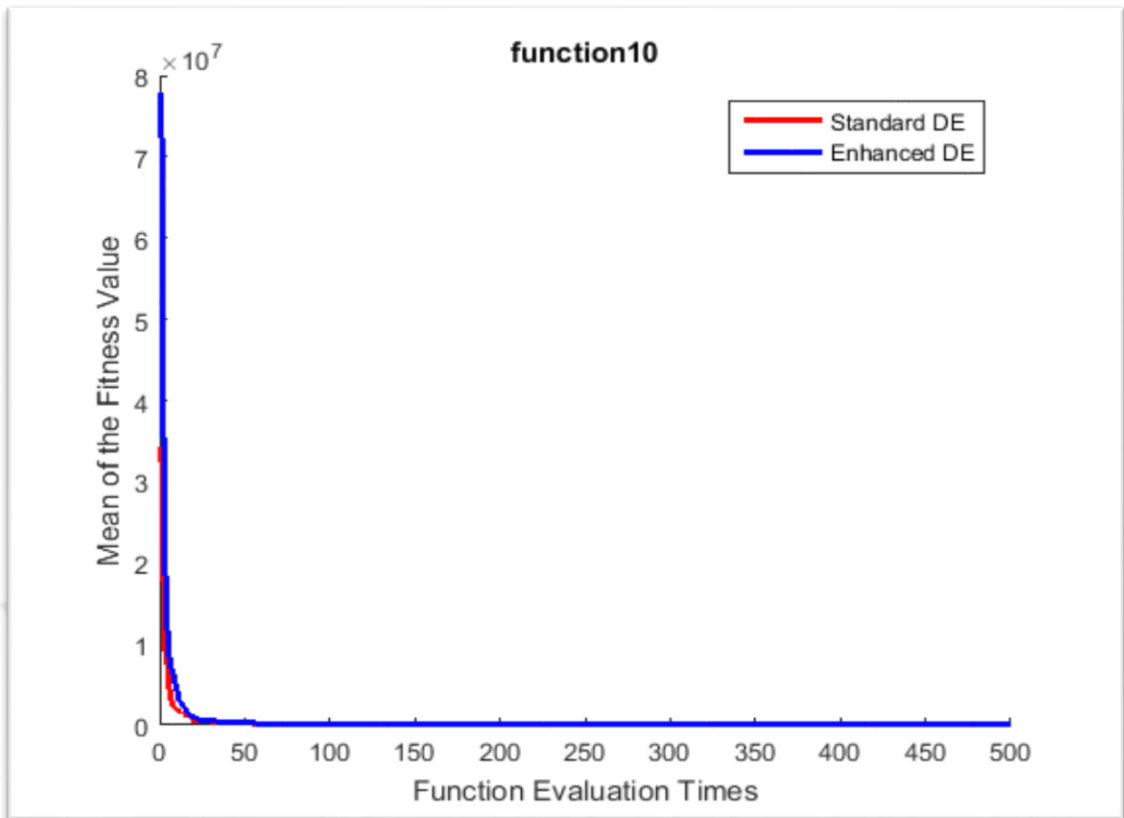


Figure 2.11 Results of Function 10

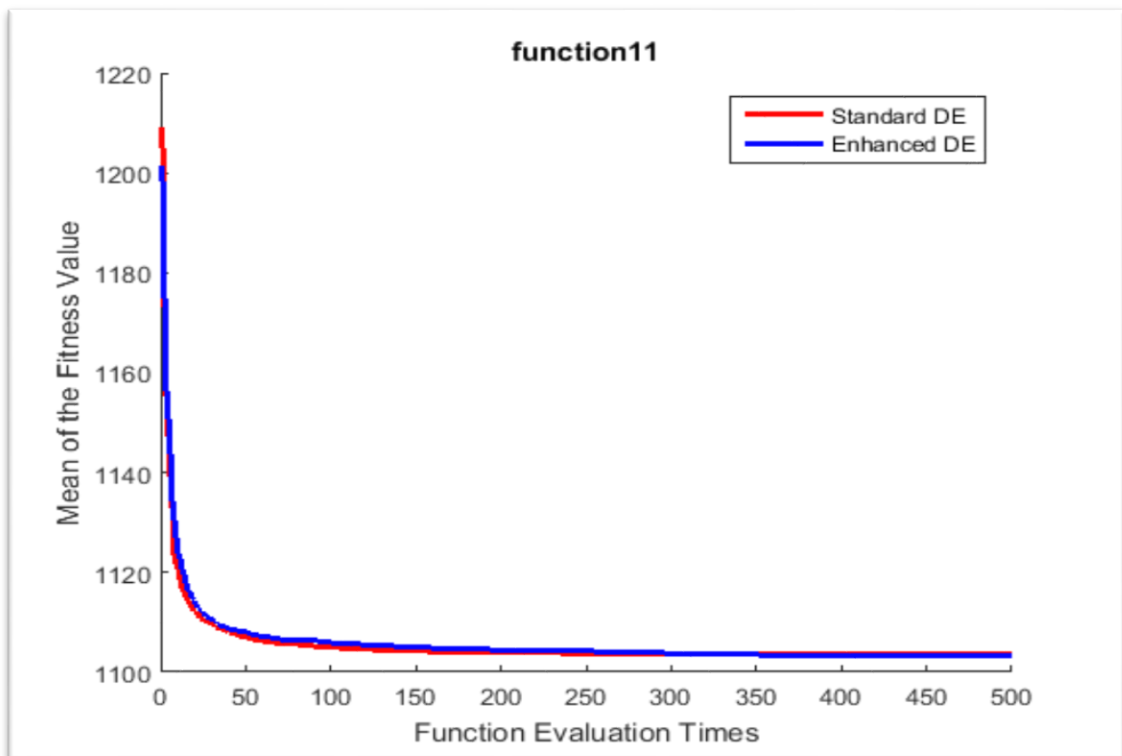


Figure 2.12 Results of Function 11

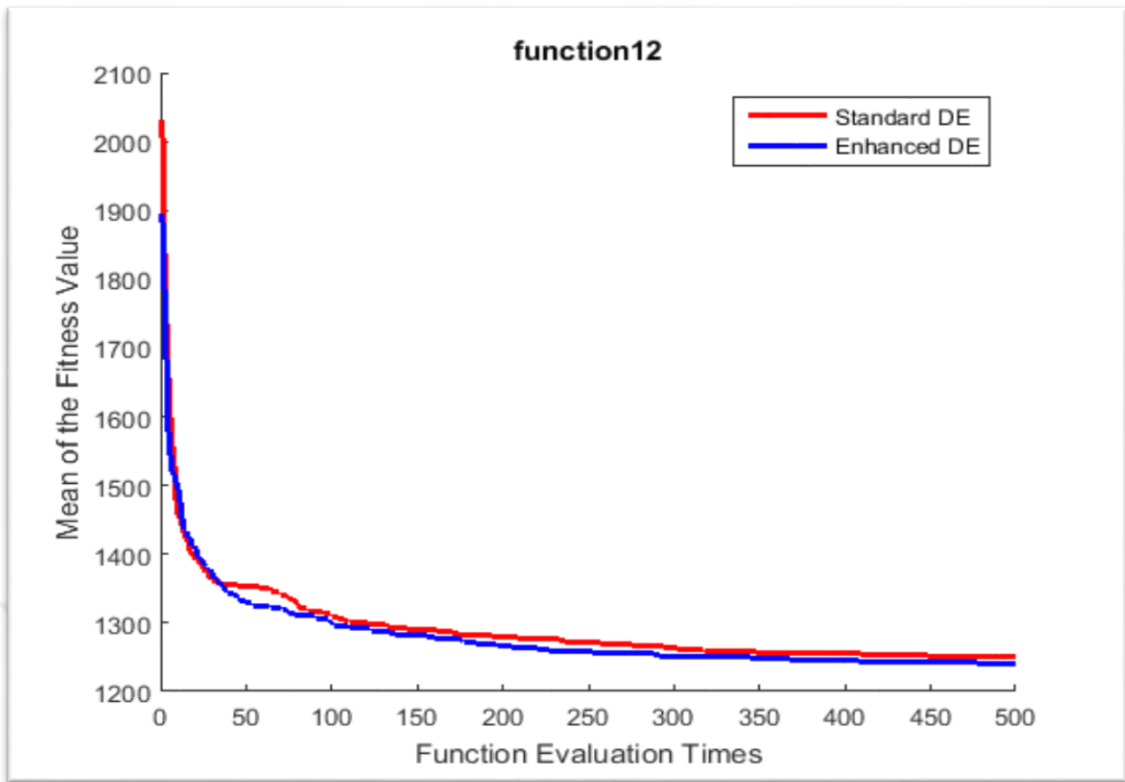


Figure 2.13 Results of Function 12

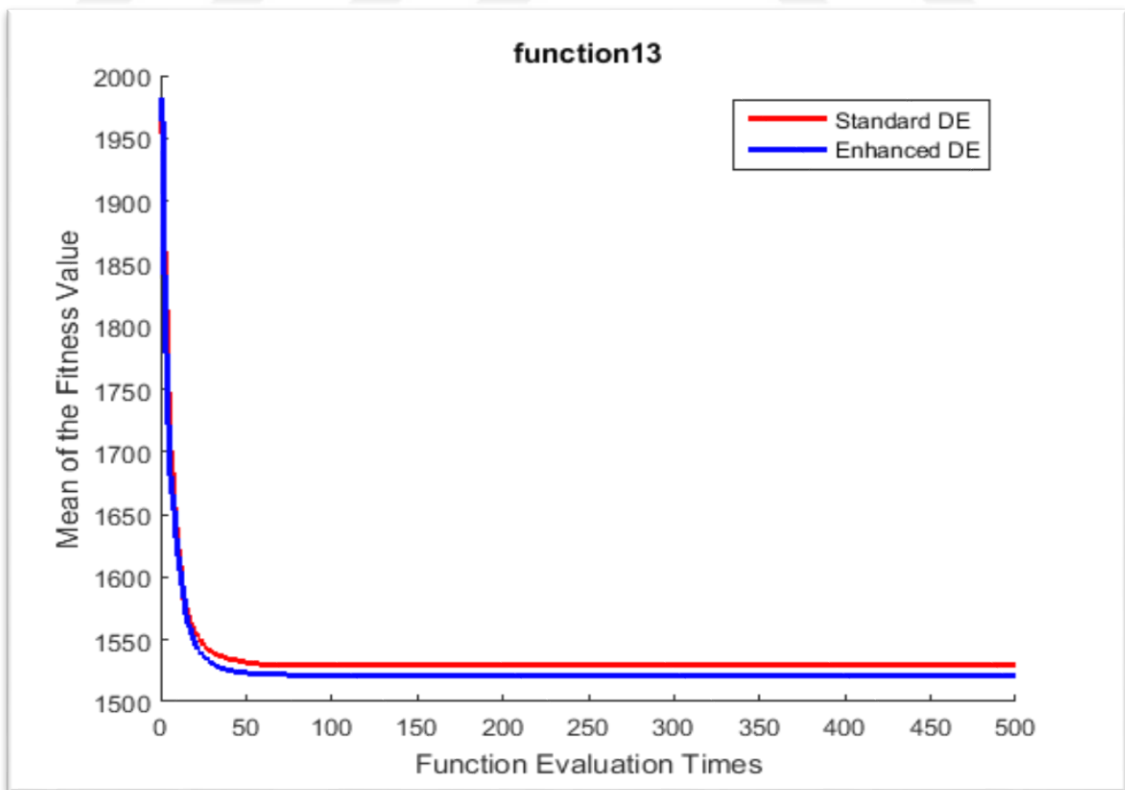


Figure 2.14 Results of Function 13

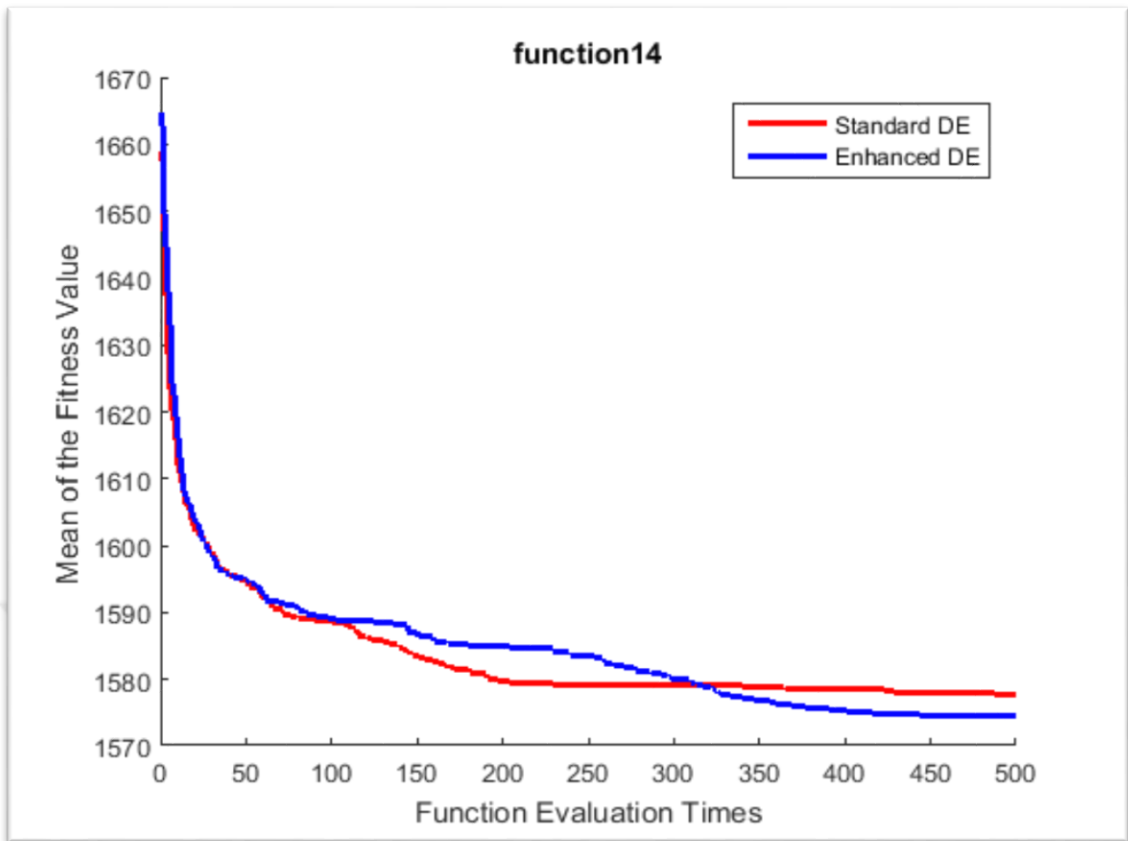


Figure 2.15 Results of Function 14

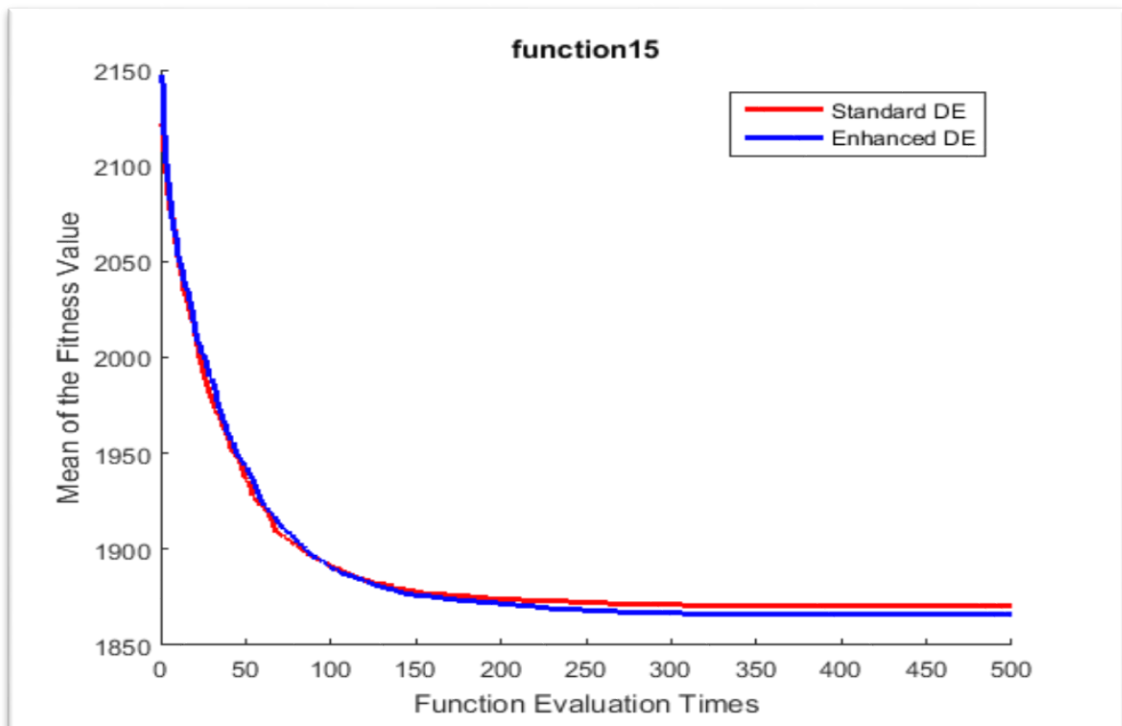


Figure 2.16 Results of Function 15

As we demonstrated in Figures 2.2...2.16, the EDE gets stable at the 50th FES. We also notice that EDE was better than the basic DE, as we have seen in Table 2.2. Now it became clear that EDE got that preference by using only fifty FES out of its 500 FES! The results tell us to set the number of iteration of EDEFS to only 50.

2.4.1 Key Notions

In this section, we will explain what the parameters of EDE means in our EDEFS approach.

Number of elements on an individual solution, or individual's size, is equal the number of features in the raw data. The number of individuals in the EDE represents the number of cases that our approach will find (assuming we have 20 individuals, we get 20 cases). The Best individual represents the highest accuracy with N number of features.

We also assigned two variables to every individual in the EDE. The first variable holds the best values of the given individual along the process lifecycle. The second one will hold the accuracy (the fitness) of the value in the first assigned variable. In this way, we will have a number of best values equal to the population size.

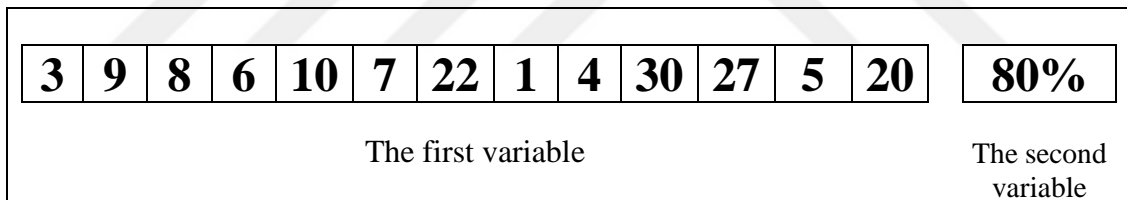


Figure 2.17 Example of the assigned variables to each individual in population

Note: The size of the first variable is not fixed. We will see how that size changes on every iteration during its lifecycle in Section 2.4.3. Just to make it more clear, the reason behind changing the size is to get as fewer features as possible.

2.4.2 Initialization

This section talks about the first step of the EDEFS lifecycle. We will also talk about what is the new with our approach and how it helps.

As we mentioned in Section 1.8.1, in the initialization step, the solutions should be filled with random numbers. Many metaheuristic approaches for Feature selection depend on representing the features with a binary vector. In this approach, the “1”

values mean that features are participating in the selected subset, whereas the “0” means the feature is not included in the selected subset.

The convergence rate with binary approach weaker than if it was in real [6]. The new idea in our approach is to not make the values in the representing vector zeros and ones. We initialize the EDE with a real number between any two boundaries. In that way, the mutation and the crossover operations are going to work better. For these reasons, we worked on making the EDEFS works with real values rather than binary.

Firstly, each individual in the population is filled with random values between any given boundaries. For example, between 5 and -5. Then the fitness of every individual will be calculated and saved in the second assigned variable. The first assigned variable of the given individual will be holding the indexes values of the initializing step. In each iteration of the initialization, the best value will be checked and saved in a variable which is called BEST.

The indexes values will be explained clearly in Section 2.4.3 and we will see how it plays a good role in our work.

Note: the first and the second assigned variables have been explained in details in Section 2.4.1 and Figure 2.17 shows an example of their values.

2.4.3 Work Flow

In this section, we will express the workflow of EDEFS in details to make it easy to understand. We will make the explanation in steps to make it easier to read. Also, we will provide samples of all the variables in our work at several steps so the reader can follow the changing in values step by step. Let us take the Wine dataset (which has 13 features, 178 instances, and 3 classes [43]) as an example and try to implement the EDEFS on it. Figure 2.18 demonstrates the pseudocode of the EDEFS algorithm.

```

Function Fitness(X) // function to calculate the accuracy of learning
{
D = Data(:,X) // extract only the desired features
Training=10-fold(D) // split the data into train and test by using 10-fold method
Testing=10-fold(D)
Accuracy=kNN(Training , Testing) // calculate the accuracy of learning using kNN
Return (Accuracy)
}
popsize = 20
The parents size = number of features in the given dataset (Data)
For i =1 to 20 do
    P(i)= rand([1,number of features in the given dataset (Data)],[-5,5])
Repeat
For i = 1 to 50 do
    For j =1 to popsize do // 20 parents chosen for popsize
        A ← copy of a random individual
        B ← copy of a random individual
        C ← copy of a random individual
        If i >=11 && Best(i-1) = =Best(i-11)
            T= Best(i-1) +  $\alpha$  (B-C)
        Else
            T = A +  $\alpha$  (B-C)
        R1 = rand [1, Size(P( j ))]
        R2 = rand [1, Size(P( j ))]
        X=P( j )
        X(from R1 to R2) = T(from R1 to R2)
        [~,Y] = sort(X) // takes the indexes of the new child after sorting it
        Z= Y (from 1 to rand [1, Size(P( j ))]) // take the values from the first
                                                    till a random value

        If Fitness(Z) > Fitness(P(j))
            P(j) = Z // Replace the parent with the kid
            If Fitness(P(j)) > Best(i-1) then
                Best(i) = P(j)

    Repeat
Repeat
Return (Best)

```

Figure 2.18 The pseudocode of the EDEFS algorithm

The First step: the population of EDE, the twenty individuals, will be filled randomly between any two given boundaries. We used -5 and 5 as the two boundaries to the initialization step. In this step, the values of every individual will be assigned using continuous uniform distribution.

Since the number of features in Wine dataset is 13, then the individual's size will be 13 as well, as we explained that in Section 2.4.1. Figure 2.18 gives an example of four individuals, out of twenty individuals in our population, values after the initialization step.

| | | | | | | | | | | | | |
|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| -0.13 | -0.64 | -0.53 | -1.93 | 0.085 | 0.107 | 3.176 | 2.948 | 1.443 | -1.21 | 3.115 | 0.328 | -1.49 |
| 3.243 | 4.826 | 2.302 | -1.56 | 0.840 | -3.92 | 4.063 | 3.796 | 3.177 | -2.39 | 0.943 | -4.77 | -0.74 |
| 3.147 | 4.057 | -3.73 | 4.133 | 1.323 | -4.02 | -2.21 | 0.468 | 4.575 | 4.648 | -3.42 | 4.705 | 4.571 |
| 2.688 | -3.32 | 3.619 | 4.898 | 0.144 | 3.842 | 0.880 | -3.45 | -3.00 | -0.93 | 2.487 | 3.255 | 2.899 |

Figure 2.19 Four individual's values after the initialization step

In this step, the fitness values for every individual will be calculated and saved in the second assigned variable. Also, the best value will be calculated and saved in the BEST variable.

The Second step: after the initialization step, we will get into the main EDE loop. Every individual in our population will pass through the mutation and the crossover operations. The output of the mutation and the crossover operation will represent a new child. We will sort the values of the new child. Then we will take the indexes of the sorting operation.

The output of the sorting operation represents the indexes of the individual values after the sorting operation. Figure 2.19 shows an example of the values of an individual and the output after implementing the sorting operation.

| | | | | | | | | | | | | |
|---|-------|-------|-------|------|------|------|------|------|-------|------|-------|-------|
| -0.13 | -0.64 | -0.53 | -1.93 | 0.08 | 0.10 | 3.17 | 2.94 | 1.44 | -1.21 | 3.11 | 0.328 | -1.49 |
| An individual's values / Input vector | | | | | | | | | | | | |
| 4 | 13 | 10 | 2 | 3 | 1 | 5 | 6 | 12 | 9 | 8 | 11 | 7 |
| The output of the sorting operation / Output vector | | | | | | | | | | | | |

Figure 2.20 Input and Output of the sorting operation

After taking the output vector of the sorting operation, the output represents the indexes of the features in the Wine dataset. Which means, the 4 value (in the output vector in Figure 2.20) represents the 4th feature in the Wine dataset and the 13 value represents the 13th feature...etc.

The Third step: The indexes of the features are available in this step. Now, we are about to decide how many features we should pick. If we want our EDEFS to select fixed number of the features, for example N, then we should pick N number of values out of the indexes vector in each iteration. For example, assume N=5 then the values 4,13,10,2, and 3 (of the output vector in Figure 2.20) will be selected.

As we previously mentioned in Section 1.7, feature selection cares about how many features to be selected. And also, we mentioned that fewer features with high accuracy are better than more features with the same accuracy. That leads us to not fix the number of features to be picked out of the indexes vector. For this reason, we made a randomized value to the number of features that we pick in each iteration.

The Fourth step: we mentioned in Section 1.9 that we will use k-NN classifier or Decision Tree classifier accuracy as a fitness function.

The Fifth step: after we sent only the given features to the classifier, we will divide it into training and testing data. We used K-fold cross validation method to separate the data into training and testing sets. We use $K=10$.

The values that have been picked out of the indexes vector will be saved in the first assigned variable.

The accuracy value will be compared with the old value which already saved from the first step. If the accuracy of this new child is higher than the parent's value, then we will replace the parent with the new child. Also, the values of the two assigned variables will be replaced with the new values.

The Sixth step: if the stop criteria are achieved then we will stop and go to the final step (Section 2.5.4), if not then we will go back to The Second Step.

2.4.4 Final Step and Showing Results

When the stop criteria in the main loop of EDE achieved, we will only have to know the highest accuracy that we found. In the beginning of Section 2.4, we were talking about the problem when we have the same accuracy with a different number of features. And also, explained that less number of features with high accuracy is better than large number of features with the same accuracy. We solved this problem by viewing the number of features with the accuracy of that given features. As a result, we let the user to decide which feature subset is the best.

The final result of the EDEFS is the final values of the two assigned variables for each individual in the last version of the population. The highest accuracy will be saved in the BEST variable. The output is represented by a structure array and each element represents the indexes of the dataset. The number of elements in the structure variable is

20 because we have 20 individuals. Table 2.3 shows the output screen of EDEFS when we implemented it on the Wine dataset by using k-NN classifier.

Table 2.3 The output of EDEFS with Wine dataset by using k-NN

| INDIVIDUAL NUMBER | NUMBER OF FEATURES | ACCURACY |
|-------------------|--------------------|----------|
| 1 | 7 | 97.674 |
| 2 | 8 | 97.674 |
| 3 | 4 | 97.674 |
| 4 | 4 | 97.674 |
| 5 | 7 | 97.674 |
| 6 | 4 | 97.674 |
| 7 | 10 | 100 |
| 8 | 8 | 93.023 |
| 9 | 5 | 95.348 |
| 10 | 6 | 100 |
| 11 | 4 | 97.674 |
| 12 | 9 | 100 |
| 13 | 2 | 93.023 |
| 14 | 9 | 97.674 |
| 15 | 7 | 97.674 |
| 16 | 4 | 97.674 |
| 17 | 8 | 97.674 |
| 18 | 8 | 97.674 |
| 19 | 6 | 95.348 |
| 20 | 8 | 95.348 |

As we see from the output (Table 2.3), there are three cases with the highest accuracy (100%). The three cases are 7, 10, and 12. In these cases, the accuracy is equal to 100% but the number of features is different. That is why we should show this table to the user to let him/her choice the best case. The 10th case probably the best because it has the highest accuracy with the fewer number of features.

The indexes of the given features are available in a structure array and Table 2.4 shows the values of that structure variable. Each case in Table 2.3 has its indexes in Table 2.4 with the same individual number. If we want to choose the 10th case in Table 2.3 then we should go to Table 2.4 and select the 10th indexes that given in the indexes column.

Table 2.4 The indexes of each case in table 2.2

| INDIVIDUAL NUMBER | THE INDEXES |
|-------------------|--------------------------|
| 1 | [3,7,10,9,4,8,6] |
| 2 | [1,8,9,10,11,2,4,7] |
| 3 | [10,3,1,9] |
| 4 | [7,11,6,10] |
| 5 | [3,11,7,10,12,1,9] |
| 6 | [3,10,6,7] |
| 7 | [12,10,4,11,1,6,7,3,9,8] |
| 8 | [3,7,10,9,11,12,6,5] |
| 9 | [7,10,6,8,9] |
| 10 | [8,9,10,6,7,5] |
| 11 | [7,10,12,9] |
| 12 | [4,10,12,1,6,11,7,3,9] |
| 13 | [10,7] |
| 14 | [7,10,12,11,4,1,3,9,8] |
| 15 | [12,10,8,4,9,1,2] |
| 16 | [10,3,12,11] |
| 17 | [6,12,10,9,7,1,3,11] |
| 18 | [7,10,12,6,9,3,1,11] |
| 19 | [11,4,8,12,1,3] |
| 20 | [12,10,4,1,11,6,9,3] |

COMPARING EDEFS TO OTHER'S WORKS

This chapter talks about the results of implementing EDEFS on different datasets. We have seen an example of the performance in CHAPTER 2. Now, we will see the performance on high dimensional datasets. After we explained the framework of EDEFS, now we have to see how far we succeed. The way to evaluate the successful is by comparing with recent work in this field of studying. Many researchers worked on implementing DE for feature selection. Some of them talk about how many features selected while others talk about the accuracy of training. We selected two recent works. The first work is [8] and the second one is [9]. The first work talks about high dimensional datasets. The second one talks about small scale datasets.

3.1 Results of High Dimensional Datasets

To evaluate the performance of EDEFS on high-dimensional datasets, we have to implement it on huge datasets. There are many high dimensional datasets available in [43], [44], and [45]. We selected twenty (20) datasets with a number of features between 2000 and 15009. Some of the twenty datasets are available at [46] and the rest of them are available at [47]. Details about the twenty datasets are explained in Table 3.1.

The work in [8] is done by employing the 1-NN classifier as a fitness function. They separate every dataset into train and test by using the 10-fold method. Since the 10-folds are generated randomly, the results will have some randomness. We also used the 10-fold cross validation method to separate our dataset into training and test/validation sets. The training accuracy depends on the training data and the training data itself depends on k-fold cross validation.

Table 3.1 Details about the high-dimensional datasets

| No | Dataset Name | No of Classes | No of sample | No of feature |
|----|----------------|---------------|--------------|---------------|
| 1 | 9_Tumors | 9 | 60 | 5726 |
| 2 | 11_Tumors | 11 | 174 | 12,533 |
| 3 | 14_Tumors | 26 | 308 | 15,009 |
| 4 | Brain_Tumor1 | 5 | 90 | 5,920 |
| 5 | Brain_Tumor2 | 4 | 50 | 10,367 |
| 6 | DLBCL | 2 | 77 | 5,469 |
| 7 | Leukemia1 | 3 | 72 | 5,327 |
| 8 | Leukemia2 | 3 | 72 | 11,225 |
| 9 | Lung Cancer | 4 | 203 | 12,600 |
| 10 | Prostate Tumor | 2 | 102 | 10,509 |
| 11 | SRBCT | 4 | 82 | 2,308 |
| 12 | Adenocarcinoma | 2 | 76 | 9,868 |
| 13 | Brain | 5 | 42 | 5,597 |
| 14 | Breast2 | 2 | 77 | 4,869 |
| 15 | Breast3 | 3 | 95 | 4,869 |
| 16 | Colon | 2 | 62 | 2,000 |
| 17 | Leukemia | 3 | 38 | 3,051 |
| 18 | Lymphoma | 3 | 62 | 4,026 |
| 19 | Nci | 8 | 61 | 5,244 |
| 20 | Prostate | 2 | 102 | 6,033 |

As a result, the accuracy value will depend on the random value of k-fold. For that reason, we take the accuracy as average of 10 iterations as in [8]. When we implemented EDEFS on the datasets in Table 3.1, the settings of the EDEFS parameters were as the following:

1. Population size = 20
2. Minimum boundary = -5
3. Maximum boundary = 5
4. Function Evaluation Times =50

We will compare our work in the concept of learning accuracy. As we mentioned lately, we use k-NN as the classifier. In [8], they used only 1-NN as a fitness function. For that reason, we are going to compare our work only by using 1-NN classifier. The work in [8] optimized the parameters of their work. We also optimized our parameters. For that reason, we will take the Table of results in [8] directly without setting the same parameters. Table 3.2 showing the results of implementing EDEFS on the datasets in Table 3.1 and compare it with [8] results.

In Table 3.2, Dataset name column contains the names of the twenty datasets. Full data accuracy column contains the accuracy of implementing 1-NN on the whole data. SSFS column contains the results of implementing the work in [8], Single Sequence Fast Feature Selection for High-Dimensional Data (SSFS), on the datasets. EDEFS column contains the results of implementing our work on the datasets. The highlighted value means it is the best by comparing it with others.

Table 3.2 The result of implementing EDEFS on high dimensional data

| No | Dataset name | No of feature | Full data accuracy | SSFS | EDEFS |
|----|----------------|---------------|--------------------|------------|-----------------|
| 1 | 9_Tumors | 5726 | 51% | 63% | 71.6667% |
| 2 | 11_Tumors | 12,533 | 82% | 83% | 90.0000% |
| 3 | 14_Tumors | 15,009 | 57% | 58% | 63.4375% |
| 4 | Brain_Tumor1 | 5,920 | 86% | 89% | 93.3333% |
| 5 | Brain_Tumor2 | 10,367 | 71% | 85% | 87% |
| 6 | DLBCL | 5,469 | 86% | 97% | 98.3333 |
| 7 | Leukemia1 | 5,327 | 89% | 99% | 98.2353% |
| 8 | Leukemia2 | 11,225 | 93% | 98% | 99.4444% |
| 9 | Lung Cancer | 12,600 | 90% | 95% | 96.1224% |
| 10 | Prostate Tumor | 10,509 | 77% | 96% | 86.4000% |
| 11 | SRBCT | 2,308 | 92% | 99% | 97.3684% |
| 12 | adenocarcinoma | 9,868 | 87% | 95% | 91.5789% |
| 13 | brain | 5,597 | 75% | 93% | 87.7778 |
| 14 | Breast2 | 4,869 | 62% | 86% | 71.5789% |
| 15 | Breast3 | 4,869 | 55% | 78% | 64.3478 |
| 16 | colon | 2,000 | 73% | 94% | 88.6667% |
| 17 | leukemia | 3,051 | 98% | 98% | 100% |
| 18 | lymphoma | 4,026 | 98% | 100% | 100% |
| 19 | nci | 5,244 | 72% | 82% | 88.4615% |
| 20 | prostate | 6,033 | 83% | 96% | 88.8000% |

It is clear from Table 3.2 that our work has better results when the dataset has a high number of features. For datasets No. 1 to 6, our work gets the best accuracy. By looking at the details in Table 3.1, the dataset No. 1 to 4 has the highest number of features. While for datasets from 10 to 16, SSFS has the highest accuracy. Datasets No. 10 to 16 has the lowest number of features. By considering this analysis of the results in Table 3.2, we can say that our work is better with high dimensional data.

We should notice that EDEFS always get higher accuracy than the full datasets. This is the main idea of using feature selection process so we can say we successfully achieved this principle. Moreover, the number of cases that EDEFS has the best result is more

than what SSFS has. After testing EDEFS on high dimensional data, now we need to test our work on small scale datasets (datasets which have less than 100 features), to see if it is able to get best results or not.

3.2 Results of Small Scale Datasets

Now we will try to implement our work (EDEFS) on a standard datasets benchmark. The recent work on this kind of dataset is [9]. They use 12 datasets as a benchmark. 3 of the datasets come with missing values so the training accuracy is lower than 50%. We will not take them into account. All the 12 datasets are available in [43]. Table 3.3 illustrates the benchmarks in more details. In [9], they use 1-NN and Decision Tree as classifiers. Firstly, we will compare our work in case we use 1-NN on the 9 datasets in Table 3.3. Also, we will compare our work by using Decision Tree as a classifier.

Datasets in Table 3.3 have number of features between 60 and 13. Testing EDEFS on this kind of datasets means we will test it on a small-scale dataset. We are trying to prove that EDEFS is a general approach and it can be implemented on any size of data. We mentioned in Section 3.1 that our EDEFS works better with high dimensional data. In this section, we will prove that our algorithm is also able to deal with small-scale datasets.

Table 3.3 Details about the small-scale datasets

| No | Dataset name | Number of samples | Number of features | Number of classes |
|----|--------------------------------|-------------------|--------------------|-------------------|
| 1 | Heart | 270 | 13 | 2 |
| 2 | Hepatitis | 155 | 19 | 2 |
| 3 | Ozone level detection | 2536 | 73 | 2 |
| 4 | Parkinson | 195 | 22 | 2 |
| 5 | Segmentation | 2310 | 19 | 7 |
| 6 | Sonar | 208 | 60 | 2 |
| 7 | Spectf | 267 | 44 | 2 |
| 8 | Wine | 178 | 13 | 3 |
| 9 | Wisc. prognostic breast cancer | 198 | 13 | 3 |

Table 3.4 shows the results of implementing EDEFS on the datasets in Table 3.3 by using 1-NN. The random values of the 10-fold cross validation may effect on the learning accuracy so taking average of number of accuracies is probably necessary. The work in [9] is simply using 1-NN classifier and also using 10-fold cross validation to separate the datasets without any consideration to the random values of the k-fold method. We also used 1-NN without any consideration to the random values of the k-fold method (without taking an average of number of accuracies). The parameters settings for our work are as follow:

1. Population size = 20
2. Minimum boundary = -5
3. Maximum boundary = 5
4. Function Evaluation Times =50

Table 3.4 shows the result of implementing EDEFS on the datasets in Table 3.3 by using only 1-NN and compare it with the results in [9]. In Table 3.4, MODE_FS is the abbreviation of Multi-Operator Differential Evolution for Feature Selection.

Table 3.4 Implementing EDEFS on datasets with 1-NN

| No | Dataset names | Full data accuracy | EDEFS Accuracy | MODE-FS |
|----|--------------------------------|--------------------|----------------|---------------|
| 1 | Heart | 75.27% | 89.55% | 92.48% |
| 2 | Hepatitis | 74.39% | 92.10% | 81.49% |
| 3 | Ozone level detection | 85.32% | 97.47% | 95.16% |
| 4 | Parkinson | 94.95% | 95.83% | 99.21% |
| 5 | Segmentation | 83.33% | 97.95% | 91.67% |
| 6 | Sonar | 84.90% | 98.03% | 88.75% |
| 7 | Spectf | 77.72% | 89.39% | 88.16% |
| 8 | Wine | 96.93% | 100% | 97.47% |
| 9 | Wisc. prognostic breast cancer | 93.16% | 87.50% | 96.58% |

The results of EDEFS in Table 3.4 are better than full data accuracy. This means our work can be considered as a good feature selection approach for small scale datasets. Also, we see in Table 3.4 that our work is better than MODE-FS in 6 datasets. These results prove that EDEFS can also be used with small scale data.

3.3 Results of Comparing the Binary and the Real Differential Evolution FS

Feature Selection is treated as a binary optimization problem in most of the researches. Since EDEFS is a real feature selection approach, there should be a comparison with a binary approach to evaluate the performance as well. Firstly, a binary DE is presented by using a V-shape transfer function as a mutation operation. The V-shape transfer function helps to adapt any real metaheuristic algorithm to work in binary search space or so-called discrete search space. Indeed, there are many differences in the main steps of the BDE from the basic DE, for example in the initialization step. Figure 3.1 shows the pseudocode of the binary feature selection approach, which is applied by using the BDE. We applied the binary version of the Differential Evolution to work as a feature selector. Figure 3.1 shows the pseudocode of the Binary Differential Evolution Feature Selector (BDEFS).

We used group of datasets (illustrated in Table 3.5) to evaluate the performance of the EDEFS and the BDEFS approach by using kNN as a fitness function and the parameter sittings as the following:

1. Population size for both EDEFS and BDEFS = 20
2. Minimum boundary for EDEFS = -5
3. Minimum boundary for BDEFS = 0
4. Maximum boundary for EDEFS = 5
5. Maximum boundary for BDEFS = 1
6. Function Evaluation Times for both EDEFS and BDEFS =50

Table 3.5 The group of datasets to test BDEFS and EDEFS

| No | Dataset names | Number of features | BDEFS | EDEFS |
|----|-----------------------|--------------------|-----------------|-----------------|
| 1 | Heart | 13 | 80.1493% | 82.0896% |
| 2 | Hepatitis | 19 | 83.9474% | 85.7895% |
| 3 | Ozone level detection | 73 | 96.0979% | 96.9352% |
| 4 | Parkinson | 22 | 92.7083% | 90.8333% |
| 5 | Segmentation | 19 | 92.0408% | 92.4490% |
| 6 | 9_Tumors | 5726 | 68.3333% | 71.6667% |
| 7 | 11_Tumors | 12,533 | 88.6842% | 90.0000% |
| 8 | 14_Tumors | 15,009 | 65% | 63.4375% |
| 9 | Brain_Tumor1 | 5,920 | 93.3333% | 93.3333% |
| 10 | Brain_Tumor2 | 10,367 | 85% | 87% |

```

Function Fitness(X) // function to calculate the accuracy of learning
{
D = Data(:,X==1) // extract only the desired features
Training=10-fold(D) // split the data into train and test by using 10-fold method
Testing=10-fold(D)
Accuracy=kNN(Training , Testing) // calculate the accuracy of learning using kNN
Return (Accuracy)
}

popsize = 20
For i =1 to 20 do
P(i)= rand([1,number of features in the given dataset (Data)], [0,1])
Repeat
For i = 1 to 50 do
For j =1 to popsize do // 20 parents chosen for popsize
X=P(j)
A ← copy of a random individual
B ← copy of a random individual
C ← copy of a random individual
For s=1 to Size(P(j))
V = TransferFunction (mean(A(s),B(s),C(s) ),P(j))
if rand [0,1] > V
X(s)=X(s)' //if X(s)=1 then X(s)'= 0 and vice versa
Repeat
Child = TwoPointsCrossover( X , P(j) )
If Fitness(Child) > Fitness(P(j))
P(j) = Child // Replace the parent with the kid
If Fitness(P(j)) > Best(i-1) then
Best(i) = P(j)
Repeat
Repeat

```

Figure 3.1 Pseudocode of the Binary Feature Selection

3.4 Results of Comparing the EDEFS with Random Feature Selection

Random Feature Selection is one of the exist approaches for feature selection. It all depends on the randomness and also uses a learning algorithm to evaluate the accuracy of learning for the selected features subsets. The experiment results showed that EDEFS is a good approach compared with the stochastic approach, even binary or others. To make sure that EDEFS is better than the random feature selection, we applied and compared one of the random feature selection approaches by using a group of datasets. Figure 3.2 shows the pseudocode of the Random Feature Selection (RFS) and Table 3.6 shows the group of datasets.


```

Function Fitness(X)
{
D = Data(:,X)
Training=10-fold(D)
Testing=10-fold(D)
Accuracy=kNN(Training , Testing)
Return (Accuracy)
}
LoadData(X)
S = Size(X) // S is the number of features in the given data
Indexes = [0] // initialize the variable which will hold the indexes
Best=0
For i = 1 to 1000 do
    N = random [ 1 , S ] //Chose the number of feature to be selected
    Features = random ( N , [ 1 , S ] ) // Chose the N features indexes
    Accuracy = Fitness(Features)
    if Accuracy > Best
        Best = Accuracy
        Indexes = Features
Repeat
Return(Indexes)

```

Figure 3.2 The pseudocode of the Random Feature Selection

Table 3.6 The group of datasets to compare RFS and EDEFS

| Datasets name | No of Features | Whole data | RFS | EDEFS |
|----------------|----------------|------------|-----|------------|
| Adenocarcinoma | 9868 | 87% | 91% | 95% |
| brain | 5597 | 75% | 87% | 93% |
| Sonar | 60 | 84.90% | 89% | 98% |
| Breast2 | 4869 | 62% | 71% | 86% |

3.5 Results of Comparing the EDEFS with Random Indexes DEFS

It is clear, from the experiment results and from the comparisons with the recent works, that EDEFS is able to achieve high accuracy. Regarding these claims, only one test is remaining. This test is simply by using the DE but with initializing its population with the indexes of the features! The population also passes through the mutation and the crossover operations but there is an additional operation after them. The additional operation is rounding the values of the new child to be between 1 and the number of features.

Figure 3.3 shows the pseudocode of the Random Indexes DEFS (RIDEFS) and illustrates the main steps. We also used a group of datasets (datasets in table 3.7) to test and compare the EDEFS with the RIDEFS.

Table 3.7 The group of datasets to compare RIDEFS and EDEFS

| Datasets name | No of Features | Whole data | RIDEFS | EDEFS |
|----------------|----------------|------------|--------|------------|
| Adenocarcinoma | 9868 | 87% | 91% | 95% |
| brain | 5597 | 75% | 87% | 93% |
| Sonar | 60 | 84.90% | 89% | 98% |
| Breast2 | 4869 | 62% | 71% | 86% |

Obviously, EDEFS still a good approach by comparing with the available approach. All the testing approved that EDEFS is applicable and acceptable to be used with high or small scale datasets.

```

Function Fitness(X) // function to calculate the accuracy of learning
{
D = Data(:,X==1) // extract only the desired features
Training=10-fold(D) // split the data into train and test by using 10-fold method
Testing=10-fold(D)
Accuracy=kNN(Training , Testing) // calculate the accuracy of learning using
kNN
Return (Accuracy)
}
popsize = 20
NumberOfFeatures = Size(Data)
For i =1 to 20 do
P(i)= rand([1, NumberOfFeatures ])
Repeat
For i = 1 to 50 do
For j =1 to popsize do // 20 parents chosen for popsize
A ← copy of a random individual
B ← copy of a random individual
C ← copy of a random individual
T = A +  $\alpha$  (B-C)
R1 = rand [1, Size(P( j ))]
R2 = rand [1, Size(P( j ))]
X=P( j )
X(from R1 to R2) = T(from R1 to R2)
X= round(X,1,NumberOfFeatures)
Z= X (from 1 to rand [1, Size(P( j ))]) // take randomly number of
features
If Fitness(Z) > Fitness(P(j))
P(j) = Z // Replace the parent with the kid
If Fitness(P(j)) > Best(i-1) then
Best(i) = P(j)
Repeat
Repeat
Return(Best)

```

Figure 3.3 The pseudocode of the Random Indexes DEFS

CONCLUSION AND FUTURE WORK

4.1 Conclusion

Feature Selection is an important subject to be studied. It helps to increase the accuracy of any learning algorithm. There are many ways to do feature selection and, in our opinion, the stochastic way is the simplest one. The simplicity of DE algorithm has made the whole implementation simple as well. Indeed, there are many metaheuristic algorithms which can be employed to work as a feature selector. The differences between any two given metaheuristic algorithms can be determined and can be analyzed by using the CEC 2015 benchmark.

EDE is a good algorithm to be used for Feature Selection. Also, it can be implemented for any related subjects since it has high exploration rate. By implementing and testing, we found that the population-base algorithms can get close to the optimal solution in a short period of time. For that reason, the number of evaluation time should not be that large. As a result, we can make the extraction of the program, or the algorithm, faster by tuning the parameters with perfect values.

EDEFS is a new and efficient approach for feature selection. The results show how strong and efficient the EDEFS algorithm with high dimensional datasets is. Also, it is suitable for low scale datasets.

4.2 Future Works

No doubt, the initialization step of the EDE effects on the total results. For that reason, we can work on finding a way to initialize the population depending on the given problem range. Also, the way to select the individuals to mutation step is very important. Therefore, we can increase the exploration rate if we can find a good way to select the effective individuals for mutation operation.



REFERENCES

- [1] Jarraya, B., and Abdelfettah B., (2012). "Metaheuristic Optimization Backgrounds: A Literature Review" *International Journal of Contemporary Business Studies* 3(12): 2156-7506.
- [2] Olafsson, S., (2006). "Metaheuristics." *Handbooks in operations research and management science* 13(6):633-654.
- [3] Luke, S., (2013). *Essentials Of Metaheuristics*, Lulu Com, New York.
- [4] Xue, B., Wenlong, F., and Mengjie Z., (2014). "Differential Evolution (De) For Multi-Objective Feature Selection in Classification", *Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, ACM*, 12 – 16 July 2014, Indian.
- [5] Wang, H., Kwong, S., and Jin, Y., (2005). "Multi-Objective Hierarchical Genetic Algorithm for Interpretable Fuzzy Rule-Based Knowledge Extraction" *Fuzzy Sets and Systems*, 149(1):149–186.
- [6] Chen, Y., Weicheng, X., and Xiufen, Z., (2015). "A Binary Differential Evolution Algorithm Learning From Explored Solutions" *Neurocomputing*, 149 (3): 1038-1047.
- [7] Clerc, M., Initialisations for Particle Swarm Optimisation, <http://clerc.maurice.free.fr/ps0>, 3 September 2016.
- [8] Boldt, F., Thomas, W., and Flávio, M., (2015). "Single Sequence Fast Feature Selection for High-Dimensional Data" *Tools with Artificial Intelligence (ICTAI)*, 2015 IEEE 27th International Conference on. IEEE, 9-11 Nov 2015, Canada.
- [9] Debie, E., Elsayed, M., and Essam, L., (2016). "Investigating Multi-Operator Differential Evolution for Feature Selection." *Australasian Conference on Artificial Life and Computational Intelligence*, Springer International Publishing, 2-5 February 2016, Australia.
- [10] Memon, N., Hicks, D., and Larsen. (2007), "Notice Of Violation Of Ieee Publication Principles Harvesting Terrorists Information From Web" in *11th International Conference on Information Visualization*, 4-6 July 2007, US.
- [11] Hand, D., (2007), "Principles of Data Mining" *Drug Safety*, 30(7): 621–622.

- [12] Cooley, R., Mobasher, B., and Srivastava, J., (1997). "Web Mining: Information And Pattern Discovery On The World Wide Web" in Proceedings of the 9th IEEE International Conference on Tools with Artificial Intelligence, 3-8 Nov 1997, Newport Beach, California.
- [13] Mac,N., (2006). Guiding rough and fuzzy-rough feature selection using alternative evaluation functions and search strategies, Ph.D. dissertation, University of Wales Aberystwyth, UK.
- [14] Fayyad, U., Piatetsky,G., and Smyth, P., (1996). "From Data Mining To Knowledge Discovery In Databases" AI Magazine, 17(3):37–54.
- [15] Mac, N., Jensen, A., and Shen, Q., (2010). "Fuzzy-Rough Approaches For Mammographic Risk Analysis", Intelligent Data Analysis, 14(2):225–244.
- [16] Diao, R,(2014) .Feature Selection with Harmony Search and Its Applications, PhD thesis, Aberystwyth University, UK.
- [17] Bellman, E., and Stuart, E., (2015). Applied Dynamic Programming, Princeton university press, London.
- [18] Dash, M., and Huan,L., (1997)."Feature Selection for Classification" Intelligent Data Analysis, 1(3):131-156.
- [19] Xing, P., Jordan, M., and Karp,M., (2001). "Feature Selection For High-Dimensional Genomic Microarray Data" ICML, 1(3):601–608.
- [20] Jensen, R., and Qiang, S., (2009),"Are More Features Better? A Response to Attributes Reduction Using Fuzzy Rough Sets", IEEE Transactions on Fuzzy Systems 17(6):1456-1458.
- [21] Lee, M., Chen, M., and Chen, J., (2001). "An Efficient Fuzzy Classifier With Feature Selection Based On Fuzzy Entropy" IEEE Trans Syst Man Cybern,31(2): 426–432.
- [22] Jensen, R., and Qiang, S., (2009)."New Approaches to Fuzzy-Rough Feature Selection" IEEE Transactions on Fuzzy Systems 17(4): 824-838.
- [23] Dash ,M., and Liu, H.,(2003). "Consistency-Based Search In Feature Selection" Artificial Intelligence, 151(1-2) :155–176.
- [24] Hall, M., (1998). Correlation-Based Feature Subset Selection For Machine Learning, Ph.D. thesis, University of Waikato, Hamilton, New Zealand.
- [25] Hsu, N., Huang,J., and Schuschel, D., (2002). "The Annigma-Wrapper Approach To Fast Feature Selection For Neural Nets" IEEE Trans. Syst., Man, Cybern. B, 32(2): 207–212.
- [26] Zhu, Z., Ong,Y., and Dash, M., (2007). "Wrapper-Filter Feature Selection Algorithm Using A Memetic Framework" IEEE Trans. Syst., Man, Cybern. B, 37(1): 70–76.
- [27] Francis, L., (2005). Data Mining Concepts, Models, Methods and Algorithms, Paperback, IEEE Press/Wiley, Canada.

- [28] Storn, R., and Kenneth, P., (1997). "Differential Evolution—A Simple and Efficient Heuristic For Global Optimization Over Continuous Spaces" ,Journal of Global Optimization 11(4): 341-359.
- [29] Bharathi, T., and Subashini, P., (2015). "Optimal Feature Subset Selection Using Differential Evolution With Sequential Extreme Learning Machine For River Ice Images" TENCON 2015-2015 IEEE Region 10 Conference. IEEE, 1-4 Nov 2015, China.
- [30] Gadat, S., and Laurent, Y., (2007). "A Stochastic Algorithm For Feature Selection In Pattern Recognition" ,Journal of Machine Learning Research 8.(2): 509-547.
- [31] Chen, Q., Liu, B., and Zhang, Q.,(2014). Problem Definitions and Evaluation Criteria for CEC 2015 Special Session on Bound Constrained Single-objective Computationally Expensive Numerical Optimization Technical Report Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou.
- [32] Gampferle, R., Muller,S., and Koumoutsakos, P., (2002).“A Parameter Study For Differential Evolution” in Advances in intelligent systems, fuzzy systems, evolutionary computation, 10(2):293-298.
- [33] Mezura,E.,Velazquez,J., and Coello,C.,(2006) , “A Comparative Study Of Different Evolution Variants For Global Optimization” in Proc. Genetic and Evolutionary Computing Conference, 8-12 July 2006,Mexico, US.
- [34] Shu,M., and Pang, H., “Improving Differential Evolution with a Successful-Parent-Selecting Framework”, IEEE Transactions On Evolutionary Computation, 19(5):717-730.
- [35] Lou, Y., Li,J., and Li,G., (2012). “A Differential Evolution Algorithm Based On Individual-Sorting And Individual-Sampling Strategies” , Comput. Inf. Syst, 8(2): 717–725.
- [36] Rahnamayan, S., Tizhoosh,R., and Salama, M.,(2007). “Quasioppositional Differential Evolution” in Evolutionary Computation, CEC 2007. IEEE Congress on. IEEE, 10(11):2229–2236.
- [37] Kazimipour, B., Li, X., and Qin, A.,(2013) ,“Initialization Methods For Large Scale Global Optimization” in Evolutionary Computation (CEC), 2013 IEEE Congress on. IEEE, 17(3): 2750–2757.
- [38] Kazimipour,B., Li, X., and Qin, A., (2014). “A Review Of Population Initialization Techniques For Evolutionary Algorithms” in Evolutionary Computation (CEC), 2014 IEEE Congress on. IEEE, 6-11 July 2014, China.
- [39] Pant,M., Ali, M., and Singh,V., (2009). “Differential Evolution Using Quadratic Interpolation For Initializing The Population,” in Advance Computing Conference, 2009. IACC 2009. IEEE International. IEEE, 10(2):375–380.
- [40] Ajeet, S., Tarun,S., and Shweta ,S.,(2016).“Differential Evolution Algorithm for Solving Computationally Expensive Optimization Problems”, Springer International Publishing Switzerland, 12(2):87:96.

- [41] Liu, H, and Motoda, H.,(2007). Computational Methods of Feature Selection, CRC Press, US.
- [42] Witten, H., and Eibe, F., (2005). Data Mining: Practical Machine Learning Tools And Techniques. Morgan Kaufmann, New Zealand.
- [43] Frank, A., and Arthur, A., (2010). UCI Machine Learning Repository, School of Information and Computer Science, Turkey.
- [44] Statnikov, A., Aliferis,C., and Tsamardinos, I., (2004). “Methods For Multi-Category Cancer Diagnosis From Gene Expression Data: A Comprehensive Evaluation To Inform Decision Support System Development”, Medinfo, 11(2):813–7.
- [45] Diaz, R., and Andr´es, S., (2005). “Variable Selection From Random Forests: Application To Gene Expression Data”, 22 -25 Jun 2005, Ithaca.
- [46] Statnikov, A., Tsamardinos, I., Dosbayev , F., www.gems-system.org, 3 June, 2016.
- [47] Supplementary material for “Gene Selection and Classification of Microarray Data Using Random Forest”, www.ligarto.org/rdiaz/Papers/rfVS/randomForestVarSel.html ,7 June 2016.

CURRICULUM VITAE

PERSONAL INFORMATION

Name Surname : Muneer HASAN
Date of birth and place : 22-1-1991 Iraq/Diyala
Foreign Languages : English
E-mail : memobarazanchi@gmail.com

EDUCATION

| Degree | Department | University | Date of Graduation |
|---------------|----------------------|-------------------|---------------------------|
| Graduate | Computer Engineering | YTU | 2016 |
| Undergraduate | Computer Science | Diyala university | 2013 |
| High School | | Ghalybya school | 2008 |

PUBLISHERMENTS

Conference Papers

- 1.** Hasan, M. M. and Altun, O., (2016). “Two Enhanced Differential Evolution Algorithms for Expensive Optimization” in Journal of Applied and Physical Sciences, 28-29 September 2016, Istanbul, Turkey.

