Adaptive Differential Evolution Algorithm for Single and Multi-Objective Numerical Optimization

Abdallah Ahmad Alaraj

Submitted to the
Institute of Graduate Studies and Research
in partial fulfillment of the requirements for the degree of

Master of Science
in
Computer Engineering

Eastern Mediterranean University
September 2019
Gazimağusa, North Cyprus

Approval of the Institute of Graduate Studies and Research

Prof. Dr. Ali Hakan Ulusoy
Acting Director

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science in Computer Engineering.

Prof. Dr. Işık Aybay
Chair, Department of Computer Engineering

We certify that we have read this thesis and that in our opinion it is fully adequate in scope and quality as a thesis for the degree of Master of Science in Computer Engineering.

Assoc. Prof. Dr. Adnan Acan
Supervisor

Examining Committee

1. Assoc. Prof. Dr. Adnan Acan
2. Asst. Prof. Dr. Zehra Borataş Şensoy
3. Asst. Prof. Dr. Ahmet Ünveren

ABSTRACT

“DE/current-to-pbest” is a new and increasingly common mutation strategy that involves an additional external archive and adaptively updates the control. This thesis introduces a novel algorithm known as JADE. The “DE/current-to-pbest” is a simplification of the typical “DE/current-to-best,” while historical data is used by the additional archive operation to provide information on progress direction. Both convergence performance and the diversity of the population are enhanced by the two operations. The control parameters are automatically updated to the appropriate values through parameter adaptation, which avoids relying on outdated information regarding the relationship between the characteristics of the optimization problems and the parameter settings.

This thesis work introduces a JADE Algorithm and examines its feasibility based on the results of CEC'17 expensive benchmark problems for single objective optimization problems and for Multi-objective optimization. The methods used in our studies are compared to different well-knows methods proposed in the related literature was conducted. The final ranking of all test problems indicate that JADE was always among the top best algorithms that were used for the same purpose.

Keywords: Multi-agent systems, Meta-heuristic algorithms, Multi-objective optimization, evolutionary optimization, Adaptive parameter control, Pareto optimality, differential evolution.

ÖZ

“DE/current-to-pbest”, harici ek bir arşiv ile kontrolü adaptif olarak güncelleyen yeni ve giderek daha da yaygın olarak kullanılan bir mutasyon stratejisidir. “DE/current-to-pbest”, özgün olan “DE/current-to-pbest” algoritmasının sadeleştirilmiş halidir. Historik veri, ilerleme yönü hakkında bilgi sağlamak amacı ile ek arşivleme işlemi tarafından kullanılır. Popülasyonun çeşitliliği ve yakınsama performansı, iki operasyon tarafından artırılmıştır. Kontrol parametreleri, optimizasyon problemlerinin karakteristikleri ve parametre ayarları arasındaki ilişki ile ilgili eski bilgilere dayanmaktan kaçınan parametre adaptasyonu ile otomatik olarak uygun değerlere güncellenmektedir.

Bu tez çalışması, JADE algoritmasını sunar ve tek amaçlı optimizasyon problemleri ile çok amaçlı optimizasyon için CEC'17 pahalı kriter problemlerinin sonuçlarını baz alarak mümkünlüğünü inceler. Çalışmalarımızda kullanılan yöntemler, literatürde bulunan bilindik yöntemlerle karşılaştırılmıştır. Tüm test problemlerinin son sıralaması JADE’in her zaman aynı amaç için kullanılan en iyi algoritmalar arasında olduğunu göstermektedir.

Anahtar kelimeler: çok ajanlı sistemler, üst-sezgisel algoritmalar, çok amaçlı eniyileme, evrimsel eniyileme, adaptif parametre kontrolü, Pareto optimalite, diferansiyel evrim.

ACKNOWLEDGMENT

I owe a number of amazing people my deepest gratitude, without whose support and inspiration this thesis would not have been possible — thank you all for coming on this journey with me. I would especially like to appreciate my supervisor Assoc.Prof.Dr.Adnan Acan. Without his encouragement, guidance, and continuous optimism, this thesis would not have been possible.

I owe my dad Eng.Ahmed , my mom Eitemad ,my sister Bayan and my brothers a great debt for their support and the many experiences that helped shape who I am. Their love and unwavering support made this body of work possible and I would like to dedicate it to them.

I would like to thank all my friends in Famagusta who gave me the necessary distractions from my research and made my stay in Cyprus memorable.

Finally, my deepest gratitude to all the thoughtful wishes of my old friends, each message and call was deeply appreciated. I would also like to thank all those friends that accompanied and helped me in the pursuit of my Master's degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

ABC	Artificial Bee Colony
CEC	IEEE Congress on Evolutionary Computation
DE	Deferential Evolution
DM	Decision Marker
EA	Evolutionary Algorithm
EP	Evolutionary Programming
GA	Genetic Algorithm
JADE	Adaptive Differential Evolution with External Archive
MOABC	Multi-Objective Artificial Bee Colony
MOEAs	Multi-Objective Evolutionary Algorithms
MOO	Multi-Objective Optimization
NSGAII	Non-Dominated Sorting Genetic Algorithm
Pbest	Personal Best
Pc	Crossover Probability
Pm	Mutation Probability

Chapter 1

INTRODUCTION

A significant amount of the real-world problems we face today in science and engineering can be treated as an optimization problem. Some with single objects others with multi-objective criteria. The conventional optimization methods are useful for finding the optimal solution or unconstrained maxima or minima of continuous and differential functions. Such methods are limited to finding local optimal solutions and are usually analytical in nature. Significant research is continuously made towards inventing novel optimization technique which are applicable in solving real life problems with the capability of population based solutions and memory update. More recently, algorithms such as Evolutionary Algorithm, Genetic Algorithms, Differential Algorithm, and Simulated Annealing algorithm which are considered to be general purpose optimization technique have become standard optimization techniques. The techniques are popular due to their ability to adapt to their ever-changing environment in an effective and efficient manner.

Differential Algorithm is a derivative free, population based global optimization algorithm. It is the most promising Evolutionary algorithm due to its powerful population-related stochastic direct search ability for finding the solutions to numeric optimization problems in a continues search environment.

1.1 Background of Study

The common challenge of most engineering application problems is solving optimization problems. They are solved using optimization algorithms such as metaheuristics, which is a type of optimization algorithm that manifested from nature. According to Sorensen and Glover [1] metaheuristic is a "high-level problem-independent algorithmic framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms." In the early 90's, research on metaheuristics grew exponentially with the introduction of several frameworks improving the initial method. Sorensen and Glover [1] stated that application of metaheuristics has existed long before the term was coined. According to their research, five periods shaped the evolution of metaheuristics; Pre-Theoretical period (until C.1940). the second phase in the early period (C.1940-1980) which introduced the formal study of heuristics. Artificial intelligence was the term used to identify the work in that period. In 1960s, evolution was highlighted as a method of solving for problem solving, using insights from natural evolution to general optimization problems. Thus, several algorithms are developed using inspiration from natural evolution. Evolutionary programming was incepted in the later years of 1960, however, population and crossover methods were not utilized. The work of Goldberg [2] ignited the evolutionary revolution. Subsequently, evolutionary methods became very widespread.

There are commonly two types of metaheuristics: the population based which produces a population of solutions, and the trajectory-based which produces a single solution. The metaheuristic algorithm uses a certain type of stochastic optimization that finds the near global optimal or global optimal solution of a problem by using a random selection. Thus, they are utilized in solving numerous optimization problems [3].

The most prevalent types of trajectory-based metaheuristic methods include Artificial Bee Colony [4], Ant Colony Optimization [5], Simulated annealing [6], Tabu search [7], Differential evolution [3], and Great Deluge algorithm Dueck, [8].

1.2 Aim of the Study

A number of real-word problems have multiple objectives that need to be optimized simultaneously. One major difference between single objective optimization and the simultaneous optimization of multiple objectives is that multi-objective optimization problems (MOPs) do not have a unique solution. Rather, the aim is to identify all good trade-off solutions to be evaluated by a decision maker according to some preferential information. JADE: Adaptive Differential Evolution with Additional External Archive is an Improved version of differential evolution (DE) algorithm intended to enhance the performance. The enhancement is achieved through the implementation of a novel mutation strategy “DE/current-to-pbest” with an additional external archive and adaptive control parameter updates So far, different strategies and models of (DE) algorithm are developed for solving difficult real- valued optimization problems.

The aim of this master thesis study is to evaluate the performance of JADE for single and multi-objective numerical optimization problems and comparatively evaluate its performance against other modern state-of-the-art metaheuristics. This will help in achieving a deep understanding of evolutionary algorithm computing within the framework of DE algorithms and experience the application of a particular method for the solution of widely used benchmark problems.

The test problems to be used in experimental evaluation will be the ones prepared for latest contests in well-known conferences, such as CEC2017 constrained and unconstrained benchmarks. In addition to these, some problem related to practical engineering applications will also be used in evaluating the DE strategies. Finally, detailed statistical analyses will be conducted to exhibit statistical significance of algorithms under consideration.

1.3 Organization of the Thesis

Chapter 2 present a brief introduction to optimization and evolutionary algorithm with a literature review on Differential Evolutionary algorithm. The methodology utilized in this thesis is presented in Chapter 3 with detail description and steps to how it was implemented. Chapter 4 covers the results and findings of the thesis with discussion of the achieved result. The thesis is concluded in chapter 5, summarizing the entire study and presenting a direction for future research based on our observed result.

Chapter 2

LITERATURE REVIEW

2.1 Optimization

Optimization is an important part of our day-to-day lives. Optimizing every aspect of our lives locally or globally provides a more productive outcome. Many engineering, economic and scientific problems require optimizing an important set of parameters. Example of such problems include the minimization of electricity losses in an electricity grid which involves optimizing component for optimal configuration, or modifying a neural network to recognize faces. Other optimization application includes finding the shortest path for rural water connection.

Optimization infers finding the one or more solution to a maximization or minimization problem of an objective function. An optimization problem is formally defined as follows:

$Maximize/Minimize\ \ f_i(x), i = 1, \dots M, X = [x_1, x_2, \dots, x_D]$

$Subject\ to$:

$$g_j(x) \leq 0, j = 1, \dots, J$$

$$h_k(x) = 0, k = 1, \dots, K$$

$$x \in [X_{min}, X_{max}]^D$$

$f_i(x)$: objective function

$g_j(x)$: inequality constraint function

$h_k(x)$: equality constraint function

A maximization problem *Maximum* $f(x)$ can easily be converted into a minimization problem as *Minimum* $-f(x)$. There are two kinds of optimization problems: Single-objective optimization problems and Multi-Objective optimization problems. A single-objective optimization problem occurs when the problem involves a single objective function, while a multi-objective optimization problem tends to solve a problem having two or more objective functions. When the optimization problem is subject to certain parameters that need to be adhered to, the problem is called a constrained optimization problem; if it does not adhere to these parameters it is referred to as an un-constrained optimization problem. Furthermore, an optimization problem that has only bound constraints are considered to be unconstrained problems.

A problem that can be modeled in form is called a Nonlinear program (NLP). The field of nonlinear is complex, thus researchers have established a special case study. A very important case study is when the constraints are all $g_j(x)$ *and* $h_k(x)$ are all linear. Problems with such characteristics are called “linear constrained optimization”. When the objective function comes in a quadratic form, it is called “Quadratic programing”. A special case occurs when the objective function and constraints comes in linear form, they are called Linear Programming (LP).

A solution that satisfies the subset of the linear program is called a global optimum solution, and it is possible to have more than one global optimum solution for an optimization problem. Optimization problems also present a local optimal solution.

Figure 2.1 illustrates a problem with two Local optimal and one global optimal solution of an optimization problem. When a problem present more than one local optimal solution, they are called multi-modal optimization problem.

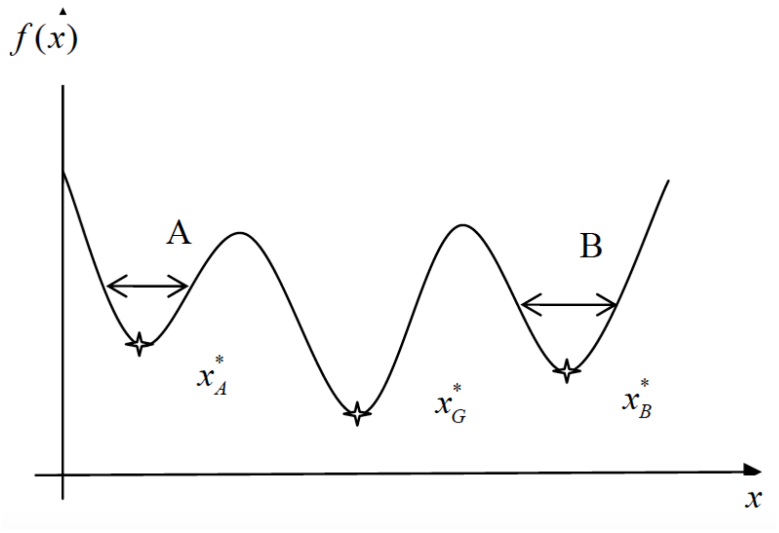


Figure 2.1: Local optima and Global optimum solution illustration

Many algorithms have been developed for finding the local optimum of an optimization problem. The most common algorithms include, Steepest decent, Quasi-Newton method, and interior-reflective Newton method. Most of the algorithms find the local optimum by starting with a point $\mathbf{x_0}$ and searching for the local optimum within the starting point.

In cases where the objective is to find the global optimum, this is achieved by repeating the process by starting at different point to achieve the best solution within the local optima. In situations where the optimization problem is highly complex with a large number of local optima solutions, it is obvious that some algorithms cannot satisfy the requirement, thus the global optimization algorithm is needed where it is possible to find the global optimum solution regardless of where the starting point $\mathbf{x_0}$ is located. The Evolutionary algorithm provides the solution for this problem.

2.2 Concept of Algorithm

The concept of algorithms is not a new phenomenon [10]. They were used by Greek mathematicians to find prime numbers in the sieve of Eratosthenes, while the Euclidean algorithm was used to identify the highest common divisor of two numbers. Effective algorithms make assumptions, show a bias toward a simple solution, trade off the costs of error against the cost of delay, and take chances [9]. An algorithm is defined as a series of steps used to solve a particular problem. This process doesn't just apply to computers and machines but also has a great influence to humans. An algorithm is a set of rules or process to be used in problem-solving operations like calculations, particularly by a computer. Processing of data is one of the basic functions of a computer. Algorithms are an essential part of computer data-processing.

Most computer programs have an algorithm that gives the basic instructions a computer should execute in an organized manner to complete a specified assignment, like printing students' report cards or calculating employees' paychecks, as well as the arrangement of complex data.

Dadaism espouses a strictly functional approach to humanity, assessing the value of human experiences based on their utility in data-processing mechanisms. If we develop an algorithm that fulfills the same function better, human experiences will lose their value. Thus, if we can replace not just taxi drivers and doctors but also lawyers, poets and musicians with superior computer programs, why should we care if these programs have no consciousness and no subjective experiences? If some humanist starts adulating the sacredness of human experience, Dadaists would dismiss such sentimental humbug [11]. ‘The experience you praise is just an outdated biochemical algorithm. In the African savannah 70,000 years ago, that algorithm was state-of-the-art. Even in the twentieth century it was vital for the army and for the economy. But soon we will have much better algorithms. Therefore, an algorithm can hereby be stated to be any operational sequence that can be simulated through a Turing-complete system [13][14][15].

Time, as well as storage is a factor that is theoretically required for a given algorithm to know its particular resource. The word 'algorithm' is rooted in a Latinized form of the name Muhammad ibn Musa al-Khwarizmi as an initial step towards algorisms. A number of different methods have been developed to analyze algorithms in an effort to achieve certain quantitative requirements. For example, virtually all classical mathematical algorithms can be described using a fixed number of English words. The performance of an algorithm is highly dependent on which trial vector generation strategy is chosen, and the values of the related parameters used in its operations knowing fully well the difficulty in selection of parameters that should be put in place [15]. In response to this, researchers developed certain techniques that allow them circumvent the problem of having to manually tune the control parameters. Searching every individual point presents an enumerative need for even moderately-complex problems to be broken down into smaller divisions to realize the best solutions. In reality, however, the overall time an algorithm needs to successfully complete its operation cannot be determined because it is not actually related to our traditional physical dimension. Algorithms related to information processing can typically read data from an input source, write it onto an output device, and store it for further processing. The data that is stored is considered an internal part of the entity executing the algorithm. The state is typically stored in at least one data structure.

2.2.1 Characteristics of Algorithms

An algorithm is defined as a series of steps used to perform a particular task. This applies to humans just as much as it does to computers and machines. The following characteristics must be present for an algorithm to be considered valid:

1. Finiteness; if the algorithm has not ended or reached a reasonable conclusion then it is really useless trying to apply it in solving a problem.

2. Another characteristic is that it should a have well-defined instructions, meaning that all the steps in the sequence should be unambiguously defined.

3. Lastly, it should be tremendously effective. The sequence must solve the required task. This should also be achievable by hand with just writing materials.

4. Uniqueness – at each step, the results must be uniquely defined and depend solely on the input and outcome of the previous step(s).

2.3 Evolutionary Algorithm

The Evolutionary strategies or Evolutionary Algorithm (EA) has become an important aspect of optimization methods for solving different search and optimize procedures. Most recent EA starts by creating a finite group of correspondence structures presented as population. The structures are identical, and together they form a generation of individuals. A string presents an individual which imitates a biological genotype. Decoding the genotype presents a result. The result contains parameters which solves the problem that is being optimized. The value corresponds to the preferred factor that most suitable for reaching the optimal or near-optimal status [16].

The concept behind the development of Evolutionary algorithm is from the natural habitat of survival of the fittest. Evolution of organisms are from two primary processes: reproduction and selection. The individual that survive and reproduce is decided form the selection process, while reproduction combines the genes of their offspring.

Evolutionary algorithm is implemented using stochastic search methods, emulating the metamorphosis evolution of biological organisms. In 1954, Barricelli, [17] simulated the evolution process. Subsequently, series of papers were published by Fraser [18] simulating the artificial selection of organisms. Bremermann [19] adopted a mutation, recombination and selection operators which form a basis for modern genetic algorithm. Holland [21, 22] popularized the technique by formally applying the EA in the adaptation in nature for applying the mechanisms in computer science.

Compared to the traditional optimization technique, the EA starts the optimization process with a population of possible solutions as opposed to a single point. At this stage, every individual in the population is evaluated. New offspring are then generated through mutation and recombination. The offspring with greater fitness values have a better chance of generating more offspring, which are then evaluated and individually selected for the subsequent generation. The process pushes the population to move towards regions where good solutions have been established.

Figure 2.2 presents a flow chart depicting a typical evolutionary algorithm.

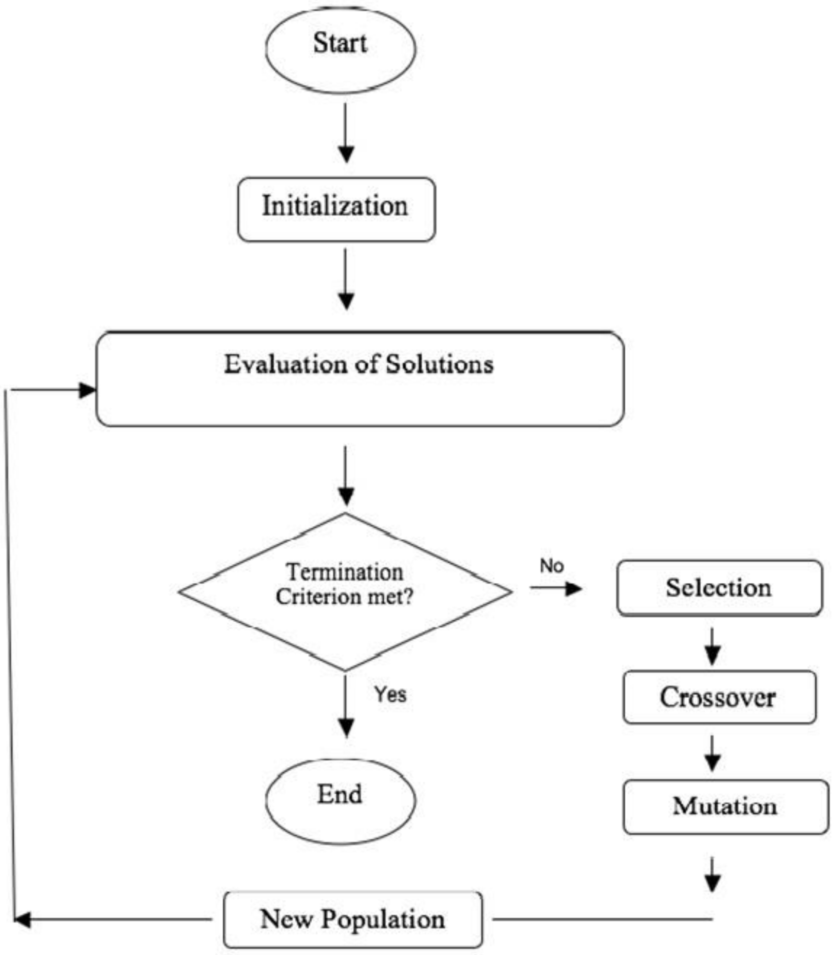


Figure 2.2: General Flow Chart of Evolutionary Algorithm [19].

The implementation procedure of EA is as follows: the generation step is broken down into recombination phase and selection phase. Figure 2.3 illustrates the division process. The strings are designated into adjacent slots during selection. The slots may be assigned randomly with the aim of shuffling the intermediate generation.

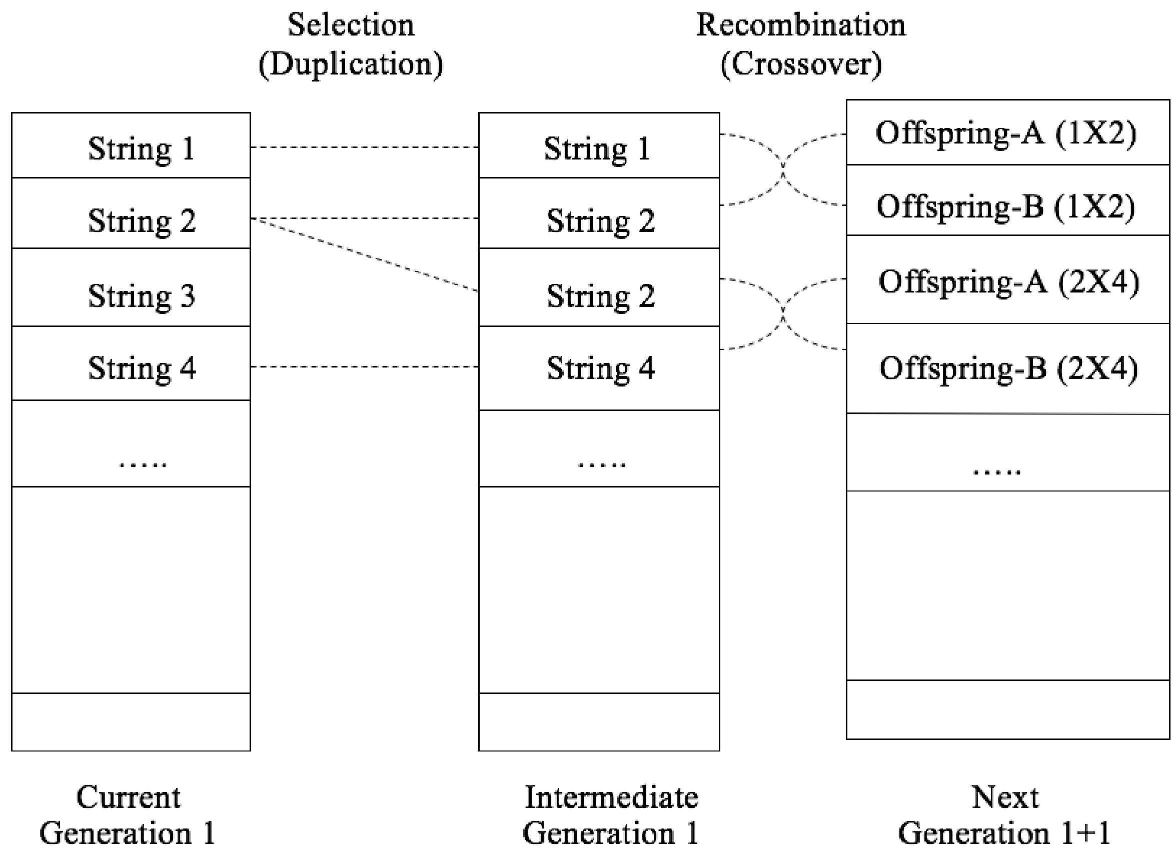
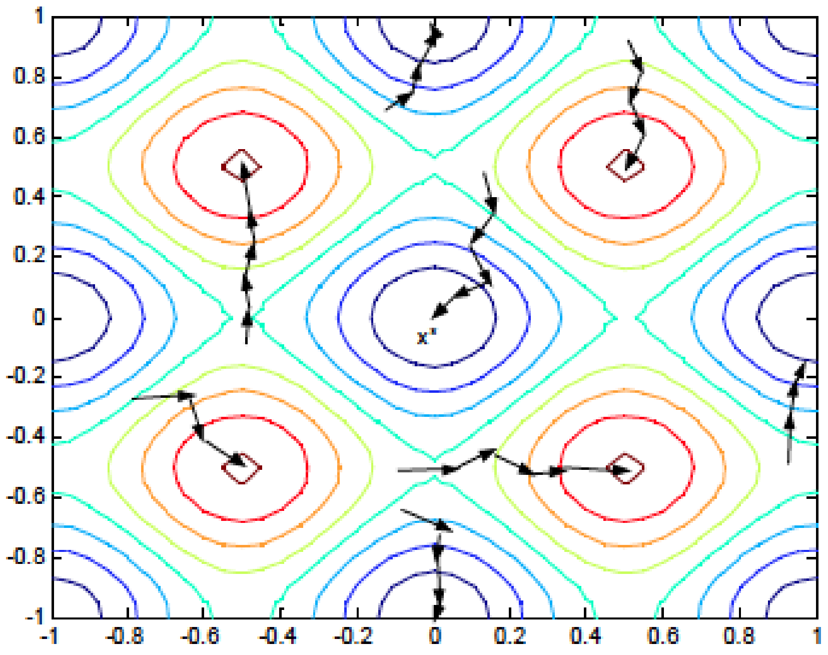


Figure 2.3: Recombination and Selection Phase of a Standard Evolutionary Algorithm

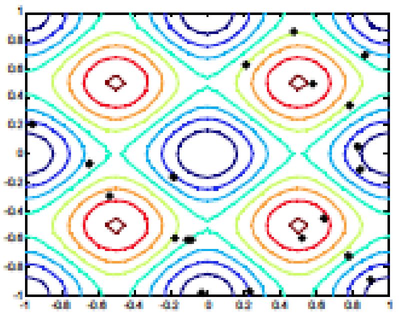
The survival of the fittest strategy produces the range with the highest fitness value. The mutation and recommendation operators, potential of large search space is created, thus, the consequences of initialization like the case of local search is minimized. EA evades the possibility of getting stuck in local search optimum when presented with a complex multi-model problem.

Figure 2.4 illustrates the search behavior of EAs and repeated local search. Another significant benefit of EAs is that they do not require detailed information on the problem’s rules, since they operate using their own rules. The evolution is executed through comparing the fitness values of the individuals. The output of a system or the results of experiments can be comparison criteria of EAs. Thus they can solve problems which do not have exact mathematic modals and cannot be solved using the traditional methods introduced before which use the first and second derivatives in the updating process. This is very useful for complex or loosely defined problems. At the same time, it eliminates the need to compute derivative information. Sometimes it is very time-consuming to numerically obtain derivative values for complex problems.

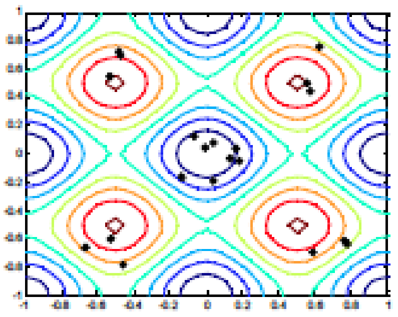
Step 1: its start with a random solution; step 2: for most problems a local search algorithm is readily available; step 3 the for the perturbation, a random move in a neighborhood of higher order than the one used by the local search algorithm can be surprisingly effective; and step 4: a reasonable first guess for the acceptance criterion is to force the cost to decrease, corresponding to a first-improvement descent in the set.



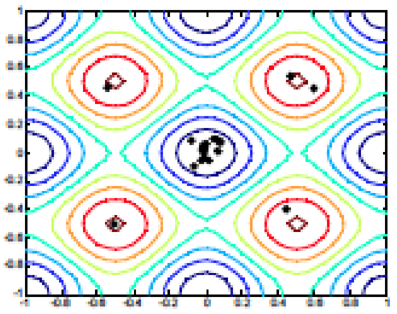
(a) Repeated Local Search



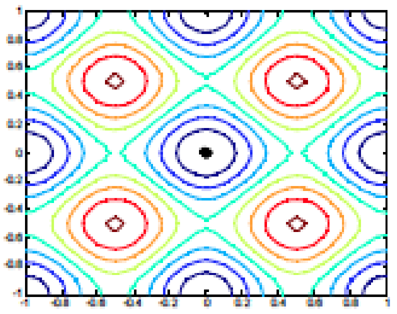
(b-1) EA Step 1



(b-2) EA Step 2



(b-3) EA Step 3



(b-4) EA Step 4

Figure 2.4: Evolutionary Algorithm vs. Repeated Local Search [22].

Generally, there are three main Evolutionary algorithm methods: Genetic algorithms (GAs) which was initially introduced by Holland [20]. In this algorithm, recombination plays an important role, while mutation acts as an assistant operator. Evolutionary programming (EPs) was developed by Fogel [23] in relation to evolving finite state machines. The Evolutionary Strategies (ES) introduced by Rechenberg and Schwefel [24]. Contrary for Gas, the main operator in ES is mutation, while recombination is the assistant operator. Koza [25] introduced and popularize the Genetic programming (GP). The primary distinction between GP and EA is the representation of solutions.

2.4 Single Objective Optimization

Single objective numerical optimization is a new approach which is widely compared to other numerical and evolutionary optimization techniques in several engineering optimization problems with numerous types of constraints. With proper and accurate examination by researchers, it was proven that this recent approach can continually outperform the other techniques by making use of relatively small sub-populations, and also regardless of any significant sacrifice in terms of performance. Single objective would be the opposite of multi-objective optimization. In other words, standard optimization with a single objective function. Multi-objective optimization means optimizing at least two conflicting objectives in relation to a set of certain constraints [26].

For this process of single-objective numerical optimization, the problem mainly concerns either minimizing or maximizing the objective function relatively based on a unit or single variable while accounting for a constraint or in an unconstrained task. The single objective numerical optimization problems consists of only one variable in the objective function. This may not be constant because there is a possibility of variations in different values of the given variable. They may vary in relative or local minimum, relative or local maximum, absolute or global minimum and absolute or global maximum.

The objective function of a single-objective optimization problem is written as ($f(x')$), which must be minimized or maximized using a number of constraints ($g\ (x')$). The general form of the formula is contained in Equation (1) as:

Minimize

$f(x')$ s.t. $g_j(x') \geq 0$

where ($j = 1,\ldots,m$) (1) $x' \in X \subset R_n$

where x' is a vector of n decision variables, $x' = (x1, x2, \ldots, xn)^T$, and X represents a feasible region. This above formula is a standard or general formula as it may be used to solve single objective numerical optimization problems with various objective functions.

2.5 Multi-Objective Optimization

In Multi-objective numerical optimization, two or more conflicting objectives are optimized simultaneously while adhering to a predetermined set of constraints. Meanwhile, in reality, the problems experienced in most cases which helps in improving one objective often cause another to degrade. Network analysis, bioinformatics, oil and gas, automobile design, aircraft design, and product & process design, are a few fields where we can see the impact and application of multi objective numerical optimization.

Optimization problems have the following characteristics:

1. There should be different decision alternatives.

2. The number of potential decision alternatives should be limited by additional constraints.

3. On the evaluation criteria, there should be different effects taken by each decision alternative.

4. The basic elements of an optimization, including decision variables, objective, and constraints.

5. Solving simultaneous equations and other constraint satisfaction problems.

Equation (2) outlines multi-objective optimization problems with a number of objective functions:

(f (x') = (f_1 (x'); f_2 (x'), … , f_k (x'))^T) can be stated as follows:

Minimize

f (x') = (f_1 (x'); f_2 (x'), … , f_k (x'))^T

s. t. g_j (x') ≥ 0

where (j = 1,…,m) (2) $x' \in X \subset R_n$

This above formula is a standard or general formula as it may be for solving multi-objective numerical optimization problems with different objective functions. In multiple objective optimization, there is a Pareto-optimal solution set. And consequently, there is a tendency of weight addition to make a tradeoff or substitution between the criteria. This shows an attribute of single objective does from the beginning onset compared to that of multi objective.

It is usually proclaimed or assumed by myth that the multi-objective numerical optimization is only suitable for problem with multiple objectives while single objective optimization is believed to be responsible for tackling problems related to single objective. Meanwhile in reality both single and multiple objective numerical optimizations can be used to solve problems with multiple objectives.

In this case, a researcher might successfully compute a very small and short set of solutions and hence present all of his findings to the decision maker who in turn makes a choice one of which can then be applied or implemented. So, in real term the researcher is not always the decision maker. A very long time can be taken to solve a particular problem but it's a fact now that the specific instance to accomplish the solution to the problem is not known long before a decision has to be made unlike some situations where the results can be affirmed even when the process has not been concluded. Apparently, when input parameters are achieved, it is only required to crosscheck the process to attain the best possible solution.

However, there are advanced techniques that can be used for multi-objective optimization problems other than the few usual common techniques used for both single and multi-objective numerical optimization as they contain multi-dimensional objectives to be satisfied. These different optimization techniques can be categorized as calculus-based techniques or numerical methods, random techniques and enumerative techniques. These techniques have made processes more advanced and comprehensible leading to solving problems accurately with lesser time to achieve good results.

2.6 Basic Operators of Multi-Objective Evolutionary Algorithm

The complexity which is the work done by an algorithm, is determined by the number of the basic operations necessary for tackling its problems. The complexity of a program that implements an algorithm is assumed to be the same, but not exactly the same as the complexity of the algorithm. By this, it means that algorithms independent on any particular implementation - programming language or computer used in actualization of the operation. Therefore, all attention will be focused mainly with the basic operations which are the number of times the basic operations have to be run depending on the available size of input.

Consequently, with the application of several theories it is concluded that the total number of steps in the operation is averagely proportional to the number of the basic operations. Studies have shown that an algorithm that is required to execute a given task usually make use of some available set of basic operations. With the possibility of estimating the amount of work done by an algorithm we usually do not take to consideration all the necessary steps like initializing certain variables and all other criteria. The main benefit of these proposed algorithms is the creation of a computer-based agent with the capability to mimic intelligent human decision-making in a dynamic game environment. This has many benefits for real-world problems provided the techniques are successfully applied, like in the application of complex systems.

The complexity which is the work done by an algorithm, is determined by the number of the basic operations necessary for tackling its problems. The complexity of a program that implements an algorithm is assumed to be the same, but not exactly the same as the complexity of the algorithm. By this, it means that algorithms independent

on any particular implementation - programming language or computer used in actualization of the operation. Therefore, all attention will be focused mainly with the basic operations which are the number of times the basic operations have to be run depending on the available size of input.

2.7 Adaptive Differential Evolutionary Algorithm

Storn and Price [27] introduced the Differential Evolution algorithm. Subsequently R. Storm presented an application of DE in designing IIR-Filter. Several applications of DE are still introduced such as, P. Thomas and D. Vernon [28] in image registration. Majority of the application of DE are directed towards image processing. In 1998, a hybrid DE algorithm was introduced to initiate recognition of DE remarkable performance or solving engineering problems Bertsimas and Tsitsiklis [6].

Differential evolution is a relatively basic, yet efficient approach to numerical optimization [29]. The bulk of recent work has been geared towards the development of more adaptive mechanisms for differential evolution because the search efficiency of differential evolution is greatly dependent on its control parameter settings. Abbass et al. [30] introduced a Pareto-frontier Differential Evolution algorithm (PDE) to solve multi objective problem through the incorporation of Pareto dominance.

The initial author later self-adapted the crossover rate of Pareto-frontier Differential Evolution algorithm; encoding each individual with the crossover rate allowed them to evolve simultaneously with other parameters. To attain the most suitable performance through the application of conventional differential evolution to a particular problem, it is advisable to perform a routine trial-and-error search or approach for the ideal trial vector generation strategy and continually adjust the values of the related control parameter. Thus, different trial vector generation strategies have a greater level of effectiveness when combined with specific control parameter values during different stages of evolution.

Differential evolution algorithm has proven to be a basic yet effective evolutionary algorithm for numerous optimization problems with applications in the real-world. This has given rise to check and actualization by many researchers so that they can be in relative term with general observations. A very tedious optimization trial is required in the trial-and-error method to adjust the control parameters, even for an algorithm whose parameters are fixed throughout the evolutionary search. The convergence performance of adaptive differential evolution algorithms have been found to be faster and more reliable than classic differential algorithms lacking parameter control for several benchmark problems. Moreover, the population of the differential evolution may switch regions within the search space over the course of the evolution, which increases the effectiveness of certain strategies associated with specific parameter settings. Therefore, it is necessary to identify inappropriate strategies and their associated parameter values at various stages of the evolution/search process in an adaptive manner. The algorithm operates using computational steps similar to those employed by a standard and classical evolution algorithm. In differential evolution, contour matching is known to be the most important feature, which means that the generation population takes up its operation in such way that the regions of the objective function surface are checked automatically as soon as they are noticed [21].

The adaptive numerical optimization procedure is used to obtain a discrete variable solution. Basically, numerical optimization techniques can be categorized as gradient-based and no gradient algorithms.

The former often lead to local optimum while the latter on the other hand converge to a global optimum but the two variables usually require a great deal of function evaluations. Numerical optimization depicts a wide range of understanding and an up-to-date explanation of the most effective and essential methods in continuous optimization. Thus, it is a response to the growing interest in optimization found in science, engineering, and business with a main focus on the methods that are most suitable for practical problems. The majority of algorithms for nonlinear optimization problems only seek out a local solution: the point at which all other viable nearby points are larger than the objective function.

They are sometimes unable to identify the global solution – that is, the point with lowest function value. Despite being necessary for many applications, global solutions are often difficult to identify and even more difficult to locate. Local solutions double as global solutions for convex programming problems, most especially for linear programs.

Both constrained and unconstrained general nonlinear problems can sometimes have local solutions that are not global solutions. From the analysis of Advances in Feedstock Conversion Technologies for Alternative Fuels and Bio products, 2019 Numerical optimization was performed using Design Expert 8.0.4 software to maximize extracted oil yield within the range set for each parameters, and a second-order polynomial equation was used to model the relationship between extract yield and the process parameters, as shown in the following equation:

$Y=\beta_0+\sum_{i=1}^{n}\beta_i x_i+\sum_{i=1}^{n}\beta_{ii}\, x_{i2}+\sum_{i=1}^{n}-1\sum_{i=1}^{n}\beta_{jl}\, x_i x_j.$ [50]

Where Y is the response variable (extract yield), x_i and x_j are coded independent variables, and β_0 ,β_i ,β_{ii} and β_{ij} are coefficients of intercept, linear, quadratic and interaction terms, respectively. A major benefit of optimization numerical methods is that it is possible to find a numerical solution for a problem even when it does not have an analytical solution. Furthermore, a numerical method only utilizes the evaluation of standard functions and their operations: addition, subtraction, multiplication and division. The advantage of numerical optimization is that it can be applied to a wide range of problems. It has also proven to be effective for reaching a local optimum for a smooth, continuous objective function.

Chapter 3

METHODOLOGY

This study employs an adaptive DE algorithms technique of metaheuristic evolutionary algorithm with a new mutation strategy to examine the results of single-objective problem optimization and multi-objective problem optimization. Adaptive DE algorithms have been used by different researchers during the past ten years. While DEs have established their capacity for exploring large search spaces, they are relatively less efficient in their ability to fine-tune the solution. This disadvantage is typically overcome through the application of local optimization algorithms to individuals in the population [31].

Adaptive Differential Evolution with Additional External Archive: JADE is proposed to enhance optimization performance in the implementation of a new "DE/current-to-pbest" mutation strategy, which includes an additional external archive and the adaptive update of control parameters. The "DE/current-to-pbest" is a simplification of the typical "DE/current-to-best," while historical data is used by the additional archive process to offer solutions on progress diversity. Both convergence performance and the diversity of the population are enhanced by the two operations.

The control parameters are automatically updated to the appropriate values through parameter adaptation, which avoids relying on outdated information regarding the relationship between the characteristics of the optimization problems and the parameter settings [32].

3.1 Differential Evaluation Operation

This section outlines the operation of differential evolution and presents the notations and terminology necessary to adequately understand the various adaptive evolutionary algorithms.

Differential evolution shadows the typical path for the algorithm. The uniform distribution $X_j^{low} \leq x_{j,i,0} \leq X_j^{up}$, for j =1,2,...,D is used to randomly generate the first population $\{x_{i,0}=(x_{1,i,0},x_{2,i,0},...,x_{D,i,0})\ |i=1,2,...,NP\}$, where D and NP represent the dimension of the problem and population size, respectively. Following initialization, DE enters an evolutionary-operation loop: mutation, crossover, and selection [32].

Mutation: Mutation vectors $V_{i.g}$ are created at each generation g using the existing population for offspring $\{x_{i,g}\ |\ i=1,2,...,NP\}$. Mutation strategies often found in the literature include:

1) “DE/rand/1”

$$V_{i,g}=X_{r0,g}+F_i\ .\ (X_{r1,g} - X_{r2,g}) \tag{3.1}$$

2) “DE/current-to-best/1”

$$V_{i,g}=X_{r0,g}+F_i\ .\ (X_{r1,g} - X_{ri,g}) + F_i\ .\ (X_{r1,g} - X_{r2,g}) \tag{3.2}$$

3) “DE/best/1”

$$V_{i,g}=X_{best,g}+F_i\ .\ (X_{r1,g} - X_{r2,g}) \tag{3.3}$$

Where the r0, r1 and r2 indices are separate integers uniformly selected from the set{1,2,...,NP}\{i}, $Xr_{1,g}$-$Xr_{2,g}$ is a difference vector used in the mutation of the matching offspring $X_{j.g}$, $X_{best,g}$ best vector for generation g, and F_i represent mutation with a range typically on the period (0,1+).

F_i =F is a static parameter utilized in classic DE to produce the mutation vectors in every generation, while each individual (i) in numerous adaptive DE algorithms is linked to its particular mutation factor F_i. The generalization of the mutation strategies outlined above is possible through the implementation of multiple difference vectors different from $Xr_{1,g} - Xr_{2,g}$, which results in a strategy known as "DE/–/k" depending on how many difference vectors are utilized.

It is noteworthy that some trial vector components might be in violation of predefined boundary constraints. This problem can be overcome through the use of penalty schemes, resetting schemes, or other possible solutions. For constrained problems, we employ a simple method in which the violating components are set in the middle of the violated bounds and the corresponding components of the parent individual [33].

$$\begin{aligned} v_{j,i,g} &= (x_j^{\text{low}} + x_{j,i,g})/2, \text{ if } v_{j,i,g} < x_j^{\text{low}} \\ v_{j,i,g} &= (x_j^{\text{up}} + x_{j,i,g})/2, \text{ if } v_{j,i,g} > x_j^{\text{up}} \end{aligned} \tag{3.4}$$

Where the jth component of the mutation vector $V_{i,g}$ and the offspring vector $X_{i,g}$ at g generation are represented by $V_{j,i,g}$ and $X_{j,i,g}$. when the best solution is either on or close to the boundary This technique will be particularly effective.

Crossover: a CR process is used in producing the last trial offspring vector $U_{i,g}$ = ($U_{1,i,g}$,$U_{2,i,g}$,...,$U_{D,i,g}$) after the mutation

$$U_{j.i.g} = \begin{cases} v_{j,i,g}\ , if\ rand\ (0,1)\ < CR_i\ or\ j = j_{rand} \\ x_{j,i,g}\ otherwise \end{cases} \tag{3.5}$$

Where the rand (a, b) it's an independently generated, random uniform number at interval [a, b] For all j and i, j_{rand} =randint (1, D) is a random number selected from 1 to D and regenerated for every i, and crossover rate CR_i ∈ [0,1] somewhat parallels mean section for components vector derived as of the Mutation coefficient. In normal differential evolution ,CR_i =CR is a generation static parameter and used in trial coefficient in every generation, while individual i's in many adaptive evolutionary algorithms each correspond to their own crossover rate CR_i.

Selection: selection process chooses the superior between offspring $X_{i,g}$ and the trial $U_{i,g}$ vectors using F values F(x). e.g., the chosen vector in a minimization problem can be given as:

$$X_{i.g+1} = \begin{cases} u_{i,g}\ ,\ if\ f\ (u_{i,g})\ < f(x_{i.g}) \\ x_{i,g}\ \ ,otherwise \end{cases} \tag{3.6}$$

which then becomes the offspring vector in the following generation And The if the trial variable $U_{i,g}$ is superior to the offspring $X_{i,g}$ procedure in (3.6) can be considered a success, thereby indicating a positive evolution progress Δ i,g =$f(X_{i,g})$–$f(U_{i,g})$. Consequently, the control parameters F_i and $CR_{\ i}$ utilized in generating $U_{i,g}$ are respectively known as the successful mutation factor and crossover probability. Outlined above is a one-to-one selection process normally fixed in various differential evolutionary algorithms, although the crossover can present in variations different from the binomial operation in (3.5). As such, the naming of classic DE algorithms is based in history; e.g., "DE/rand/1/bin", is indicative of its "DE/rand/1" binomial crossover process and mutation technique.

3.2 Pseudo-Code for JADE in Single Objective Optimization Problems

This section provides a brief overview of recent adaptive DE algorithms involving the dynamic update of control parameters over the course of the evolutionary search. It outlines the convenient adaptation mechanism proposed in JADE, which is a novel DE algorithm, used in implementing the “DE/current-to-pbest” mutation strategy with an additional archive and adaptive control of F and CR. JADE implements identical crossover process and the selection process as defined in (3.5) and (3.6).

3.2.1 DE/current-to-pbest

“DE/rand/1” it’s the first one type of DE mutation technique (3.1), (3.2), and is widely accepted in the literature as the most successful scheme [34]. However, “DE/best/2” might better than “DE/rand/1”, even as “DE/best/1” is preferred for many official problems [35]. Furthermore, it has been argued in [36] that incorporating some   best solution info and the use of “DE/current-to-pbest/1” can be beneficial for an algorithm.

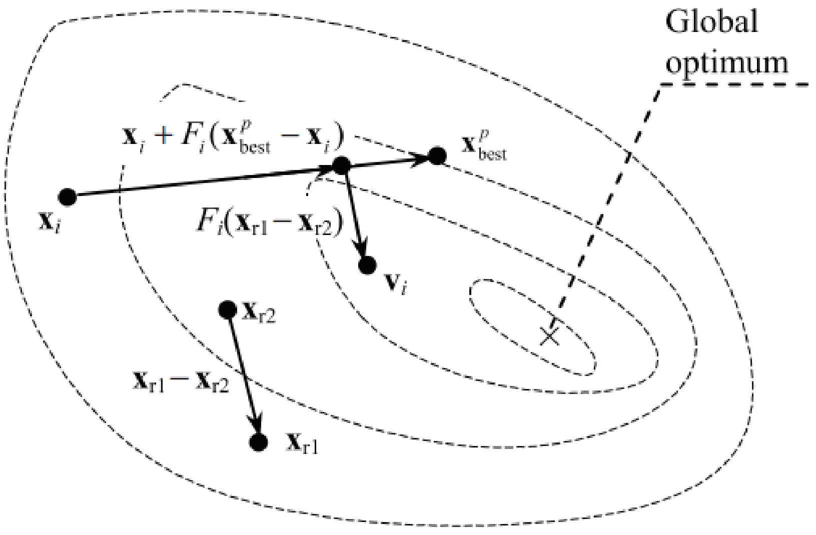

Global optimum
$\mathbf{x}_i + F_i(\mathbf{x}^p_{\text{best}} - \mathbf{x}_i)$
$\mathbf{x}^p_{\text{best}}$
$\mathbf{x}_i$
$F_i(\mathbf{x}_{r1} - \mathbf{x}_{r2})$
$\mathbf{v}_i$
$\mathbf{x}_{r2}$
$\mathbf{x}_{r1} - \mathbf{x}_{r2}$
$\mathbf{x}_{r1}$

Figure 3.1: “DE/current-to-pbest/1” mutation technique [32].

The figure above illustrates the “DE/current-to-pbest/1” mutation strategy implemented in JADE and the Optimization problem outlines are represented by dashed curves. V_i represents the mutation variable created for the individual X_i through related F_i mutation factor uniformly chosen form the set {1,2,….,NP}\{i} and F_i is the mutation factor that associate with X_i and is re-generate at each generation by the adaptation process introduced later in (3.5).DE/current-to-pbest is indeed a generalization of DE/current-to-best .any of the 100p% solution can be randomly chosen to play the role of the single best solution in DE/current-to-best while inferior solution when compared to the current population provide additional information about the promising progress direction .Donta A as the set of archived inferior solution and P as the current population.

The incorporation of the solution in the DE search speeds up convergence. However, this information could also have adverse effects, such as premature convergence as a result of the reduction in the diversity of the population. Due to the faster but not as consistent divergence performance of prevailing technique (3.3) and (3.4), a novel mutation technique, known as "DE/current-to-pbest" with the additional archive, is suggested as the possible for adaptive algorithm that suggested in this thesis.

As Figure 3.1 illustrates, a mutation vector "DE/current-to-pbest/1" (without archive) that generated using:

$$V_{i,g}=X_{i,g}+F_i \ . (X_{best,g}^{p} - X_{i,g}) + F_i \ . (X_{r1,g} - X_{r2,g}) \tag{3.7}$$

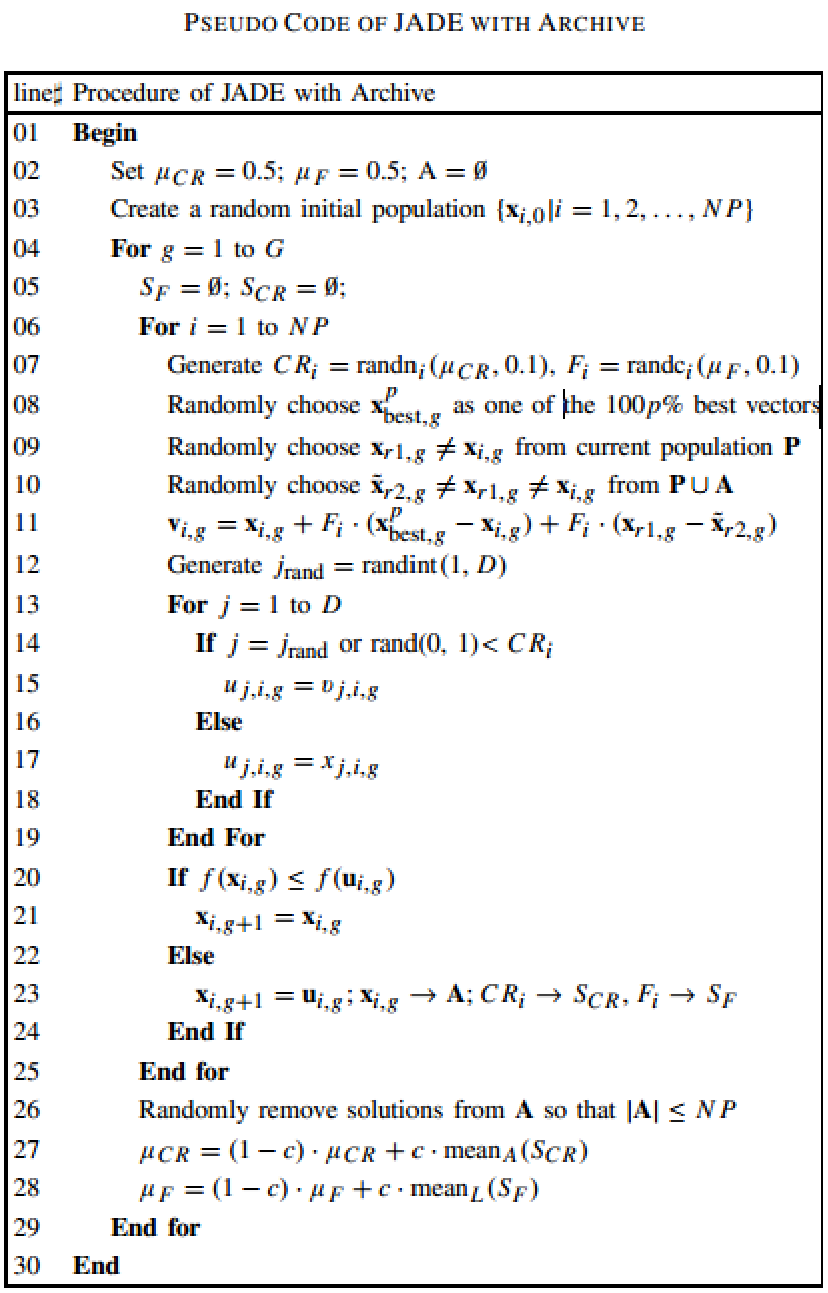
where $X_{best,g}^{p}$ is selected randomly as one of the top 100p% individuals in current population with p ∈ [0,1], with F_i denotes the xi-associated mutation rate, which it's re-generated through an adaptation technique described later in (3.11) at each generation. "DE/current-to-pbest" is essentially a general form of "DE/current-to-best". In "DE/current-to-best", any more than 100p% sets can be selected at random to perform the function of single optimal set.

Additional information regarding the likely progress direction can be provided by comparing recently-explored inferior solutions to the existing population. With A denoting the set in archived low-grade sets and P as the existing population, The mutation vector in “DE/current-to-pbest/1” with archive its build as follows:

$$V_{i,g} = X_{i,g} + F_i \ . \ (X^{p}_{best,g} - X_{i,g}) \ + \ F_i \ . \ (X_{r1,g} - \tilde{X}_{r2,g}) \qquad (3.8)$$

where $X_{i,g}$, $X_{r1,g}$ and $X^{\ p}_{best,g}$ “ they selected by **P** in a manner similar to (3.7), while $\tilde{X}_{r2,g}$ is selected at random from the union of P and P (**P∪A**).

The operation of the archive is simplified to reduce the amount of computation required. Initially empty, the archive is filled with parent solutions that did not pass the selection process in (3.6) after each generation. Once it reaches a particular threshold, NP, solutions are removed from the archive at random to keep its size from surpassing said threshold. (3.7) is evidently a unique case of (3.8) in that the size of the archive is set at zero.



PSEUDO CODE OF JADE WITH ARCHIVE

```
line# Procedure of JADE with Archive
Begin
    Set $\mu_{CR} = 0.5$; $\mu_F = 0.5$; $A = \emptyset$
    Create a random initial population $\{\mathbf{x}_{i,0} | i = 1, 2, \ldots, NP\}$
    For $g = 1$ to $G$
        $S_F = \emptyset$; $S_{CR} = \emptyset$;
        For $i = 1$ to $NP$
            Generate $CR_i = \text{randn}_i(\mu_{CR}, 0.1)$, $F_i = \text{randc}_i(\mu_F, 0.1)$
            Randomly choose $\mathbf{x}^p_{\text{best},g}$ as one of the $100p\%$ best vectors
            Randomly choose $\mathbf{x}_{r1,g} \neq \mathbf{x}_{i,g}$ from current population $\mathbf{P}$
            Randomly choose $\tilde{\mathbf{x}}_{r2,g} \neq \mathbf{x}_{r1,g} \neq \mathbf{x}_{i,g}$ from $\mathbf{P} \cup \mathbf{A}$
            $\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F_i \cdot (\mathbf{x}^p_{\text{best},g} - \mathbf{x}_{i,g}) + F_i \cdot (\mathbf{x}_{r1,g} - \tilde{\mathbf{x}}_{r2,g})$
            Generate $j_{\text{rand}} = \text{randint}(1, D)$
            For $j = 1$ to $D$
                If $j = j_{\text{rand}}$ or rand(0, 1)< $CR_i$
                    $u_{j,i,g} = v_{j,i,g}$
                Else
                    $u_{j,i,g} = x_{j,i,g}$
                End If
            End For
            If $f(\mathbf{x}_{i,g}) \leq f(\mathbf{u}_{i,g})$
                $\mathbf{x}_{i,g+1} = \mathbf{x}_{i,g}$
            Else
                $\mathbf{x}_{i,g+1} = \mathbf{u}_{i,g}$; $\mathbf{x}_{i,g} \rightarrow \mathbf{A}$; $CR_i \rightarrow S_{CR}$, $F_i \rightarrow S_F$
            End If
        End for
        Randomly remove solutions from $\mathbf{A}$ so that $|\mathbf{A}| \leq NP$
        $\mu_{CR} = (1 - c) \cdot \mu_{CR} + c \cdot \text{mean}_A(S_{CR})$
        $\mu_F = (1 - c) \cdot \mu_F + c \cdot \text{mean}_L(S_F)$
    End for
End
```

Figure 3.2: Pseudo JADE Code with additional Archive [32].

The archive offers solutions regarding the progress direction and also has the capability to improve population diversity. Furthermore, as the adaptation process parameters (3.11) and (3.13) later show, the best F values that help enhance the diversity of the population are encouraged.

Consequently, despite its bias in the way of a possible optimum (potentially a local optimum mini), “DE/current-to-pbest/1” proposed cannot be trapped to a local minimum. From a comparative standpoint, the search region of “DE/rand/1” is relatively small and has no apparent biases, while the search region of “DE/current-to-pbest/1” with an archive is quite large and has biases towards favorable progress directions. Figure 3.2 contains the pseudo code of JADE.

3.2.2 Parameter Adaptation

The crossover probability CR_i of every individual X_i in each generation g is generated independently in accordance with a regular distribution having a mean of μCR, 0.1 standard deviation

$$CR_i = randn_i(\mu\ c\ r\ ,0.1) \tag{3.9}$$

And after that it’s become to [0, 1]. S_{CR} the arrangement of success CR_i ’s rates in g generation. μCR is initially set at 0.5 and subsequently will updated after finishing every generation using the following equation:

$$\mu\ CR = (1 - c) . \mu\ CR + c . \mathrm{mean}_A\ (S_{CR}) \tag{3.10}$$

Where c is positive constant between 0 to 1 and $\mathrm{mean}_A(\cdot)$is the usual arithmetic mean. In a similar manner, the mutation rate F_i for every individual x_i in every generation g is generated independently based on a Cauchy distribution with location parameter μF and scale parameter 0.1

$$F_i = \mathrm{randc}_i\ (\mu\ \mathrm{F}, 0.1) \tag{3.11}$$

And then regenerated if $F_i \leq 0$ or truncated to 1 if $F_i \geq 1$. S_F denotes the set of effective mutation rates in generation g.

The position of parameter μF of the Cauchy distribution is initially set at 0.5 and every time updated following in every generation as

$$\mu \text{ F} = (1 - \text{c}) \,.\, \mu \text{ F} + \text{c} \,.\, \text{mean}_L \,(S_F) \tag{3.12}$$

And the $\text{mean}_L(\cdot)$ its represent the Lehmer mean

$$mean_L(S_F) = \frac{\sum_{F \in S_F} F^2}{\sum_{F \in S_F} F} \tag{3.13}$$

3.2.3 Descriptions for Parameter Adaptation

μCR adaptation is guided by the notion that the individuals generated by better control parameters have a greater chance of survival and should therefore be propagated to subsequent generations. In practice, this involves recording the previses success crossover rate and using them as a controller when generating new CR_i's. The standard deviation in (3.9) is set to a fairly minor figure to ensure that the adaptation functions efficiently; for example, in the rare instance of an infinite standard deviation, the value of μCR will not affect the truncated normal distribution. Relative to CR, the adaptation of μF involves two unique operations. In the first, F_i's are generated in accordance with a truncated Cauchy distribution.

Unlike normal distribution, a Cauchy distribution is better able to improve the diversity of mutation variable , thereby preventing convergence from occurring prematurely as is often the case with mutation technique (such as “DE/best”, “DE/current-to-best”, and “DE/current-to-pbest”) when mutation variable are overly focused nearby to definite value.

In the second operation, μF adaptation attaches greater significance to bigger effective mutation rate through Lehmer mean as it’s declared in (3.13), as opposed to an mathematics mean, which is the case in μCR adaptation.

As a result, Lehmer mean also useful in the propagation of large mutation factors, which leads to improvements in the progress rate. In contrast, the mutation factor's optimal value tends to be larger than the arithmetic mean of S_F , which results in a smaller μF value and ultimately premature convergence. The lesser μF is expected to the inconsistency between the rate of success and the rate of progress in DE search. In fact, an important similarity between the "DE/current-to-pbest" with small F_i and a (1+1) evolution strategy (ES) [37] is that the offspring generated by both tend to be in the area around the base vector. The (1+1) ES typically has a higher probability of success the smaller the mutation variance (a proven fact for sphere and corridor functions [37], [38]), although a near-zero variance value results in a trivial evolution progress. One way to overcome this shortcoming involves placing greater significance on successful mutation rate to speed up the evolutionary search process, which is both simple and effective.

In regards to (3.10) and (3.12), parameter adaptation does not occur is the constant c = 0. Moreover, a successful CR_i or F_i has a lifespan of approximately 1/c generations and so the previous estimation of μCR or μF is decreased by "$(1-c)^{1/c}$→1/e≈37%", at c is near enough to zero.

3.2.4 Parameter Settings to Discussion

F and CR are the two control parameters in classic DE that need to be user-adjusted. These problem-dependent parameters require a relatively tiresome process of trial and error to determine their appropriate value for specific problems. In contrast, the new c and p parameters in JADE are not particularly sensitive to different problems based on their role in JADE: c regulates parameter adaptation, while p regulates how greedy the mutation strategy is. The best JADE results are usually found under the following conditions:

$1/c \in [5, 20]$ and $p \in [5\%, 20\%]$; i.e., the μCR and μF values have a lifespan with a range of 5-20 generations, and top 5–20% high-quality solutions in the mutation are considered [32].

3.2.5 Adding External Archive to JADE

The algorithm proposed here uses both internal population and external archive.

Using a decomposition-based strategy, it is able to allow its working population evolve while maintaining the external archive using domination-based sorting. The information extracted from the external archive is utilized in the generation of a new solution using genetic operator, first solution selected from the archive using roulette wheel method and by select one solution from archive and second solution will generate randomly then modified both solution by crossover or mutation.

3.2.6 Update External Archive

The external archive provides information about the progress direction and is also capable of improving the diversity of the population , **A** denote as the set of archive inferior solution and **P** as set of successful current population archive .

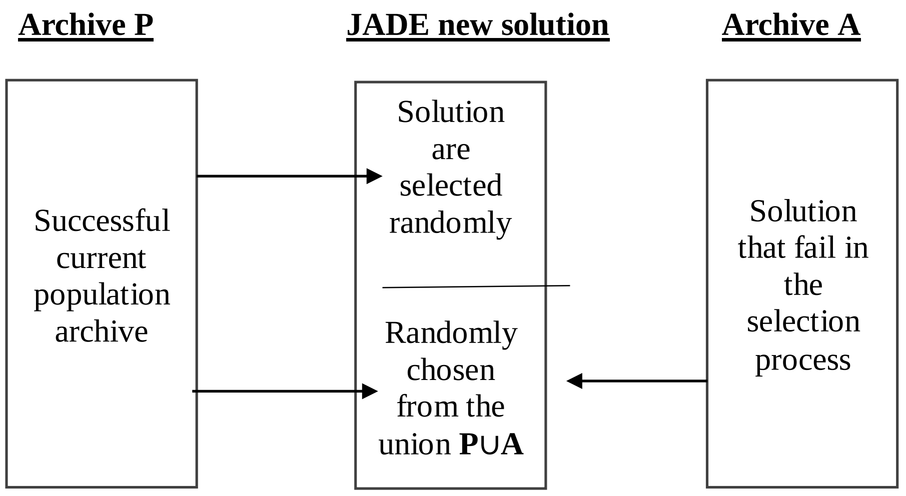

Archive P
JADE new solution
Archive A
Successful current population archive
Solution are selected randomly
Randomly chosen from the union **P**∪**A**
Solution that fail in the selection process

Figure 3.3 Illustrate Updating the Archive in JADE Algorithm

where the new solutions are built by using both archive some solution are selected randomly as one of the top individuals form current population **P** while the rest is randomly chosen from the union **P**∪**A** of the current population and the archive . The archive operation is made very simple to avoid significant computation overhead The archive is initiated to be empty. Then, after each generation, the parent solutions that fail in the selection process are added to the archive. If the archive size exceeds a certain threshold, say NP,then some solutions are randomly removed from the archive to keep the archive size at NP.

3.3 JADE for Multi-Objective Problems

MOO has real-world uses in various fields, including finance, engineering, and science [39], due to the direct links between the outcomes of the optimization and cost prices, profit margins, and other factors that affect safety, performance, and the environment, amongst others. There is no simple way to compare different outcomes using a single dimension due to the involvement and competition between numerous non-commensurable objectives/criteria.

For example, financial managers need to account for both risk and return when making investment decisions; air traffic controllers need to balance between satisfying the preferences of stakeholders and reducing systemic airspace congestion. The presence of such multiple (and often conflicting) objectives in real-world decision-making applications presents a unique challenge to researchers. This necessitates that more effort, beyond conventional techniques such as linear and nonlinear programming, be put into solving such problems [38].

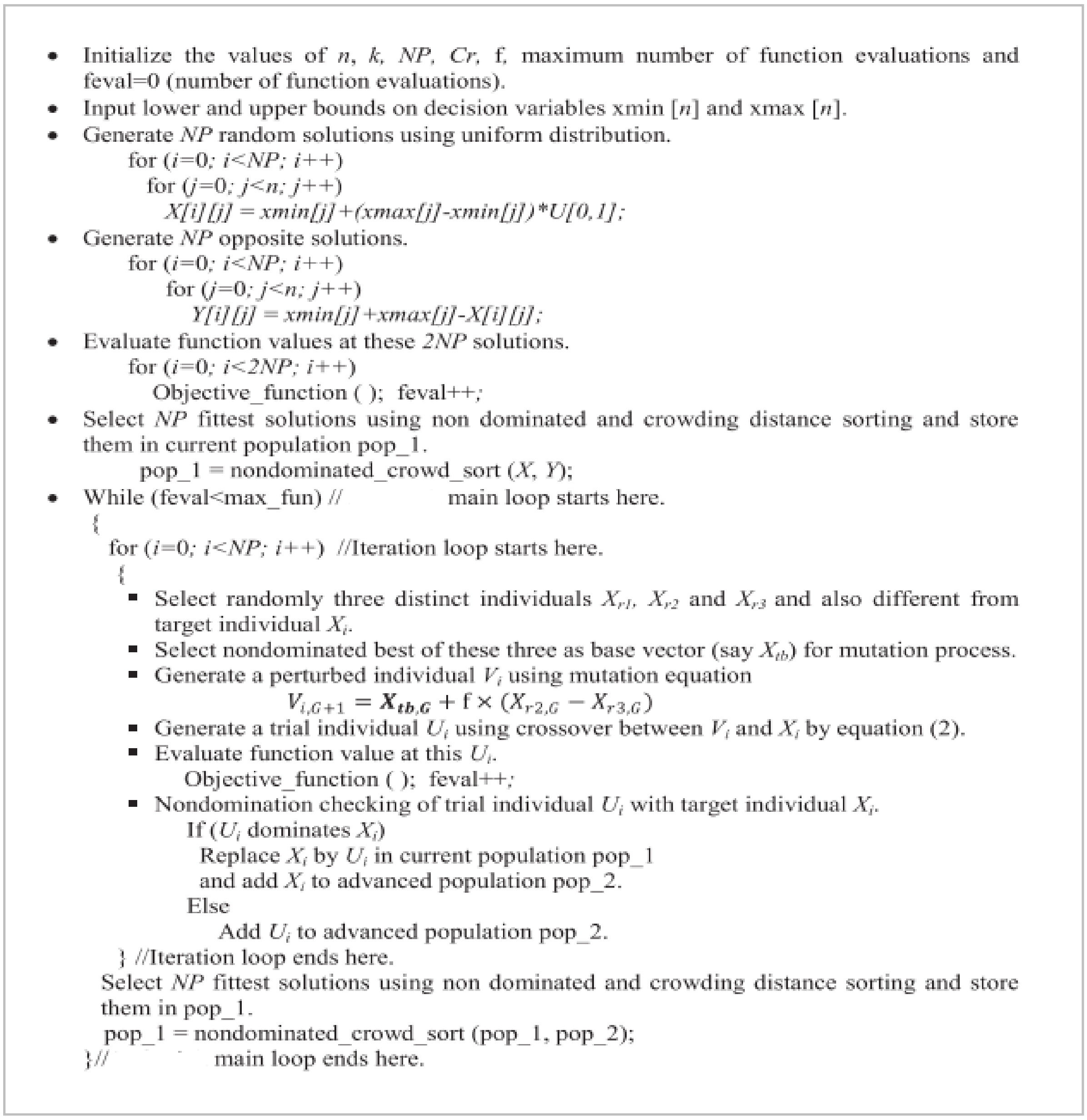
The design of multi-objective evolutionary algorithms has to primary objectives: maximizing the diversity and spread of solutions, and to enhance the speed of convergence to the Pareto-optimal front. The first issue is addressed in the literature through the use of different mechanisms for maintaining diversity, which primarily depend on crowding density estimates for solutions or by calculating crowding distance as the sum of the distance between point on both sides of a solution at all dimension in the problem space. The techniques regard solutions that’s have either high crowding densities or small crowding distances as substandard and more likely to be eliminated to enhance the diversity of the population.

MOJaDE addresses the issues raised above. It is a multi-object algorithm, whose updated there control parameters in a self-adaptive way, thereby bypassing user interaction both prior and during the optimization process. In so doing, it prevents the algorithm's performance from being adversely affected by wrongly-set parameter values. Explored inferior solutions are stored in a newly-introduced external archive provided their differences with the existing population show promise in regards to the direction of the optimal solution Pareto front. It is noteworthy that this archive is distinct from those in other MOEAs to store their best non-dominated solutions. Furthermore, crowding distance is calculated based on a fairness measure with a preference for solutions with a near-uniform yet large distance between them and their nearest neighbors. This is particularly important for the distribution of solutions after computing the non-dominated sets, which it could be speedily motivated in the direction of the optimal solution Pareto sets. Additionally, control parameters with the self-adaptive nature of the algorithm's is help to avoid issues with parameter tuning for problems with diverse characteristics [40].

3.3.1 JADE Operation for Multi-Objective

The DE crossover be able to perform identically for both single and multi-objective optimization. Due to the challenges unique to multi-objective optimization, however, there is a need to reconsider the selection and mutation processes. Firstly, despite the selection requirement that unique solutions in a multi objective dimension space be comparable, it is impossible to guarantee this using the rank dominance comparison since this establishes the order between the solutions. To overcome this, a second metric like crowding density can be introduced to facilitate the comparison of two solutions tied in the dominance comparison.

Secondly, the original meaning of the best-solution information utilized in a mutation technique is lost as it can no longer serve as a guide towards the optimum. This problem typically arises after a few generations, particularly when the objective space dimension is elevated, leading solutions to become non-dominated from one another. As such, crowing density is used to identify the best solutions rather than dominance. Consequently, the population is moved to the least-populated regions using a regular greedy mutation strategy, although these are not automatically nearer to the real Pareto front [38].



```
• Initialize the values of n, k, NP, Cr, f, maximum number of function evaluations and
  feval=0 (number of function evaluations).
• Input lower and upper bounds on decision variables xmin [n] and xmax [n].
• Generate NP random solutions using uniform distribution.
      for (i=0; i<NP; i++)
        for (j=0; j<n; j++)
          X[i][j] = xmin[j]+(xmax[j]-xmin[j])*U[0,1];
• Generate NP opposite solutions.
      for (i=0; i<NP; i++)
          for (j=0; j<n; j++)
            Y[i][j] = xmin[j]+xmax[j]-X[i][j];
• Evaluate function values at these 2NP solutions.
      for (i=0; i<2NP; i++)
         Objective_function ( );  feval++;
• Select NP fittest solutions using non dominated and crowding distance sorting and store
  them in current population pop_1.
       pop_1 = nondominated_crowd_sort (X, Y);
• While (feval<max_fun) //             main loop starts here.
   {
     for (i=0; i<NP; i++)  //Iteration loop starts here.
      {
       ▪ Select randomly three distinct individuals $X_{r1}$, $X_{r2}$ and $X_{r3}$ and also different from
         target individual $X_i$.
       ▪ Select nondominated best of these three as base vector (say $X_{tb}$) for mutation process.
       ▪ Generate a perturbed individual $V_i$ using mutation equation
                       $V_{i,G+1} = \boldsymbol{X_{tb,G}} + \text{f} \times (X_{r2,G} - X_{r3,G})$
       ▪ Generate a trial individual $U_i$ using crossover between $V_i$ and $X_i$ by equation (2).
       ▪ Evaluate function value at this $U_i$.
            Objective_function ( );  feval++;
       ▪ Nondomination checking of trial individual $U_i$ with target individual $X_i$.
            If ($U_i$ dominates $X_i$)
              Replace $X_i$ by $U_i$ in current population pop_1
              and add $X_i$ to advanced population pop_2.
            Else
                 Add $U_i$ to advanced population pop_2.
      } //Iteration loop ends here.
     Select NP fittest solutions using non dominated and crowding distance sorting and store
     them in pop_1.
      pop_1 = nondominated_crowd_sort (pop_1, pop_2);
   }//          main loop ends here.
```

Figure 3.4: MOJaDE Pseudo-Code

3.3.2 Pareto Dominance and Crowding Density

In an M-objective minimization problem, it is defined that an objective vector **f** = (f_1, f_2, …, f_M) dominates another vector **g** =(g1, g2, …, gM) if

$$\forall i \in \{1,2,\ldots,m\}: F_i \leq g_i \; and \; \exists i \in \{1,2,\ldots,m\}: F_i < g_i \qquad (3.14)$$

If it is dominant, the objective vector f (and its matching decision vector) is taken to be superior to vector g (and its matching decision vector). The definition of Pareto dominance makes it evident that it is possible to not be able to compare two objective vectors (i.e., neither dominates the other).

A Pareto optimal solution is one that is not dominated by others in the problem space. Similarly, the Pareto-optimal front is a set containing these non-dominated solutions. Generally speaking, there is a near-unlimited number of Pareto optimal solutions. However, an EA is only capable of searching a small set of archetypal solutions, which should be diverse and dispersed across the Pareto-optimal front. This is typically attained using various methods to approximate the crowding density of solutions: more crowded solutions are typically eliminated to help maintain population diversity.

3.3.3 Selection Operation

The MOJADE selection operation involves comparing the Pareto dominance of solutions, as well as estimating their crowding density at a lexicographic order. As such, solutions are considered superior to others if they dominate or have a smaller crowding density in the event of a tie in dominance comparison. Specifically, the selection of NP vectors in MOJADE uses a three-step comparison of available (P') 2NP offspring and their trial vectors. Chosen vectors constitute the offspring population for the subsequent generation. The initial step involves comparison of the dominance of each parent-trial vector pair. The non-dominated vector is then removed from the available pool P' in a manner similar to the DE comparison process in single-objective. Following this, P' has a scope ranging from NP to 2NP.

In the next step, P' it’s reduces to a smaller size using the dominance relationship among all solutions. In the initial stage, every non-dominated solutions are given rank 1 to determine the solutions with the highest fitness value. The remaining solutions are then taken into consideration with the non-dominated ones being assigned rank 2. This procedure is repeated until rank values have been assigned to no less than NP vectors, after which the rest vectors are instantly eliminated from P'.

After that its involves estimating the crowding density of solution with the worst rank values. The most crowded solutions are eliminated, followed by updates of the crowding density of the remaining solutions until the scope of P' equals NP. The density is estimated on the basis of the principles that a solution has a lower crowdedness if the distance between it and its nearest neighbors is larger or more similar to the uniform. To this end, the most common fairness measure in network engineering [41] used to preserve the regularity of competing variables while simultaneously maximizing their summation is an ideal way to estimate crowding density.

Consider the (p,α)-proportionally fairness measure in [41]. With a usual p=1 and,$\alpha \geq 0$ it is possible to compute the crowding distance d_i of a solution i as

$$d_i = \begin{cases} \sum_{i=1}^{k} \log d_{ij} & if\ \alpha = 1 \\ (1-\alpha)^{-1} \sum_{i=1}^{k} d_{ij}^{1-\alpha} & otherwise \end{cases} \tag{3.15}$$

where d_{ij} denotes the Euclidean distance of solution *i* to its *j*-th nearest neighbor in *P*'. It is not difficult to show that d_i is the aggregate distance if α= 0.and it is equal product distance ,If α = 1

$$\mathrm{d_i} = \prod_{\mathrm{i=1}}^{\mathrm{k}} \mathrm{d_{ij}}\ ; \tag{3.16}$$

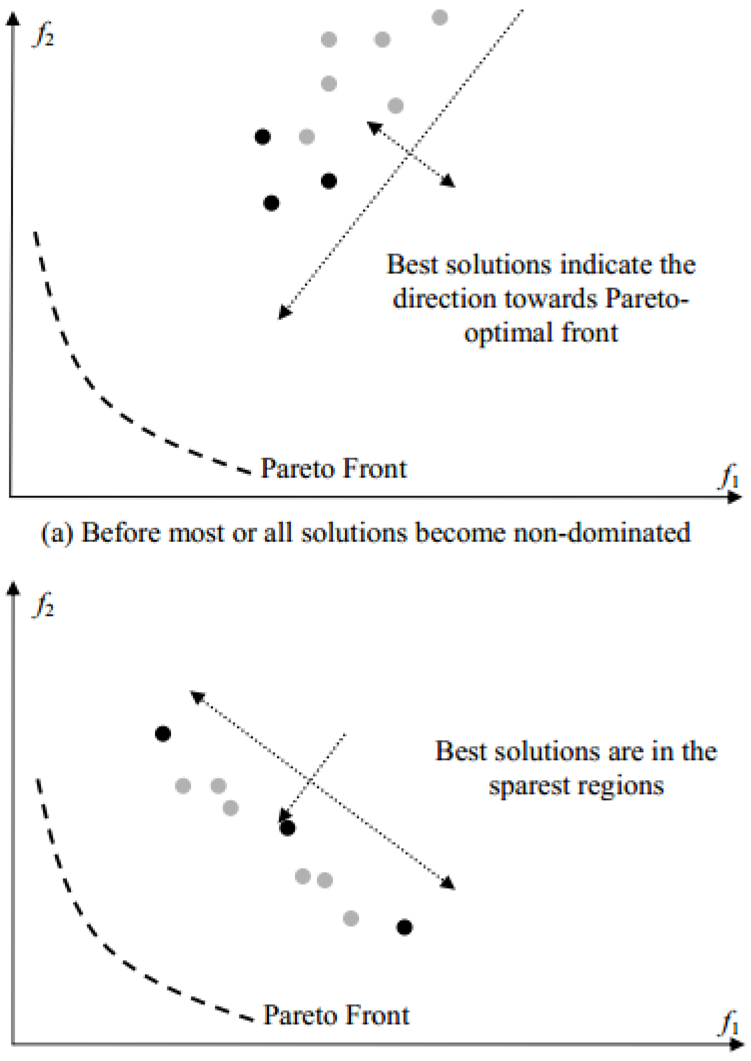
And its harmonic distance when α= 2

$$\mathrm{d_i} = \frac{1}{\frac{1}{\mathrm{d_{i,1}}}+\frac{1}{\mathrm{d_{i,2}}}+\cdots+\frac{1}{\mathrm{d_{i,2(m-1)}}}} \tag{3.17}$$

And max-min distance when $\alpha \to \infty$

$$d_i = \max(\min(d_{i,1}, d_{i,2}, \ldots, d_{i,k})) \tag{3.18}$$

It is noteworthy that both product and harmonic are accepted in DE [42] as the crowding distance.


f_2
Best solutions indicate the direction towards Pareto-optimal front
Pareto Front
f_1
f_2
Best solutions are in the sparest regions
Pareto Front
f_1

(a) Before most or all solutions become non-dominated

(b) After most or all solutions become non-dominated

Figure 3.5: Best Solution Places Compare with Other Solutions [38].

Figure 3.5 above illustrates the relative location of worst solutions (gray dots) and the best (dark dots) in the problem space.

The bidirectional arrows and unidirectional indicate the path of the progress towards the Pareto front and the distribution of the population, respectively.

3.3.4 Mutation Operation

After a few generations, most or all individuals in the population will typically have become non-dominated. In the event of this, the best solutions are identified primarily based on crowding density. The population is guided in the direction of the sparsest region with the best solutions as part of a usual greedy strategy.

This situation differs from single-objective optimization and even the earlier stages of multi-objective optimization, in which the direction of the optimal solutions is indicated by the best solutions. This is illustrated in Fig. 3.5 where the best solutions are evidently unable to indicate a promising direction when they are non-dominated by other solutions. We propose a new approach of utilizing directional information based on the previously-explored inferior solutions in the optimization process. The external archive A is then used as a storage space for parent individuals recently eliminated from the population due to their domination by other solutions in the initial phases for selection process. After that, as an improvement on the mutation strategy in (3.5), the following formulation is used to generate the mutant vector:

$$\boldsymbol{v_{i,g}} = \boldsymbol{x_{i,g}} + \boldsymbol{F_i}\left[\left(\boldsymbol{x^{p}_{best,g}} - \boldsymbol{x_{i,g}}\right) + \left(\boldsymbol{\tilde{X}_{r1,g}} - \boldsymbol{x_{r2,g}}\right)\right] \tag{3.19}$$

where $\tilde{X}_{r1,g}$is a randomly-selected vector by the combination of the archive with the parent population, while $X_{r2,g}$ and $X^{p}_{best,g}$are selected from the parent population in the way as in (3.5). It is clear that (3.19) falls back to (3.5) if the archive is blank. $\tilde{X}_{r1,g}$ is selected from and archive if it is not empty, while the difference between $\tilde{X}_{r1,g}$ and $\tilde{X}_{r2,g}$ indicates the direction towards the optimum. It is important to note that this method is distinct from the one proposed in [43], where solutions in the current population are compared to obtain direction information despite differences in their dominance ranks (which essentially means that direction information will be lacking if all solutions are non-dominated). Furthermore, the operation in (3.19) is distinct from many other MOEAs where saving the best non dominant solution in an archive .However, similar to these archive-based strategies, our method is also helpful to enhance the diversity of solutions, other than its main benefit of providing direction information. Simple archive operations are used to make the process less complex.

Initially empty, parent individuals that failed the initial steps of the selection process are added to the archive after each generation. Solutions are eliminated at random to maintain the size of the archive when it reaches a predefined threshold like 2NP [44].

3.3.5 External Archive

In contrast to single objective optimization, MOJaDE are more likely to keep a group of non-dominated solutions. Due to the lack of preference information in multi-objective optimization, no solution be able to superior to others. Consequently, JADE algorithm utilize an external archive as way to accurately document the pareto optimal set non dominated vectors encountered during the adaptation method [54,55].

3.3.6 Diversity

The success of MODE can be attributed to its capability to uncover a group of non-dominated solution ("Pareto optimal solutions") from one iteration. Evolutionary algorithms need to conduct a multimodal search, which includes a variety of unique potential solutions, as a way to determine a reliable approximation of the Pareto optimal set from a single optimization run. As such, the efficiency of MODE considerably relies on the availability of a diverse population [53].

3.3.7 Update External Archive

New solutions are consistently assigned to the external archive over the course of the evolution. The decision of new solution remains in the external archive or not is based on a comparison between it and every other pareto optimal set (non-dominated 0solution in the archive, the size of which is limited.

Each individual in our algorithm search for a new solution in each generation. The new solution is allowed into the external archive if it is found to dominate the original individual. Conversely, if the original individual dominates the new solution, it is not permitted into the external archive. If neither of the two solutions dominates the other, one of them is chosen at random to add into external archive, which is updated after each generation. If the number of Pareto optimal set is greater than the pre-determined archive size, crowding distance [54] is then used to delete any extra members.

Chapter 4

EXPERIMENTAL RESULT AND EVALUATIONS

Execution evaluation of the algorithm suggested, and the display of the comparable success set apart from the standard meta heuristics is to be undertaken within the difficulties of CEC 2017 [45].

Although the definitions, categorizations and characteristics (fitness landscape) are not described here, the functional benchmarks are clearly explained in the references. To ensure an equitable and comparative evaluation, the independent runs, and the stopping criteria of the function evaluations will be identical to those of the corresponding references. Likewise, the proposed algorithmic parameter methodology will remain the same in all test functions; throughout the program executions, there will be no interactive intervention. Test function integrity also dictates that the number of variables, in respect of the test functions, also obtain to that of the corresponding references.

4.1 CEC'17 Expensive Optimization Test Problems

4.1.1 Common Definitions

All test functions are minimization problems defined as follows in equation (4.1):

$$\min f(x), x = [x_1, x_2, \dots., x_D]^T \tag{4.1}$$

Where D is the number of decision variable. All search ranges and Dimension are clearly explained in the references [46].

4.1.2 Results

Our Proposed algorithm was tested distinctly for optimizing CEC2017 single objective problems in 30 Dimension [46] .The results intended to demonstrate a large improvement from the JADE solutions. Each problem have been Averaged over 10 runs.

Table 4.1: Summary of CEC'17 Optimization Test Problem

	No.	Functions	F_i^*=$F_i(x^*)$
Unimodal Functions	1	Shifted and Rotated Bent Cigar Function	100
	2	Shifted and Rotated Sum of Different Power Function	200
	3	Shifted and Rotated Zakharov Function	300
Simple Multimodal Functions	4	Shifted and Rotated Rosenbrock's Function	400
	5	Shifted and Rotated Rastrigin's Function	500
	6	Shifted and Rotated Expanded Scaffer's F6 Function	600
	7	Shifted and Rotated Lunacek Bi_Rastrigin's Function	700
	8	Shifted and Rotated Non-Continuous Rastrigin's Function	800
	9	Shifted and Rotated Levy Function	900
	10	Shifted and Rotated Schwefel's Function	1000
Hybrid Functions	11	Hybrid Function 1(N=3)	1100
	12	Hybrid Function 2(N=3)	1200
	13	Hybrid Function 3(N=3)	1300
	14	Hybrid Function 4(N=4)	1400
	15	Hybrid Function 5(N=4)	1500
	16	Hybrid Function 6(N=4)	1600
	17	Hybrid Function 6(N=5)	1700
	18	Hybrid Function 6(N=5)	1800
	19	Hybrid Function 6(N=5)	1900
	20	Hybrid Function 6(N=6)	2000
Composition Function	21	Composition Function 1 (N=3)	2100
	22	Composition Function 1 (N=3)	2200
	23	Composition Function 1 (N=4)	2300
	24	Composition Function 1 (N=4)	2400
	25	Composition Function 1 (N=5)	2500
	26	Composition Function 1 (N=5)	2600
	27	Composition Function 1 (N=6)	2700
	28	Composition Function 1 (N=6)	2800
	29	Composition Function 1 (N=3)	2900
	30	Composition Function 1 (N=3)	3000

Table 4.2 : Best Result of Improved JADE Algorithm in Dimension 30 Over 10 Runs.

Function Number	Optimal Solution	JADE	Error
1	100	100	0.00E+00
3	300	300	0.00E+00
4	400	400	0.00E+00
5	500	500	0.00E+00
6	600	600	0.00E+00
7	700	710.9892139	1.10E+01
8	800	800	0.00E+00
9	900	900	0.00E+00
10	1000	1006.322455	6.32E+00
11	1100	1100	0.00E+00
12	1200	1200	0.00E+00
13	1300	1300	0.00E+00
14	1400	1400	0.00E+00
15	1500	1500	0.00E+00
16	1600	1600	0.00E+00
17	1700	1700	0.00E+00
18	1800	1800	0.00E+00
19	1900	1900	0.00E+00
20	2000	2000	0.00E+00
21	2100	2102.5727	2.57E+00
22	2200	2200	0.00E+00
23	2300	2300.000028	2.80E-05
24	2400	2403.814833	3.81E+00
25	2500	2504.837405	4.84E+00
26	2600	2603.37198	3.37E+00
27	2700	2703.231896	3.23E+00
28	2800	2806.122754	3.33E+00
29	2900	2902.831896	2.83E+00
30	3000	3004.245148	4.25E+00

The results of the analyses of Table 4.2 revealed an apparent improvement in the quality of solutions, which obviously tend to get closeness to the optimal values.The findings of our experiment with JADE are consistent to some extent with the past studies on Problem optimization.

Both of the Unimodal functions results in JADE algorithm in Dimension 30 reached optimal solutions without any small differences from optimality. Multimodal functions were mixed between problems which had very high differences from optimal solutions; problem no. 7, and Composite functions that included problem no. 10, had very small differences from optimal solutions.

While the rest of the problems' results in the same category reached to the optimal solutions. Finally, Hybrid functions and Composition function which included problems from no.21 to no.30, reached near-optimal solutions with relatively small differences from optimality.

Table 4.3: IGD Values Obtained by Improved JADE and it are 3 Competitors for 30 Test Function .

Function Number	JADE	BCO	LSHADE	SPA
1	**0.00E+00**	0.00E+00	0.00E+00	0.00E+00
3	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
4	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
5	**0.00E+00**	**0.00E+00**	3.0E+00	1.8E+00
6	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
7	1.10E+01	**1.04E+01**	1.2E+01	1.2E+01
8	**0.00E+00**	**0.00E+00**	2.4E+00	1.9E+00
9	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
10	6.32E+00	**2.50E-01**	2.2E+01	2.2E+01
11	**0.00E+00**	**0.00E+00**	4.1E-01	**0.00E+00**
12	**0.00E+00**	1.87E-01	7.7E+01	1.2E+02
13	**0.00E+00**	1.44E-01	3.2E+00	3.6E+00
14	**0.00E+00**	2.18E-02	1.7E-01	2.0E-02
15	**0.00E+00**	5.37E-03	1.7E-01	2.7E-01
16	**0.00E+00**	2.05E-01	4.1E-01	5.2E-01
17	**0.00E+00**	8.10E-01	1.7E-01	1.2E-01
18	**0.00E+00**	1.30E-01	2.8E-01	2.4E+00
19	**0.00E+00**	1.74E-01	1.1E-02	5.5E-02
20	**0.00E+00**	**0.00E+00**	1.5E-02	1.8E-01
21	2.57E+00	**1.09E-02**	1.6E+02	1.6E+02
22	**0.00E+00**	**0.00E+00**	1.0E+02	1.0E+02
23	**2.80E-05**	2.47E-01	3.0E+02	3.0E+02
24	3.81E+00	**8.25E-03**	3.2E+02	2.9E+02
25	4.84E+00	**2.33E-01**	4.1E+02	4.2E+02
26	3.37E+00	**2.97E-02**	3.0E+02	3.0E+02
27	3.23E+00	**1.90E-01**	3.9E+02	3.2E+02
28	3.33E+00	**6.50E-01**	3.6E+02	4.0E+02
29	2.83E+00	**0.00E+00**	2.3E+02	2.3E+02
30	4.25E+00	**1.53E-03**	7.8E+04	4.1E+04

Table 4.3 shows the best IGD values for 30 Test Problem. The results gained JADE algorithm as against with the solution from [47][48]. IGD score of obtained solutions found by JADE approach are very small. That means JADE Algorithm can discover a well spread sets and high quality solution in objective range for each problems.

By examining the comparison between error rates demonstrated in Table 4.3, it can be concluded that the highest number of ***best*** problem optimization results belong to the JADE and BCO Algorithm. The table showed superior performance of JADE from optimizing results of 22 out of 30 problems, which is the highest between all the methods from literature. In problems number (12, 13,14,15,16,17,18,19, and 23), the error rates of JADE appeared to be very close to optimality. The rest of the problems' results varied between generally small differences and extreme differences from the optimal values.

Table 4.4: average of IGD Values Obtained by improved JADE* with Addition Archive and Normal JADE without Archive, Competitors for same 30 Test Function.

Function Number	JADE*	JADE
1	0.00E+00	0.00E+00
3	0.00E+00	0.00E+00
4	0.00E+00	0.00E+00
5	0.00E+00	0.00E+00
6	0.00E+00	0.00E+00
7	1.10E+01	1.10E+01
8	0.00E+00	0.00E+00
9	0.00E+00	0.00E+00
10	**6.32E+00**	9.82E+00
11	0.00E+00	0.00E+00
12	0.00E+00	0.00E+00
13	0.00E+00	0.00E+00
14	0.00E+00	0.00E+00
15	0.00E+00	0.00E+00
16	0.00E+00	0.00E+00
17	0.00E+00	0.00E+00
18	0.00E+00	0.00E+00
19	0.00E+00	0.00E+00
20	0.00E+00	0.00E+00
21	**2.57E+00**	2.77E+00
22	0.00E+00	0.00E+00
23	**2.80E-05**	2.00E-01
24	**3.81E+00**	2.67E+01
25	**4.84E+00**	4.33E+01
26	**3.37E+00**	3.00E+02
27	**3.23E+00**	7.58E+01
28	**3.33E+00**	3.00E+02
29	**2.83E+00**	3.02E+01
30	**4.25E+00**	3.53E+01

Table 4.4 shows the average of IGD values for 30 Test Problem. The results gained JADE* algorithm with addition archive as against the solution of normal JADE without archive.

By examining the comparison between error rates demonstrated in Table 4.4 , it can be concluded that adding archive improves the solution in problem number (10,21,23,24,25,26,27,28,29 and 30).

4.1.3 Friedman Ranking Test

The ***Friedman Test*** is a non-parametric statistical test developed by Milton Friedman. It is used to check the statistical similarities in treatments across multiple test attempts. The procedure involves ranking each row together, then considering the values of ranks by columns [51]. The P-value indicator represents the difference between the ranked functions statistically. The smaller the p-value is, the bigger the statistical differences between the ranked methods are [52].

The ranking procedure was used in order to assess the quality of the proposed JADE. A comparison among the three proposed variants EA algorithm in ***dimension 30*** opposed to the JADE results, and between EA algorithm proposed in ***dimension 30*** with literature studies was conducted using ***Friedman test***.

Table 4.5: Friedman Ranking between JADE and literature EA algorithm in D30.

Rank	Function
1	BCO
1	JADE
2	SPA
3	LSHADE

P-value = 1.08087e-06

In dimension 30, BCO and JADE had the same level of performance according to Friedman Test ranking in Table 4.5. Both of version BCO and JADE had the best rank before SPA followed by the original LSHADE.

4.2 CEC'17 Test Problems for Multi-objective problem

Set of benchmarks through 'Extended shifted 'and 'Extended rotated'. More information for each problem are presented in [49].

Where the average inverted generational distance (IGD) value of each generation over 30 independent runs calculated by using all NDS values.

4.2.1 Results

Table 4.6: Min, Max, Average, Standard Deviation of IGD Values and Number of Function Evaluation of JADE in 30 Runs.

Function	Average	Min	Max	Std
ZDT1	9.54E-04	6.56E-04	1.72E-03	2.61E-04
ZDT2	1.51E-02	1.03E-03	9.00E-02	2.12E-02
ZDT3	3.11E-03	2.00E-03	5.99E-03	8.93E-04
ZDT4	9.47E-04	6.69E-04	1.51E-03	2.05E-04
ZDT6	4.43E-03	1.52E-03	1.55E-02	3.09E-03
DTLZ1	6.70E-04	5.81E-04	8.15E-04	6.39E-05
DTLZ2	1.94E-03	1.63E-03	2.37E-03	1.96E-04
DTLZ3	5.04E-01	1.08E-01	1.18E+00	2.70E-01
DTLZ4	5.70E-01	8.60E-02	1.41E+00	3.58E-01
DTLZ5	4.85E-04	4.11E-04	7.09E-04	6.68E-05
DTLZ6	2.06E-02	5.08E-04	2.39E-02	5.59E-03
DTLZ7	3.21E-03	1.68E-03	8.05E-03	1.62E-03
WFG1	2.15E-03	2.20E-03	2.12E-03	2.66E-05
WFG2	9.45E-03	7.89E-03	1.17E-02	1.06E-03
WFG3	2.94E-03	1.81E-03	5.69E-03	9.10E-04
WFG4	5.78E-03	4.06E-03	1.12E-02	1.53E-03
WFG5	5.38E-03	4.34E-03	7.14E-03	6.82E-04
WFG6	6.45E-03	4.75E-03	1.34E-02	1.76E-03
WFG7	7.35E-03	4.91E-03	1.34E-02	2.37E-03
WFG8	5.37E-03	4.52E-03	8.17E-03	8.74E-04
WFG9	6.78E-03	5.01E-03	1.29E-02	1.87E-03

It can be seen from Table 4.6 that JADE is a robust and successful approach explain with small ‘IGD’ score and their standard deviation.

Tables 4.7, 4.8, 4.9, 4.10, 4.11, 4.12, 4.13 and 4.14 clarify the rating of problems in “CEC2017” and JADE with respect to the IGD values. From [47] [48], MOEA/D, SMPSO, GDE3, MOCell and SPEA2 are the best five approaches in the contest in order .The defender of this contest was “MOCell”. From all our result can see that JADE acted better than “MOCell” in all test problem .The proposed JADE takes the first rank in most of test problems.

Table 4.7: IGD Values Obtained by JADE and it are 11 Competitors for ZDT1, ZDT2 and ZDT3.

Rank	ZDT1	IGD	ZDT2	IGD	ZDT3	IGD
1	**MOJaDE**	**9.54E-04**	**MOCell**	**3.79E-03**	**MOJaDE**	**3.11E-03**
2	SMPSO	3.67E-03	SMPSO	3.79E-03	SMPSO	4.28E-03
3	MOCell	3.68E-03	AbYSS	3.82E-03	OMOPSO	4.35E-03
4	OMOPSO	3.71E-03	OMOPSO	3.83E-03	GDE3	4.36E-03
5	AbYSS	3.72E-03	SPEA2	3.89E-03	SPEA2	4.84E-03
6	GDE3	3.77E-03	GDE3	3.91E-03	NSGAII	5.38E-03
7	SPEA2	3.92E-03	CellDE	4.36E-03	MOCell	6.17E-03
8	IBEA	4.10E-03	NSGAII	4.89E-03	CellDE	1.02E-02
9	NSGAII	4.83E-03	MOEA/D	9.13E-03	AbYSS	1.50E-02
10	CellDE	4.83E-03	IBEA	9.41E-03	MOEA/D	1.72E-02
11	PAES	1.17E-02	PAES	1.46E-02	IBEA	2.97E-02
12	MOEA/D	1.25E-02	MOJaDE	1.51E-02	PAES	5.61E-02

Table 4.8: IGD Values Obtained by JADE and it are 11 Competitors for ZDT4, ZDT6 and DTLZ1.

Rank	ZDT4	IGD	ZDT6	IGD	DTLZ1	IGD
1	**MOJaDE**	**9.47E-04**	**MOCell**	**3.00E-03**	**MOJaDE**	**6.70E-04**
2	SMPSO	3.71E-03	OMOPSO	3.01E-03	SPEA2	2.02E-02
3	MOCell	3.84E-03	SMPSO	3.03E-03	GDE3	2.33E-02
4	SPEA2	4.07E-03	AbYSS	3.05E-03	MOEA/D	2.54E-02
5	AbYSS	4.41E-03	GDE3	3.12E-03	NSGAII	2.61E-02
6	NSGAII	4.93E-03	SPEA2	3.17E-03	AbYSS	2.73E-02
7	PAES	7.34E-03	CellDE	3.43E-03	SMPSO	2.82E-02
8	MOEA/D	1.43E-01	MOEA/D	4.16E-03	MOCell	2.86E-02
9	GDE3	4.72E-01	MOJaDE	4.43E-03	PAES	5.86E-02
10	IBEA	6.26E-01	NSGAII	4.76E-03	CellDE	1.60E-01
11	CellDE	4.24E+00	IBEA	5.16E-03	IBEA	1.81E-01
12	OMOPSO	4.92E+00	PAES	7.07E-03	OMOPSO	1.18E+01

It is apparent from Table 4.7 and Table 4.8 that the MOJaDE is the most competitive algorithm and obtained the best values on 4 problems (ZDT1, ZDT3 , ZDT4 and DTLZ1), while the MOCell algorithm computed the second best fronts regarding this indicator on this evaluated problems. MOEA/D, SMPSO,AbYSS, GDE3 and OMOPSO have a similar performance, while CellDE and PAES has relatively poorer results in regards to this indicator.

Table 4.9: IGD Values Obtained by JADE and it are 11 Competitors for DTLZ2, DTLZ3 and DTLZ4.

Rank	DTLZ2	IGD	DTLZ3	IGD	DTLZ4	IGD
1	**MOJaDE**	**1.94E-03**	**SMPSO**	**1.15E-01**	**MOEA/D**	**5.49E-02**
2	SPEA2	5.42E-02	PAES	1.91E-01	AbYSS	6.05E-02
3	GDE3	6.28E-02	NSGAII	2.93E-01	NSGAII	6.39E-02
4	CellDE	6.61E-02	SPEA2	3.38E-01	OMOPSO	6.48E-02
5	MOCell	6.68E-02	AbYSS	3.94E-01	GDE3	6.57E-02
6	MOEA/D	6.71E-02	MOJaDE	5.04E-01	SMPSO	6.80E-02
7	AbYSS	6.88E-02	IBEA	5.11E-01	CellDE	7.71E-02
8	NSGAII	6.88E-02	MOCell	7.55E-01	MOCell	1.35E-01
9	OMOPSO	6.88E-02	MOEA/D	1.17E+00	SPEA2	1.37E-01
10	SMPSO	7.17E-02	GDE3	2.25E+00	IBEA	2.10E-01
11	IBEA	1.22E-01	CellDE	8.51E+00	PAES	3.99E-01
12	PAES	3.15E-01	OMOPSO	1.15E+02	MOJaDE	5.70E-01

Table 4.10: IGD Values Obtained by JADE and it are 11 Competitors for DTLZ5, DTLZ6 and DTLZ7.

Rank	DTLZ5	IGD	DTLZ6	IGD	DTLZ7	IGD
1	**MOJaDE**	**4.85E-04**	**OMOPSO**	**3.89E-03**	**MOJaDE**	**3.21E-03**
2	MOCell	4.05E-03	SMPSO	3.93E-03	SPEA2	6.96E-02
3	AbYSS	4.08E-03	GDE3	4.15E-03	GDE3	7.47E-02
4	SMPSO	4.09E-03	CellDE	4.54E-03	NSGAII	7.64E-02
5	OMOPSO	4.13E-03	PAES	7.13E-03	SMPSO	8.52E-02
6	GDE3	4.19E-03	MOEA/D	9.36E-03	OMOPSO	8.68E-02
7	SPEA2	4.33E-03	SPEA2	1.25E-02	CellDE	1.23E-01
8	NSGAII	5.42E-03	NSGAII	1.35E-02	MOEA/D	1.90E-01
9	PAES	6.83E-03	MOJaDE	2.06E-02	MOCell	2.45E-01
10	CellDE	8.56E-03	IBEA	5.75E-02	AbYSS	3.94E-01
11	MOEA/D	1.04E-02	AbYSS	7.89E-02	IBEA	3.99E-01
12	IBEA	1.93E-02	MOCell	7.55E-01	PAES	8.87E-01

It is apparent from Table 4.9 and Table 4.10 that the MOJaDE is the most competitive algorithm, having the best values on 3 problems (DTLZ2, DTLZ5 and DTLZ7), while the SMPSO algorithm computed the second best fronts regarding to this indicator in the evaluated problems. MOEA/D, SPEA2, AbYSS and OMOPSO performed similarly, while PAES and IBEA performed poorly in regards to this indicator.

Tables 4.8, 4.9 and 4.10 shows the best IGD values for all three objective function problems DTLZ1 to DTLZ7.The results gained JADE algorithm as against with the solution from [47][48]. IGD score of obtained solutions found by JADE are very small. That means JADE Algorithm can discover a well spread sets and high quality solution in objective range for all objective function for each problems. Tables illustrate the ranking of all three objectives DTLZ1 to DTLZ7 .From tables can be see that JADE acted better than other approaches in most of problems. JADE takes the first position in most of previous test problems.

Table 4.11: IGD Values Obtained by JADE and it are 11 Competitors for WFG1, WFG2 and WFG3.

Rank	WFG1	IGD	WFG2	IGD	WFG3	IGD
1	**MOJaDE**	**2.15E-03**	**MOJaDE**	**9.45E-03**	**MOJaDE**	**2.94E-03**
2	GDE3	5.07E-02	GDE3	1.00E-02	MOCell	1.38E-01
3	CellDE	8.73E-02	OMOPSO	1.03E-02	OMOPSO	1.38E-01
4	NSGAII	1.96E-01	SMPSO	1.07E-02	SMPSO	1.39E-01
5	IBEA	2.89E-01	CellDE	1.14E-02	GDE3	1.39E-01
6	MOEA/D	3.21E-01	SPEA2	3.58E-02	IBEA	1.39E-01
7	MOCell	3.46E-01	NSGAII	3.75E-02	AbYSS	1.39E-01
8	SPEA2	3.71E-01	MOCell	4.93E-02	SPEA2	1.39E-01
9	AbYSS	7.32E-01	MOEA/D	4.97E-02	NSGAII	1.41E-01
10	OMOPSO	8.36E-01	AbYSS	6.21E-02	CellDE	1.42E-01
11	SMPSO	1.10E+00	IBEA	9.84E-02	MOEA/D	1.43E-01
12	PAES	1.25E+00	PAES	3.06E-01	PAES	1.67E-01

Table 4.12: IGD Values Obtained by JADE and it are 11 Competitors for WFG4, WFG5 and WFG6.

Rank	WFG4	IGD	WFG5	IGD	WFG6	IGD
1	**MOJaDE**	**5.78E-03**	**MOJaDE**	**5.38E-03**	**MOJaDE**	**6.45E-03**
2	MOCell	1.04E-02	AbYSS	6.59E-02	OMOPSO	1.26E-02
3	AbYSS	1.04E-02	MOCell	6.62E-02	SMPSO	1.28E-02
4	GDE3	1.08E-02	OMOPSO	6.62E-02	GDE3	1.30E-02
5	SPEA2	1.27E-02	SMPSO	6.63E-02	CellDE	1.45E-02
6	NSGAII	1.36E-02	CellDE	6.64E-02	MOEA/D	1.90E-02
7	PAES	1.55E-02	GDE3	6.64E-02	SPEA2	2.31E-02
8	CellDE	1.61E-02	SPEA2	6.67E-02	NSGAII	3.49E-02
9	IBEA	2.02E-02	NSGAII	6.81E-02	IBEA	5.39E-02
10	MOEA/D	2.22E-02	MOEA/D	6.82E-02	MOCell	6.32E-02
11	OMOPSO	2.30E-02	PAES	6.97E-02	AbYSS	9.32E-02
12	SMPSO	2.69E-02	IBEA	7.28E-02	PAES	9.74E-02

Table 4.13: IGD Values Obtained by JADE and it are 11 Competitors for WFG7, WFG8 and WFG9.

Rank	WFG7	IGD	WFG8	IGD	WFG9	IGD
1	**MOJaDE**	**7.35E-03**	**MOJaDE**	**5.37E-03**	**MOJaDE**	**6.78E-03**
2	OMOPSO	1.17E-02	SMPSO	1.03E-02	GDE3	1.35E-02
3	MOCell	1.17E-02	SPEA2	1.05E-02	SMPSO	1.35E-02
4	SMPSO	1.19E-02	MOCell	1.17E-02	SPEA2	1.45E-02
5	AbYSS	1.19E-02	CellDE	1.21E-02	MOEA/D	1.45E-02
6	GDE3	1.24E-02	OMOPSO	1.26E-02	CellDE	1.45E-02
7	SPEA2	1.29E-02	AbYSS	1.30E-02	NSGAII	1.55E-02
8	CellDE	1.43E-02	IBEA	1.32E-02	IBEA	1.74E-02
9	IBEA	1.55E-02	GDE3	1.36E-02	MOCell	2.01E-02
10	NSGAII	1.62E-02	PAES	6.95E-02	AbYSS	2.03E-02
11	PAES	1.95E-02	MOEA/D	8.30E-02	PAES	2.06E-02
12	MOEA/D	2.02E-02	NSGAII	9.90E-02	OMOPSO	3.04E-02

By observing Table 4.11, 4.12 and 4.13 carefully, we find that MOJaDE is the most competitive algorithm and obtained the best values on all problems, while the GDE3 algorithm computed the second best fronts regarding this indicator in the evaluated problems. MOEA/D, SMPSO, AbYSS, MOCell and OMOPSO performed similarly, while PAES and IBEA performed poorly in regards to this indicator.

The IGD results from WFG1-WFG9 test problems generated by JADE algorithms are listed in Tables 4.11, 4.12 and 4.13, it is clearly shown that JADE achieves the best performance among its all other competitors in three objective function problems.

In addition, Tables illustrate the ranking of all three objective function problems WFG1 to WFG9. From tables can be see that the JADE performed better than other algorithms in all problems and takes the first position in all test problems.

4.2.2 Friedman Ranking Test

Friedman Test is a non-parametric statistical test developed by Milton Friedman more details are mentioned in 4.1.3 .

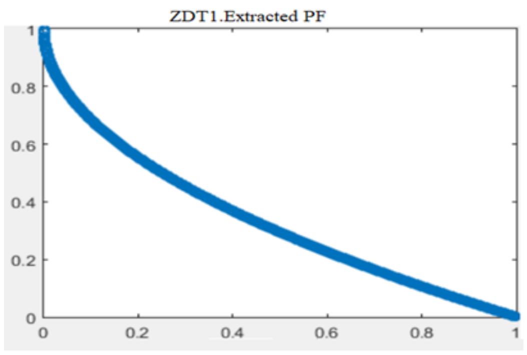
Table 4.14: Friedman Ranking between MOJaDE and Literature in D30.

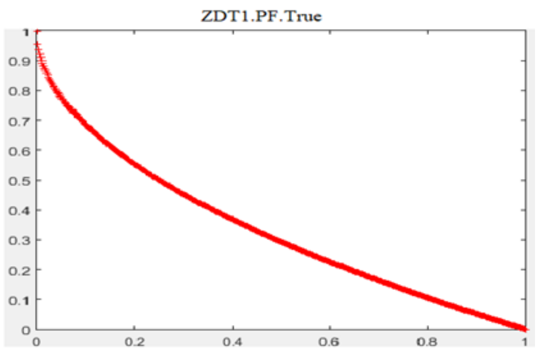
Algorithm	**Mean rank**
MOJaDE	1.646
OMOPSO	3.2451
MOCell	3.7351
SMPSO	2.8
AbYSS	4.4652
GDE3	2.2573
SPEA2	5.1341
CellDE	5.7131
IBEA	6.7
NSGAII	7.1463
PAES	8.1521
MOEA/D	6.1242

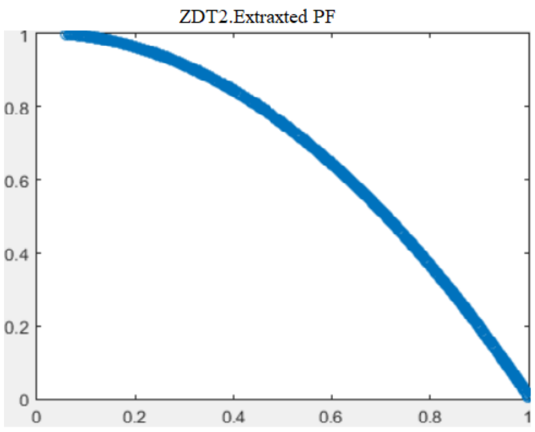
P-value = 4.5764E-17

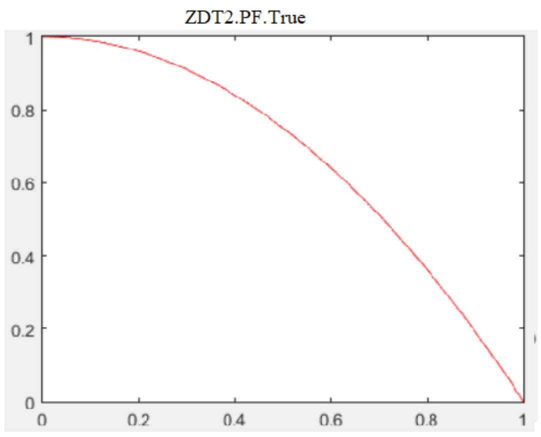
Table 4.14 ranking results showed that between all literature results in dimension 30, compared with JADE. The proposed JADE method showed the best performance overall.

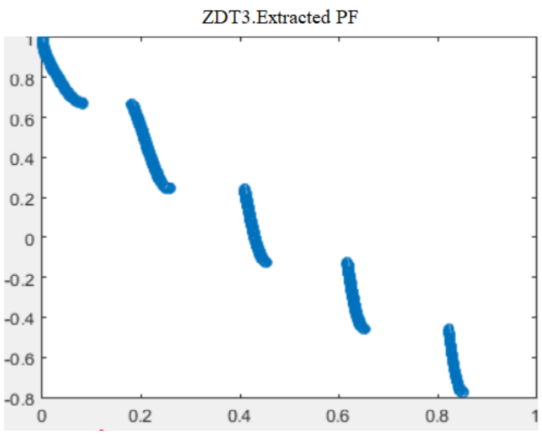
Comparisons between JADE with the second best acting algorithm in the competition called SMPSO denote that the JADE is more efficacious than SMPSO in 16 test problems over all 21, while SMPSO just win in 5 test problems.

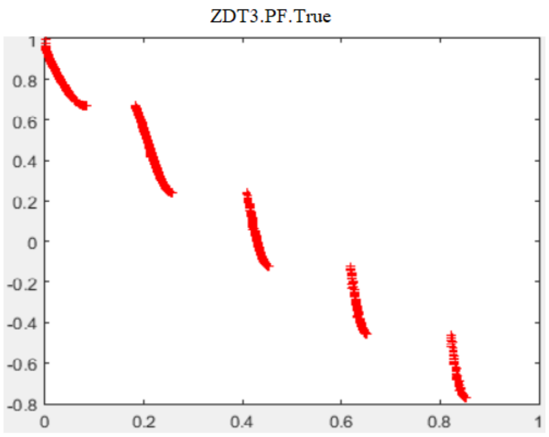


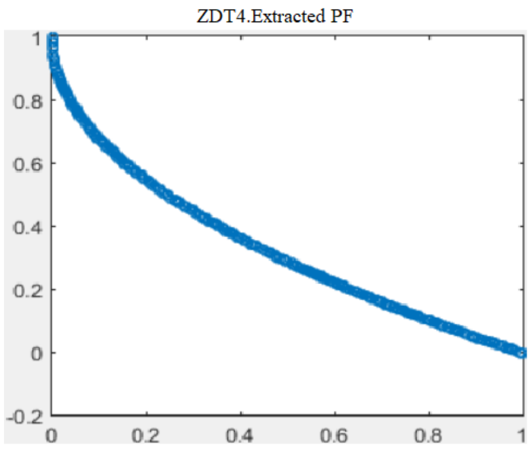


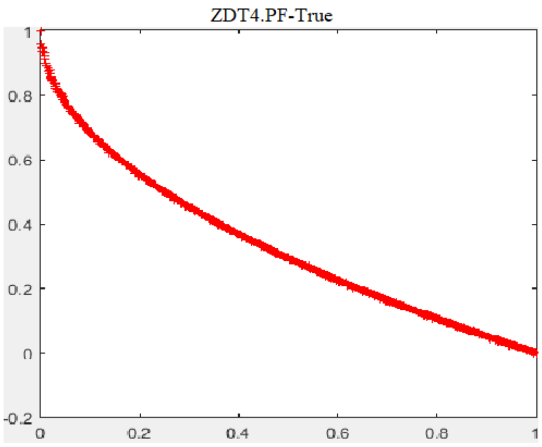

ZDT2.Extraxted PF

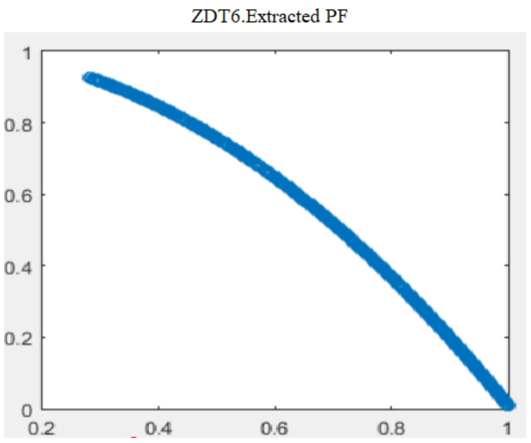

ZDT2.PF.True

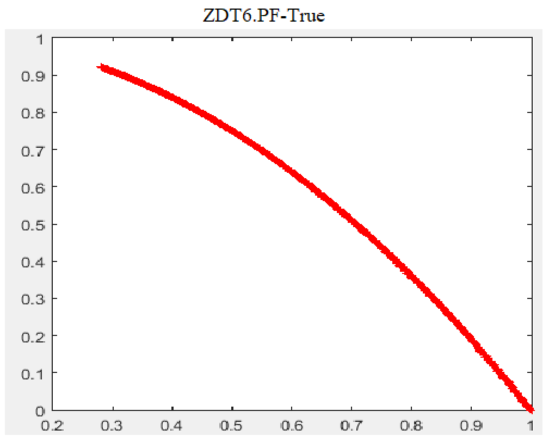

ZDT3.Extracted PF

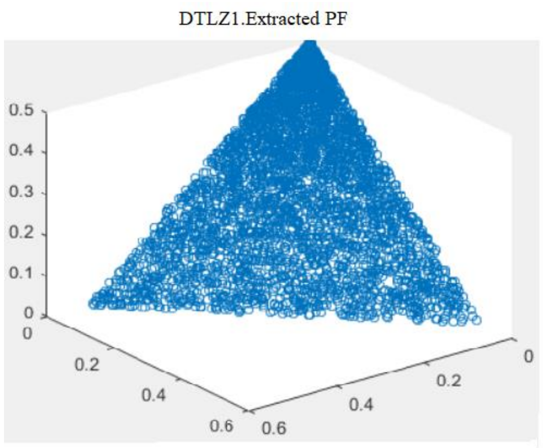

ZDT3.PF.True

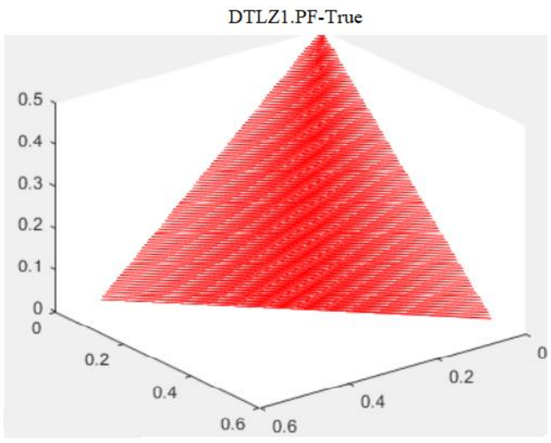

ZDT4.Extracted PF

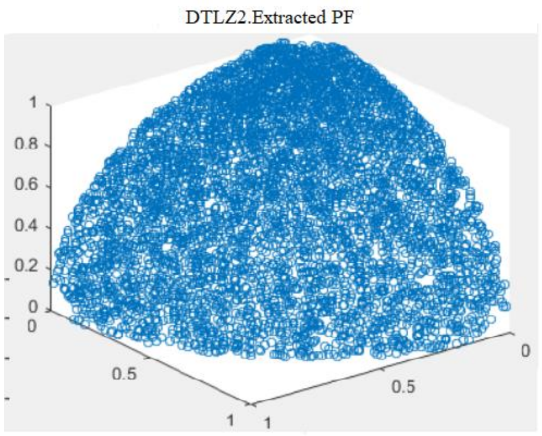

ZDT4.PF-True

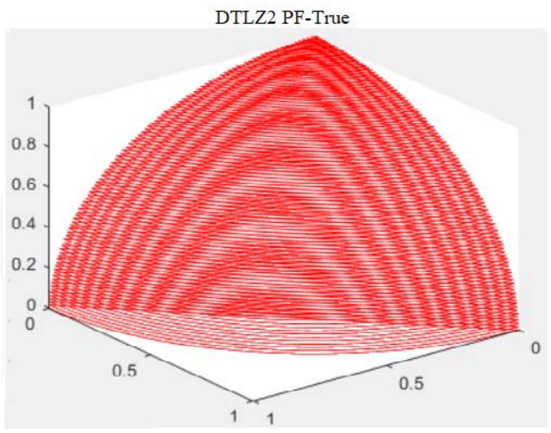

ZDT6.Extracted PF

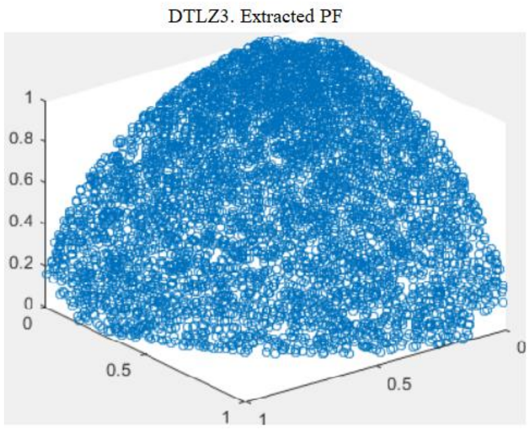

ZDT6.PF-True

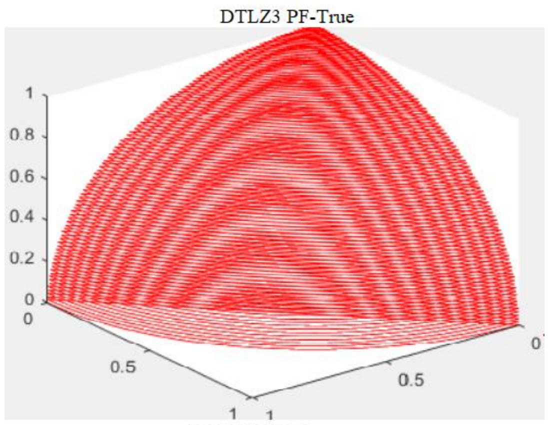

DTLZ1.Extracted PF

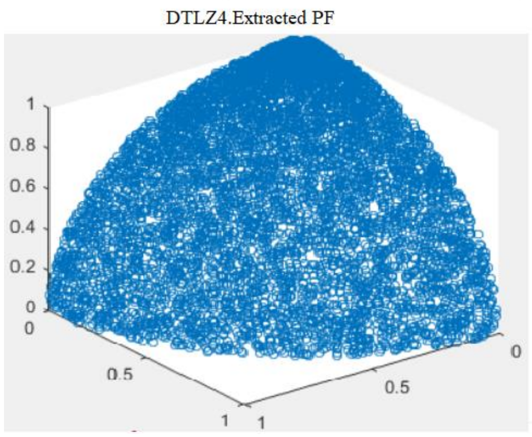

DTLZ1.PF-True

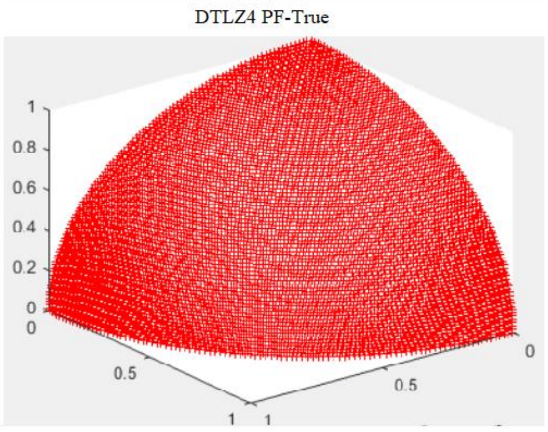

DTLZ2.Extracted PF


DTLZ2 PF-True

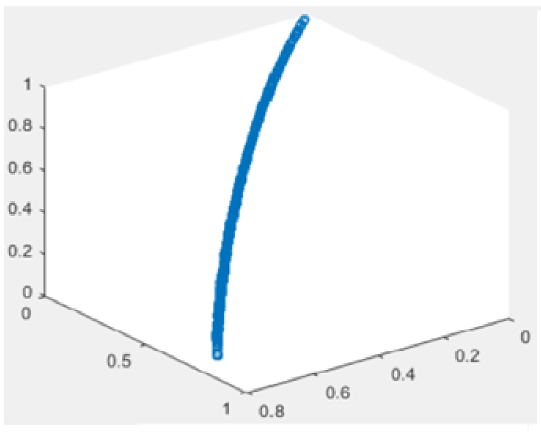

DTLZ3. Extracted PF


DTLZ3 PF-True

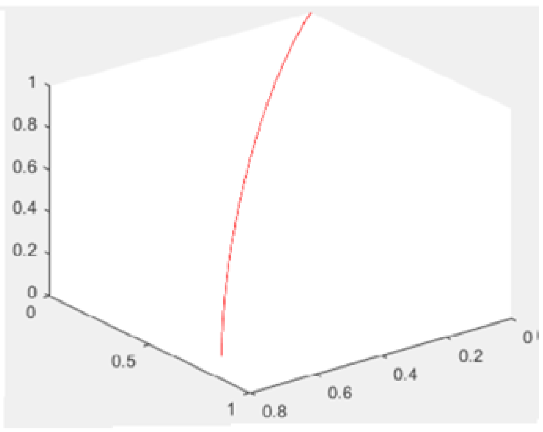

DTLZ4.Extracted PF


DTLZ4 PF-True

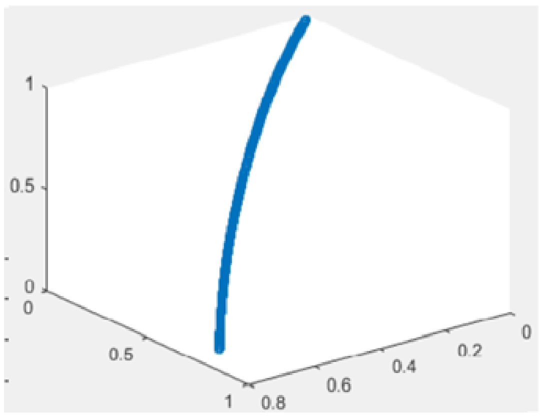
DTLZ5.Extracted PF



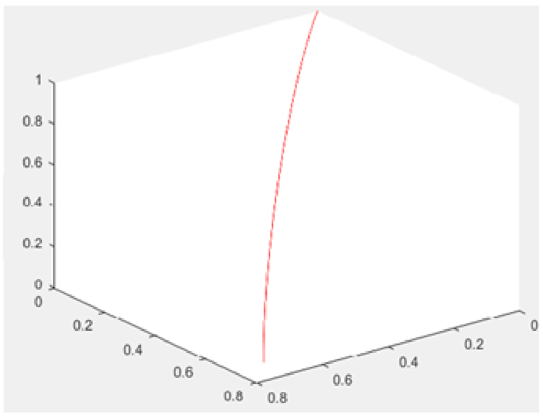
DTLZ5 PF-True



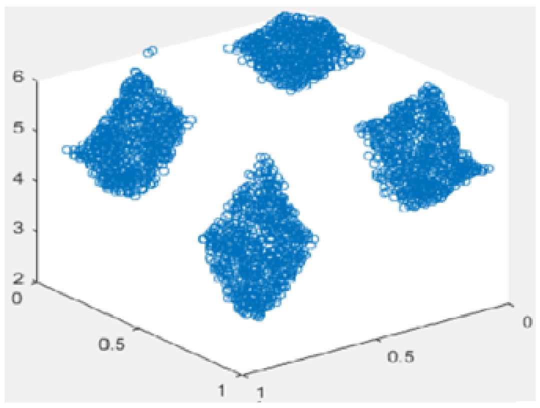
DTLZ6 Extracted PF



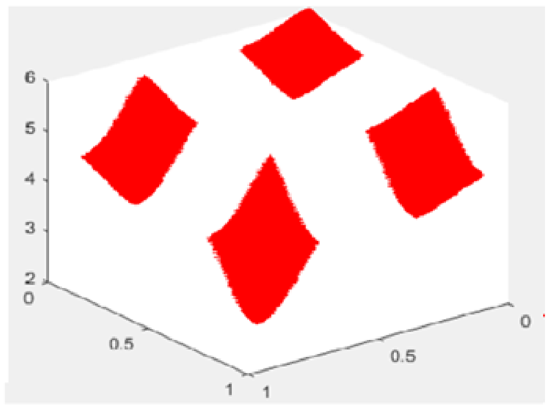
DTLZ6 PF-True

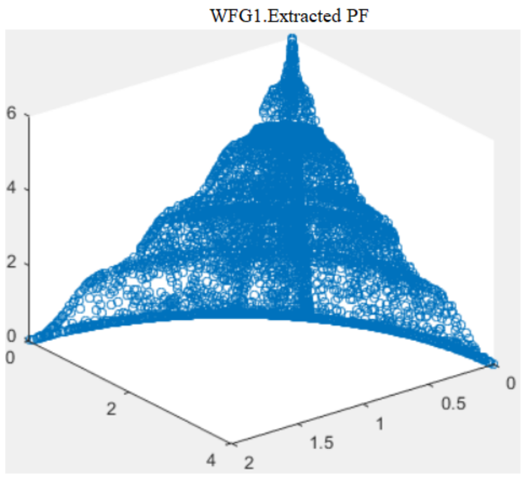


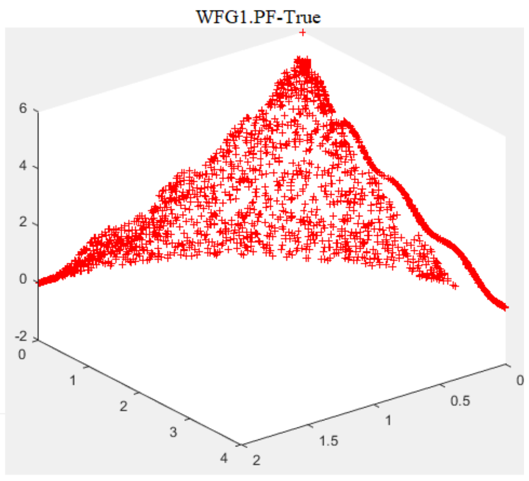
DTLZ7. Extracted PF

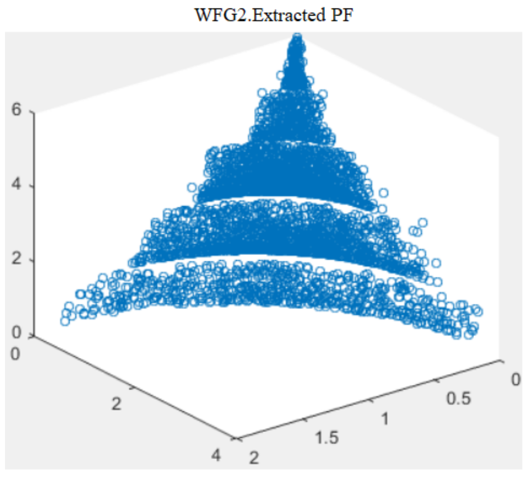


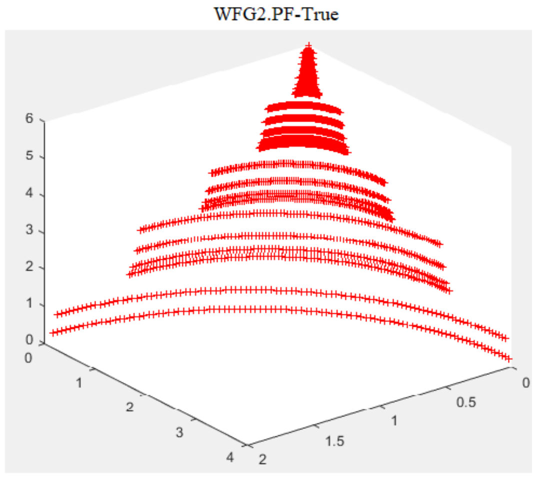
DTLZ7 PF-True




WFG1.Extracted PF
6
4
2
0
0
2
4
2
1.5
1
0.5
0


WFG1.PF-True
6
4
2
0
-2
0
1
2
3
4
2
1.5
1
0.5
0


WFG2.Extracted PF
6
4
2
0
0
2
4
2
1.5
1
0.5
0


WFG2.PF-True
6
5
4
3
2
1
0
0
1
2
3
4
2
1.5
1
0.5
0

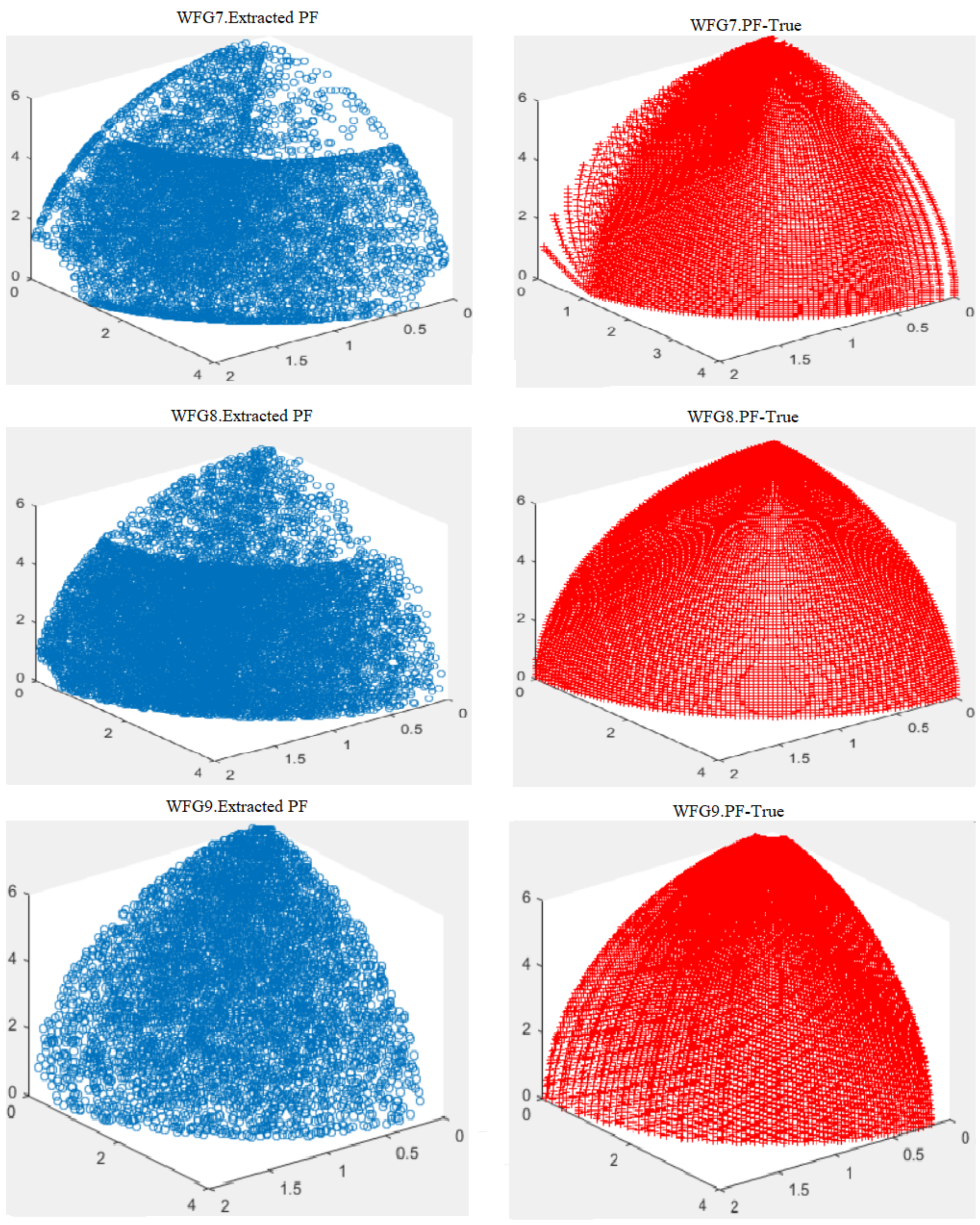


Figure 4.1: The Plots of Best Computed Pareto-Fronts and PF-True

Figure 4.1 illustrate the plots of computed Pareto-optimal set gained by JADE and Pareto Front True shared as a result of the competition .Plots present that the Pareto-optimal set found by JADE Algorithm quite close to PF-True and has a good spread.

Chapter 5

CONCLUSION

The optimization performance of an evolutionary algorithm can be enhanced through parameter adaptation, which automatically adjusts the control parameters to appropriate values over the course of the evolutionary search. Mutation Technique like “DE/current-to-pbest” are typically used together in an effort to bolster the rate of convergence, while ensuring that the algorithm remains highly reliable. This encouraged our interest in JADE, a differential evolutionary parameter “current-to-pbest”. with “current-to-pbest,”, the sets of the best solutions got the information where it’s used in balancing the population diversity and mutation greediness. The JADE parameter adaptation involved the evolution of mutation factors and crossover probabilities on the basis of their past successes. An external archive was also introduced for use in storing the already-explored inferior solutions, while the differences between them and the current population was used as a guide towards the optimum.

Tested against classic benchmark functions found in the literature, JADE consistently displayed a well and enhanced competitive degree of performance optimization in regards to the rate of convergence and reliability relative to other algorithms.

We suggested a novel differential evolution algorithm, MOJaDE for multi objective optimization through the self-adaptive control of parameters and the utilization of information gotten from archived inferior solutions.

REFERENCES

[1] Sörensen, K., & Glover, F. W. (2013). Metaheuristics. *Encyclopedia of operations research and management science*, 960-970.

[2] Goldberg, R. B., Barker, S. J., & Perez-Grau, L. (1989). Regulation of gene expression during plant embryogenesis. *Cell*, *56*(2), 149-160.

[3] Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4) 341-359.

[4] Ebrahimi, E., Monjezi, M., Khalesi, M. R., & Armaghani, D. J. (2016). Prediction and optimization of back-break and rock fragmentation using an artificial neural network.and a bee colony algorithm. *Bulletin of Engineering Geology and the Environment*, *75*(1), 27-36.

[5] Dorigo, M., Caro, G. D., & Gambardella, L. M. (1999). Ant algorithms for discrete .optimization. *Artificial life*, *5*(2), 137-172.

[6] Bertsimas, D., & Tsitsiklis, J. (1993). Simulated annealing. *Statistical science, 8*(1), 10-15.

[7] Chelouah, R., & Siarry, P. (2000). A continuous genetic algorithm designed for the global optimization of multimodal functions. *Journal of Heuristics*, *6*(2), 191-213.

[8] Dueck, G. (1993). New optimization heuristics: The great deluge algorithm and the record-to-record travel. *Journal of Computational physics*, *104*(1), 86-92.

[9] Cooke, Roger L. (2005). *The History of Mathematics: A Brief Course*. John Wiley & Sons. ISBN 978-1-118-46029-0.

[10] Humphreys, J. (2016). Book Review: Algorithms to live by: The computer science . of human decisions by Brian Christian and Tom Griffiths. *Philosophy of Coaching: An International Journal, 1*(1), 122-124. doi:10.22316/poc/01.1.10.

[11] Deus, H. (2017). A brief history of tomorrow. *Yuval Noah Harari. Vintage, UK*, 512.

[12] Minsky , M. L. (1967). *Computation.* Englewood Cliffs : Prentice-Hall.

[13] Savage, J. E., & Wloka, M. G. (1988, June). A parallel algorithm for channel routing.In *International Workshop on Graph-Theoretic Concepts in Computer Science* (pp. 288-303). Springer, Berlin, Heidelberg.

[14] Gurevich,Y.(2000).Sequential abstract-state machines capture sequential algorithms. *ACM Transactions on Computational Logic*, *1*(1), 77-111.

[15] Rogers, S. E., Chang, J. L., & Kwak, D. (1987). A diagonal algorithm for the method of pseudocompressibility. *Journal of Computational Physics, 73*(2), 364-379.

[16] Beni, G., & Wang, J. (1989). Swarm Intelligence in Cellular Robotic Systems Proceed. NATO Advanced Workshop on Robots and Biological Systems, Tuscany, Italy, June 26-30. *Y.: NATO*.

[17] Barricelli, N. A. (1954). Esempi numerici di processi di evoluzione. *Methodos, 6*(21-.... 22), 45-68.

[18] Fraser, Alex, "Simulation of genetic systems by automatic digital computers. I. Introduction". Aust. J. Biol. Sci. vol. 10, pp. 484-491, 1957.

[19] Bremermann, H. J. (1958). *The evolution of intelligence: The nervous system as a model of its environment*. University of Washington, Department of Mathematics.

[20] J. H. Holland, "Outline for a logical theory of adaptive systems," Journal of the Association for Computing Machinery, vol. 3, pp. 297-314, 1962.

[21] J. H. Holland, Adaptation in natural and artificial systems. The University of Michigan Press, Ann Arbor, MI, 1975.

[22] Liang, J. J. (2008). *Novel particle swarm optimizers with hybrid, dynamic & adaptive neighborhood structures* (Doctoral dissertation, PhD thesis, Nanyang Technological University, Singapore).

[23] L. J. Fogel, "Toward inductive inference automata," In Proceedings of the International Federation for Information Processing Congress, pp. 395-399, 1962.

[24] I. Rechenberg, "Cybernetic solution path of an experimental problem," Royal Aircraft Establishment, Library translation No. 1122, Farnborough, Hants., UK, August 1965.

[25] J. R. Koza, "Genetic Programming: A Paradigm for Genetically Breeding Populations of Computer Programs to Solve Problems," Stanford University Computer Science Department technical report STAN-CS-90-1314, 1990.

[26] Ahmadi-Nedushan, B. (2012). An optimized instance based learning algorithm for estimation of compressive strength of concrete. *Engineering Applications of Artificial Intelligence, 25*(5), 1073-1081.

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

[28] Thomas, P., & Vernon, D. (1997, September). Image registration by differential evolution. In *Irish Machine Vision and Image Processing Conference, Magee College, University of Ulster, Ireland* (pp. 221-225).

[29] Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for Differential Evolution. *2013 IEEE Congress on Evolutionary Computation*..doi:10.1109/cec.2013.6557555.

[30] Abbass, H. A., Sarker, R., & Newton, C. (2001, May). PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems..In *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)* (Vol. 2, pp. 971-978). IEEE.

[31] Martinez-Estudillo, A. C.; Hervas-Martinez, C.; Martinez-Estudillo, F. J.; Garcia-Pedrajas, N. (2005). *Hybridization of Evolutionary Algorithms and Local Search by means of a Clustering method*. IEEE transactions on Systems, Man, and Cybernetics,Part B (Cybernetics), vol 36, issue 3. DOI:10.1109/TSMCB.2005.860138.

[32] S. M. I. A. C. S. I. Jingqiao Zhang, "JADE: Adaptive Differential Evolution with Additional External Archive" *IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION*, vol. 13, p. 18, 2009.

[33] K. Price, R. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, 1st ed. New York: Springer-Verlag, Dec. 2005.

[34] B. V. Babu and M. M. L. Jehan, “Differential evolution for multiobjective optimization,” in *Proc. IEEE Congr. Evol. Comput*. Dec. 2003, pp. 2696–2703.

[35] U. Pahner and K. Hameyer, “Adaptive coupling of differential evolution and multiquadrics approximation for the tuning of the optimization process,” *IEEE Trans Magnetics*, vol. 36, no. 4, pp. 1047–1051, Jul. 2000.

[36] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello, “Modified differential evolution for constrained optimization,” *in Proc. IEEE Congr. Evol. Comput.*, Vancouver, BC, Jul. 2006, pp. 25–32.

[37] T. Back, *Evolutionary Algorithms in Theory and Practice: Evolution Strategies,Evolutionary Programming, Genetic Algorithms*.NewYork: Oxford University Press, 1996.

[38] H. G. Beyer,*Theory of Evolution Strategies*. New York: Springer-Verlag, Apr. 2001.

[39] J. Zhang and A. C. Sanderson, "Self-adaptive multi-objective differential evolution with direction information provided by archived inferior solutions," *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence,* p. 10, 2008.

[40] C. A. Coello Coello and G. B. Lamont, *Applications of Multi-Objective.Evolutionary Algorithms,* World Scientific, 2004.

[41] Jingqiao Zhang and Arthur C. Sanderson, "JADE: Self-Adaptive Differential Evolution with Fast and Reliable Convergence Performance," *IEEE Congress on Evolutionary Computation, Sept 2007, Singapore*.

[42] J. Mo and J. Walrand, "Fair end-to-end windows-based congestion control,"*IEEE/ACM Trans. On Networking*, vol. 8, no. 5, pp. 556-567, University of California at Berkeley Oct. 2000.

[43] V. L. Huang, P. N. Suganthan and S. Baskar, "Multiobjective Differential Evolution with External Archive and Harmonic Distance-Based Diversity Measure," Technical Report, Nanyang Technological University, Singapore, Dec., 2005.

[44] Iorio, A and Li, X.(2006), "Incorporating Directional Information within a Differential Evolution Algorithm for Multiobjective Optimization", in Proceeding of Genetic and Evolutionary Computation Conference 2006 (GECCO'06), eds. M. Keijzer, et al., p. 691-697, ACM Press.

[45] P. N. Suganthan, Performance Assessment on Multi-objective Optimization Algorithms. Nanyang Technological University, 2017.

[46] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger & S. Tiwari,"Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session onReal-Parameter Optimization," Technical Report, Nanyang Technological University, Singapore, May 2017 and KanGAL Report #2017017, IIT Kanpur, India, 2017.

[47] k. Elhalawany, "Bee Colony Optimization for Single and Multi-Objective Numerical Optimization," M. S. thesis, Eastern Mediterranean University, Gazimağusa, North Cyprus, IL, 2019.

[48] R. Akbari, R. Hedayatzadeh, K. Ziarati and B. Hassanizadeh, *A multi-objective artificial bee colony algorithm*. Swarm and Evolutionary Computation, 2012.

[49] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y.-P. Chen, A. Auger & S. Tiwari,"Problem Definitions for Performance Assessment on Multi-objective Optimization Algorithms," Technical Report, Nanyang Technological University,Singapore, May 2017 and KanGAL Report #2017017, IIT Kanpur India, 2017.

[50] C. Y. See, Y. H. Chan and S. Yusup, "Advances in Feedstock Conversion Technologies for Alternative Fuels and Bioproducts," Kumamoto University,Kumamoto, Japan, Woodhead Publishing Series in Energy, 2019, pp. 281-298.

[51] *Friedman Ranking Test* website: http://vassarstats.net/textbook/ch15a.html

[52] Sherinov, Z.; Ünveren, A. (2017). *Multi-objective Imperialistic Competitive Algorithm with Multiple Non-Dominated Sets for the Solution of Global Optimization Problems*. Soft Compute. DOI: https://doi.org/10.1007

[53] J. D. Knowles and D. W. Corne, “Approximating the nondominated front using the Pareto Archived Evolution Strategy,” *Evolutionary computation*, vol. 8, no. 2, pp. 149–172, 2000.

[54] K. Deb, A. Pratap, S. Agarwal and T. Meyarivan, *A fast and elitist multiobjective genetic algorithm: NSGA-II*. IEEE Transactions on Evolutionary Computation,182–197, 2002.