9-20-2019

Cultural Algorithm based on Decomposition to solve Optimization Problems

Ramya Ravichandran
University of Windsor

Cultural Algorithm based on Decomposition to solve Optimization Problems

By

Ramya Ravichandran

A Thesis
Submitted to the Faculty of Graduate Studies
through the School of Computer Science
in Partial Fulfillment of the Requirements for
the Degree of Master of Science
at the University of Windsor

Windsor, Ontario, Canada

2019

Cultural Algorithm based on Decomposition to solve Optimization Problems

By

Ramya Ravichandran

APPROVED BY:

_____

K. Tepe
Department of Electrical and Computer Engineering

_____

S. Samet
School of Computer Science

_____

Z. Kobti, Advisor
School of Computer Science

September 20th, 2019

DECLARATION OF CO-AUTHORSHIP / PREVIOUS PUBLICATION

1. Co-authorship

I hereby declare that this thesis incorporates material that is the result of research conducted under the supervision of Dr. Ziad Kobti. In all cases, the key ideas, primary contribution, experimental designs, data analysis, and interpretation were performed by the author, and the contribution of the co-author was primarily through the proofreading of the published manuscripts.

I am aware of the University of Windsor Senate Policy on Authorship, and I certify that I have properly acknowledged the contribution of other researchers to my thesis, and have obtained written permission from each of the co-author(s) to include the above material(s) in my thesis.

I certify that, with the above qualification, this thesis, and the research to which it refers, is the product of my work.

2. Previous Publication

This thesis includes one original paper that has been previously submitted for publication in peer-reviewed journals, as follows:

Section	Publication title/ Full citation	Publication status
3, 4	Ramya Ravichandran and Ziad Kobti " Solving Dynamic Multi-Objective Optimization Problem Using Cultural Algorithm based on Decomposition." In 2019 International Symposium on Computing and Artificial Intelligence (ISCAI 2019), Vancouver, Canada.	Accepted

I certify that I have obtained a written permission from the copyright owner(s) to include the above published material(s) in my thesis. I certify that the above material describes work completed during my registration as a graduate student at the University of Windsor.

3. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any propriety rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my thesis, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent

that I have included copyright material that surpasses the bounds of fair dealing within the meaning of Canada Copyright Act, I certify that I have obtained written permission from the copyright owner to include such material in my thesis.

I declare that this is a true copy of my thesis, including any final revisions, as approved by my thesis committee and the Graduate Studies Office, and that this thesis has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Decomposition is used to solve optimization problems by introducing many simple scalar optimization subproblems and optimizing them simultaneously. Dynamic Multi-Objective Optimization Problems (DMOP) have several objective functions and constraints that vary over time. As a consequence of such dynamic changes, the optimal solutions may vary over time, affecting the performance of convergence. In this thesis, we propose a new Cultural Algorithm (CA) based on decomposition (CA/D). The objective of the CA/D algorithm is to decompose DMOP into a number of subproblems that can be optimized using the information shared by neighboring problems. The proposed CA/D approach is evaluated using a number of CEC 2015 optimization benchmark functions. When compared to CA, Multi-population CA (MPCA), and MPCA incorporating game strategies (MPCA-GS), the results obtained showed that CA/D outperformed them in 7 out of the 15 benchmark functions.

DEDICATION

I would like to dedicate this thesis to my family

Father: Ravichandran Egappan

Mother: Janagi Ravichandran

Sister: Aishwarya Ravichandran

ACKNOWLEDGMENTS

There are many people I would like to thank for the successful completion of my master thesis. First and foremost, I would like to express my sincere gratitude to my supervisor, Dr. Ziad Kobti, for his continuous support of my research study, for his patience, motivation, enthusiasm, and immense knowledge. He helped me in accomplishing my goals and has provided valuable guidance in improving research skills. It was a great pleasure to work and discuss with him. I would also like to appreciate the time he dedicated for me and the funding he provided me to complete this thesis. Without his support, this thesis would not be complete.

I would also like to thank my committee members – Dr. Tepe and Dr. Samet – whose inputs and suggestion have given a better shape to my research. I also would like to thank the support of Mrs. Melissa, Mrs.Christine, and Mrs. Gloria for helping with various academic issues.

I am very thankful to my parents and friends who gave me the strength, moral support, and unconditional love, which kept me motivated to pursue my research. Last but not least, I would like to thank God for giving this excellent opportunity.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

EA	Evolutionary Algorithms
CA	Cultural Algorithms
SOP	Single Optimization Problem
MOP	Multi-Objective Problem
DMOP	Dynamic Multi-Objective Optimization Problem
DOP	Dynamic Optimization Problem
GA	Genetic Algorithms
DE	Differential Evolution
POS	Pareto Optimal Solutions
POF	Pareto Optimal Front
CR	Convergence Ratio
NSGA-II	Non-Dominated Sorting Genetic Algorithm – II
FPS	Feed-Forward Population Strategy
PPS	Prediction-based Population Strategy
MOEA/D-DP	Differential Prediction incorporated by Multi-Objective Optimization Evolutionary Algorithm based on Decomposition
KF	Kalman Filter
SGEA	Steady-State and Generational Evolutionary Algorithm
MRP-MOEA	Multiple Reference Point – MOEA
CAEP	Cultural Algorithm with Evolutionary Programming
GPM	Genotype Phenotype Mapping

MPCA	Multi Population Cultural Algorithm
HMPCA	Hetrogeneous Multi Population Cultural Algorithm
TM	Tchebycheff Method
RP	Reference Point Method

Chapter 1

Introduction

1.1 Background

Optimization problems involve finding one or more effective and efficient solution(s) from a pool of feasible solutions. In other words, it involves finding the best solution by maximizing the desired factors and minimizing undesired factors [30]. Evolutionary Algorithms (EA) have been widely used by researchers to solve complex optimization problems. EA contains a search space which focuses on the optimization of the problem and searches for the best possible solution [14]. The search space in EA comprises exploration and exploitation operators. Exploration means finding new points in different areas of the search spaces, which has not been investigated before. On the other hand, exploitation is the process of improving and combining the traits of the currently known solutions [31]. The solutions generated can be near-optimal or optimal. While EAs are successfully applied to various types of optimization problems, they undergo specific issues such as immature convergence and diversity over

generations. Diversity can be maintained between the population by using Cultural Algorithms (CA). CA is a class of EA which is most likely used to solve multi-objective problems (MOP). Introducing decomposition of the MOP in CA can address the issue of immature convergence (finding solutions as close as possible to Pareto optimal front) as decomposing the problem into many subproblems, enhances the search for best solutions which shows good potential for better results.

Decomposition is a traditional and primary method used to solve multi-objective problems. As the name suggests, it decomposes a multi-objective optimization into many simple scalar optimization subproblems, and also optimizes these problems simultaneously [1]. Using Decomposition strategies in CA can provide a balance between the exploration and exploitation in the Evolutionary algorithms. It can make efficient use of the knowledge obtained from the sub-problem(s) to decide whether to co-operate with another sub-problem and generate excellent results. The combination of these two different fields can cover the significant aspects of diversity, immature convergence, escaping from local optima, exploration, and exploitation. This combination can lead to better results and efficiently solve optimization problems.

1.2 Problem Definition

Ongoing research focuses on solving optimization problems which have a single objective function commonly known as Single-Objective Optimization Problem (SOP). Formally, consider an optimization problem denoted as follows [1]:

$$min/max \qquad f(x), \tag{1.1}$$

$$subject\ to \qquad g_j\ (x) \geq 0, \qquad \text{j} = 1,2, \ldots, \text{J};$$

$$h_k\ (x) = 0, \qquad \text{k} = 1,2, \ldots, \text{K}.$$

where, $x = (x_1, x_2, \ldots, x_n)^T$ is a vector of n decision variables, $g_j\ (x)$ and $h_k\ (x)$ are equality and inequality constraints respectively. However, real-world problems usually involve one or more objectives to be optimized, and this is termed: Multi-Objective Optimization Problem (MOP) [2]. A typical example of MOP is the problem of buying a car [3]; we tend to select one which has maximum comfort and minimum cost; these issues considered here are called objective functions. A car with maximum comfort usually has a higher cost; whereas a car with minimum cost sacrifices comfort. Objective functions conflict with each other making this problem exciting and challenging to solve.

MOPs have to solve these conflicting and competing objectives. The solutions obtained are known as Pareto Optimal Solutions (POS) or Non-

Dominated Solutions. A set of POS is called a Pareto Front (PF), if it is represented graphically and forms a clear-cut curve by joining all the optimal solutions in the objective space. Mathematically, an MOP is expressed as [3]:

$$\begin{aligned} &min/max \quad f_m(x), \quad && \text{m} = 1,2,\ldots,\text{M}; \\ &subject\ to \quad g_j(x) \geq 0, \quad && \text{j} = 1,2,\ldots,\text{J}; \\ &\qquad\qquad h_k(x) = 0, \quad && \text{k} = 1,2,\ldots,\text{K}. \end{aligned} \tag{1.2}$$

where, $x = (x_1, x_2, \ldots, x_n)^T$ is a vector of n decision variables, $g_j(x)$ and $h_k(x)$ are equality and inequality constraints, respectively, m is the number of objectives (m ≥ 2). The MOP finds multiple optimal solutions which have a wide range of values for the objective functions, and later choosing one optimal solution with the help of higher-level information. If there is no further information about the problem; then it will be challenging to choose one solution over all other optimal solutions which are now equally important.

Therefore, there are two goals in solving MOP, namely: Convergence and Diversity. Thus, *Convergence*. Finding a set of solutions as close as possible to the POF and *Diversity*. Finding a set of solutions as diverse as possible. There are other basic concepts of optimization problems such as [4]:

Pareto dominance: A solution $x = (x_1, x_2, \ldots, x_n)$ dominates (denoted by $\prec$) another solution $y = (y_1, y_2, \ldots, y_n)$ if and only if $f(x)$ is comparatively less than $f(y)$. That means, $\forall m \in \{1, \ldots, M\}$, we have $f_m(x) \leq f_m(y)$ and $\exists m \in \{1, \ldots, M\}$, where $f_m(x) < f_m(y)$.

Pareto optimal solutions: A solution $x = (x_1, x_2, \ldots, x_n)$ is said to be an optimal solution if and only if there is no $y = (y_1, y_2, \ldots, y_n)$ that y dominates x.

Pareto optimal set: Given MOP $f(x)$, the Pareto optimal set $P = \{x \in \Omega \mid \nexists y \in \Omega, f(y) \prec f(x)\}$, which is also known as non-dominated solutions as discussed.

Pareto front: Given MOP $f(x)$ and its Pareto Optimal set P, the Pareto front PF is $\{f(x), x \in P\}$.

1.2.1 Dynamic Multi-Objective Optimization

In the real world, a dynamic change to an optimization problem should be taken into account; where the objective functions, constraints, as well as the decision variables may change with respect to time [5]. Furthermore, considering our car purchasing problem mentioned earlier, it is possible that some desirable cars may not be available for sale at the moment or anymore; or there are newer car models available in the market; or the price of your desired car has gone up over time, etc. All these culminate to Dynamic Optimization Problems (DOPs).

When numerous competing objective functions and constraints change with respect to time simultaneously in real-world DOPs [3], the problem is called a Dynamic Multi-Objective Optimization Problem (DMOP). As a result, the POS and PF may vary with regard to time. In this thesis, we consider the following DMOPs [6]:

$$min\ F\ (x, t) = (f_1(x, t), f_2(x, t)\ , \ldots , f_m(x, t))^{\mathrm{T}} \tag{1.3}$$

$$subject\ to\ x \in \Omega$$

where m is the number of objectives, $t = 0, 1, 2\ldots$ is the discrete-time instant, $x = (x_1, x_2\ , \ldots , x_n)^T$ is the decision variable vector, and Ω represents the decision space. The objective function $F\ (x, t)$ have m time-dependent objective functions that vary periodically.

There exist a variety of algorithms or optimization techniques used in solving DMOPs such as (1.3). Some of the classical methods to solve a MOP is to group all the objective functions into a single function. In other words, the conversion of MOP to SOP. A few traditional methods are the weighted-sum method [4], the ε-constraint method [41], and the goal-programming method [4]. There are specific difficulties which may accompany the classical optimization methods. In the weighted-sum method, the shape of the curve for the Pareto Optimal front is sensitive. The required knowledge concerning the problem for these traditional methods is not available. Also, there are other

optimization methods to solve DMOPs, such as particle swarm optimization [9] and artificial immune systems [10].

Another method for solving the optimization problems is using the Evolutionary Algorithms (EAs). An advantage of EA over traditional methods for a MOP is that the former operates over a set of solutions at a time. This method performs satisfactorily well when dealing with DMOP. Therefore, applying EAs has grabbed the attention of researchers. In 1966, the first method to solve the application of dynamic environments using EA was introduced. However, it became widely known and used in the late 80′*s* Dealing with DMOP is complicated and as a result of the dynamism, algorithm design for a DMOP is different from that of a static MOP. As discussed the goals of the MOP algorithm is to have fast convergence as well as be able to track the loss in diversity during an environmental change. Therefore, many other new additional techniques were introduced to maintain diversity in the population-based methods. When dealing with DMOPs, the main goal is not only to converge to a well-diversified Pareto Front but also to rapidly track down the PF as it changes over time. The proposed algorithm should have a high convergence rate.

1.3 Evolutionary Computation

Evolutionary Computation (EC) is a set of algorithms which are inspired by the evolution of the biological model. EC is one of the branches of artificial intelligence, which is used for metaheuristic and stochastic optimization of complex problems [14]. Evolutionary algorithm (EA) is a subset of EC; hence, they are also known as optimization algorithms. There is numerous algorithm which comes under EA, such as:

1. Genetic Algorithms
2. Differential Evolution
3. Cultural Algorithms
4. Coevolution

The standard underlying concept in all the evolutionary algorithm is the same: given a set of the population, which in environmental pressure causes natural selection. The fitness function evaluates each candidate, and only the better candidate survive the next generation, eliminating the worst ones. Each individual is evolved by using mutation and recombination operators. Mutation is applied to only on one individual and as a result, we get a new candidate whereas in recombination two individuals (called parents) are selected and it

results in the new generation of one or more new candidates (called offsprings). Mutation and recombination operators generate a new set of candidates (offsprings) which replace the existing old individuals for the next generation. This process repeats until the stopping criteria are met (number of generations, CPU time). Figure 1.1 represents the pseudo-code for the evolutionary algorithm [23].

When an algorithm incorporates genetics in the process of evolution is known as Genetic Algorithms (GA). GAs are a heuristic search algorithm which is based on evolutionary ideas of natural selection. GA was first proposed by Holland [24], which is inspired by the Darwin theory of evolution and biological genetics. GA was used by many researchers to solve optimization problems. However, a simple GA converges into a single optimum, and it is not suitable for multi-objective optimization. GA evolves complex problems by coevolution, which also includes explicit notions of modularity to provide a fair chance to complex problems to evolve in the form of co-adopted subcomponents. The structure for complex problems is noted when there is a need for rule hierarchies in classifier systems and subroutines in genetic programming [25]. When two or more individuals reciprocally affect each other in evolution, then it is known as Coevolution. The main disadvantage of coevolution is that it has a good chance of losing diversity among the population.

```
Evolutionary Algorithm();
    Initialize population;
    Evaluate initial population;
    WHILE convergence criteria IS NOT satisfied, DO
        Selection technique;
        Crossover operations;
        Mutation operators;
        Evaluation operators;
        Update Population;
    END WHILE
```

Figure 1.1 Pseudo-code for EA [23]

Differential Evolution (DE) is also an EA which was introduced by Storn and Price [25] to solve global optimization problems. DE was designed to solve continuous problems but also works excellent in combinatorial optimization problems. Even on the continuous domain, it cannot be applied directly, but overall, it shows good performance on optimization problems and some permutation problems [27, 28]. DE is popular among the EA due to its robust search space exploration. In DE, the differential formulation mechanism is used to generate offspring from the population. All of the above-discussed EAs are used to solve complex optimization problems, but none of them uses knowledge of the individual to solve. To apply the knowledge possessed by the individual or population, Reynolds [12] designed the Cultural Algorithms (CA).

Cultural Algorithms extracts knowledge and uses them to direct the search process. A huge number of successful applications of CA exhibits the performance of knowledge-based EA. The search mechanism is improved by amending the extracted knowledge into a CA. Therefore it leads CA to find better solutions with excellent quality and also improves the convergence rate. The inspiration for CA is from human cultures and beliefs. Unlike the other EAs, CA has two search spaces: the population space and belief space. Population space consists of individuals in the population, and belief space consists of the knowledge of the best individual in the population of the current generation. There are five knowledge components of CA, such as situational, topographical, historical, normative, and domain. It is discussed briefly in Chapter 2.

1.4 Decomposition

There are several approaches for converting a MOP for the approximation of the Pareto Front into a number of scalar optimization problems. Decomposition is similar to the traditional and primary method used to solve multi-objective problems. As the name suggests, it decomposes a multi-objective optimization into many simple scalar optimization subproblems and also optimizes these problems simultaneously [1]. Information of several neighboring subproblems is used to solve a subproblem. The idea of decomposition has been started to involve in the current state-of-the-art in DMOPs algorithm. For example, a decomposition algorithm consists of a set of scalar optimization

problems in which the objectives are the aggregation of the objectives in DMOP. A scalar optimization algorithm is applied to these scalar optimization problems in a chain based on the coefficients of aggregation, the solution obtained from the previous problem is set as the starting point for the next subproblem to be solved. This is done because the next aggregation objective is just slightly different from the previous one.

In 1979, Hwang and Masud presented the classification of decomposition methods according to the participation of the decision-maker [15]. There are four classes, namely, no-preference methods, priori methods, posteriori methods, and interactive methods. Interactive methods are the most advanced class out of the four methods mentioned. Interactive methods are believed to produce the most satisfactory results. The detailed discussion about the types of decomposition methods is presented in Chapter 2.

1.5 Research Motivation

The primary motivation of this research has come from observing the techniques to solve complex optimization problems. While working on the optimization problems, we found there are many algorithms which can be used. The main problem with most of the algorithm was that they were less general and more problem-specific. Mostly the existing algorithm tries to solve these

problems in static rather than a dynamic way. After working in this topic, we realized the Cultural Algorithms, shows many potentials to solve complex optimization problems, and they also resemble the human culture. Exchange of knowledge between the individuals in the environment can help them to explore and exploit conditions around them more precisely. We implement this idea by introducing specific strategies such as breaking the DMOP into many subproblems for achieving a better quality of results and convergence performance. In this thesis, we focus on implementing different decomposition strategies in CA for better performance. Convergence based approaches try to make use of past information for thriving better tracking performance.

1.6 Thesis Statement

In this thesis, the goal is to improve the convergence performance and track the optima. We are aiming to achieve this objective by introducing Decomposition strategy into Cultural Algorithm, which uses belief space to store past information about each individual. This past information is used to thrive in better performance for the convergence. These algorithms also use domain knowledge that will lead to faster convergence. Complexity in DMOPs makes it challenging to handle them. The decomposition strategy will help to handle the complexity of the problem. We will evaluate our method using the CEC 2015 Benchmarks and analyze the results.

1.7 Thesis Contribution

In our work, we aim to develop and evaluate different decomposition strategies to improve the results of Dynamic Multi-Objective Optimization Problem. Different decomposition techniques are compared with each other to evaluate and identify the better method on its performance on optimizing the complex problems. In our work, we hypothesize that decomposition techniques incorporated in CA will lead to improving the performance in DMOP through accelerating the convergence. In our study, we hypothesize when a DMOP is decomposed into many subproblems by using one of the specific strategies proposed that will affect the whole population and improve the performance. In order to evaluate the efficiency of the proposed algorithm, the Convergence Ratio (CR) measure is incorporated. We have developed our framework based on the work done by Cao [6], Parikh [19] and implemented different decomposition techniques incorporated by cultural algorithms. CEC 2015 [29] expensive benchmark functions have been used to test our framework and compare it with existing algorithms. Testing is done on both 10 and 30-dimensional functions of CEC. The function consists of different types of simple, multimodal, hybrid, and composite functions.

1.8 Thesis Outline

The rest of the thesis/research work is organized as follows

In Chapter III, I discuss the related work/literature review in the field of optimization problems using different techniques.

In Chapter III, I introduce Evolutionary Computation and explain its working in detail. We also introduce CA and types of Decomposition methods that are used in this research.

In Chapter IV, I explain the proposed approach, which makes it possible to utilize evolutionary techniques in complex optimization problems.

In Chapter V, I present the experimental setup and results with its assumption.

In Chapter VI, I compare our methods with state-of-the-art techniques and deeply analyze the results. We also compare the results of different decomposition techniques.

In Chapter VII, I conclude the research by providing insights for future work.

Chapter 2

Literature Review

This chapter consists of all the related work obtained for the establishment of fundamental ideas, developing our framework, and the structure of our thesis. In this section, we explain the literature related to Dynamic Multi-Objective Optimization Problem (DMOP), Cultural Algorithms, and Decomposition strategies. The first section consists of the traditional methods used to solve DMOPs. The second section of this chapter consists of evolutionary methods to solve DMOPs. The third section consists of literature for cultural algorithms solving DMOPs.

2.1 Traditional methods to solve DMOPs

In general, for Multi-Objective Optimization Problem (MOP), it is intuitive to propose the aggregation of the different objective functions into a single one. In order to generate an emblematic approximation of the whole PF, the user must perform several runs with different parameter settings. In the following, we will

explain some classic methods for handling MOPs. Cohen [41] classified them into the following two types:

1. Generating methods
2. Preference-based methods

In the *generating methods*, a handful of non-dominated solutions are produced, and one solution from the obtained non-dominated solutions is chosen. No prior knowledge about the relative importance of each solution is given. On the other hand, *preference-based methods*, some known information/preference for each objective function is used in the optimization process. Meitten [1] further fine-tuned the above classification into four different classes.

1. No-preference methods
2. Posteriori methods
3. A priori methods
4. Interactive methods

The *no-preference methods* do not obtain any information about the importance of the objective function, but intuition is used to find a single optimal solution. It is vital to note that although no preference information is used, these methods do not make any attempt to find multiple Pareto Optimal Solution (POS).

Posteriori methods utilize preference information of each objective function and iteratively process a set of Pareto Optimal Solution (POS). The classical method of generating POS requires some knowledge on algorithmic parameters which ensure us in a finding a POS. This method is expensive and computationally demanding. It is challenging to represent POS if the objective functions are two or more. Some of the techniques include the weighted sum method, the Ɛ-constraint method, and the hybrid method.

Priori methods use more preference information about the objective function and also finds one preferred POS. The expected solution may be too optimistic or pessimistic. It is hard to express a preference without knowing the problem well. One of the most common methods in this class is goal programming.

Interactive methods use the preference information progressively or iteratively throughout the optimization process. A minimum knowledge is needed in advance. The main aspect of this approach is that during the optimization process, the user is required to provide some information about the direction of search, weight vectors, reference points, and other factors. Since the information is collected iteratively, these techniques are becoming popular in practice. There are many types of interactive methods; we use the Tchebycheff method and Reference point method in this thesis, which will be discussed in detail later.

2.1.1 The weighted sum

The weighted sum method [4] converts the MOP to SOP (Single Optimization Problem) by forming a linear aggregation of the objectives as follows:

$$Minimize \quad F(x) = \sum_{m=1}^{M} w_m f_m(x), \tag{2.1}$$

$$subject\ to \quad g_j(x) \geq 0, \quad j = 1,2,\ldots,J;$$

$$h_k(x) = 0, \quad k = 1,2,\ldots,K;$$

where w_m ($\in [0,1]$) is the weight of the *m-th* objective function. Solving 2.1 with varying weighted-coefficient sets provides a set of Pareto Optimal Solutions (POS). The weight of an objective function is usually chosen in proportion to the objective's relative significance in the problem considered. The major strength of this method is its efficiency and its simplicity, whereas the main disadvantage of this method is its difficulty in determining the significant weights for the corresponding problem.

2.1.2 The ε-constraints method

In 1971, Haimes [41] reformulated the MOP by just keeping one of the objectives and restricting the other objectives within the user-specified values. The problem is as follows:

$$Minimize \quad f_\mu(x), \tag{2.2}$$

$$subject\ to \quad f_m(x) \leq \varepsilon_m, m = 1,2,\dots,M\ and\ m \neq \mu$$

$$g_j(x) \geq 0, \quad j = 1,2,\dots,J;$$

$$h_k(x) = 0, \quad k = 1,2,\dots,K;$$

where, ε_m represents an upper bound of the value of f_m also, need not necessarily mean a small value close to zero. This method is done by optimizing an individually selected Subjective function (f_μ) while keeping the remaining (M-1) objectives values less than or equal to some user-specified thresholds (ε_m). Different POS values can be obtained for different threshold values. The solution for 2.2 mostly depends on the chosen ε vector. The chosen value should lie within the maximum and minimum values of the individual objective function.

2.1.3 The goal programming method

The primary idea in goal programming [7] is to find solutions which achieve a predefined target (goal) for one or more objective functions. Let *f(x)* be the objective function, x be the solution vector. In goal programming, a target value G is selected for every objective function by the user, and the task is to find a solution of the objective, which is equal to G. The problem is formulated as follows:

$$Minimize\ f(x) = \sum_{i=1}^{M} w_i |f_i(x) - G_i| \tag{2.3}$$

where w_i represents the weighting coefficient of the i^{th} objective such that $\sum_{i=1}^{M} w_i = 1 \; and \; w_i \geq 0, \forall i = \{1, \ldots, M\}$. The implementation of this method is simple, but its major drawback is its sensitivity to the weighting coefficients and the target value defined by the user.

2.2 Evolutionary methods

As mentioned earlier, Evolutionary Algorithms (EA) are population-based methods that are inspired by biological evolution. Finding and maintaining multiple solutions in one single simulation is a unique nature of evolutionary optimization methods. In this thesis, we are dealing with the presence of dynamism, the design for dynamic multi-objective optimization problem (DMOP) is different from that of multi-objective optimization (MOP) for static problems. The algorithm should not only have a fast convergence performance but also be able to address diversity loss when there is an environmental change in order to explore the new search space. In literature, many approaches have been proposed to handle the environmental changes, and they can be categorized into three approaches as follows:

1. Convergence-based approaches
2. Diversity-based approaches
3. Prediction-based approaches

2.2.1 Convergence-based methods

The main goal of these approaches is to achieve a fast convergence performance so that the tracking ability of the algorithm is guaranteed. For better tracking ability, these approaches make use of the past information, mainly when the new Pareto Optimal Solution (POS) is similar to the past POS or the environment change exhibits some regular patterns [6]. Making a note of relevant past information might help track the new POS as soon as possible [8]. The reuse of past information is closely related to the type of environmental change involved and hence, can be helpful for various purposes. These are also known as memory-based methods because past information is used and helps in evolving population when needed. In 2010 Wang and Li [32] proposed new DMOP test problems and also a new multi-strategy ensemble Multi-objective Evolutionary Algorithm (MS-MOEA) where the convergence speed is accelerated using a new offspring generation mechanism based on adaptive genetic and differential operators. The algorithm also uses a Gaussian mutation operator and a memory-like strategy to reinitialize the population when change occurs. Several memory-based dynamic environment techniques have been introduced in [33]. The major drawback of this approach is that memory is very dependent on diversity, and hence, it should be used with the combination of diversity-based techniques.

2.2.2 Diversity-based methods

It mainly focuses on maintaining the population diversity. Generally, the diversity of the population can be handled by increasing diversity using mutation of selected old solutions or some random generation of new solutions upon detection of environmental change or deploying multi-population methods [13], [16]. Good diversity helps obtain promising search regions. In [53], Deb presented an extended version of the Nondominated Sorting Genetic Algorithm (NSGA-II) [54] by introducing diversity at each environmental change detection. There were two approaches discussed in the paper; the first version introduced the diversity by replacing the population with new randomly created solutions. In the second version, diversity is promised by replacing the population with mutated solutions. In 2015 Azzouz [2], proposed a different version of the above algorithm to deal with dynamic constraints by replacing the constraint-handling mechanism with a much elaborated and self-adaptive penalty function. The major drawback of diversity-based methods is the difficulty to determine the useful amount of diversity needed. Because when the diversity high, it will resemble restarting the optimization process, whereas less diversity leads to slow convergence.

2.2.3 Prediction-based approaches

When the behavior of the dynamic problem follows a regular pattern, a prediction model is usually used to exploit the past information and anticipate

the location of the new optimal solutions. In [34], the authors proposed a new technique known as Feed-forward Prediction Strategy (FPS) to estimate the location of the optimal solution in DMOPs. In this method, a prediction set is placed in the neighborhood to accelerate the discovery of the next optimum. This set is formed by selecting two points as vertices and tracking and predicting them as the next step optimum. In FPS, only two points of the optimal solutions are predicted. In [20], the authors proposed to predict the number of Pareto optimal solutions in the decision space once changes are detected. Then, the individuals in the reinitialized population are generated around these predicted points. Later in [35], the authors proposed a new prediction strategy known as the dynamic predictive gradient strategy to predict the direction and magnitude of the changes in location of the Pareto optimal solutions.

More recently, in [36], the authors proposed a prediction model to predict the whole population rather than some isolated points. This approach is known as the Population Prediction Strategy (PPS) consists of dividing the optimal solutions into two parts: a center point and manifold. When a change is detected, the next center point is predicted using a sequence of center points maintained throughout the search progress, and the previous manifold is used to predict the next manifold. Then, the new population consists of the predicted center point and manifold. In 2018, [6] proposed a differential prediction model which is incorporated into MOEA based on decomposition (MOEA/D-DP) to solve DMOPs. The differential prediction model is used to forecast the shift vector in the decision space of the centroid in the population. This method uses only three

historical locations of the centroid. After the detection of environmental change, half of the population is forecasted their new location in the decision space by using the DP model and the others retain their old position. In [16], the authors propose a new approach to predict the POS in DMOPs called dynamic MOEA based on Kalman Filter (KF). KF is a set of mathematical equations which provides a well-ordered computational means to predict the state of a process, and also it minimizes the mean of the squared error. The efficiency of all the mentioned models relies on the accuracy of the predicted POS locations. If the actual locations and predicted locations are far in the decision space, then the prediction model will not be valued.

2.2.4 Other Evolutionary Methods

In [37], presented an artificial life inspired EA for DMOP in the case of unpredictable parameter changes. Contrast to the classical EAs such as Genetic Algorithms (GA) where Darwinian theory is considered as a type of intelligence, the proposed method that life and interactions among the individuals in the population in a changing environment are itself a type of intelligence to be exploited. The major drawback of this method is the slow convergence speed because the algorithm progresses individual by individual. In [17], a new algorithm was proposed by the authors known as the Steady-state and Generational Evolutionary Algorithms (SGEA), which is the combination of fast and standard tracking ability of steady-state algorithms and proper diversity maintenance of generational algorithms. When an environmental change is

detected, the proposed algorithm responds to the change in a steady-state manner. If there is environmental change detection, it reuses a portion of outdated solutions with good distribution and relocates many solutions close to the new Pareto Front (PF). The relocation is based on the information collected from the previous environments and new environment. Thus adaptability of this algorithm is expected to bring good tracking ability. In [38], the authors proposed multiple reference point-based MOEA (MRP-MOEA) that deals with dynamic problems with undetectable change. This algorithm does not detect changes. It uses the new reference point-based dominance relation, ensuring the guidance of the search towards the optimal PF.

2.2.5 Culturally evolved methods

The first CA to solve MOP was developed by Caello and Becerra [39]. The proposed algorithm was known Cultural Algorithm with Evolutionary Programming (CAEP). The belief space of MOP is constructed using the normative component and a grid. The number of non-dominated solutions for each cell is recorded in the grid. This information is utilized so that the non-dominated solutions are distributed uniformly along the Pareto front. The normative component is updated at regular intervals, whereas the grid is updated every generation. Updating the grid means simply recalculating the number of non-dominated solutions each cell. Selection of the new population in the

population space is adapted to make use of the grid information. Tournament selection is used and applied to the parents and offspring.

On the other hand, Saleem and Reynolds [42] presented us that CAs naturally contain self-adaptive components. The belief space in CA is dynamic, which makes CA suitable for tracking optima in dynamically changing environments. The belief space stores information from previous and current environmental states. Environmental history is stored in a table that consists of the following information about each environment: the location of the best solution, the fitness value of that solution, and the change magnitude in each dimension. This information is used by the dynamic influence function to introduce diversity in the population, proportional to the magnitude of change. In [44], the authors proposed a method to enhance the migration efficiency in Multi-Population Cultural Algorithm (MPCA). A novel MPCA adopting knowledge migration was proposed. Knowledge extracted from the evolution process of each sub-population directly reflects the information about dominant search space. By migrating knowledge among the sub-population at regular intervals, the algorithm realizes effective communication with low cost.

In [43], the authors proposed two new dynamic dimension approaches to improve the efficiency of the Heterogeneous Multi-Population Cultural Algorithm (HMPCA). The two approaches are Top-Down Strategy and Bottum-Up Strategy.

The first one starts with the local CA designed to optimize all the dimensions of the problem and recursively split the dimensions between two newly generated local CA. The second one starts with the idea of merging the dimensions of two local CAs when they reach to the no improvement threshold. This approach begins with the number of local CAs, and each CA is designed to optimize only one dimension. The number of initially generated local CAs is equal to the number of problem dimensions. Recently [19], provided us with knowledge migration strategies in MOP. It provides a variety of migration strategies which are inspired by the game theory model. This strategy was incorporated to increase diversity and avoid premature convergence. It also provides us with a significant migration to the population in the environment. Migration can depend on the individual choice; the decision of best individuals in the subpopulation or also by negotiating among the population. Game theory strategies were integrated with the MPCA. The proposed approach has two belief space, namely local and global belief space.

Recently [55], the authors proposed a method to tackle the distance between the parents to produce offspring in MOP, because it is not easy to produce an offspring in high-dimensional objective space. They proposed an elite gene-guided (EGG) reproduction operator. This was designed by three models: disturbance (Dr), exchange (Er) and inheritance to generate offsprings. To tackle MOPs, a small value of Dr and a large value of Er showed overall better performance.

Chapter 3

Evolutionary Computation

Optimization is a process which is used to minimize or maximize an objective function until an optimum or a satisfactory solution is found [3]. There exist many optimization problems where the computational time required to find the optimal solution is exponentially high. Evolutionary Computation contains a set of evolutionary algorithms (EA) that can find optimal or near-optimal solutions in polynomial time [14]. There are several Evolutionary Computation algorithms such as:

1. Evolutionary algorithm
2. Genetic algorithm
3. Cultural algorithm

3.1 Evolutionary Algorithm

Evolutionary algorithms are metaheuristic optimization algorithms which use mechanisms inspired by Darwin's theory of biological evolution[31]. Evolutionary algorithms (EAs) are a subset of those methods which has been successfully used in the past for optimization problems. They are population-based algorithms using the concepts of mutation, crossover, natural selection, and survival of the fittest, to refine a set of candidate solutions iteratively in a cycle [46]. In EAs the population is randomly initialized over specific search space which is called the initial population. Then it incorporates evolutionary operators which include mutation and crossover. This operator creates new offsprings (children) from the parent in the population. The selection operator selects the people with higher fitness from the parent and offspring, which serves as the population for the next generation. The leftover individuals are discarded from the people. This process continues, until the termination criteria are fulfilled, which can be either reaching a maximum number of predefined generations or CPU time. EA is based on the simplified model of biological evolution [47]. While solving a problem, a particular environment can be created where potential solutions can evolve. Parameters of the problem shape up the atmosphere, which helps to develop the right answer. EAs are a group of a probabilistic algorithm which is similar to the biological systems and artificial systems. Optimization using evolutionary algorithms also involves understanding the concepts of phenotypes, genotypes, objective function, fitness function, and search operations. The following definitions are stated below [46].

Definition 1. (Phenome)

The set of all the elements x that can be the solution of the optimization problem is known as the problem space or the phenome X.

Definition 2. (Phenotype)

The elements $x \in X$ of the phenome are known as the phenotypes.

Although we need to find the optimal phenotypes, the phenotypes are represented in mathematical terms so that it is possible to compute their score and execute different search operations. This representation of phenomes is known as genomes.

Definition 3. (Genome)

The set of all elements g which can be processed by the search operations in an optimization problem is known as the search space or the genome G.

Definition 4. (Genotype)

The elements $g \in G$ of the genome are known as genotypes.

A genotype may consist of many parameters, where each parameter may represent a specific property of the genotype. These parameters are known as genes. Genes can be binary, where its values can be either 0 or 1, or real coded, where its value is an actual number. The cost of a gene is known as an allele.

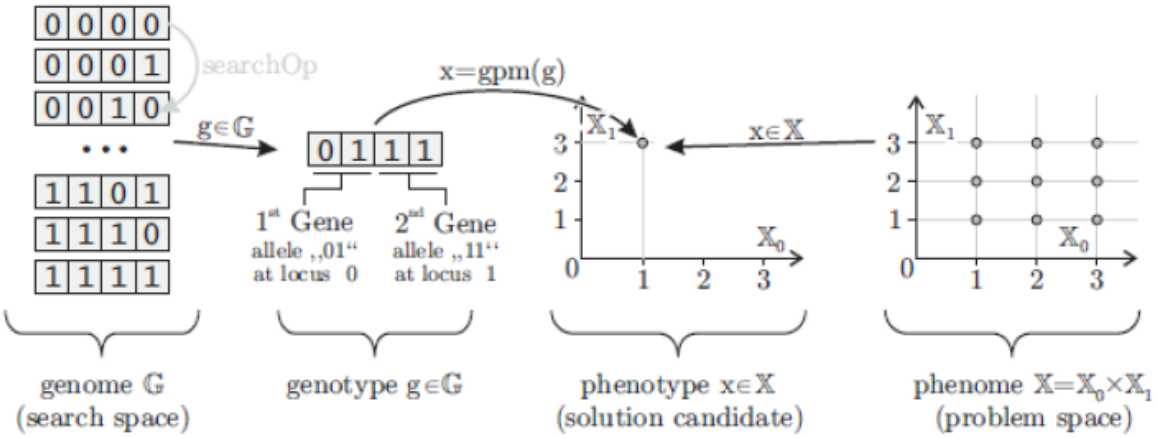

Figure 3.1: Relation between Genomes and Phenomes [46]

The phenomes (problem space) contains a set of a point on the Cartesian plane from which the optimum position is to find for a particular optimization problem. This problem space represents through genomes (search space), which is computationally easier to optimize. Each genotype present in the genome has binary genes. Once the optimal genotype is found, it is mapped into the corresponding optimal phenotype using a genotype-phenotype mapping (GPM) function.

3.2 Genetic Algorithms

One of the most standard evolutionary algorithms is Genetic Algorithms (GA). Genetic Algorithms, first proposed by John Holland [24] and popularized by the works of Goldberg [48], can find the right solutions to problems that were otherwise computationally intractable. They are heuristic search techniques that

start with a random population and, based on the fitness evaluation, selects individuals that will produce the successor population. This process iterates until a stopping criterion reached. GA helps in searching for solutions, even when the domain knowledge is minimum [47].

3.2.1 Selection

Selection is one of the main operators in EAs, and it directly relates to the Darwin theory of survival of the fittest. Selection is applied to the population for two reasons: (1) Selection of the new population – At the end of each generation a new population of candidate solutions is selected to serve as the population of next generation. The new population can be from the offspring or the combination of both parent and offspring. (2) Offsprings are produced from the application of crossover and mutation operators. In terms of *crossover,* 'superior' individuals will have more opportunities to reproduce to ensure that the offspring have the genetic material of the best individuals. On the other other hand *mutation,* selection mechanism focuses on 'weak' individuals. The hope is that the mutation of weak solutions will result in better traits to weak individuals, which increases their chances of survival [14]. They select the best individuals in the current generation based on their fitness. The individuals who are fitter are chosen, and the weaker are discarded from the production. The fitter individuals have a high chance of passing knowledge from the current generation to the next

generation. Many selection operators have been developed. Let us discuss some essential operators in detail.

Random Selection is the most straightforward selection operator. Each individual has the same probability to be selected: $1/n_s$, where n_s is the population size. Fitness information is not needed, which makes that the best and worst individuals have the same probability of selection for the next generation.

Proportional Selection was proposed by Holland [24]; the selection is based on the most-fit individuals. A probability distribution proportional to the fitness is created, and the individuals are selected through sampling the distribution [14].

$$\varphi_s(\mathbf{x}_i(t)) = \frac{f_\Upsilon(\mathbf{x}_i(t))}{\sum_{l=1}^{n_s} f_\Upsilon(\mathbf{x}_l(t))} \tag{3.1}$$

where n_s is the population size, and $\varphi_s(x_i)$ is the probability that x_i will be selected; $f_\gamma(x_i)$ is the scaled fitness of x_i, it produces a positive floating-point value. There are two popular sampling methods used in proportional selection: roulette wheel sampling and stochastic universal sampling. *Roulette wheel sampling* is an example of a proportional selection operator where fitness values are normalized. Then the probability distribution can be visualized as the roulette wheel, where the size of each slice is directly proportional to the normalized selection probability of an individual. Selection can be similar to the rotation of a

roulette wheel and recording which slice ends up at the top, and then the corresponding individual is selected. Since the selection is directly proportional to the fitness, a strong individual may dominate in producing offspring, and this limits the diversity of the new population.

Tournament Selection selects a group of individuals n_{ts} randomly from the population where $n_{ts} < n_s$ (n_{ts} is the size of tournament selection population). The performance of the selected n_{ts} individuals are compared, and the best individual is selected from the group. For crossover with two parents, the selection is carried out twice, one for each parent. When the tournament size is not too large, tournament selection prevents the best individual from dominating. Whereas if the tournament size is too small, there are chances that corrupt individuals are selected.

3.2.2 Crossover Operation

In a crossover operation, specific genes of one individual are exchanged with the genes present in the same position as the other individual to produce two new individuals. A segment of genes is swapped between the parents to create their offspring and not single genes. The simplest of all is the *single point crossover* where a random crossover point is selected, and the bitstrings after that point are swapped between the two parents. In the *multi-point crossover*,

two or more crossover points are selected randomly, and every alternate bitstring sequence is swapped. In the *uniform crossover* [50], there exists a probability distribution for each gene. This distribution indicates the probability with which a gene should be exchanged. Here p_x is the bit-swapping probability. If $p_x = 0.5$ then each bitstring as an equal chance to be swapped.

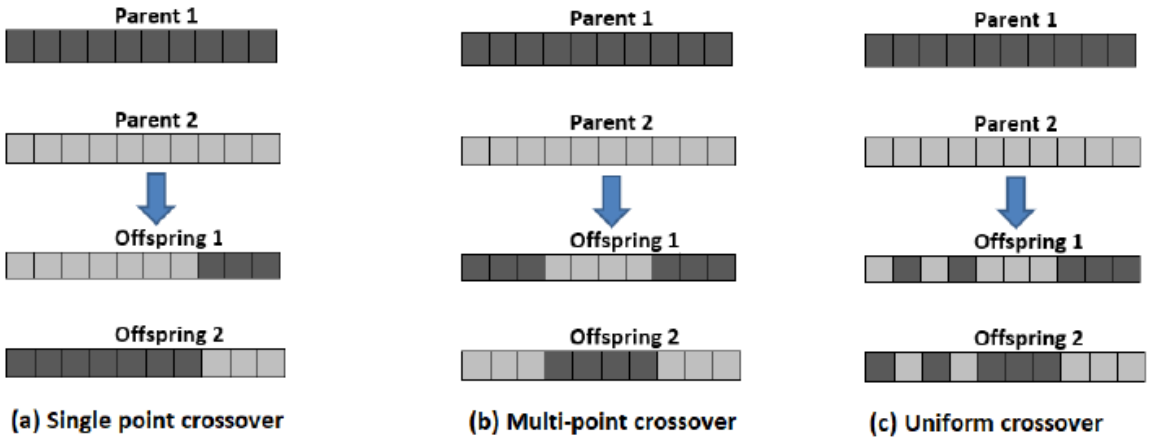

Figure 3.2: Types of crossover operations [14]

3.2.3 Mutation Operation

The main goal of mutation is to introduce genetic material into the existing individual; this adds diversity to the genetic characteristics of the population. The mutation is applied at a specific probability p_m, to each gene of the offspring, which produces the mutated offspring. It is also known as the mutation rate, which is generally a small value, $p_m \in [0,1]$, this is to ensure some good solutions are not biased too much. Some of the mutation operators are developed [24].

Uniform (Random) mutation, where the bits are chosen randomly, and corresponding bits are negated. *Inorder mutation*, two points are selected randomly, and only the bits between these points undergo random mutation. The *Gaussian mutation* was proposed mainly for binary representation of the floating-point value. The bitstring, which represents a decision variable, can be converted back to floating-point value and mutated with Gaussian noise. Poisson distribution is used to draw chromosomes randomly to determine to mutate the genes. The bitstring of these genes is converted. To each of the floating-point value, the step size is added $N\ (0, \sigma_j)$, where σ_j is 0.1 of the range of that decision variable. Gaussian mutation showed superior results in bit flipping.

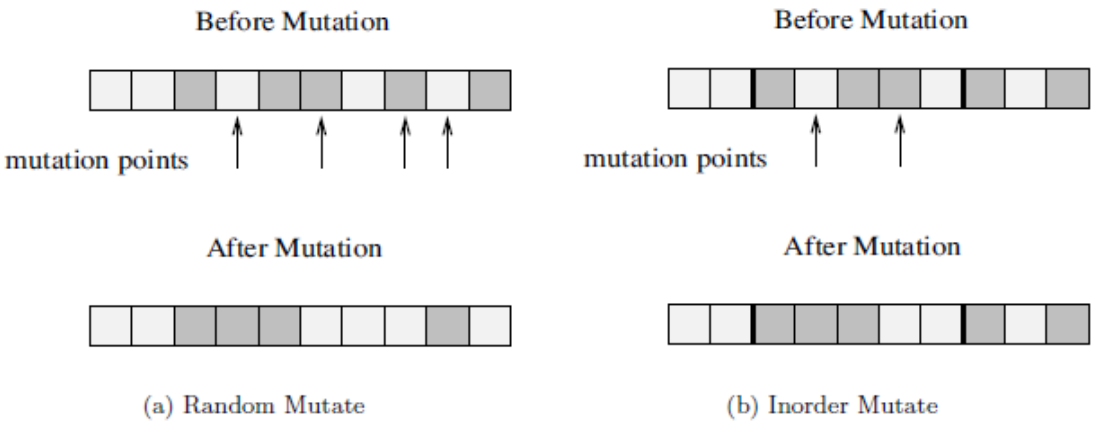

Figure 3.3: Types of Mutation [14]

3.3 Cultural Algorithms

The search process in the standard EAs is unbiased; it uses only a little or no domain knowledge to direct the search process [50]. The performance of the EAs can be improved considerably by using domain knowledge; it makes the search process biased. In 1994 Reynolds [12], proposed Cultural Algorithm (CA). CA is one of the popular types of EA which incorporates knowledge to guide the search process. A vast number of successful applications of CA exhibits the performance of knowledge-based EA. The search mechanism is improved by amending the extracted knowledge into a CA. Therefore it leads CA to find better solutions with high quality and also improves the convergence rate. In [14] Engelbrecht defines culture as *"Culture is the sum total of the learned behavior of a group of people that are generally considered to be the tradition of that people and is transmitted from generation to generation."*

Fig. 3.4 illustrates the underlying architecture of CA. As depicted in the figure, CA maintains two search spaces: *the population space* like all the other EAs is represented by the individuals. Each individual will have a set of features independent from each other, which is used to determine its fitness. This space will be managed by an EAs such as GA or DE. CA has one more space known as the *belief space*. The belief space stores and updates all the extracted knowledge over generations. At each generation, these two spaces communicate with each other using a communication protocol. The protocol defines two communication

channels. One is the *acceptance function* which selects a group of individuals to adapt the set of beliefs and the second one is *influence function*, which defines a way that all the individuals in the population are influenced by the beliefs. The knowledge circulation is carried out as follows:

i. The belief space will receive the top best individuals from the generation *g* in the population space using acceptance function.
ii. The belief space knowledge is updated
iii. In the following generation *g+1*, the knowledge updated in the belief space is sent through the influence function to the population space.
iv. The population space integrates the knowledge to generate offspring from generation *g* and creates the next generation *g+1*.
v. Now, the best individuals of g+1 are sent to the belief space and update its knowledge.

This routine continues until the algorithm ends. It seems like the population space of a CA works like any other EA, but it uses knowledge-based evolutionary operators than random ones.

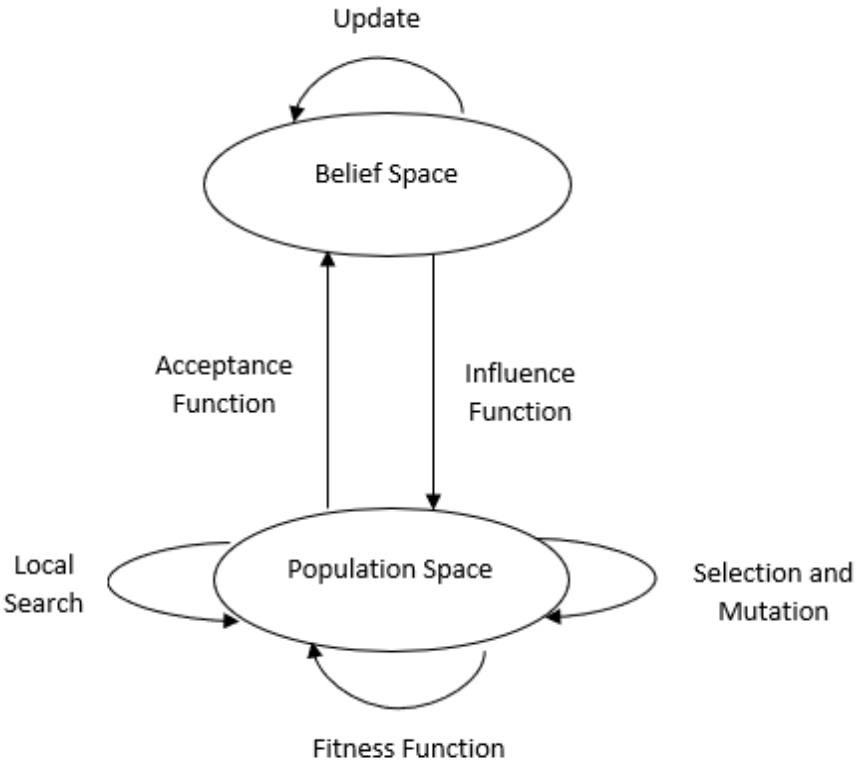

Figure 3.4: Architecture of CA [12]

3.3.1 Belief Space

The belief space is the central component where knowledge or beliefs of the individuals in the population space is stored. This knowledge searches biased towards a particular direction, resulting in a significant reduction of the search space. The belief space is updated after each iteration by the fittest individuals. The belief space has been classified into five basic categories [12]:

Situation knowledge component tracks the best solution found at every generation.

Normative knowledge component provides specific standards for individual behavior, which are used as guidelines for mutational adaptation to individuals. It also maintains a set of intervals, one for each dimension of the problem solved.

Domain knowledge component, it differs from the situational knowledge in that knowledge is not re-initialized at every generation but archives all the best solutions since the evolution began.

History knowledge component, it maintains a sequence of information about the environmental changes. It is mostly used in problems where search landscapes may change.

Topographical knowledge component, the search space is represented as a multidimensional grid. Information such as the frequency of the individual that occupies the cell is stored.

3.3.2 Population Space

Population component is the space which consists of the individual in the population. The population space of CA is similar to that of GA. There are two functions which allow the individual to communicate between population space

to belief space and vice versa. The acceptance function transfers the best individual of the population space into belief space. After that, the belief space updates its knowledge and updates the population space by making use of influence function. The individuals in the population space make use of this knowledge to generate individuals for the next generation [51].

Chapter 4

Proposed Approach

In this section, we have introduced the pseudo-code and framework of our proposed algorithm. We will discuss the design, belief space, and population space of the algorithm.

4.1 Cultural Algorithm to solve DMOPs

We propose the use of Cultural Algorithms (CA) combined with decomposition strategies to solve DMOPs. We have incorporated two different decomposition strategies with the CA. Implementing these strategies can improve the performance in DMOP through accelerating the convergence. Now we will discuss the Cultural Algorithms to solve DMOPs. The problems we focused on solving have n decision variables and k objective functions. The population space consists of a set of individuals, which contains n decision variables of the problem

to be solved. The population is initialized randomly, which consists of p individuals. There is an external memory introduced into the algorithm which collects all the non-dominated individuals found along the process. The final content of this file is the set of solutions produced by the algorithm. The size of the external memory is q, which is the number of non-dominated we aim to obtain. Next, we will discuss the structure of belief space and the remaining steps of the algorithm.

4.1.1 Structure of Belief Space

The belief space consists of three parts: Situational Knowledge, Normative Knowledge, and Environmental History to adjust the belief space and influence the population. Mathematically, belief space can be represented as:

$$B(t) = [N(t), S(t), E(t)] \tag{4.1}$$

where $B(t)$ represents the belief space at generation t $N(t)$, $S(t)$ and $E(t)$ represent the Normative, Situational, and Environmental History knowledge respectively. Each of these components is updated simultaneously and influence each individual of the next generation.

Situational Component: Let $x_{best}(t)$ represent the individual having the best fitness value at generation t. The situational component is updated as follows:

$$S(t+1) = \begin{cases} x_{best}(t) & if\ x_{best}(t) > S(t) \\ S(t) & otherwise \end{cases} \quad (4.2)$$

The property of storing the individual is known as elitism. Elitism guarantees that the EA will converge.

Normative Component: It consists of two parts, namely - the phenotypic normative part and a grid which is used to prioritize the generations of non-dominated solutions that are uniformly distributed along the Pareto front.

The phenotypic normative part consists of the lower and upper bounds, $l_{fi} and\ u_{fi}$, for each objective function (i = *1,..., k*) within which the grid will be built. The grid is used to place each non-dominated solution in some coordinate system where the values of the objective function are used to place each solution. Once we have the intervals, we need to know the number of identical sub-intervals to apply to each of the objective function s_i with i = *1,...,k* so now the grid can be built in the objective space. For each cell, the number of non-dominated solutions within that cell is recorded.

l_{f1}	u_{f1}	l_{f2}	u_{f2}	…	l_{fk}	u_{fk}

Figure 4.1: Phenotypic normative part

In order to *initialize the belief space*, we need to have an initial population, because we will use the non-dominated individuals from the population. It has been proven that any population with a size greater than zero; there will be at least one non-dominated individual [52]. The initialization of the phenotypic part of the belief space includes finding the extreme of each objective function for the non-dominated solutions of the initial population. These extremes are stored in $l_{fi} and\ u_{fi}$, so that the grid is constructed in the location of non-dominated solutions. The initialization of the grid part of the belief space involves setting/initializing the number of non-dominated solutions within each cell is to 0.

In order to *update the belief space,* the grid is updated at each generation, whereas the phenotypic normative part is updated at regular intervals, $g_{normative}$, where $g_{normative}(20)$ is a parameter defined by the user. The grid is updated by the increase in the number of non-dominated solutions by the number of individuals added to the external memory in the current generation. The update of the grid is very simple, and that is why we update it every generation. The acceptance function is used to update this part of the belief space; it uses the population of the external memory and only chooses the new individuals within the population. The update of phenotypic part is not done at every generation because it involves

the reconstruction of the grid, and which will affect the computational efficiency of the algorithm. The population of the external memory is used to implement this update.

Environmental History Component: Since we are dealing with dynamic problems, the belief will include the environmental history component. This is a data table for each environment, and it consists of the information such as the location of the best solution, the fitness value of the solution, the change in magnitude in each dimension, and the following change in the fitness value.

4.1.2 Influence Functions

Once the belief is updated, it is used to influence the population for the next generation. To allow CA to adjust rapidly to environmental changes, we use an influence function, which introduces diversity to the population by mutating the population proportional to the magnitude of the change. The step size is calculated as follows [42]:

$$\Delta x_{ij}(t) = \frac{2|f(\mathbf{x}_i(t)) - \tilde{f}(\mathbf{x}_i(t))|}{\hat{f}(t)} U_{ij}(0,1) \tag{4.3}$$

where f represents landscape before the change $\tilde{f}$ represents the changes in landscape, $\hat{f}(t)$ is the best fitness value stored in the history table. This indicates a large step size for large environment changes; thereby increasing the diversity.

4.1.3 Mutation Operator

The information stored in the belief space belongs to the objective space of our problem. The mutation parameters are given as the input to the algorithm, which is provided by the user. The Gaussian mutation operator adopted in our algorithm is as follows [39]:

$$x_i' = [x_i + N(0, \sigma)] * F \qquad (4.4)$$

where: x_i is the i-th variable of the individual x, x_i' is the i-th variable of new individual x' obtained after applying the mutation operator and F is the scalar factor. $N(\mu, \sigma)$ is a normal distribution of the random variable that has mean μ and a standard deviation σ. In our case, μ will always be zero, and σ be the parameter provided by the user. Mutation is applied to $i = 1, \ldots, n$, and operates on the main population. At the end of this process, the population size will $2p$.

4.1.4 Selection Operator

Tournament selection is carried out using the main population of size *2p*. Each individual is confronted against *c* individuals who are randomly chosen from the main population. The rules for tournament selection are as follows:

- If an individual dominates its competitor (contender *c*), then the dominating individual wins.
- If none of the competitors are non-comparable or if their objective values of the function are same, then:
  - If one of the individuals lie outside the grid, then that individual is selected.
  - If both lie within the cell, then the individual which lies in the less populated cell is selected.
- If none of the above cases satisfy, then the fittest individual is selected.

The first rule is straightforward; we are just giving preference to the non-dominated individuals. In the second rule, the influence of the belief space in decisions is appreciated during the tournament. Once the tournaments are done, we select the individual with maximum victories to be part of the next generation. The decisions taken in the tournament selection will be influenced by the information stored in the belief space.

4.1.5 Environmental Change Detection

When the problem changes, 10% of the population are chosen randomly for re-evaluation to detect environmental changes [20]. This is carried out by computing the average objective function values and comparing them with the previous and current generations. If they differ, then an environmental change has occurred. Therefore, the system responds to the change by reconstructing the population for only randomly chosen 50% individuals [6].

4.2 Decomposition Strategy

There is a wide variety of methods for accomplishing MOP, but none can be said to be superior to all the others. When selecting a method, the specific features of the problem to be solved should be taken into consideration. Consequently, the input from the decision-maker is essential. Therefore, Hwang (1979), classified the different methods according to the participation of the decision-maker which has been discussed in Chapter 2.4

4.2.1 Tchebycheff Method

This method was proposed by Steur (1989), it is one of the types of interactive methods. The design of this method is to be user-friendly for the

decision-maker. To find a set of Pareto optimal solutions or non-dominated solutions, preference information from the decision-maker should be obtained iteratively. The preference information is distinguished by two user-specified inputs. One is known as the utopian objective vector or ideal solution (z^*) to the DMOP. The second one is the weight vector (λ_i: $i = 1,2,\ldots,n$) which assigns the relative preferences to n objectives. The mathematical model is represented as follows [1]:

$$\begin{aligned} &\text{Minimize} && Max_{i\ldots,n}[\lambda_i|z_i^* - z_i|] \\ &\textit{subject to} && g_j(x) \leq 0 \; \forall j = 1,2,\ldots,m \\ & && \lambda_i \in R^n | \lambda_i \in \{0,1\} \\ & && \textstyle\sum_{i\in n} \lambda_i = 1 \\ & && x \in X \end{aligned} \tag{4.5}$$

where z_i is one of the n-objectives being maximized and z_i^* is the corresponding i-th utopian objective vector value or the ideal solution, λ_i is the i-th weight vector value. $g_j(x)$ is the j-th constraint of the original problem, m is the total number of constraints, R^n is the objective space. X is the decision space and x is the decision vector. For each Pareto optimal point x^* , there is a weight vector λ such that x^* is the optimal solution of (4.5) and each optimal solution of (4.5) is also a POS of (1.3). Therefore, we are able to obtain different Pareto optimal solutions by simply altering the value of the weight vector. Different solutions can be obtained with different weight vectors.

The problem considered for our approach is defined here as follows: let $\lambda^1, \ldots, \lambda^N$ be considered the set of evenly spread weight vectors, a MOP will be decomposed into *N* scalar optimization subproblems, and the *j-th* subproblem is as follows [6]:

$$minimize\ g\left(x\middle|\lambda^j, z^*\right) = \max_{1\le i\le m}\{\lambda_i^j\,|f_i(x) - z_i^*\}$$

$$subject\ to\ x \in X \tag{4.6}$$

where $\lambda^j = \lambda_1^j, \ldots, \lambda_m^j$ and objective vector is $z^* = (z_1^*, \ldots, z_m^*)^T$ for a minimization problem $z_i^* = min\{f_1(x)\},\ x \in X$, for each $i = 1, \ldots, m$. The proposed approach CA/D minimizes the N-scalar subproblems simultaneously. In CA/D a neighborhood λ^j is defined by a set of closest weight vector in $\lambda^1, \ldots, \lambda^N$. The *j-th* subproblem neighborhood consists of all the subproblems with the weight vectors from the neighborhood λ^j. The population of the subproblem consists of the best solution found so far for every subproblem. This is the modified version of MOEA/D-DP [6]. We explicitly use cultural algorithm (CA) which aims at providing better results by using the belief space component and also the history knowledge component which was mentioned earlier to keep track of the environmental changes occurred. Our algorithm for CA/D-TM works as follows:

Algorithm 1: CA/D-TM

Step I: Initialization

1. Set the generation counter, $T = 0$.
2. Initialize a population $x^1, \ldots, x^N$ and the population space P(0).
3. Generate and initialize the belief space *B(0).*
4. Select the neighborhood (subproblem) $S(i) = \{i_1, \ldots, i_H\}$, where H are close

to the weight vectors λ^j.

5. Compute the Euclidean distance between any two weight vectors and work out the R closest weight vectors for each weight vector. For each $i = 1, \ldots, N$, set $C(i) = \{i_1, \ldots, i_R\}$ where $\lambda^{j_1}, \ldots, \lambda^{j_R}$ are the R closest weight vectors to λ^j.
6. Initialize an objective vector $z^* = (z_1, \ldots, z_m)$.
7. Initialize the external memory q.

Step II: Environmental change detection

1. Re-examine 1/10 * N individuals which are randomly chosen from the population and calculate their average objective function value $\bar{F} = \{\bar{f}_1, \ldots, \bar{f}_m\}$ and have a comparison of these with the previous generation. If the values of objective function are different continue; otherwise go to Step III.
2. Responding to the change, reconstruct the population P for randomly chosen 50% individuals in the objective space.
3. Output the population P, objective functions F and increment $T = T+1$.
4. Re-examine the new population and update their objective vector z^*

Step III: Update

For $i = 1, \ldots N$, do

1. Reproduction – Select two indexes a, b from $C(i)$ and then output the new solution y from x^a and x^b by using the genetic operators. (Mutation and Crossover).
2. Evaluate the new solution y, if it is out of the boundary of the decision space and produce y'
3. Apply Tournament selection – Randomly choose c contenders and do tournaments.
4. Select p individuals with max victories to produce the population of next generation.
5. Update the population space $P(T)$, *if* $g(y|\lambda^p, z^*) < g(x^p|\lambda^p, z^*)$ *then* $x^p = y$.
6. Add the new non-dominated individuals to the external memory of size q
7. Update the belief space $B(T)$ using the individuals added to external memory.
8. Update the objective vector z^* for every $i = 1, \ldots, m$, $z_i > f_i(y')$ then set , $z_i = f_i(y')$.

Step IV: Stopping criterion

1. If the stopping conditions are met then stop and output the population otherwise go to step II.

4.2.3 Reference Point Method (RP)

This method was proposed by Wierzbicki (1981); it is also a part of interactive methods. The reference point is a feasible or infeasible point in the objective space which will be reasonable or desirable to the decision-maker. The reference point is based on aspiration levels. In this method, the pareto optimal solutions are based on the reference points, not on function values or weighting vectors. It should be noted that RPs are not used to present user preference or to guide the search process but to predefine the search directions covering the entire search spaces in order to accelerate the convergence speed [38]. The goal of the RP method is to derive achievement functions having minimal solutions at weakly, ε-properly or Pareto optimal solution closest to a given aspiration level based on solving a scalarizing problem. Given a reference point $\bar{z}$ for an M-objective optimization problem $f_1(x), \ldots, f_i(x)$ with $x \in S$, the following is the mathematical representation [40]:

$$Minimize \qquad Max_{i=1}^{M}[w_i(f_i(x) - \overline{z_i})] \tag{4.4}$$

$$subject\ to\ x \in S$$

where, w_i is the i-th component of a chosen weight vector, which is used for scalarizing the objectives. An appropriate form of achievement function must also be selected. For an RP, the closest Pareto solution is the target solution for this method. The location of the RP makes the algorithm to focus on a specific region in the Pareto front, where the use of a weight vector is to make a fine trade-off

among the objectives and focuses the algorithm to obtain a single POS and trading-off the other objectives. Thus, the RP provides higher-level information about the region to focus; and on the other hand, the weight vector provides more detailed information about what to converge on the Pareto front. Our algorithm for CA/D-RP is stated below:

Algorithm 1: CA/D-RP

Step I: Initialization

1. Set the generation counter, $T = 0$.
2. Initialize a population $x^1, \ldots, x^N$ and the population space P(0).
3. Generate and initialize the belief space $B(0)$.
4. Compute the Euclidean distance between any two weight vectors and work out the R closest weight vectors for each weight vector. For each $i = 1, \ldots, N$, set $C(i) = \{i_1, \ldots, i_R\}$ where $\lambda^{j_1}, \ldots, \lambda^{j_R}$ are the R closest weight vectors to λ^j.
5. Initialize a reference point $\bar{z} = (z_1, \ldots, z_m)$.
6. Initialize the external memory q.

Step II: Environmental change detection

1. Re-examine 1/10 * N individuals which are randomly chosen from the population and calculate their average objective function value $\bar{F} = \{\overline{f_1}, \ldots, \overline{f_m}\}$ and have a comparison of these with the previous generation. If the values of objective function are different continue, otherwise go to Step III.
2. Responding to the change, reconstruct the population P for randomly chosen 50% individuals in the objective space.
3. Output the population P, objective functions F and increment $T = T+1$.
4. Re-examine the new population and update their reference point $\bar{z}$.

Step III: Update

For $i = 1, \ldots N$, do

1. Reproduction – Select two indexes a, b from $C(i)$ and then output the new solution y from x^a and x^b by using the genetic operators. (Mutation and Crossover).
2. Evaluate the new solution y, if it is out of the boundary of the decision space

and produce y'

3. Apply Tournament selection – Randomly choose c contenders and do tournaments.
4. Select p individuals with max victories to produce the population of next generation.
5. Update the population space $P(T)$.
6. Add the new non-dominated individuals to the external memory of size q.
7. Update the belief space $B(T)$ using the individuals added to external memory.
8. Update the reference point $\bar{z}$ for every $j = 1, \ldots, m$, $z^j = \bar{z} + (z' - \bar{z}) \cdot e^j$ (j-th unit vector)

Step IV: Stopping criterion

1. If the stopping conditions are met then stop and output the population; otherwise go to step II.

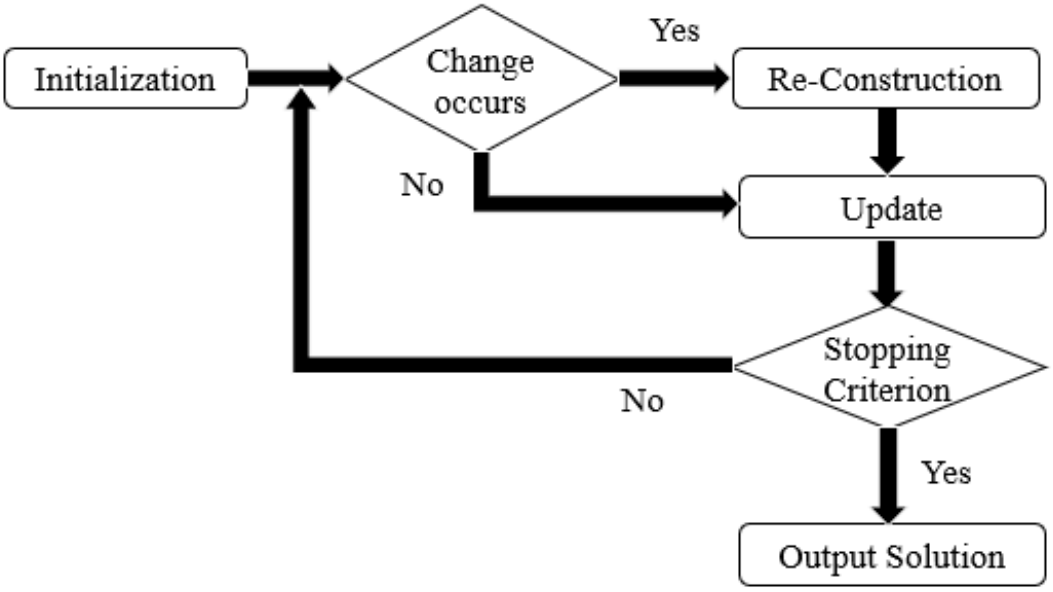

Figure 4.2: Flowchart of the Algorithm

Chapter 5

Benchmark Functions and Experiments

In this chapter, we will discuss the benchmark functions used in testing the proposed algorithm in comparison to existing algorithms, and describe the details of the experimental setup.

5.1 Benchmark Optimization Functions

Benchmark optimization problems are often used to evaluate the performance of any optimization algorithm. The are used to evaluate the characteristics of the algorithms such as convergence, robustness, precision, and general performance. For our proposed algorithm, we have used the CEC 2015 benchmark functions to evaluate and compare them with the existing algorithms [29]. These functions are briefed in the following section[1]. There are 15 minimization functions. Functions may be either convex or non-convex. The test

[1] These content are taken from https://al-roomi.org/multimedia/CEC_Database/CEC2015/RealParameterOptimization/ExpensiveOptimization/CEC2015_ExpensiveOptimization_TechnicalReport.pdf

functions are dimension-wise scalable. For our experiments, the different functions used are as follows:

1. Unimodal Functions
2. Multi-modal Functions
3. Hybrid Functions
4. Composite Functions

Table 5.1: Summary of the CEC 2015 Benchmark problems [29].

Categories	**No.**	**Functions**	**Related Basic Functions**
Unimodal Functions	F1	Rotated Bent Cigar Function	Bent Cigar Function
	F2	Rotated Discus Function	Discus Function
Simple Multi-modal Functions	F3	Shifted and Rotated Weierstrass Function	Weierstrass Function
	F4	Shifted and Rotated Schwefel's Function	Schwefel's Function
	F5	Shifted and Rotated Katsuura Function	Katsuura Function
	F6	Shifted and Rotated	HappyCat Function

		HappyCat Function	
	F7	Shifted and Rotated HGBat Function	HGBat Function
	F8	Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function	Griewank's Function Rosenbrock's Function
	F9	Shifted and Rotated Expanded Scaffer's F6 Function	Expanded Scaffer's F6 Function
Hybrid Functions	F10	Hybrid Function 1 (N=3)	Schwefel's Function Rastrigin's Function High Condition Elliptic Function
	F11	Hybrid Function 2 (N=4)	Griewank's Function Rosenbrock's Function Scaffer's F6 Function Weierstrass Function
	F12	Hybrid Function 3 (N=5)	Katsuura Function HappyCat Function Griewank's Function Rosenbrock's Function Schwefel's Function Ackley's Function
Composite Functions	F13	Composite Function 1 (N=5)	Rosenbrock's Function High Condition Elliptic Function Bent Cigar Function Discus Function
	F14	Composite Function 2 (N=3)	Schwefel's Function Rastrigin's Function High Condition Elliptic Function

	F15	Composite Function 3 (N=5)	HGBat Function Weierstrass Function Schwefel's Function Rastrigin's Function High Condition Elliptic Function

5.1.1 Unimodal Functions

The functions are the extension of the primary functions. They are shifted and rotated.

$$o_{i1} = [o_{i1}, o_{i2}, \ldots, o_{iD}]^T \quad (5.1)$$

is the shifted global optimum, which is randomly distributed in $[-80,80]^D$. All the test functions are scalable and shifted to o.

F1 (Rotated Bent Cigar Function): This is the extended version for bent cigar function. The properties of this function are non-separable, dimension-wise scalable, and unimodal.

$$f(x_1, \ldots, x_n) = f_1(M(x - o_1)) + 100 \quad (5.2)$$

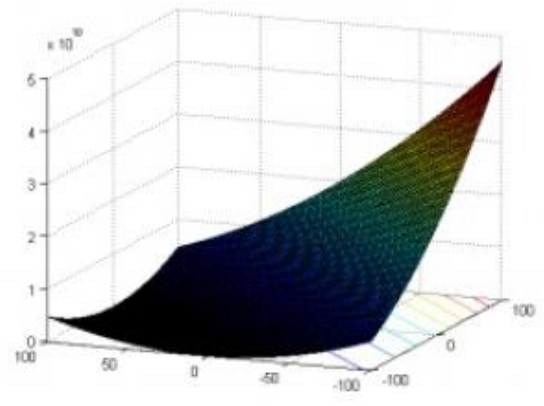

Figure 5.1: 3D-Map for Rotated Bent Cigar Function [29]

F2 (Rotated Discus Function): This function is the extended version of discus function. The properties of this function are non-seperable, unimodal, and dimension-wise scalable.

$$f(x_1, \ldots, x_n) = f_2(M(x - o_2)) + 200 \qquad (5.3)$$

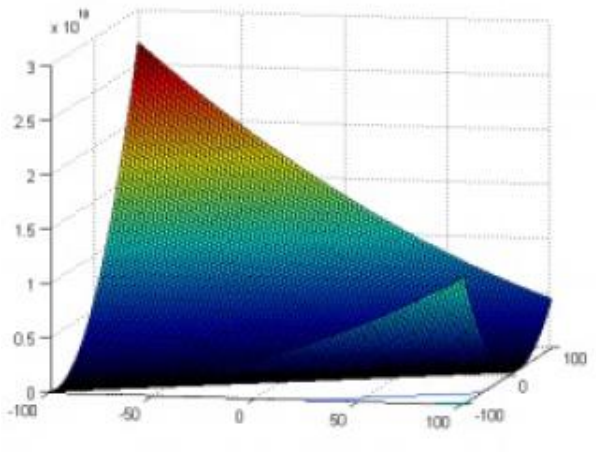

Figure 5.2: 3D – Map for Rotated Discus Function [29]

5.1.2 Simple Multi-Modal Functions

F3 (Shifted and Rotated Weierstrass Function): This is an extended version for Weierstrass function. The properties of this function are non-separable, dimension-wise scalable, and multi-modal.

$$f(x_1, \dots, x_n) = f_3(M(\frac{0.5\,(x - o_3)}{100})) + 300 \qquad (5.4)$$

F4 (Shifted and Rotated Schwefel's Function): This function is an extended version of Schwefel's function. The properties of the function are non-separable, multi-modal, and dimension-wise scalable.

$$f(x_1, \dots, x_n) = f_4(M(\frac{1000\,(x - o_4)}{100})) + 400 \qquad (5.5)$$

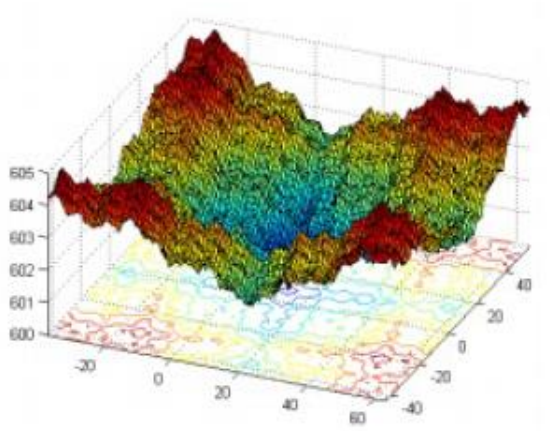

Figure 5.3: 3D – Map for Rotated and Shifted Schwefel's Function [29]

F5 (Rotated and Shifted Katsuura Function): This is an extended version of Katsuura function. The properties of the function are non-separable, multi-modal, and dimension-wise scalable.

$$f(x_1, \dots, x_n) = f_5(M(\frac{5\,(x - o_5)}{100})) + 500 \quad (5.6)$$

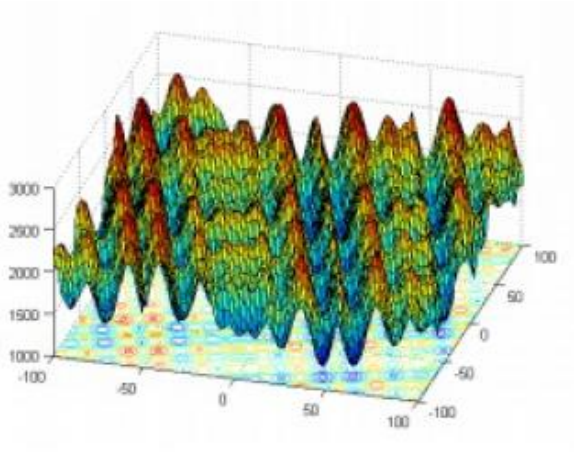

Figure 5.4: 3D -Map for Rotated and Shifted Katsuura Function [29]

F6 (Rotated and Shifted HappyCat Function): This is an extension of HappyCat function. The properties of this function are separable, dimension-wise scalable, and multi-modal.

$$f(x_1, \dots, x_n) = f_6(M(\frac{5\,(x - o_6)}{100})) + 600 \quad (5.7)$$

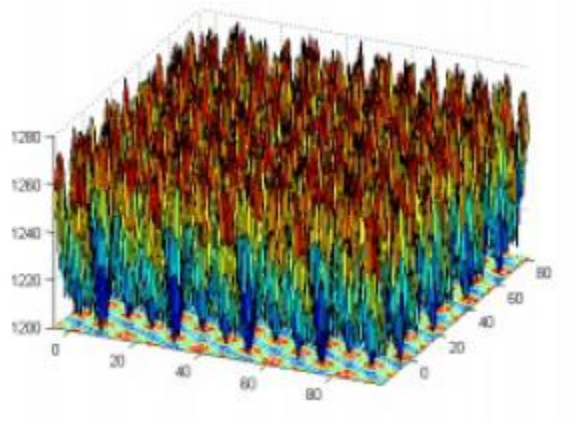

Figure 5.5: 3D – Map for Rotated and Shifted HappyCat Function [29]

F7 (Rotated and Shifted HGBat Function): This is an extended version of the HGBat function. The properties of the this function are multi-modal, non-separable and dimension-wise scalable.

$$f(x_1, \dots, x_n) = f_7(M(\frac{5\,(x - o_7)}{100})) + 700 \qquad (5.8)$$

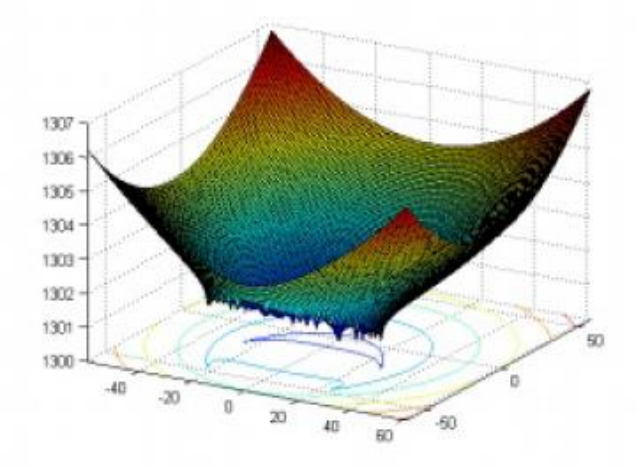

Figure 5.6: 3D – Map for Rotated and Shifted HGBat Function [29]

F8 (Shifted and Rotated Expanded Griewank's plus Rosenbrock's Function): This function is an extended and expanded version of two functions: Griewank's and Rosenbrock function. The properties of this function are non-separable, dimension-wise scalable, and multi-modal.

$$f(x_1, \dots, x_n) = f_8(M(\frac{5\,(x - o_8)}{100})) + 800 \qquad (5.9)$$

F9 (Shited and Rotated Expanded Scaffer's F6 Function): This function is an expanded and extended version of Scaffer's F6 function. The properties of this function are non-separable, dimension-wise scalable, and multi-modal.

$$f(x_1, \ldots, x_n) = f_9(M(x - o_9) + 1) + 900 \quad (5.10)$$

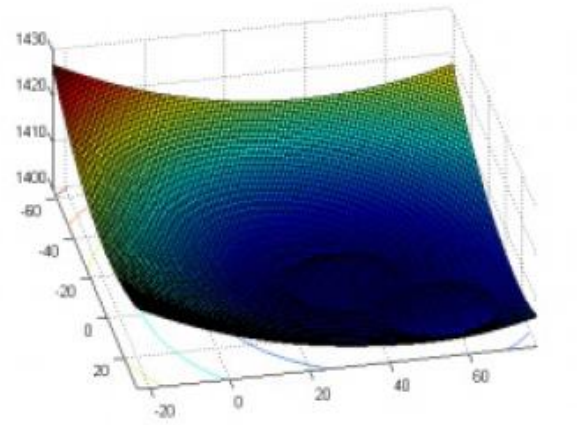

Figure 5.7: 3D – Map for Shifted and Rotated Scaffer's F6 Function [29]

5.1.3 Hybrid Functions

Hybrid functions resemble real-world optimization problems, comprising different set of variables possessing different properties. Similarly, in hybrid functions, all the variables are divided into some subsets, and each subset will have different basic functions operating on them.

$$F(x) = g_1(M_1 z_1) + g_2(M_2 z_2) + \ldots + g_N(M_N z_N) + f^*(x) \quad (5.11)$$

$F(x)$: Hybrid Function

$g_i(x)$: *i-th* basic function used to construct the hybrid function.

N: Number of basic functions

$$z = [z_1, z_2, \ldots, z_N],\ z_1 = \left[y_{s_1}, y_{s_2}, \ldots, y_{s_m}\right],\ z_2 = \left[y_{s_{m+1}}, y_{s_{m+2}}, \ldots, y_{s_{m+n2}}\right], \ldots,$$

$$z_N = [y_{s_{\sum_{i=1}^{N-1} n_i+1}}, y_{s_{\sum_{i=1}^{N-1} n_i+2}}, \ldots, y_{s_D}] \qquad (5.12)$$

$y = x - o_i$, S = *randperm (1:D)*

p_i: used to control the percentage of $g_i(x)$

n_i: dimension for each basic function $\sum_{i=1}^{N} n_i = D$

$$n_1 = [p_1 D], n_2 = [p_2 D], \ldots, n_{N-1} = [p_{N-1} D], n_N = D - \sum_{i=1}^{N-1} n_i \qquad (5.13)$$

F10 (Hybrid Funtion 1) (N = 3)

p = [0.3, 0.3, 0.4]

g_1 : High Conditioned Elliptic Function

g_2 : Modified Schwefel's Function

g_3 : Rastrigin's Function

F11 (Hybrid Function 2) (N = 4)

p = [0.2, 0.2, 0.3, 0.3]

g_1 : Weierstrass Function

g_2 : Griewank's Function

g_3 : Scaffer's F6 Function

g_4 : Rosenbrock's Function

F12 (Hybrid Function 3) (N = 5)

p = [0.1, 0.2, 0.2, 0.2, 0.3]

g_1 : Modified Schwefel's Function

g_2 : HappyCat Function

g_3 : Auckley's Function

g_4 : Katsuura Function

g_5 : Expanded Griewank's plus Rosenbrock's Function

5.1.4 Composite Functions

$$F(x) = \sum_{i=1}^{N}\{\omega_i * [\lambda_i g_i(x) + bias_i]\} + F^* \quad (5.14)$$

$F(x)$: Composite Function

$g_i(x)$: *i-th* basic function used to construct the composite function.

N: Number of basic functions

o_i: new shifted optimum position for each $g_i(x)$, define the global and local optima's position

$bias_i$: defines which optimum is the global optimum

σ_i: used to control each $g_i(x)$'s coverage range, a small σ_i gives a narrow range for that $g_i(x)$

λ_i: used to control each $g_i(x)$'s height

w_i: weight for each $g_i(x)$'s, calculated as below:

$$W_i = \frac{1}{\sqrt{\sum_{j=1}^{D}}} exp\left(\frac{\sum_{j=1}^{D}(x_j - o_{ij})^2}{2D\sigma_i^2}\right) \quad (5.15)$$

Then normalize the weight $\omega_i = w_i / \sum_{i=1}^{n} \omega_i$

So when $x = o_i, \omega_j = \begin{cases} 1 & j = i \\ 0 & j \neq i \end{cases}$ $for\ j = 1,2, \ldots, N,\ f(x) = bias_i + f^*$

The optimum which has the smallest bias value is the global optimum. The composite function merges the properties of the subfunction better and maintains continuity around the global/optima. For some composite function, the hybrid functions are used as the basic functions.

F13 (Composite Function 1) (N = 5)

λ = [1, 1e-6, 1e-26, 1e-6, 1e-6]

σ = [10, 20, 30, 40, 50]

bias = [0, 100, 200, 300, 400]

g_1 : Rotated Rosenbrock's Function

g_2 : Rotated Bent Cigar Function

g_3 : Expanded Griewank's plus Rosenbrock's Function

g_4 : High Conditioned Elliptical Function

g_5 : Rotated Discus Function

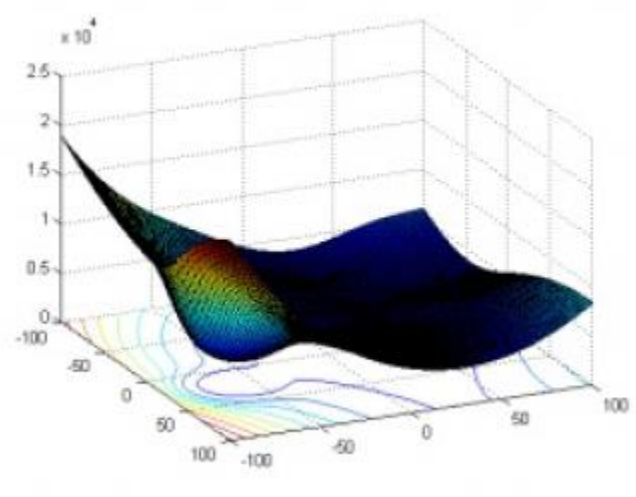

Figure 5.8: 3D – Map for Composite Function 1 [29]

F14 (Composite Function 2) (N = 3)

λ = [0.25, 1, 1e-7]

σ = [10, 20, 30]

bias = [0, 100, 200]

g_1 : Rotated Rastrigin's Function

g_2 : Rotated Schwefel's Function

g_3 : High Conditioned Elliptic Function

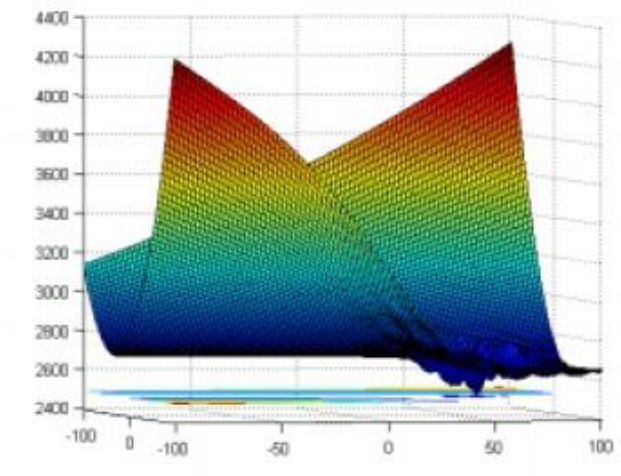

Figure 5.9: 3D – Map for Composite Function 2 [29]

F15 (Composite Function 3) (N = 5)

λ = [10, 10, 2.5, 2.5, 1e-6]

σ = [10, 10, 30, 40, 50]

bias = [0, 100, 200, 300, 400]

g_1 : Rotated Weierstrass Function

g_2 : Rotated Rastrigin's Function

g_3 : Rotated High Conditioned Elliptic Function

g_4 : Rotated Schwefel's Function

g_5 : Rotated Rosenbrock's Function

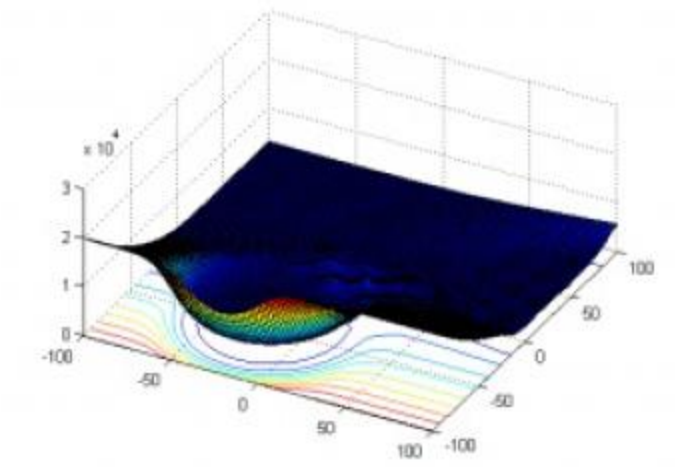

Figure 5.10: 3D – Map for Composite Function 3 [29]

5.2 Experimental Setup

Our proposed algorithm is compared with the performance of existing algorithms such as Cultural Algorithm (CA), Multi-Population Cultural Algorithm (MPCA), and MPCA incorporated by Game Theory Model (MPCA-GS) [19]. The proposed strategies: Tchebycheff method and Reference-Point method are compared with each other, and also with the above mentioned well-known algorithms. The algorithms are abbreviated as follows:

M1: Cultural Algorithm

M2: Multi-Population Cultural Algorithm

M3: Multi-Population Cultural Algorithm with Game theory model

M4: Cultural Algorithm with Tchebycheff method

M5: Cultural Algorithm with Reference point method

The five algorithms mentioned above are compared with each other. To make a fair comparison, the parameters used for all the algorithms are the same. The values of the parameters used are listed in Table 5.2. All the algorithms are tested individually 20 times on all the functions to get an exact solution. The performance of the proposed algorithm was evaluated using the following information:

Mean Value (Mean): This is the mean value of the solution gotten maximum generation in 20 runs.

Standard Deviation Value (Std): The standard deviation of the mean value.

Best Individual Value (Best): This is the best individual in the whole population of all the generations.

Average number of generation (Gen): This is the average number of generations required to find the best solution.

Parameters	Values
Size of the Population (*N)*	100
Number of Generations	100
Size of Neighbourhood	20
Crossover Probability (*CP)*	0.5
Scalar Factor (*F*)	0.5
Polynomial Mutation Rate ($p_{m)}$	$1/n$ (n=number of dimensions)
Independent Runs	20
Dimensions	10 & 30
Change of Severity (n_t)	20
Change of Frequency (τ_t)	10

Table 5.2 Parameter values for the algorithm

There are different types of DMOPs based on severity, frequency, and predictability of changes. In our approach, we concentrate on the frequency and severity-based changes. The change in Frequency (τ_t) means how often the environment changes. Therefore, there is an environmental change every 10 generations. The change in Severity (n_t) refers to how severe the problem changes. The change can be either small or large. If the severity is small, it is easier to converge to the optimal solution since information acquired from the previous generation can be reused to accelerate the convergence. Otherwise, the problem may be completely unrelated to the previous one [45].

5.3 Results and Analysis

In this section, a relative comparison with respect to all benchmark models/algorithm is accomplished via experiments. The comparisons are carried out in both low dimension (10D) and high dimension (30D) for all the benchmark problems discussed in section 5.1.

	F1	F2	F3	F4	F5	F6	F7
M1							
Mean	**1.12E09**	6.76E05	3.19E02	5.06E03	5.15E02	6.16E02	7.48E02
Std	5.18E02	4.65E03	0.74E00	2.01E02	0.65E00	2.07E00	5.60E01
Best	1.43E09	5.66E05	3.11E02	**1.64E03**	5.02E02	6.01E02	7.97E02
Gen	70	16	72	97	43	83	73
M2							
Mean	1.44E09	**5.22E05**	3.10E02	1.22E03	**5.02E02**	6.01E02	**7.11E02**
Std	3.15E02	9.47E03	0.71E00	8.15E01	0.21E00	0.19E00	2.97E00
Best	1.75E09	**5.07E04**	3.06E02	4.40E03	**5.01E02**	**6.00E02**	7.00E02
Gen	83	31	83	69	62	94	74
M3							
Mean	1.55E09	5.70E05	3.10E02	1.07E03	**5.02E02**	**6.02E02**	**7.11E02**

Std	9.92E02	1.49E02	1.05E00	9.46E01	0.24E00	0.22E00	2.86E00
Best	2.01E09	**5.04E04**	**3.05E02**	3.77E03	**5.01E02**	**6.00E02**	**7.00E02**
Gen	97	29	59	91	54	93	86
M4							
Mean	1.40E09	6.67E04	**3.03E02**	5.27E03	**4.99E02**	**6.02E02**	**7.11E02**
Std	0.26E01	5.35E01	0.65E00	0.75E00	0.10E01	0.25E00	0.28E01
Best	2.96E09	5.94E04	**3.05E02**	5.96E03	**5.01E02**	**5.99E02**	7.28E02
Gen	84	40	55	87	23	36	69
M5							
Mean	1.40E09	6.99E05	3.27E02	**1.03E03**	**5.02E02**	**5.98E02**	**7.11E02**
Std	3.56E03	4.82E02	0.87E00	2.01E01	0.58E00	0.30E00	2.85E00
Best	**1.27E09**	3.91E05	4.57E02	4.65E03	**5.01E02**	**6.00E02**	7.97E02
Gen	90	35	61	91	26	42	74

Table 5.3: M1-M5 on F1-F7 for 10D

	F8	F9	F10	F11	F12	F13	F14	F15
M1								
Mean	7.79E04	9.05E02	3.83E03	4.70E03	1.88E03	2.46E03	7.87E04	4.43E04
Std	2.43E01	0.29E00	4.63E01	2.63E02	1.07E01	1.62E03	1.16E02	3.28E03

Best	8.66E04	**9.04E02**	3.77E03	1.43E03	1.97E03	2.60E03	8.31E04	2.00E04
Gen	99	93	89	94	84	98	91	71
M2								
Mean	**6.83E03**	9.04E02	3.01E03	1.11E03	1.48E03	1.66E03	5.61E03	1.98E03
Std	3.67E03	0.11E00	4.11E01	2.73E00	5.73E01	1.53E01	1.61E00	1.12E02
Best	8.04E03	**9.04E02**	**3.23E03**	1.10E03	1.29E03	1.62E03	5.60E03	1.56E03
Gen	80	75	81	90	69	97	93	57
M3								
Mean	7.07E03	9.04E02	2.03E03	**1.11E03**	**1.48E03**	1.66E03	5.61E03	2.02E03
Std	4.66E03	0.10E00	5.72E02	1.90E00	5.51E01	1.67E01	2.50E00	5.12E01
Best	**8.03E03**	**9.04E02**	3.49E03	**1.10E03**	1.24E03	**1.61E03**	5.60E03	1.93E03
Gen	80	79	86	86	61	97	93	69
M4								
Mean	7.95E03	**9.01E02**	**2.01E03**	**1.11E03**	**1.48E03**	**1.42E03**	**5.59E03**	**1.77E03**
Std	2.87E03	0.23E00	4.10E02	2.54E00	1.98E02	1.45E02	1.75E02	1.32E02
Best	8.01E03	9.99E02	3.58E03	1.57E03	**1.01E03**	**1.61E03**	5.98E03	**1.43E03**
Gen	77	96	84	73	62	92	71	56
M5								
Mean	7.56E03	9.82E02	**2.01E02**	1.85E03	**1.33E03**	**1.42E03**	5.98E01	**1.77E03**

Std	2.93E00	0.17E00	3.19E02	2.67E02	1.45E03	1.97E03	1.63E01	2.02E02
Best	**8.03E03**	**9.04E02**	3.52E02	1.91E03	1.24E03	**1.58E03**	**1.25E03**	2.41E03
Gen	72	93	81	83	59	90	77	65

Table 5.4: M1-M5 on F8-F15 for 10D

As seen in table 5.3 and 5.4; the mean, standard deviation, best individual, and average generations have been recorded. The results of our proposed method are tabulated against the different existing algorithms, aforementioned. The results are for 10 dimensions. In abide to ensure equity/balance in our comparisons, all the parameters used are similar. All the best individuals and similar values are highlighted in the table (5.3 and 5.4). With respect to our results, M4 outperforms 7 out of the 15 benchmark functions. Thus, giving 4 functions with similar results (F6, F7, F11, F12), but the number of generations required by the proposed M4 approach to converge is lesser when compared with other existing methods. Accordingly, we analyze the results in the next chapter.

On the other hand, another proposed approach M5 outperforms only 6 out of the 15 functions. Same as M4, it gives similar results for two functions (F5 and F7), and the number of generations required to converge is relatively the same as the existing methods. It can be observed that this method performs well on composite and hybrid functions as against multi-modal and unimodal functions. The explanation will be discussed in chapter 6.

	F1	F2	F3	F4	F5	F6	F7
M1							
Mean	2.60E09	**1.04E05**	3.37E02	3.32E03	6.57E02	6.51E03	7.04E02
Std	1.05E07	1.29E03	1.35E00	5.32E02	0.39E00	0.94E00	3.55E01
Best	**1.03E09**	2.69E05	3.45E02	5.81E03	5.04E02	6.02E03	9.35E02
Gen	95	22	78	87	51	75	85
M2							
Mean	2.60E09	1.27E05	**3.35E02**	2.28E03	6.02E02	6.42E03	**7.03E02**
Std	4.99E03	5.50E02	1.27E00	2.53E02	0.45E00	6.34E00	1.10E01
Best	2.60E09	1.71E05	3.36E02	**2.29E03**	5.02E02	6.01E03	7.04E02
Gen	98	31	87	88	36	51	96
M3							
Mean	2.99E09	1.05E05	3.38E02	**2.56E03**	6.02E02	6.45E03	7.04E02
Std	4.49E09	3.78E04	1.62E00	2.64E02	0.36E00	0.44E00	1.13E01
Best	3.65E09	**1.19E05**	3.36E02	2.59E03	**5.02E02**	6.01E03	7.05E02
Gen	97	35	71	98	44	53	98
M4							
Mean	**1.75E09**	1.05E05	3.45E02	**2.56E03**	**5.59E02**	**6.31E03**	7.04E02
Std	2.70E03	2.98E03	1.98E00	2.99E02	0.99E00	0.52E00	1.92E02

Best	2.56E09	**1.19E05**	**3.06E02**	2.59E03	**5.02E02**	**5.99E03**	7.65E02
Gen	76	10	63	99	33	83	79
M5							
Mean	**1.75E09**	1.06E09	3.08E02	**2.23E03**	**5.59E02**	**6.31E03**	7.05E02
Std	2.98E02	5.54E09	1.84E00	2.26E02	0.85E00	0.94E00	1.72E02
Best	2.65E09	**1.17E09**	3.16E02	2.61E03	**5.02E02**	6.00E03	**7.02E02**
Gen	73	21	67	95	47	87	82

Table 5.5: M1-M5 on F1-F7 for 30D

	F8	F9	F10	F11	F12	F13	F14	F15
M1								
Mean	1.14E04	9.67E02	2.05E06	1.58E03	1.94E03	3.65E03	1.99E03	2.82E03
Std	4.67E02	0.20E00	8.84E03	4.37E02	3.58E02	3.52E03	4.70E03	8.58E03
Best	1.78E04	9.14E02	3.62E06	1.48E03	**1.07E03**	4.27E03	2.88E03	7.28E03
Gen	91	81	42	10	42	91	91	57
M2								
Mean	2.09E04	9.15E02	1.74E06	1.44E03	2.05E03	1.68E03	1.67E03	2.81E03
Std	4.34E04	0.85E00	1.98E03	4.52E01	7.65E03	6.97E01	2.71E01	2.15E01
Best	2.01E04	9.13E02	3.58E06	1.41E03	1.94E03	1.68E03	1.65E03	2.81E03

Gen	98	89	49	69	88	93	94	35
M3								
Mean	**1.01E04**	**9.14E02**	**1.15E06**	1.51E03	2.05E03	**1.69E03**	1.63E03	2.77E03
Std	4.94E03	0.91E00	2.38E01	3.38E01	3.45E03	7.22E01	3.91E01	3.01E01
Best	2.10E04	9.14E02	**1.50E06**	1.15E03	2.05E03	1.69E03	1.63E03	2.78E03
Gen	91	87	81	59	69	90	95	50
M4								
Mean	1.54E04	**9.14E02**	1.50E06	**1.41E03**	**1.85E03**	**1.69E03**	**1.52E03**	**2.72E03**
Std	4.26E02	0.98E00	2.63E03	3.45E03	2.98E02	9.67E03	2.32E03	3.51E01
Best	**1.61E04**	**9.01E02**	3.32E06	**1.13E03**	**1.07E03**	1.69E03	**1.61E03**	**2.56E03**
Gen	82	75	74	73	36	79	84	47
M5								
Mean	3.98E04	9.14E02	1.17E06	**1.41E03**	**1.98E03**	**1.63E03**	**1.52E03**	**2.77E03**
Std	2.36E02	0.96E00	3.84E03	3.65E03	2.54E02	9.78E03	2.21E03	3.49E01
Best	1.91E04	9.99E02	3.63E06	1.98E03	1.28E03	**1.67E03**	1.67E03	2.69E03
Gen	95	88	66	74	38	67	97	52

Table 5.6: M1-M5 on F8-F15 for 30D

Additionally, as observed in table 5.5 and 5.6; the mean, standard deviation, best individual, and average generations have been recorded. The results of our proposed method are tabulated against the different existing

algorithms, aforementioned. The results are for 30 dimensions. In order to ensure a relatively fair comparison, all the parameters used are similar. All the best individuals and similar values are highlighted in the table(5.5 and 5.6). With regard to our results, M4 outperforms 7 out of the 15 benchmark functions. Thus, resulting in 3 functions with similar results (F4, F9, F13), but the number of generations acquired by the proposed M4 approach to converge is lesser in comparison to the existing methods. We analyze the results in the next chapter.

On the other hand, another proposed approach M5 outperforms 8 out of the 15 benchmark functions. Same as M4, it gives similar results for 2 functions (F9 and F15), and the number of generations required to converge is relatively same as the existing methods. It can be observed that this method performs well on composite and hybrid as opposed to multi-modal and unimodal functions. The explanation will be discussed in chapter 6.

Chapter 6

Discussion, Comparisons, and Analysis

In this chapter, we will discuss the different proposed strategies in the context of their characteristics. We will compare all the algorithms with each other and the convergence speed to reach the near-optimal solution.

6.1 Comparison between M2, M3, and M4

We will compare M2 Vs. M3 Vs. M4. M4 is the Tchebycheff method, and we will compare it with the other two algorithms, MPCA, and MPCA with Game theory model on the 30-Dimensional problem.

Figure 6.1 demonstrates the performance of M4 with M2 and M3 on function F5, regarding the fitness value against the number of generations. We can observe from the figure that for M4 there is a fluctuation in fitness up to some generations. This is because of the environmental change that is occurring by the parameter (change in severity) mentioned in table 5.2. While for other algorithms like M2 and M3 it takes longer number of generations than M4 to

reach the optimal solution. The decomposition strategy helps to solve multi-objective problems as it uses the information from the neighbourhood subproblem.

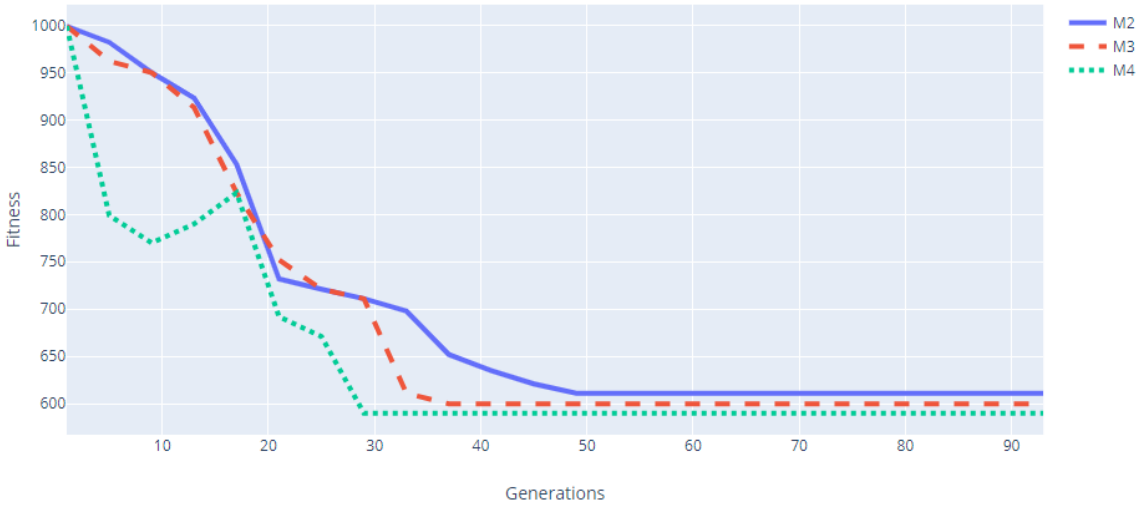

Figure 6.1: Convergence performance of M2, M3 and M4 for F4 (30D)

6.2 Comparison between M1, M3, and M4

Figure 6.2 demonstrates the performance of M4 with M1 and M3 on function F11, regarding the fitness value against the number of generations. This function is hybrid in nature and hence will have many local minima. It is a complex, non-convex and non-separable function. It is difficult for the individuals in the population to escape from the local optima and explore the new search space for a good optimal solution. This is the reason for the algorithm to have same solution over generations. This is the reason M1 attains a optimal

solution and gets stuck to it. It falls into the local optima and follows the same pattern for the remaining generations. While M3 gets a good solution and then again obtains global minima. The knowledge of the past generation inherited by the next generation makes then fall into global minima. The graph of M4 shows that it takes time by the individual to attain the optimal solution. The decomposition is done by the dominating individuals of the population which allows the whole population to make better decisions and escape from the local optima.

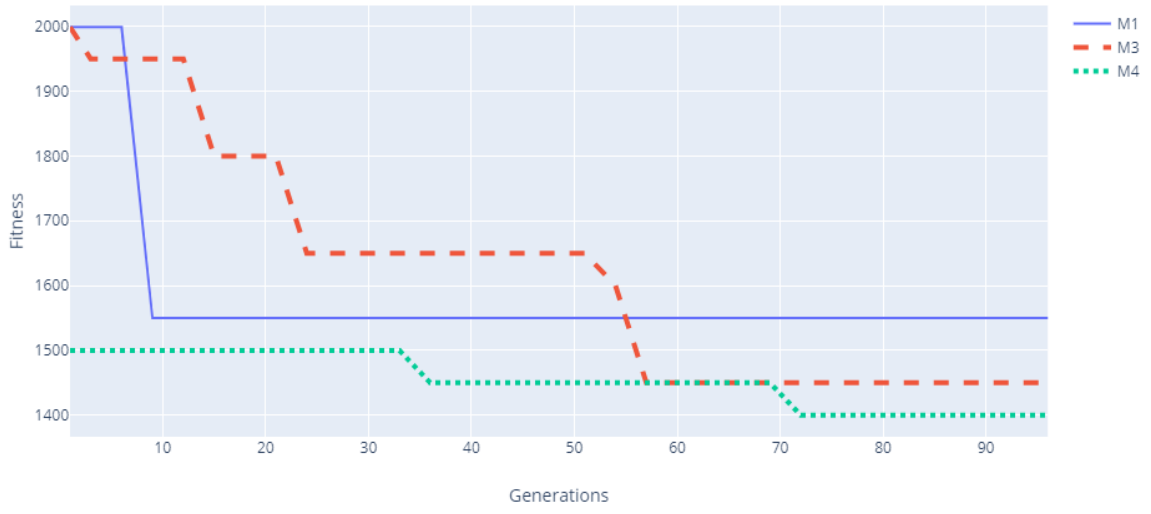

Figure 6.2: Convergence performance of M1, M3 and M4 for F11 (30D)

6.3 Comparison between M2, M3, and M5

We will compare M2 Vs. M3 Vs. M5. M5 is the Reference point method, and we will compare it with the other two algorithms, Multi Population Cultural

Algorithm (MPCA), and MPCA with Game theory model on the 30-Dimensional problem.

The figure 6.3 demonstrates the comparison of M2, M3 and M5 on function F6. All the algorithms almost shows similar pattern in optimizing the function. M2 reaches the optimal solution (converges) in 60 generations. Whereas M3 gradually converges to the optimal solution but it does not reach the optimal solution. M5 gives the best result as the reference point method helps the individuals to explore the unsearched space with a better optimal solution. The convergence speed is not high because the population also posses diversity, avoiding it to move towards the best solution.

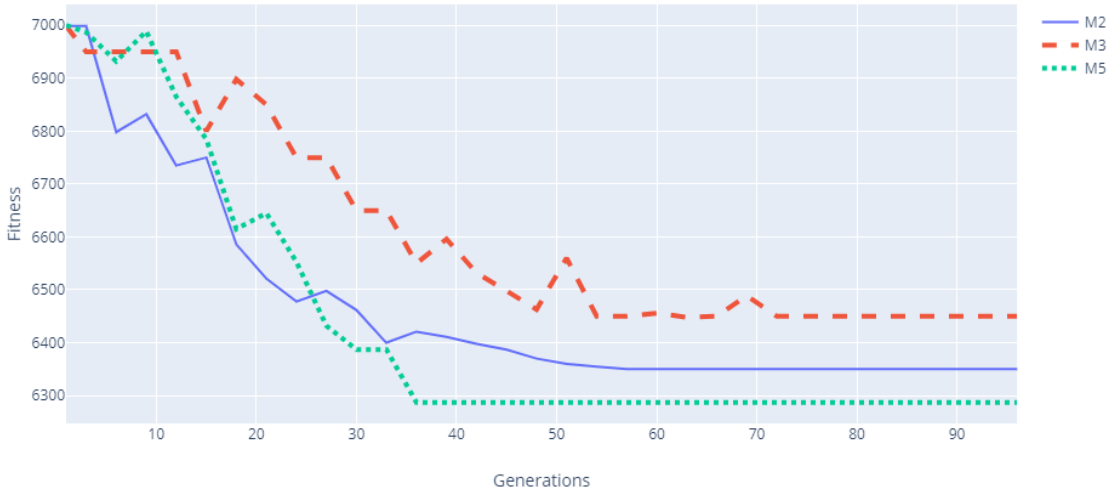

Figure 6.3: Convergence performance of M2, M3 and M5 for F6 (30D)

6.4 Comparison between M1, M2, and M5

We will compare M1 Vs. M2 Vs. M5. M5 is the Reference point method, and we will compare it with the other two algorithms Cultural Algorithm (CA), and MPCA with Game theory model on the 30-Dimensional problem.

The figure 6.4 demonstrates the comparison of M1, M2 and M5 on function F10. All the algorithms follow similar pattern except for M1 which is CA. CA finds an optimal solution but in the next generation there is not enough exploration of the search space. While M2 and M5 follows a similar pattern for certain number of generations but later M2 keeps fluctuating, but M5 continues to find a better solution in the search space.

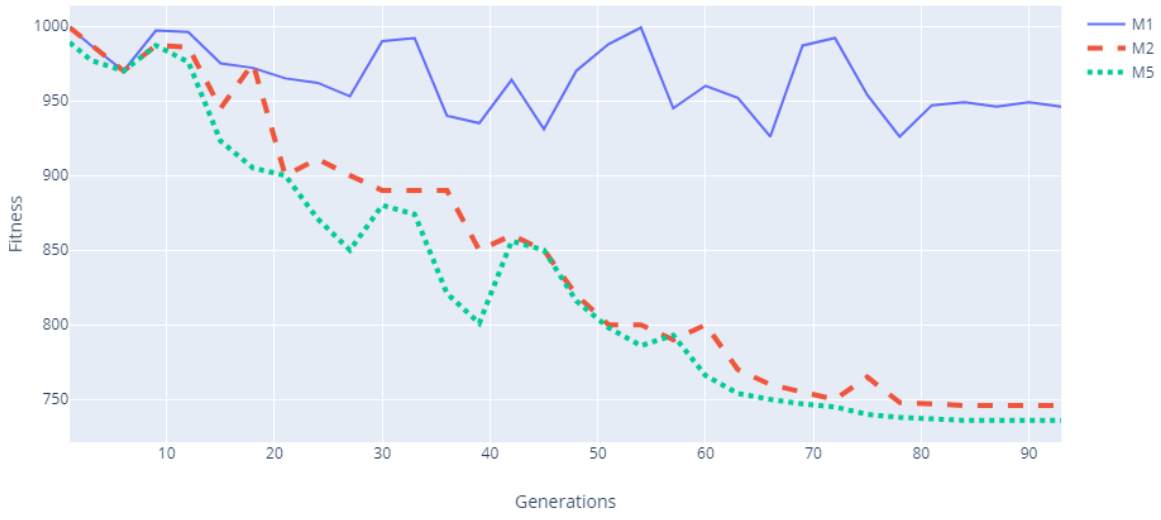

Figure 6.4: Convergence performance of M1, M2 and M5 for F10 (30D)

6.5 Time Complexity

While analyzing an algorithm, we mostly consider the time complexity. The complexity depends upon the genetic operators, representation of the individuals and the population. The algorithm starts with the individual representation. The algorithm can decode a individual in linear time with the complexity of O(n), where n is the number of individuals. Here the run time complexity for each iteration in our algorithm can be represented as O(gs), where g is the number of iterations and s is the population size. Since we are using Cultural Algorithm we need to consider the complexity for updating the belief space and it can be represented as O(B).

Chapter 7

Conclusion and Future Work

We proposed Cultural algorithm incorporating decomposition strategies to solve Dynamic Multi-Objective Optimization Problems. The decomposition methods used in this thesis are the Tchebycheff method and Reference Point method. We have compared CA/D with the traditional CA and MPCA and also MPCA with game strategies, respectively.

Our primary focus was to show the convergence performance of the population. To witness the convergence impact, we introduced decomposition into the cultural algorithm. It can be observed that the search space has been explored and exploited which makes the proposed algorithm to escape from the local optima. We have used the CEC 2015 benchmark functions to evaluate the proposed algorithm and compared them with the aforementioned. Our experimentation results and analysis show that CA/D outperforms on most of the complex test problems and gives similar kind of results or equivalent on the others.

The major limitation of our model is that using Tchebycheff method for extremely large datasets because, the evaluation of the objective function values may be strenuous[1] and great deal of calculation is needed. The RP method convergence may not be fast, if the decision maker is not determined to find the optimal solutions.

In the near future, we intend to evaluate the proposed algorithm with more complex dynamic benchmark functions possessing higher dimensions. More decomposition strategies can be introduced into other evolutionary algorithms. The proposed approach shows good results on complex problems such as hybrid and composite functions. This procedure could be applied to real world applications Team formation problem (TFP) – to select multiple individuals that match the required set of skills must be chosen to maximize one or more social positive parameter. Another example such as reducing hospital readmissions to cut the healthcare costs and also recommendation systems, decision support systems. Apparently, all of the real world applications are dynamic in nature and complex.

REFERENCES

[1] Kaisa Miettinen. 1999. Nonlinear Multiobjective Optimization. Vol. 12. Springer Science & Business Media.

[2] Radhia Azzouz, Slim Bechikh, and Lamjed Ben Said. 2015. Multi-objective optimization with dynamic constraints and objectives: new challenges for evolutionary algorithms. In Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation Conference (GECCO). ACM, 615–622.

[3] Kalyanmoy Deb. 2001. Multi-Objective Optimization Using Evolutionary Algorithms. Vol. 16. John Wiley & Sons.

[4] K Deb. 1995. Optimization methods for engineering design. (1995).

[5] Tim Blackwell and Jürgen Branke. 2006. Multiswarms, exclusion, and anticonvergence in dynamic environments. IEEE transactions on evolutionary computation 10, 4 (2006), 459–472.

[6] Leilei Cao, Lihong Xu, Erik D Goodman, Shuwei Zhu, and Hui Li. 2018. A differential prediction model for evolutionary dynamic multiobjective

optimization. In Proceedings of the Genetic and Evolutionary Computation Conference. ACM, 601–608.

[7] Abraham Charnes, William W Cooper, and Robert O Ferguson. 1955. Optimal estimation of executive compensation by linear programming. Management science 1, 2 (1955), 138–151.

[8] Renzhi Chen, Ke Li, and Xin Yao. 2018. Dynamic multiobjectives optimization with a changing number of objectives. IEEE Transactions on Evolutionary Computation 22, 1 (2018), 157–171.

[9] Jingxuan Wei and Liping Jia. 2013. A novel particle swarm optimization algorithm with local search for dynamic constrained multi-objective optimization problems. In 2013 IEEE Congress on Evolutionary Computation. IEEE, 2436–2443.

[10] Coello, Carlos A. Coello, and Nareli Cruz Cortés. "Solving multiobjective optimization problems using an artificial immune system." *Genetic Programming and Evolvable Machines* 6.2 (2005): 163-190.

[11] Andrzej Jaszkiewicz. 2002. On the performance of multiple-objective genetic local search on the 0/1 knapsack problem-a comparative experiment. IEEE Transactions on Evolutionary Computation 6, 4 (2002), 402–412

[12] Robert G Reynolds. 1994. An introduction to cultural algorithms. In Proceedings of the third annual conference on evolutionary programming. World Scientific, 131–139.

[13] Ziad Kobti et al. 2013. Heterogeneous multi-population cultural algorithm. In 2013 IEEE Congress on Evolutionary Computation. IEEE, 292–299.

[14] Andries P Engelbrecht. 2007. Computational intelligence: an introduction. John Wiley & Sons.

[15] Qingfu Zhang and Hui Li. 2007. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on evolutionary computation 11, 6 (2007), 712–731.

[16] Arrchana Muruganantham, Kay Chen Tan, and Prahlad Vadakkepat. 2016. Evolutionary dynamic multiobjective optimization via Kalman filter prediction. IEEE transactions on cybernetics 46, 12 (2016), 2862–2873.

[17] Shouyong Jiang and Shengxiang Yang. 2017. A steady-state and generational evolutionary algorithm for dynamic multiobjective optimization. IEEE Transactions on Evolutionary Computation 21, 1 (2017), 65–82.

[18] Leilei Cao, Lihong Xu, Erik D Goodman, and Hui Li. 2019. Decomposition-based evolutionary dynamic multiobjective optimization using a difference model. Applied Soft Computing 76 (2019), 473–490.

[19] Panth Parikh and Ziad Kobti. 2017. Comparative strategies for knowledge migration in Multi Objective Optimization Problems. In 2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE).

IEEE, 1–5

[20] Aimin Zhou, Yaochu Jin, Qingfu Zhang, Bernhard Sendhoff, and Edward Tsang. 2007. Prediction-based population re-initialization for evolutionary dynamic multi-objective optimization. In International Conference on Evolutionary MultiCriterion Optimization. Springer, 832–846.

[21] Dilpreet Singh et al 2018. A Multilevel Cooperative Multi-Population Cultural Algorithm. In 2018 IEEE Innovations in Intelligent Systems and Applications (INISTA). IEEE, 1-5

[22] Stephen P Boyd and Lieven Vandenberghe. Convex optimization (pdf). *Np: Cambridge UP*, 2004.

[23] Agoston E Eiben, James E Smith, et al. Introduction to evolutionary computing, volume 53. Springer, 2003.

[24] John H Holland. Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence. MIT press, 1992.

[25] Rainer Storn and Kenneth Price. Differential evolution- a simple and efficient heuristic for global optimization over continuous spaces. Journal of global optimization, 11(4):341-359, 1997.

[26] Massimiliano Vasile, Edmondo Minisci, and Marco Locatelli. An inationary differential evolution algorithm for space trajectory

optimization. IEEE Transactions on Evolutionary Computation, 15(2):267-281, 2011.

[27] Godfrey Onwubolu and Donald Davendra. Scheduling ow shops using differential evolution algorithm. European Journal of Operational Research, 171(2):674-692, 2006.

[28] Bin Qian, Ling Wang, Rong Hu, Wan-Liang Wang, De-Xian Huang, and Xiong Wang. A hybrid differential evolution method for permutation flow-shop scheduling. The International Journal of Advanced Manufacturing Technology, 38(7-8):757-777, 2008.

[29] JJ Liang, BY Qu, PN Suganthan, and Q Chen. Problem definitions and evaluation criteria for the cec 2015 competition on learning-based real-parameter single objective optimization. Technical Report201411A, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, 2014.

[30] Boyd, Stephen, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[31] Upadhyayula, Santosh. "Dominance in multi-population cultural algorithms." (2015).

[32] Wang, Yu, and Bin Li. "Multi-strategy ensemble evolutionary algorithm for dynamic multi-objective optimization." *Memetic Computing* 2.1 (2010): 3-24.

[33] Branke, Jürgen. "Memory enhanced evolutionary algorithms for changing optimization problems." *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*. Vol. 3. IEEE, 1999.

[34] Hatzakis, Iason, and David Wallace. "Dynamic multi-objective optimization with evolutionary algorithms: a forward-looking approach." *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006.

[35] Koo, Wee Tat, Chi Keong Goh, and Kay Chen Tan. "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment." *Memetic Computing* 2.2 (2010): 87-110.

[36] Zhou, Aimin, Yaochu Jin, and Qingfu Zhang. "A population prediction strategy for evolutionary dynamic multiobjective optimization." *IEEE transactions on cybernetics* 44.1 (2013): 40-53.

[37] Amato, P., and M. Farina. "An ALife-inspired evolutionary algorithm for dynamic multiobjective optimization problems." *Soft Computing: Methodologies and Applications*. Springer, Berlin, Heidelberg, 2005. 113-125.

[38] Azzouz, Radhia, Slim Bechikh, and Lamjed Ben Said. "A multiple reference point-based evolutionary algorithm for dynamic multi-objective optimization with undetectable changes." *2014 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2014.

[39] Coello, Carlos A. Coello, and Ricardo Landa Becerra. "Evolutionary multiobjective optimization using a cultural algorithm." *Proceedings of the 2003 IEEE Swarm Intelligence Symposium. SIS'03 (Cat. No. 03EX706)*. IEEE, 2003.

[40] Deb, Kalyanmoy, and J. Sundar. "Reference point based multi-objective optimization using evolutionary algorithms." *Proceedings of the 8th annual conference on Genetic and evolutionary computation*. ACM, 2006.

[41] Cohon, Jared L. *Multiobjective programming and planning*. Vol. 140. Courier Corporation, 2004.

[42] Saleem, Saleh, and Robert Reynolds. "Cultural algorithms in dynamic environments." *Proceedings of the 2000 Congress on Evolutionary Computation. CEC00 (Cat. No. 00TH8512)*. Vol. 2. IEEE, 2000.

[43] Kobti, Ziad. "Heterogeneous multi-population cultural algorithm with a dynamic dimension decomposition strategy." *Canadian Conference on Artificial Intelligence*. Springer, Cham, 2014.

[44] Guo, Yi-nan, et al. "A novel multi-population cultural algorithm adopting knowledge migration." *Soft computing* 15.5 (2011): 897-905.

[45] Azzouz, Radhia, Slim Bechikh, and Lamjed Ben Said. "A dynamic multi-objective evolutionary algorithm using a change severity-based adaptive population management strategy." *Soft Computing* 21.4 (2017): 885-906.

[46] Weise, Thomas. "Global optimization algorithms-theory and application." *Self-Published Thomas Weise* (2009).

[47] Bhullar, Amangel. "Improving Quality of the Solution for the Team Formation Problem in Social Networks Using SCAN Variant and Evolutionary Computation." (2018).

[48] Booker, Lashon B., David E. Goldberg, and John H. Holland. "Classifier systems and genetic algorithms." *Artificial intelligence* 40.1-3 (1989): 235-282.

[49] Chittle, Joshua, and Ziad Kobti. "A New Dimension Division Scheme for Heterogeneous Multi-Population Cultural Algorithm." *The Twenty-Seventh International Flairs Conference*. 2014.

[50] Xue, Zhengui, and Yinan Guo. "Improved cultural algorithm based on genetic algorithm." *2007 IEEE International Conference on Integration Technology*. IEEE, 2007.

[51] Kobti, Ziad, R. G. Reynodls, and Tim Kohler. "A multi-agent simulation using cultural algorithms: The effect of culture on the resilience of social systems." *The 2003 Congress on Evolutionary Computation, 2003. CEC'03.*. Vol. 3. IEEE, 2003.

[52] Van Veldhuizen, David A. *Multiobjective evolutionary algorithms: classifications, analyses, and new innovations*. No. AFIT/DS/ENG/99-01. AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING, 1999.

[53] Deb, Kalyanmoy, and S. Karthik. "Dynamic multi-objective optimization and decision-making using modified NSGA-II: a case study on hydro-thermal power scheduling." *International conference on evolutionary multi-criterion optimization*. Springer, Berlin, Heidelberg, 2007.

[54] Deb, Kalyanmoy, et al. "A fast and elitist multiobjective genetic algorithm: NSGA-II." *IEEE transactions on evolutionary computation* 6.2 (2002): 182-197.

[55] Zhu, Qingling, et al. "An Elite Gene Guided Reproduction Operator for Many-Objective Optimization." *IEEE transactions on cybernetics* (2019).

VITA AUCTORIS

NAME:	Ramya Ravichandran
PLACE OF BIRTH:	Chennai, India
YEAR OF BIRTH:	1996
EDUCATION:	Bachelor of Technology in Information Technology, Vel Tech Multi Tech Dr.RR & Dr.SR Engineering College, Avadi, India, 2017
	Master of Science in Computer Science, University of Windsor, Windsor, ON, 2019