3-2-2021

H-MPGWO: A Hierarchical Multi-Population Grey Wolf Optimization Framework

Nimish Verma
University of Windsor

H-MPGWO: A Hierarchical Multi-Population Grey Wolf Optimization Framework

By

Nimish Verma

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

H-MPGWO: A Hierarchical Multi-Population Grey Wolf Optimization Framework

by

Nimish Verma

APPROVED BY:

17 December 2020

Declaration of Co-Authorship/Previous Publication

I. Co-Authorship

I hereby declare that this thesis incorporates material that is a result of joint research, as follows: Chapter 2 of the thesis is co-authored with Dr. Pooya Moradian Zadeh, and Dr. Ziad Kobti, under their joint co-supervision. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by the author, and the contribution of co-authors was primarily through providing feedback on the refinement of ideas and editing of the 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 own work.

II. Previous Publication

This thesis includes 1 original paper that has been previously published/submitted for publication in peer reviewed journals, as follows:

Section	Full citation	Publication status
2.1, 2.3	N. Verma, P. Zadeh, Z. Kobti ”A Novel Cooperative Hierarchical Multi-Population Optimization Algorithm” Expert Systems with Application (2021)	Submitted

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.

III. General

I declare that, to the best of my knowledge, my thesis does not infringe upon anyone's copyright nor violate any proprietary 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 copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) 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

Various fields, such as engineering, physics, and economics, require optimization in the real world. Various meta-heuristic methods have gained popularity in recent decades to solve these optimization problems; evolutionary algorithms are one of the ways to solve these problems. This class of algorithms deal with a generation of candidate solutions that are evolved until a stopping criterion is achieved. Researchers are improving these algorithms' performance by introducing new ensemble strategies to tackle a variety of problems. This thesis focuses on creating a novel co-operative multi-population framework to solve single and bi-objective problems based on the hunting strategies and hierarchical structures of grey wolves. The structure of this framework allows to overcome several defects and improves the information flow and convergence of the search process.

The framework is evaluated using IEEE's Congress of Evolution Congress benchmarks for single-objective real parameter optimization (2013) and unconstrained multi-objective optimization problems (2009). The performance is compared with the traditional grey wolf optimization algorithms and state-of-the-art for single and multi-objective optimization.

Dedication

Every challenging work needs self-efforts as well as guidance of elders especially those who are very close to our hearts.

I dedicate this humble effort to my sweet and loving

Father and Mother,

To my wonderful sister Ridhi, thank you for the encouragement.

To Pixie and Pogo, for being the best listeners.

Acknowledgements

I owe a debt of gratitude to Dr. Pooya and Dr. Kobti, for their constant help and motivation. This thesis work is possible due to their vision and knowledge. As my teachers and mentors, they have taught me more than I could ever give them credit for. Thank you for your patience, motivation, enthusiasm and immense knowledge. I am also thankful to my thesis committee members, Dr. Razavi Far, and Dr. Samet, for providing me extensive professional and personal guidance, which helped me learn a great deal about both scientific research and life in general. Nobody has been more important to me in the pursuit of this thesis than my family members and colleagues. Special thanks to my friends in Windsor for their support and care throughout my coursework. Most importantly, I wish to thank all the faculties and staff of the School of Computer Science for the environment and support.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Optimization

Optimization is the art of making better decisions; it is a technique of finding the best set of solution(s) for a given objective function under given constraints or limitations [3]. It is significant as it finds a feasible solution in a limited amount of time (or resources). In single-objective optimization problems, the best solution based on one objective function (minimization) [1].

$$Minimize\ f(x) \qquad x = [x_1, x_2, \dots,\ x_D]$$
$$L_i \leq x_i \leq U_i,\ i = 1, 2, \dots, D$$

where D is the dimension of the problem, x is a variable in the solution space and $[L_i$,$U_i]$ are boundaries of the ith dimension. An example of this kind of problem can be finding the best value of x, for which cost is minimum.

Multi-objective problems are more complex than single-objective problems since we deal with more than one objective function. The objective functions are represented by F(x) where-

$$Minimize\ F\left(x\right)=f_1\left(x\right),\ f_2\left(x\right),\ldots,f_k(x) \qquad x=[x_1,x_2,\ldots,\ x_D]$$
$$L_i\leq x_i\leq U_i,\ i=1,2,\ldots,D$$

The functions f_1, f_2,...,f_k are the objective functions that need to be optimized [2]. In case of multi-objective problems, k is usually 2 or 3, if $k > 4$, then the problem is classified as a many-objective problem.

In multi-objective optimization there exists more than one optimal solutions, unlike single-objective optimization. To compare candidate solutions, the concept of Pareto dominance is commonly used. Consider $X_1 = (x_1^1, x_2^1, \ldots, x_D^1)$ and $X_2 = (x_1^2, x_2^2, \ldots, x_D^2)$. According to Pareto dominance, X_1 is said to be dominated by X_2 if and only if:

$$\forall i\in(1,2,\ldots,k): f_1(k)\geq f_2(k)$$
$$\exists i\in(1,2,\ldots,k): f_1(k)> f_2(k)$$

where k is the number of objective functions f_i. Thus, the set of optimal solutions (or Pareto front) is the set of non-dominated solutions, illustrated in Figure 1.1.

While there exists many dynamic optimization problems, in which there exists dynamic variables whose values change over time, in this thesis we will be restricting our scope to static optimization problems.

Some applications of optimization are optimizing warehouse location to minimize shipping time, optimizing the structural design to maximize load-bearing, optimizing the stock portfolio to maximize profits. It is evident that optimization has a wide range of applications and is ubiquitous, hence it is an important research topic. When it comes to solving these optimization problems there are various methods such as linear programming, simulated annealing, evolutionary algorithms and many more. This research focuses on evolutionary algorithms, because of their advantages, which

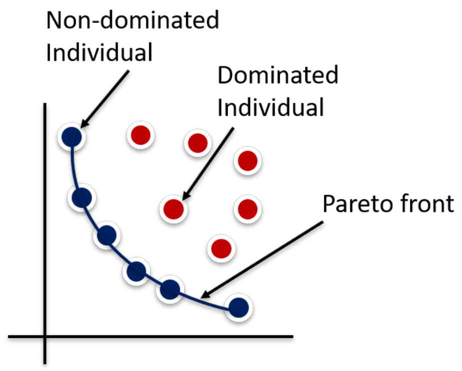


Figure 1.1: Illustration of Pareto Front

are discussed in the later sections.

1.2 No-Free Lunch

In 1997, the no-free lunch (NFL) theorem was proposed in the field of mathematics [4]. It states that if an algorithm performs excellently on a specific subset of problems, then it will surely suffer a loss of performance on another subset of problems. Therefore, to compare the equivalence of two algorithms, their performance needs to be averaged over all variety of problems.

Theoretically, the NFL states that it is not possible to have an algorithm efficient for all problems. Nevertheless, in practice, recent studies have shown that free-lunches are possible when an ensemble of strategies is used to make the algorithm more versatile [5, 6]. Thus, a meticulously designed algorithm with a dynamic ensemble can be considered efficient for the considered subset of problems.

1.3 Evolutionary Algorithms

As mentioned earlier, there are numerous ways to solve optimization problems, evolutionary algorithm (EA) is one of them. EAs are a subset of evolutionary computation, and have gained much popularity in recent years for meta-heuristic optimization. These algorithms are generally inspired by biological evolution and evolve a generation of the population until the stopping criteria are achieved. The goal of these algorithms is to find a near-optimal solution. The advantages of using EAs are, and not limited to:

1. **Robustness**- They are able to deal with many solutions at once, and hence are able to achieve near-optimal solutions in a shorter time than their traditional counterparts, which take a long time to provide the optimal solution.

2. **Easy to implement**- EAs can be treated as a black-box method, and do not require any implementation other than basic solution representation and search operators.

3. **Suitability for multi-objective problems**- It is an inherent quality of EAs, that by making little or no changes they can be used for single and multi objective problems,

4. **Parallel and Distributed**- EAs can be run in parallel, and even distributed really efficiently. Because of the fact that each individual is independently evaluated, it is possible to distribute the task of evaluating n individuals to n processors, and in turn save time.

A general high-level flowchart of evolutionary algorithms is shown in Figure 1.2. The common underlying concept of this set of algorithms is - a population of an individual is initialized and their fitness is evaluated according to an objective function. The best ones are selected, while the worst ones may be discarded, and the next generation of the population is generated by applying various operators such as mutation and recombination.

One of the first EA to be proposed was by John H. Holland in 1975, called genetic algorithm [7]. The basic concept behind it is similar to that of genes in biology, a solution is represented as a vector of real or integer values, and the gene is mutated

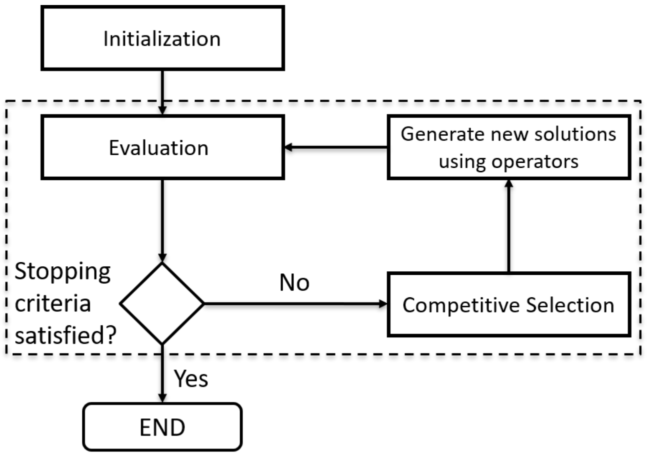

Initialization
Evaluation
Generate new solutions using operators
Stopping criteria satisfied?
No
Competitive Selection
Yes
END

Figure 1.2: Flowchart for Evolutionary Algorithms

and produces offspring population throughout the evolution. The concept is survival of the fittest, where the more fit genes survive to the next iteration, and are chosen to produce the offspring population. In order to also incorporate randomness in the population, some genes are randomly mutated, and may provide to be even more fitter than their original version.

Some other examples of evolutionary algorithms are Particle Swarm Optimization (PSO) [8], Grey Wolf Optimization (GWO) [9], and Differential Evolution (DE) [10].

1.4 Research Motivation

This thesis was motivated while studying various optimization algorithms for complex problems. There is a plethora of options available when it comes to evolutionary algorithms for optimization problems. But most of the them are inflexible and problem-specific. After reviewing various algorithms, we realized that grey wolf optimization (GWO) algorithm [9] has a lot of potential due to its simplicity and initial results. Unlike other various other algorithms, where an individual is assigned various other attributes, in GWO only the position and fitness is calculated for each individual. We realized the defects discovered in GWO [11] can be improved upon by improvising the algorithm. After studying various surveys done on ensemble strategies, we were inspired to employ the use of a multi-population strategy to improve the performance of our algorithm. Thus, we inspire our multi-population structure from the hierarchical structure of the grey wolves. Our initial results on single-objective problems showed promising results, which led us to implement this model on multi-objective problems, and provide better performance when compared to the state of the art models.

1.5 Hypothesis

The hypothesis for this thesis is - If we incorporate a multi-population ensemble to the traditional grey wolf optimization algorithm then we expect to see a better performance in terms of both convergence and time, as seen in the literature review. Additionally, by introducing new components, namely - elite group and mutation operator, we expect to reform the defects, studied in [11, 12].

This will be measured by comparing results on standard benchmark test suites for single and bi-objective problems, that are provided by the IEEE's Congress of Evolutionary Computation (CEC); for single-objective problem we will use CEC 2013 benchmark for real parameter optimization [1], for multi-objective problem we will use CEC 2009 benchmark for unconstrained bi-objective problems [2]. The comparison for single-objective problem will be done by comparing the fitness of best individual achieved from each algorithm, and with some statistical tests will be performed to support the claim. In case of multi-objective problem, we compare the obtained Pareto fronts, graphically and statistically using two metrics, GD and hypervolume.

1.6 Thesis Contribution

This thesis makes the following contributions:-

1. The main contribution is co-evolving dynamic multi-population framework for

single objective and bi-objective optimization problems, inspired by the hierarchical structure of grey wolves in [9, 13]. The structure is analogously adopted in our populations and allows dynamic leader selection in every generation. To the best of our knowledge, this is the first multi-population co-evolving algorithm that mimics the grey wolves' pack and hunting behaviour.

2. Additionally, owing to the co-evolving and dynamic nature, the proposed framework also aims to overcome the problem of getting stuck in local optima [12], and low convergence for non-zero optima [11].

3. We also introduce two components to our novel local search algorithm - namely elite group and mutation operator. These components are introduced in order to work on the defects discussed, and further improve the performance of the local search algorithm.

4. Finally, this thesis compares and studies the comparison done between various state of the arts using IEEE's CEC Benchmark for unconstrained single and bi-objective problems.

1.7 Thesis Outline

The chapters of this thesis is organized in the following manner:

Chapter 1 describes the background information, motivation, and contributions of the research work.

Chapter 2 describes the literature survey in the field of EA inspired from grey wolves, and EA that have a multi-population structure.

Chapter 3 explains the proposed framework for single objective problems, briefly lists functions in the benchmark (CEC 2013 test suite) used, and the results obtained.

Chapter 4 explains the proposed framework for multi objective problems, information about the CEC 2009 test suite of unconstrained bi-objective problems with the actual Pareto fronts, and the results achieved on the same.

Chapter 5 discusses the results obtained, and provides a discussion of the complexity and limitations of the framework.

Chapter 6 concludes the thesis and provides the future work.

Chapter 2

Background Study

This chapter comprises related work that has inspired us to build this framework, design the architecture, and propose it. Multi-population strategy can be divided into two major types - Cooperative and competitive [5].

In co-operative multi-population, the populations co-evolve with a mutual flow of information. The populations are treated equally, irrespective of their performance, and are provided with the same types of operators. The populations usually have the same size and can be run in parallel, thus reducing these algorithms' time complexity. In these algorithms, the flow of information is necessary to maintain both diversity and convergence. The information can be exchanged in various ways, such as migrating individuals and dynamic merging and regrouping.

On the other hand, in a competitive multi-population strategy, the populations compete for more resources. In one case, the populations start with equal population size and depending on the strategy's performance followed by each population. The size is decreased/increased. In the other case, each operator/strategy is assigned a small-sized indicator population and a large-size reward population. By judging operators' performance in the indicator population, the reward population is assigned to different operators.

Recent studies that implement a multi-population architecture prove that multi-population architecture improves EA's performance significantly due to an increase in exploration ability and convergence rate [16**?**].

The remaining section classifies the research based on the number of objectives in the optimization problem- single objective and multi-objective.

2.1 Single-Objective Optimization

In [14], a memory-based multi-population genetic learning model is proposed to find the shortest path in a dynamic graph. The strategy used to maintain multi-population is dynamic in nature. A density-based top-down clustering is used to divide the population into various sub-populations automatically. The number of sup-population is also dynamic, which depends on the number of current peaks in the population. The approach consists of two factors, the solution with high fitness value should be a cluster center, and two cluster centers should be far from each other.

In [15], in which a heritage-dynamic cultural algorithm is proposed to solve single-objective optimization functions. It employs a novel approach in maintaining the multi-population. The term "Heritage" is introduced, which implies that an individual has weighted connections with one or multiple sub-populations. Thus, the term "Multi-Population" is dropped, as implied in the term "Heritage". The influence of heritage is passed down from the predecessor individuals to their children in an additive manner. This allows an individual to belong to multiple populations at once.

In [16], a multi-swarm cooperative particle swarm optimization (MCPSO) is proposed by the authors, in which slave swarms co-evolve and return their best individuals. A new term is added in the velocity update equation of the master swarm, which follows the best individual returned from the swarm populations. This research proposes two models - collaborative and competitive; according to the results reported, the competitive model performed better than the collaborative.

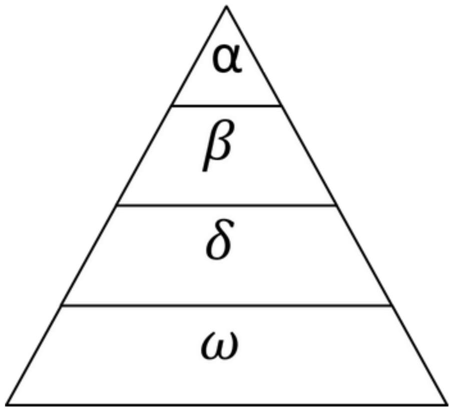


Figure 2.1: Hierarchical Structure of Grey Wolves

2.1.1 Grey Wolf Optimization

Like various other swarms and biological evaluations, grey wolves (Canis lupus) and their packs have been the source of inspiration of a subset of evolutionary algorithms. These algorithms are mainly based on the hunting mechanism and the hierarchical structure of grey wolves.

In a pack of grey wolves, there are 4 types of individuals - alpha α, beta β, delta δ and omega ω [9, 13]. An alpha wolf is considered as the leader of the pack, responsible for main decision making, and the first one to prey. The beta wolf is a subsidiary to the alpha, and assists it in making the decisions. The lowest ranking wolves are called omegas. And, the wolf that does not fit in any category is considered as delta wolf. In nature, they are usually the elders, scouts or sentinels. Figure 2.1 shows the hierarchical structure of these wolves.

Additional to the hierarchical structure, the hunting mechanism of grey wolves is also important, which can be divided into three major steps [13] -

1. Approaching: In this step, the pack covers a lot of ground and explores the area. This can be considered as an exploration phase in the analogy of EA.
2. Encircling: This is the second step in which many pack members, and in some cases, all, surround the prey, trapping it so that the wolves can harass it.

3. Attacking: The last step of the hunting mechanism is attacking the prey. After encircling, the wolves move closer towards the prey, still surrounding the prey, and ultimately finishing it. This step is analogous to final convergence in EA.

S. Mirjalili et al. first proposed Grey Wolf Optimization (GWO) algorithm in 2014 [9] for single-objective problems. As shown in Figure 2.2, the authors proposed a model which states that the fittest solution for a given problem be named as alpha (α). The second and third best solutions be called as beta (β) and delta (δ), respectively. The rest of the population is collectively called the omega (ω) wolves. The pack is led by α , β , δ and the ω wolves will follow them. In fact, during the evolution process, the updating of the wolves is done in such a way that they follow the positions of the most fit three wolves name- α , β, and δ.

After selecting the top 3 wolves based on fitness value: α , β, and δ, the position of the population is updated in such a way that they follow these leaders.

Consider, t as the current iteration, $\vec{X_p}$ as the position of prey, and $\vec{X}$ as the position of a grey wolf. Therefore, the displacement vector $\vec{X}(t+1)$ for prey p is given as,

$$\vec{D_p} = |\vec{C}.\vec{X_p}(t) - \vec{X}(t)|$$
$$\vec{X}(t+1) = \vec{X_p}(t) - \vec{A}.\vec{D_p}$$

For every wolf in the population the new position is given by

$$\vec{X}(t+1) = \frac{\vec{X_1} + \vec{X_2} + \vec{X_3}}{3}$$

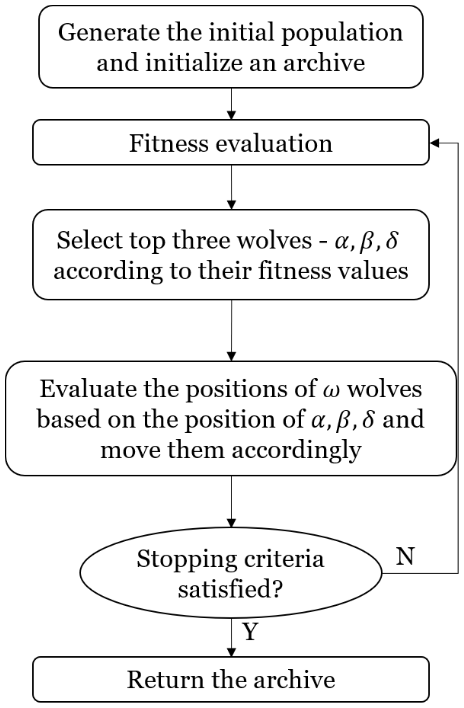

Generate the initial population and initialize an archive
Fitness evaluation
Select top three wolves - α, β, δ according to their fitness values
Evaluate the positions of ω wolves based on the position of α, β, δ and move them accordingly
Stopping criteria satisfied?
N
Y
Return the archive

FIGURE 2.2: Classic GWO architecture

where, $\vec{X}_1, \vec{X}_2.\vec{X}_3$ are the displacement vectors for $p = \alpha, \beta, \delta$ respectively.

$$\vec{D}_\alpha = |\vec{C}_1.\vec{X}_\alpha - \vec{X}|, \vec{D}_\beta = |\vec{C}_1.\vec{X}_\beta - \vec{X}|, \vec{D}_\delta = |\vec{C}_1.\vec{X}_\delta - \vec{X}|$$
$$\vec{X}_1 = \vec{X}_\alpha - \vec{A}.\vec{D}_\alpha, \vec{X}_2 = \vec{X}_\beta - \vec{A}.\vec{D}_\beta, \vec{X}_3 = \vec{X}_\delta - \vec{A}.\vec{D}_\delta$$

Other studies address the use of GWO in other scientific fields. In [17], the original GWO is modified so that it becomes a discrete process instead of continuous. By making it a discrete process, the algorithm can be used in feature selection, where the problem is to either choose a feature or not, which is a binary option. That is why such modifications of GWO are known as Binary Grey Wolf Optimization (BGWO). Various other fields such as community detection, Knapsack problems, and some special optimization problems require binary-operated algorithms. BGWO has been modified and adapted in various researches other than feature selection such as community detection [18], text classification [19], and knapsack [20].

It has been discovered that GWO has a defect in which when the optimum solution is non-zero, the population tends to have a significant drop in convergence rate [11]. This may be because the variable that migrates the individuals during the evolution converges to zero, but when the case is of a non-zero optimum point, it explores randomly. This might also be the cases with other optimization algorithms [11].

Furthermore, like other evolutionary algorithms, GWO also has a problem of getting stuck in the local optima [12]. Additionally, to the best of our knowledge, no work introduces a multi-population strategy inspired by the grey wolves. We believe that by incorporating co-evolving multi-population, we can tackle the defects above and improve the exploration and exploitation abilities during the search process.

2.1.2 Evaluation Metric for Single-Objective Optimization

The metric for single-objective optimization to compare two algorithm's performance is the best fitness value achieved. Since there is only one objective, we can directly compare the best individuals' values achieved from various algorithms and see which is better.

To incorporate the randomness of the evolutionary algorithm, the algorithm is run for various independent runs, and then the average of the best fitness is considered. For example, each algorithm can be run for thirty independent runs, and then the average of best values can be compared.

Non-Parametric Tests

Non-parametric tests are conducted to compare two distributions, which may not be presumed as normal. To better understand the distributions and compare if they are derived from the same sample or not, we can conduct various tests. Some of these tests are introduced below:

1. Friedman's Test - Proposed by Milton Friedman [21], this test is a pre-hoc analy-

sis used to detect differences in variables. If the null hypothesis is rejected, then it means that there exists at least one treatment that is significantly different from the rest.

2. Wilcoxon's Test - This is a posthoc analysis that performs a pair-wise comparison of different distributions. It is used to measure the mean rank between two populations and attempts to prove that the two populations are samples of the same distribution.

3. Kruskal-Walli's test - This pre-hoc test is usually used for consistency, and its null hypothesis states that the variables come from the same distribution. This test is also based on the rank system but can take more than two groups at once.

2.2 Multi-Objective Optimization

For multi-objective problems, the state of the art methods considered are non-dominated sorted algorithm II (NSGA-II) [22], and multi-objective evolutionary algorithm based on decomposition (MOEAD) [23]. NSGA-II is an upgraded version of NSGA [24], in which the population is repeatedly sorted based on their Pareto dominance, and ranks of obtained fronts are generated. The top individuals are considered based on this sorting. These individuals generate an offspring population using the genetic algorithm operators such as crossover, mutation, and tournament. Another version of this model is proposed as NSGA-III [25], which is for many-objective problems precisely, and out of the scope of this thesis.

MOEAD is based on the decomposition of a problem into small problems and optimizing these sub-problems simultaneously. Each sub-problem is optimized by using the neighbouring subproblems' information only and has shown to have a lower complexity compared to NSGA-II. This has motivated the use of decomposition in optimization and the application of this algorithm in many areas. The problem with

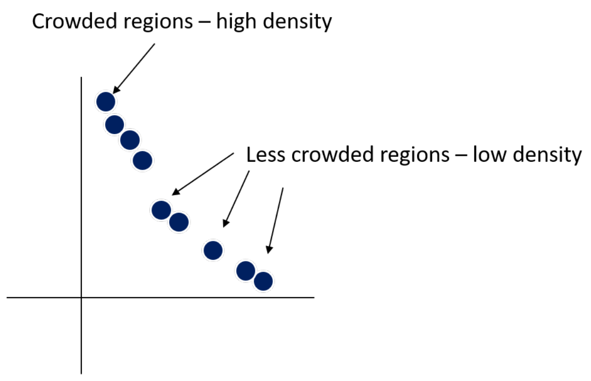

Crowded regions – high density
Less crowded regions – low density

Figure 2.3: Density of archive members decides probability for leader selection and removal when archive overfills

MOEAD is that it requires knowledge of the problem that needs to be optimized to divide it into subproblems in a better way. Since each sub-problem needs to be assigned a weight, these weights need to be initialized efficiently to get better results.

There are some variants of particle swarm optimization (PSO) as well that are used in multi-objective optimization. Multi-objective particle swarm optimization (MOPSO) was first proposed in 2002 [26], which uses an adaptive archive, and a leader selection based on cell density of archive members, as illustrated in figure 2.3. In 2019, a multi-objective version of PSO was proposed using a co-evolving multi-population structure [27]. Two different models are proposed, competitive and collaborative model. The master population may or may not consider the best individual from the slave population in the competitive model but always considers it in the collaborative model.

A multi-objective model for GWO (MOGWO) was first posed in [28]. In this paper, the authors added an archive to store all the non-dominated Pareto optimal solutions to make the algorithm multi-objective. The authors also introduced the concept of leader selection in the archive to select alpha, beta, and delta leaders in the archive. The concept of leader selection is based on roulette wheel selection. The probability of selection is inversely proportional to the cell density in which a candidate belongs; MOPSO inspires this. This way, the leaders from sparsely dense cells are selected, allowing more exploration of the population. The archive used in

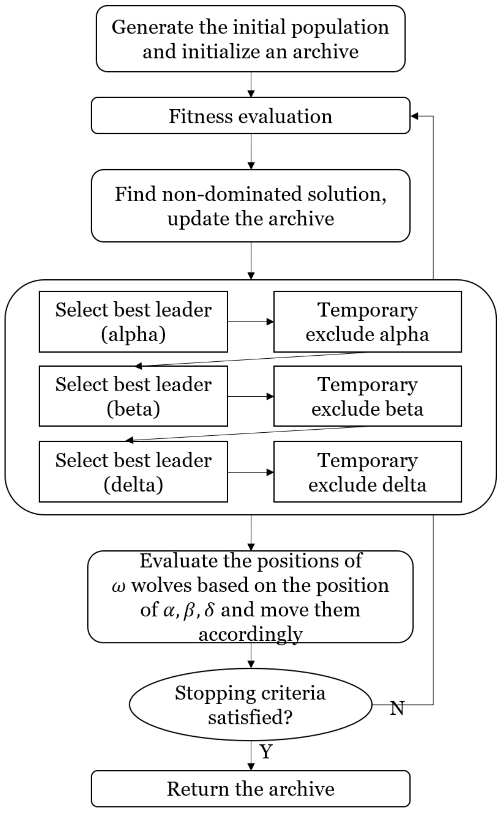
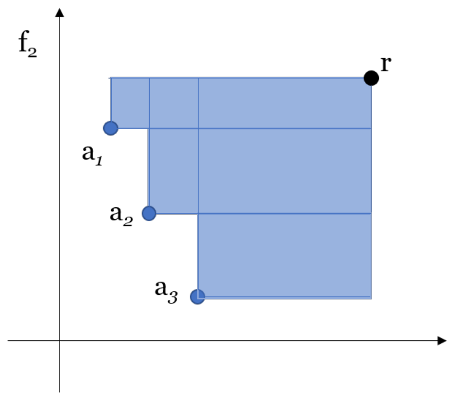

Generate the initial population and initialize an archive
Fitness evaluation
Find non-dominated solution, update the archive
Select best leader (alpha)
Temporary exclude alpha
Select best leader (beta)
Temporary exclude beta
Select best leader (delta)
Temporary exclude delta
Evaluate the positions of ω wolves based on the position of α, β, δ and move them accordingly
Stopping criteria satisfied?
N
Y
Return the archive

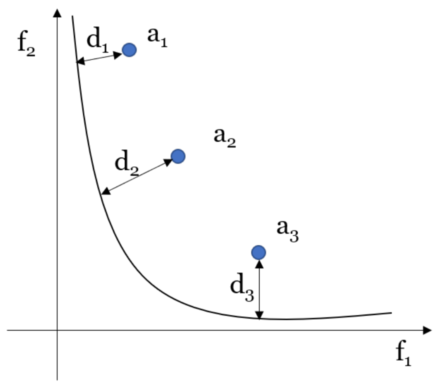
Figure 2.4: Classic MOGWO architecture

this research is inspired by the adaptive archive proposed in multi-objective particle swarm optimization (MOPSO) [26]. The architecture of the traditional MOGWO is illustrated in figure 2.4.

This work has been further modified in the literature and applied in various fields. Hybrid MOGWO proposed in [29] modifies individuals' encoding process and introduces a crossover and mutation operator. This model is specifically designed to solve dynamic scheduling in the welding industry. In [30], a bi-objective problem of wind energy conversion is optimized by finding an optimal operating point of a fuzzy controller to maximize the power output and alleviate the loads. Another research done in [31] uses a modified version of MOGWO to optimize a hybrid artificial neural network used to predict the strength of silica fume concrete.



(A) Hypervolume



(B) Generational Distance

FIGURE 2.5: Performance Indicators

2.2.1 Evaluation Metrics for Multi-Objective Optimization

Unlike single-objective optimization, where the best individual's fitness can compare the performance, multi-objective optimization problems have various metrics available to compare the obtained Pareto fronts (set of non-dominated solutions). We will be introducing some of these metrics that are in the scope of this thesis.

Generation Distance

Generational Distance (GD) [32] is one of the most used evaluation metrics that calculate the mean distance between an obtained Pareto front to the actual Pareto front. This metric requires knowledge about the optimal Pareto front in order to compare the efficiency of an algorithm.

Consider a set of points obtained by an algorithm as $A = \{a_1, a_2, \ldots, a_n\}$ for a problem with optimal Pareto front $Z = \{z_1, z_2, \ldots, z_n\}$, then GD is calculated by:

$$GD(A) = \frac{1}{|A|}(\sum_{i=1}^{|A|} d_i^2)^{1/2} \tag{2.1}$$

where d_i is the distance between point a_i and its nearest point in Z. GD is illustrated in figure 2.5a

Hypervolume

Hypervolume [33] is another performance indicator that calculates the volume/area covered by the obtained Pareto front with respect to a reference point. Unlike GD, Hypervolume does not require knowledge about the optimal Pareto front, But due to its complexity in the calculation, GD requires more time. Additionally, the answer depends on the position of the reference point. The reason to use Hypervolume is that it tells two things about a Pareto front, the spread and the closeness to the optimal regions.

Though it does not directly compare the distance between the obtained Pareto front and the actual Pareto front, it tells how farther a front is from the reference point. The greater the area covered, the farther it is from the reference point. Another note to be made for Hypervolume is that the reference point chosen must be greater than the maximum values observed in each objective. For example, if the worse value for $f1$ is 5, and for $f2$ is 7, then the chosen reference point r must be (x, y) where x $\geq$ 5 and y $\leq$ 7. The values of hypervolume change when the reference point is moved. There is thus much study on choosing the best possible reference point, which is slightly worse than the nadir point [34].

For minimization problems, greater Hypervolume means better performance. Figure 2.5b illustrates hypervolume and its calculation. In order to perform a multi-dimensional comparison, we consider both Hypervolume and GD for comparison.

Chapter 3

Proposed Model for Single Objective Optimization

This chapter describes the model proposed for single-objective problem in detail, the single objective benchmark functions used, the comparisons done with the state of the art algorithms and the traditional GWO for single objective problems.

3.1 Model Architecture

Generally, as shown in Figure. 3.1, our proposed approach can be divided into five main steps.

1. Initialization: The population consists of n individuals which is initialized in a randomly uniform manner in the solution space. Assume $X_t = \{I_1, ..., I_n\}$ represents a population at the iteration t with n individuals, I_i, where $1 <= i <= n$. Each individual represents a potential solution for a given problem. Hence, the initial population can be defined as X_0 with n random individuals. We also initialize three variables $\vec{A}$, $\vec{C}$, and a parameter, which are be used to migrate the individuals in

GWO as described in [9].

$$\vec{A} = 2a.\vec{r_1} - a \quad \text{and} \quad \vec{C} = 2\vec{r_2} \tag{3.1}$$

value of a is decreased linearly from 2 to 0 over the course of iterations and $\vec{r_1}$ and $\vec{r_2}$ are random vectors between [0, 1].

$$a = 2 - i\frac{2}{N_t} \tag{3.2}$$

where i is the current iteration number and N_t is the total number of iteration.

2. Evaluation and Division: In the second step, the quality and suitability of the individuals in the population is evaluated using the objective function, $f(I)$, and are ranked accordingly. In case of minimization- if $f(I_x) < f(I_y)$ then I_x is a better than I_y.

The population is then divided into 4 fixed-size populations based on their fitness scores, namely Alpha population (A), Beta population (B), Delta population (Δ), and Omega population (Ω). Thus, $X_t = \mathrm{A}_t \cup \mathrm{B}_t \cup \Delta_t \cup \Omega_t$ and $\mathrm{A}_t \cap \mathrm{B}_t \cap \Delta_t \cap \Omega_t = \varnothing$

The initial population is divided in such a way that individuals in Alpha population are fitter than individuals in Beta population, which are fitter than individuals in Delta population, and so on. Alpha population is the group of individuals that have better performance, and hence play the lead role in the optimization process, Beta population is subordinate to the Alpha population and support them in decision making. Additionally, Omega population is non-dominant and gives in to other populations by following them, and Delta population includes the rest of the individuals that dominate Omega population but submit to Alpha and Beta populations.

As a brief example to illustrate the way the populations are divided, assume we have a population consisting of the following eight individuals of dimension 2.

$$X_1 = [-4, 1];\ X_2 = [-2, 1];\ X_3 = [0, 1];\ X_4 = [4, 2];$$

$$X_5 = [0, 0.9];\ X_6 = [1.5, 1];\ X_7 = [0, 3];\ X_8 = [2.5, 1.5]$$

Now, assume the objective function is a sphere function which is $x^2 + y^2 = 0$. After calculating the fitness values, the algorithm will sort them based on their performance. Since this is a problem of minimization, the lesser the value of the fitness score of an individual, the better it is.

Now, the initial population is divided into 4 populations. This is done by sorting them according to the fitness function and assigning them population such as the individuals in Alpha are better than the individuals in Beta, which are better than the individuals in Delta. The remaining individuals are put in Omega popuation. **Alpha Population**-{ X_3, X_5 } **Beta Population**-{ X_2, X_6 } **Delta Population**-{ X_7, X_8 } **Omega Population**-{ X_1, X_4 }.

Thus, this is how the populations are divided in the proposed framework. It is to be noted that in practice, each population needs to have at least 4 individuals, this is because in each population, there is a unique alpha, beta, delta individual; consequently the population follow these top 3 individual according to the GWO algorithm [9]. This process is known as 'evolving' the population.

3. Evolving Alpha, Beta, and Delta Populations: After forming the populations, the third step consists of running the local optimization process. We propose a modified grey wolf algorithm, which is run independently and concurrently on each of the Alpha, Beta and Delta populations. Consequently, three best individuals will be produced at the end of this step. We call the best one global alpha, the second one global beta and the last one global delta. These individuals will form the elite group and act as new global leaders for the Omega's search process. In fact, to mitigate the

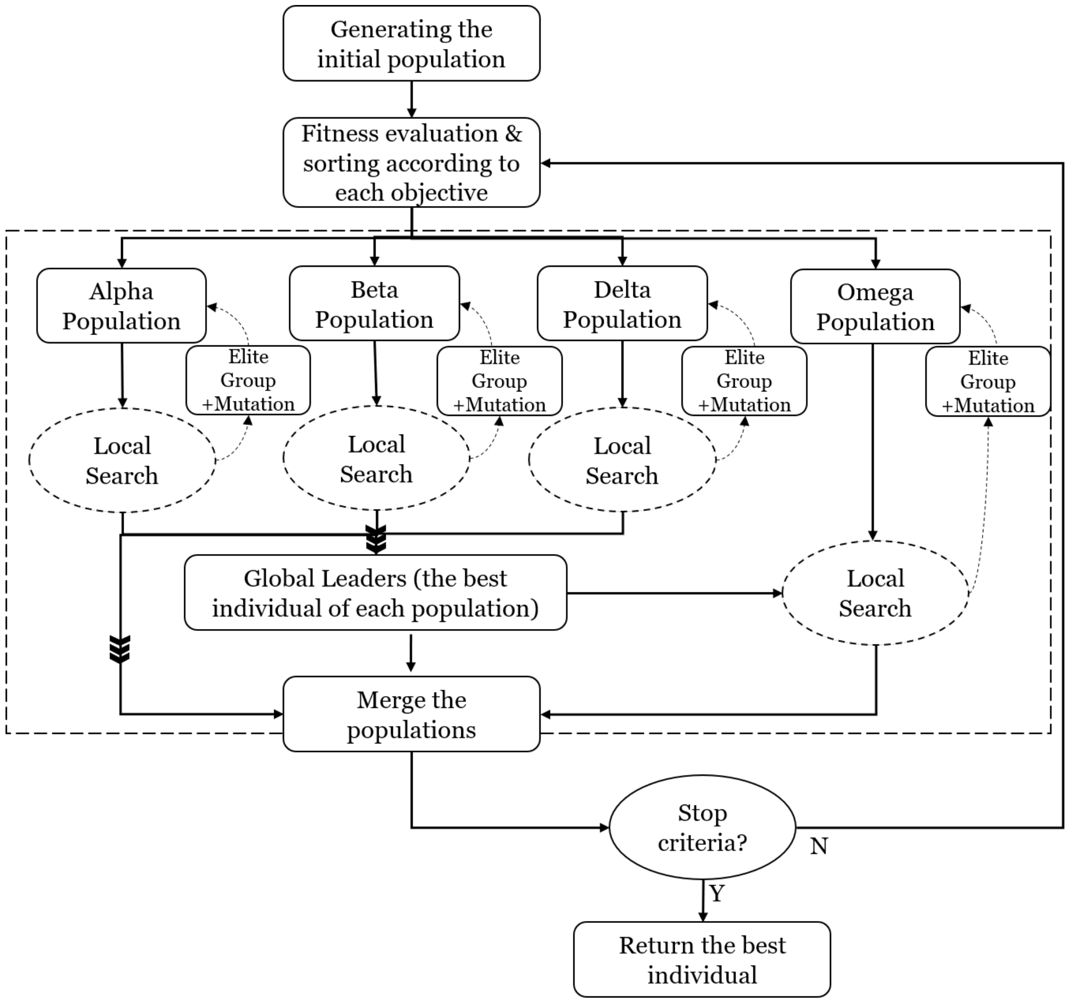


FIGURE 3.1: Proposed Framework Architecture for Single Objective Problems

effects of the defect discussed in the Section 2, and avoid the population exploring away from the optima points, we use an elite group of individuals, which records individuals that had best accuracy in previous iteration.

In other words, guide leaders are basically the best individuals from each population- Alpha, Beta, and Delta populations. Since we apply modified GWO in these populations, it is possible that the best individual from alpha may not be as fit as the best individual from the beta population. Hence, these leaders will be sorted according to their fitness scores, and then are labelled as global alpha, global beta, and global delta.

4. **Evolving Omega Population:** In this step, a modified GWO is run in the

Algorithm 1: Proposed Framework for Single Objective Optimization

```
Input  : $n$ : The total number of individuals in a population
         $N_t$ : Max. Number of Iteration
         $m$: the portion/size of each population
Output: $I_\alpha$ : a near-optimal solution
         $f(I_\alpha)$ : the best fitness value
1  Initializing the population(n) randomly;
2  Elite ← {Ø} ;
3  i ← 1 ;
4  while $i <= N_t$ do
5      Sort the population based on the objective function;
6      Divide the initial population into 4 populations where top m% individuals
         belong to Alpha population, next m% to Beta population and so on..;
7      Run Algorithm 2 in all populations (Alpha, Beta and Delta populations);
8      Find the best solutions of each population and add them to the Leaders
         set;
9      Sort the leaders to obtain Global Alpha, Global Beta, Global Delta;
10     Run Algorithm 2 in Omega population with the guide leaders;
11     Elite ← alpha of each population;
12     population ← population ∪ Elite;
13     for all individuals do
14         if rand <= 20% then
15             Perform random mutation;
16         end
17     end
18     i++;
19 end
20 Return the Global alpha and its fitness value;
```

Omega population which uses the global alpha, global beta, and global delta as the leaders to guide the search direction, hence maintaining the hierarchical structure. As a result of this process the best individual from the Omega population is generated and will be added to the elite group.

5. Merging and Enhancing Population: In the final step, the populations are merged together into one global population, which is then mutated based on a random probabilistic mutation operator. The output will be then send again to the fitness evaluation phase and the cycle continues until the stop condition is met where the best produced individual will be returned as the output of the algorithm. The reason to use the mutation operator here, is to escape from local optima during the search process. In fact, we can interpret it in a way that, there exists some individuals in each population which are not following the leaders completely.

The algorithm for our proposed Multi-Population Grey Wolf Optimization, Algorithm 1, is a type of cooperative multi-population ensemble with information flow using elite member, and periodic dynamic regrouping. The elite member consists of the best individuals from each population - Alpha, Beta, Delta, and Omega.

3.1.1 Local Search Algorithm

In our thesis, we propose a search algorithm that performs a single-objective optimization in a local population. This algorithm, as mentioned earlier, is inspired from the hunting patterns seen in grey wolves. Additionally, to mitigate the effects of the defects seen in GWO [11], an elite group and a mutation operator are incorporated in the algorithm as well.

Furthermore, to allow a mutual information flow between population, knowledge from other population is used to enhance the search process. This is done by introducing an optional parameter in our local search algorithm that takes guide leaders, which are the leaders from the other populations. If the fitness of the guide leaders is more than the best fitness in the population, then these leaders steer the search process of the population.

This local search algorithm can be seen in 2. The mutation operator, as shown, is a probabilistic operator that mutates a random dimension of mp percentage of the population. The elite group consists of the best individual from the last iteration, this is done in order to have a constant convergence of our algorithm, since GWO converges away in case of non-zero optima [11].

Algorithm 2: The Proposed Local Search for Single Objective Optimization

```
Input  : $Population$ of size n
         $N_t$ : Max Number of iteration
         $\vec{X}_\alpha$, $\vec{X}_\beta$, and $\vec{X}_\delta$ Guide leaders (optional input)
Output: The last evolved set of population &
         $X_\alpha$ : The best found optimal solution &
         $f(x_\alpha)$ : The best fitness value
1  Initialize $a$, $A$, and $C$ using Equations. 3.1, and 3.2;
2  if guide leaders are given then
3      add them into the population;
4      sort the population;
5  end
6  Find the $\alpha$, $\beta$, and $\delta$ individuals based on their fitness value;
7  Elite ← {Ø};
8  i ← 1;
9  while $i <= N_t$ do
10     foreach individual ∈ Population do
11         Update individuals' Position using GWO equations;
12     end
13     Update $a$, $A$, and $C$ using Equations 3.1, and 3.2;
14     Evaluate fitness value of individuals;
15     Update new $\alpha$, $\beta$, and $\delta$;
16     Elite ← $\alpha$;
17     population ← population ∪ Elite;
18     for all individuals do
19         if rand <= 20% then
20             Perform random mutation;
21         end
22     end
23     i++;
24 end
25 Return the population set, $\alpha$, and its fitness value
```

3.2 Evaluation

This section describes the evaluation done to evaluate the single-objective proposed algorithm. The section is divided into three parts - comparison with state of the arts, performance of proposed algorithm at different dimensions, and non-parameteric statistical tests.

All experiments in this project have been conducted in the following environmental setup and parameters:

- Windows 10, Core i5 9th Gen, GTX 1650, 8GB RAM
- Python 3.5.0
- IDE Used: PyCharm Professional 2019.3
- Size of the population - 100 individuals
- Number of iterations in the proposed algorithm- 25 outer iterations and 15 local iterations
- Number of iterations for other state-of-the-art methods - 100
- Number of independent runs - 30

All the comparisons are made based on the average and standard deviation of the results obtained after the 30 independent runs.

For the standardized testing environment, test functions from the CEC 2013 test suite of 'Single Objective Real-Parameter Numerical Optimization Problems' [1] were used. This test suite compromises of mathematical functions that have D input dimensions and a single objective function. The test consists of three categories of minimization functions: "Unimodal," "Multimodal Function", and "Composite Functions". The bounds of search space are [-100, 100]^D where D is the dimension. For this test suite, there is a shifted global optima denoted by $o = [o_1, \ldots, o_D]^T$.

$M_1, M_2, \ldots, M_10$ are rotational matrix that are provided with the test suite. Λ^{α} is the diagonal matrix with $\lambda_{ii} = \alpha^{\frac{i-1}{2(D-1)}}$. T^{β}_{asy} and T_{osz} are constants which are provided by IEEE. The functions are summarized in Table 3.1

	No.	Functions	Minima (Fi*)
Unimodal Functions	1	Sphere Function	-1400
	2	Rotated High Conditioned Elliptic Function	-1300
	3	Rotated Bent Cigar Function	-1200
	4	Rotated Discus Function	-1100
	5	Different Powers Function	-1000
Basic Multimodal Functions	6	Rotated Rosenbrock's Function	-900
	7	Rotated Schaffers F7 Function	-800
	8	Rotated Ackley's Function	-700
	9	Rotated Weierstrass Function	-600
	10	Rotated Griewank's Function	-500
	11	Rastrigin's Function	-400
	12	Rotated Rastrigin's Function	-300
	13	Non-Continuous Rotated Rastrigin's Function	-200
	14	Schwefel's Function	-100
	15	Rotated Schwefel's Function	100
	16	Rotated Katsuura Function	200
	17	Lunacek Bi_Rastrigin Function	300
	18	Rotated Lunacek Bi_Rastrigin Function	400
	19	Expanded Griewank's plus Rosenbrock's Function	500
	20	Expanded Scaffer's F6 Function	600
Composite Functions	21	Composition Function 1 (n=5, Rotated)	700
	22	Composition Function 2 (n=3, Unrotated)	800
	23	Composition Function 3 (n=3, Rotated)	900
	24	Composition Function 4 (n=3, Rotated)	1000
	25	Composition Function 5 (n=3, Rotated)	1100
	26	Composition Function 6 (n=5, Rotated)	1200
	27	Composition Function 7 (n=5, Rotated)	1300
	28	Composition Function 8 (n=5, Rotated)	1400

TABLE 3.1: Summary of the 28 IEEE CEC 2013 Benchmark Functions for single-objective problems [1]

The following are the functions used:

5 UniModal Functions

Sphere Function

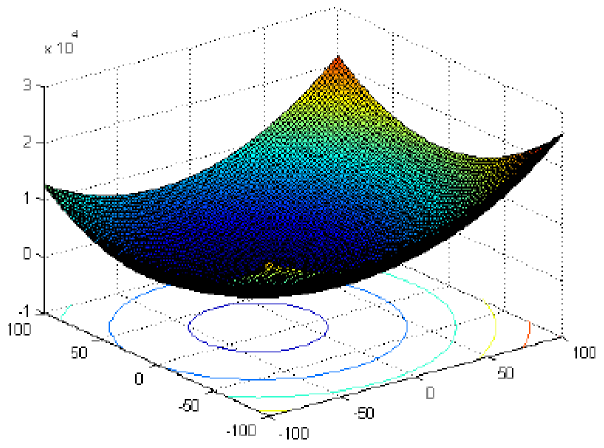


FIGURE 3.2: 3-D Map for UF-1 [1]

$f_1(x) = \sum_{i=1}^{D} z_i^2 + f_1*, z = x - o$

Rotated High Conditioned Elliptic

$f_2(x) = \sum_{i=1}^{D} (10^6)^{\frac{i-1}{D-1}} z_i^2 + f_2*, z = x - o$

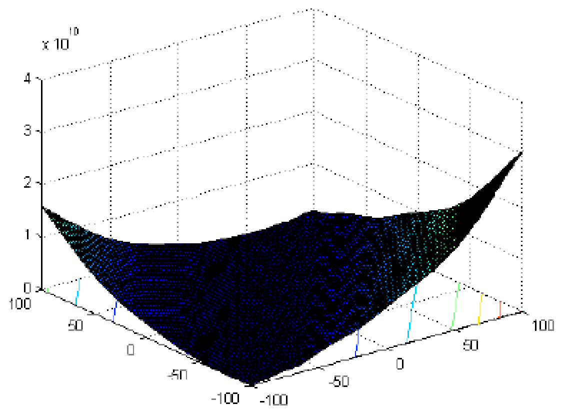


FIGURE 3.3: 3-D Map for UF-2 [1]

Rotated Bent Cigar Function

$f_3(x) = z_1^2 + 10^6 \sum_{i=2}^{D} z_i^2 + f_3* \ z = M_2 T_{azy}^{0.5}(M_1(x - o))$

Rotated Discus Function

$f_4(x) = 10^6 z_1^2 + \sum_{i=2}^{D} z_i^2 + f_4*, z = T_{osz}(M_1(x - o))$

Different Power Function

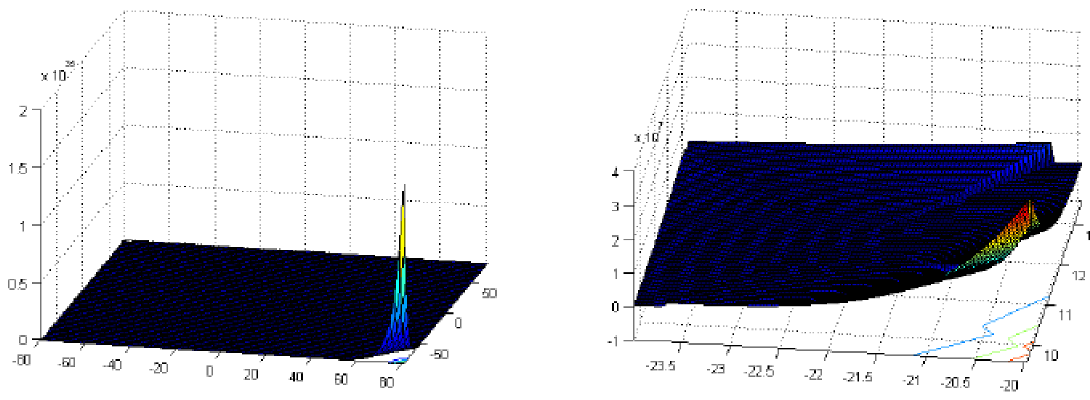


FIGURE 3.4: 3-D Map for UF-3 [1]

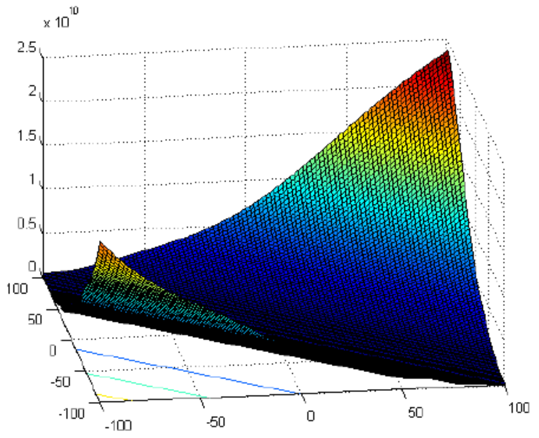


FIGURE 3.5: 3-D Map for UF-4 [1]

$$f_5(x) = \sqrt{\sum_{i=1}^{D} |z_i|^{2+4\frac{i-1}{D-1}}} + f_5*, z = (x - o)$$

15 Basic Multimodal Functions

Rotated Rosenbrock's Function

$$f_6(x) = \sum_{i=1}^{D-1} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right) + f_6*$$

$$z = \mathbf{M}_1\left(\frac{2.048(x-0)}{100}\right) + 1$$

Rotated Schaffers F7 Function

$$f_7(\mathbf{x}) = \left(\frac{1}{D-1}\sum_{i=1}^{D-1}\left(\sqrt{z_i} + \sqrt{z_i}\sin^2\left(50 z_i^{0.2}\right)\right)\right)^2 + f_7^*$$

$$z_i = \sqrt{y_i^2 + y_{i+1}^2} \text{ for } i = 1, \ldots, D, y = \Lambda^{10}\mathbf{M}_2 T_{asy}^{0.5}\left(\mathbf{M}_1(x - o)\right)$$

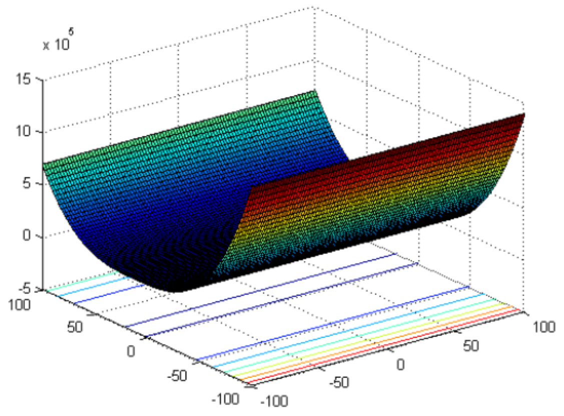


FIGURE 3.6: 3-D Map for UF-5 [1]

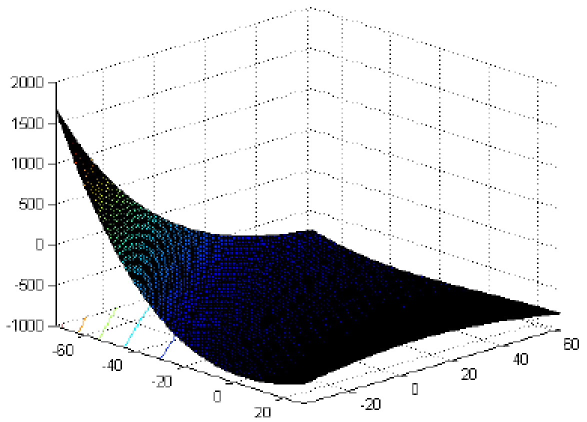


FIGURE 3.7: 3-D Map for UF-6 [1]

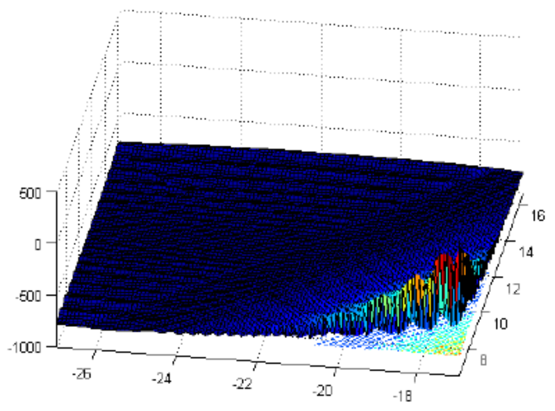


FIGURE 3.8: 3-D Map for UF-7 [1]

Rotated Ackley's Function

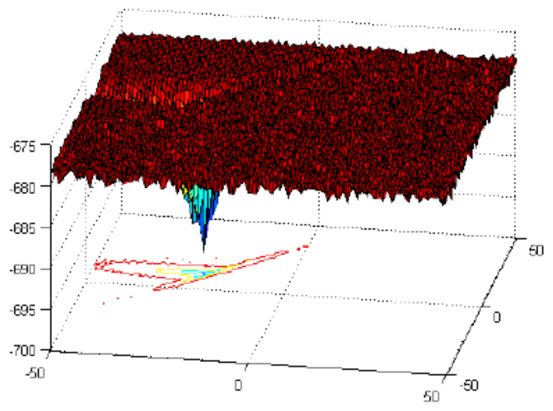


FIGURE 3.9: 3-D Map for UF-8 [1]

$$f_8(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} z_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi z_i\right)\right) + 20 + e + f_8*$$

$$z = \Lambda^{10} M_2 T_{asy}^{0.5}(M_1(x-o))$$

Rotated Weierstrass Function

$$f_9(x) = \sum_{i=1}^{D}\left(\sum_{k-0}^{k\max}\left[a^k\cos\left(2\pi b^k\left(z_i+0.5\right)\right)\right]\right) - D\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\cdot 0.5\right)\right] + f_9^*$$

$$\mathrm{a} = 0.5, \mathrm{b} = 3, \mathrm{kmax} = 20, \quad z = \Lambda^{10}\mathbf{M}_2 T_{ay}^{0.5}\left(\mathbf{M}_1\frac{0.5(x-o)}{100}\right)$$

Rotated Griewank's Function

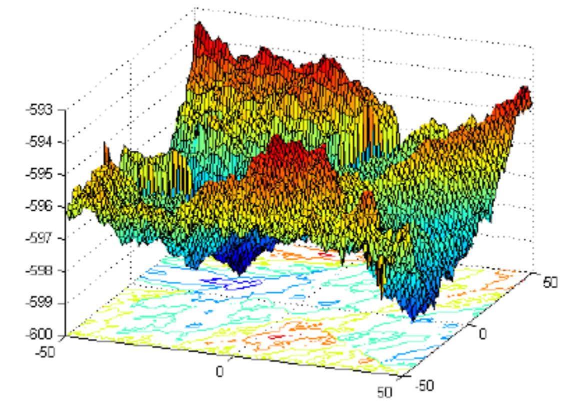


FIGURE 3.10: 3-D Map for UF-9 [1]

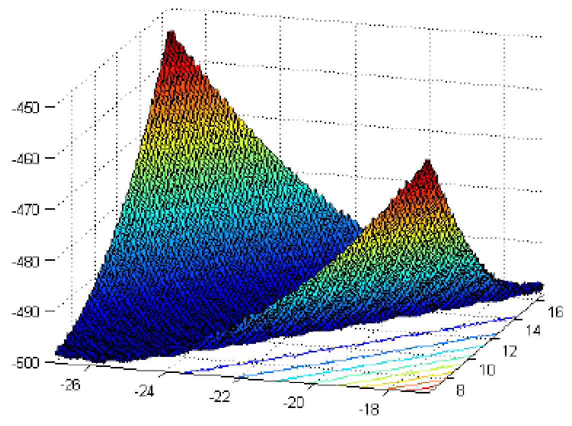


FIGURE 3.11: 3-D Map for UF-10 [1]

$$f_{10}(x) = \sum_{i=1}^{D} \frac{z_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{z_i}{\sqrt{i}}\right) + 1 + f_{10}*, \quad z = \Lambda^{100}\mathbf{M}_1 \frac{600(x-o)}{100}$$

Rastrigin's Function

$$f_{11}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right) + f_{11}* \ z = \Lambda^{10} T_{asy}^{0.2}\left(T_{osz}\left(\frac{5.12(x-o)}{100}\right)\right)$$

Rotated Rastrigin's Function

$$f_{12}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos\left(2\pi z_i\right) + 10\right) + f_{12}*$$

$$z = \mathbf{M}_1 \Lambda^{10} \mathbf{M}_2 T_{asy}^{0.2}\left(T_{osz}\left(\mathbf{M}_1 \frac{5.12(x-o)}{100}\right)\right)$$

Non-Continuous Rotated Rastrigin's Function

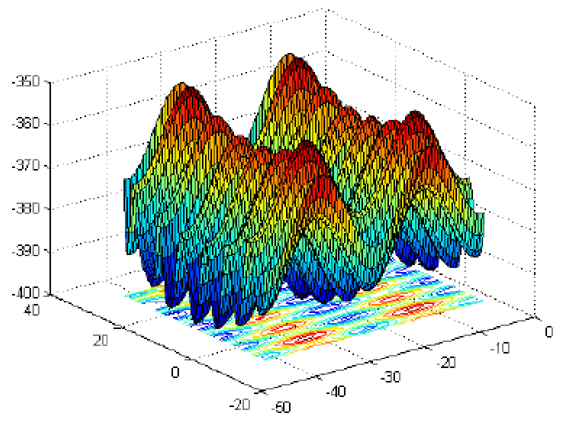


FIGURE 3.12: 3-D Map for UF-11 [1]

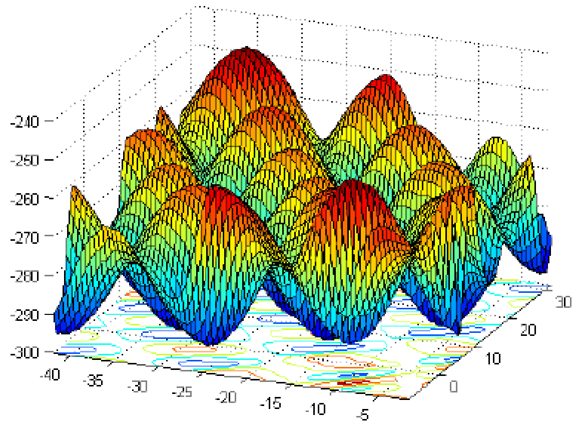


FIGURE 3.13: 3-D Map for UF-12 [1]

$$f_{13}(x) = \sum_{i=1}^{D} \left(z_i^2 - 10\cos(2\pi z_i) + 10\right) + f_{13}*$$

$$\widehat{x} = \mathbf{M}_1 \frac{5.12(x-o)}{100}, y_i = \begin{cases} \hat{x}_i & \text{if } |\hat{x}_i| \leq 0.5 \\ \text{round}(2\hat{x}_i)/2 & \text{if } |\hat{x}_i| > 0.5 \end{cases} \quad \text{for } i = 1, 2, \ldots, D$$

$$z = \mathbf{M}_1 \Lambda^{10} \mathbf{M}_2 T_{\text{asy}}^{0.2}\left(T_{\text{osz}}(y)\right)$$

Schwefel's Function

$$f_{14}(z) = 418.9829 \times D - \sum_{i=1}^{D} g(z_i) + f_{14}*$$

$$z = \Lambda^{10}\left(\frac{1000(x-o)}{100}\right) + 4.209687462275036\text{e} + 002$$

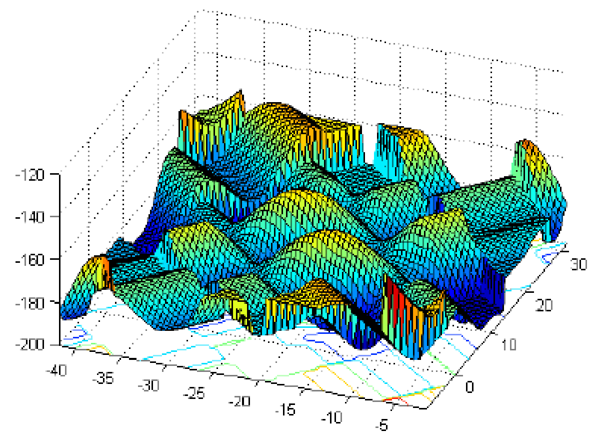


FIGURE 3.14: 3-D Map for UF-13 [1]

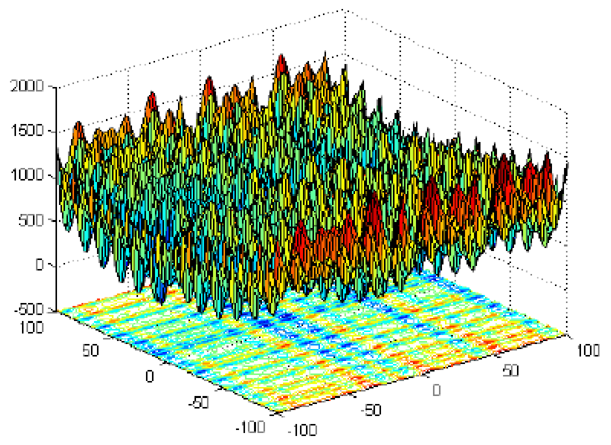


FIGURE 3.15: 3-D Map for UF-14 [1]

$$g\left(z_i\right)=\begin{cases} z_i \sin\left(\left|z_i\right|^{1/2}\right) & \text{if } \left|z_i\right|\leq 500 \\ \left(500-\bmod\left(z_i,500\right)\right)\sin\left(\sqrt{500-\bmod\left(z_i,500\right)\,|}\right)-\frac{\left(z_i-500\right)^2}{10000D} & \text{if } \left|z_i\right|>500 \\ \left(\bmod\left(\left|z_i\right|,500\right)-500\right)\sin\left(\sqrt{\bmod\left(\left|z_i\right|,500\right)-500\,|}\right)-\frac{\left(z_i+500\right)^2}{10000D} & \text{if } z_i<-500 \end{cases}$$

Rotated Schwefel's Function

$$f_{15}(z)=418.9829\times D-\sum_{i=1}^{D} g\left(z_i\right)+f_{15}*$$

$$z=\Lambda^{10}\mathbf{M}_1\left(\frac{1000(x-o)}{100}\right)+4.209687462275036\text{e}+002$$

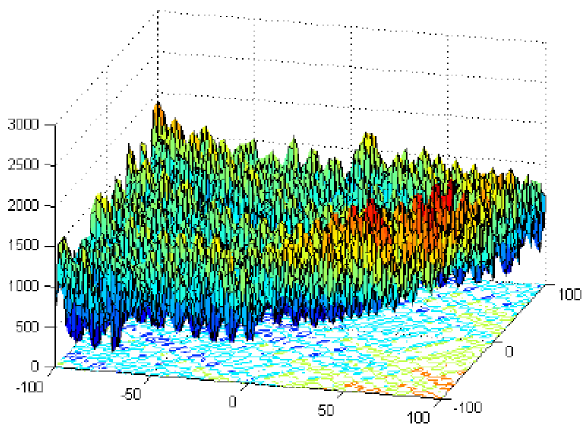


FIGURE 3.16: 3-D Map for UF-15 [1]

$$g(z_i) = \begin{cases} z_i \sin\left(|z_i|^{1/2}\right) & \text{if } |z_i| \le 500 \\ (500 - \text{mod}(z_i, 500)) \sin\left(\sqrt{500 - \text{mod}(z_i, 500)\ |}\right) + \frac{(z_i-500)^2}{10000D} & \text{if } |z_i| > 500 \\ (\text{mod}(|z_i|, 500) - 500) \sin\left(\sqrt{\text{mod}(|z_i|, 500) - 500\ |}\right) + \frac{(z_i+500)^2}{10000D} & \text{if } z_i < -500 \end{cases}$$

Rotated Katsuura Function

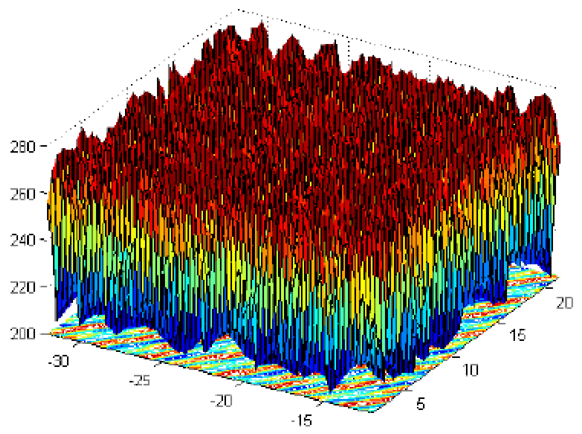


FIGURE 3.17: 3-D Map for UF-16 [1]

$$f_{16}(x) = \frac{10}{D^2} \prod_{i=1}^{D} \left(1 + i \sum_{j=1}^{32} \frac{\left|2^j z_i - \text{round}\left(2^j z_i\right)\right|}{2^j}\right)^{\frac{10}{D^{12}}} - \frac{10}{D^2} + f_{16}*$$

$$z = \mathbf{M}_2 \Lambda^{100} \left(\mathbf{M}_1 \frac{5(x-o)}{100}\right)\Big)$$

Lunacek Bi-Rastrigin Function

$$f_{17}(x) = \min\left(\sum_{i=1}^{D} (\widehat{x}_i - \mu_0)^2, dD + s \sum_{i=1}^{D} (|\ \hat{x}_i - \mu_1)^2\right) + 10\left(D - \sum_{i=1}^{D} \cos(2\pi \widehat{z}_i)\right) +$$

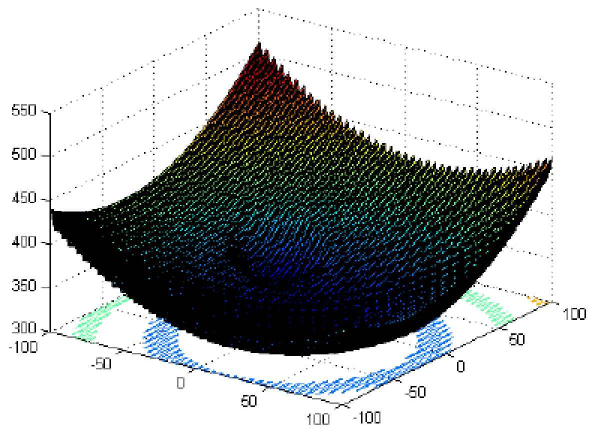


FIGURE 3.18: 3-D Map for UF-17 [1]

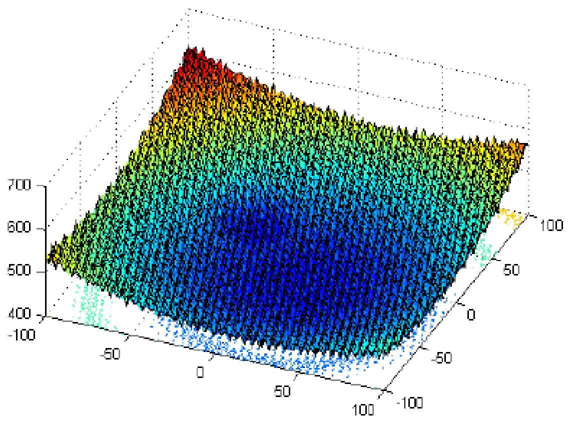


FIGURE 3.19: 3-D Map for UF-18 [1]

$f_{17}*$

$\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}, s = 1 - \frac{1}{2\sqrt{D+20}-8.2}, d = 1}$

$y = \frac{10(x-o)}{100}, \widehat{x}_i = 2\,\mathrm{sign}\left(x_i^*\right) y_i + \mu_0$, for $i = 1, 2, \ldots, D$

$z = \Lambda^{100}\left(\widehat{x} - \mu_0\right)$

Rotated Lunacek Bi-Rastrigin Function

$f_{18}(x) = \min\left(\sum_{i=1}^{D}\left(\hat{x}_i - \mu_0\right)^2, dD + s\sum_{i=1}^{D}\left(\widehat{x}_i - \mu_1\right)^2\right) + 10\left(D - \sum_{i=1}^{D}\cos\left(2\pi\varepsilon_i\right)\right) +$
$f_{18}*$

$\mu_0 = 2.5, \mu_1 = -\sqrt{\frac{\mu_0^2 - d}{s}, s} = 1 - \frac{1}{2\sqrt{D+20}-8.2}, d = 1$

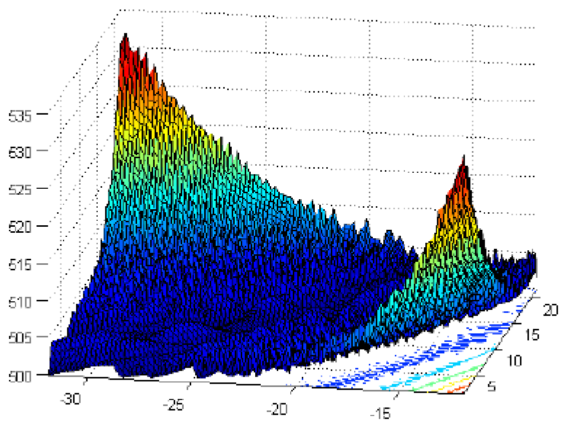


FIGURE 3.20: 3-D Map for UF-19 [1]

$y = \frac{10(x-o)}{100}, \hat{x}_i = 2\,\text{sign}\left(y_i^*\right) y_i + \mu_0$, for $i = 1, 2, \ldots, D, z = \mathbf{M}_2 \Lambda^{100} \left(\mathbf{M}_1 \left(\hat{x} - \mu_0\right)\right)$

Expanded Griewank's + Rosenbrock's Function

Basic Griewank's Function: $g_1(x) = \sum_{i=1}^{D} \frac{x_i^2}{4000} - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$

Basic Rosenbrock's Function:

$g_2(x) = \sum_{i=1}^{D-1} \left(100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right)$

$f_{19}(x) = g_1\left(g_2\left(z_1, z_2\right)\right) + \ldots + g_1\left(g_2\left(z_{D-1}, z_D\right)\right) + g_1\left(g_2\left(z_D, z_1\right)\right) + f_{19}*$

$z = \mathbf{M}_1\left(\frac{5(x-o)}{100}\right) + 1$

Expanded Scaffer's F6 Function

Scaffer's F6 Function: $g(x, y) = 0.5 + \frac{\left(\sin^2\left(\sqrt{x^2+y^2}\right) - 0.5\right)}{\left(1 + 0.001\left(x^2+y^2\right)\right)^2}$

$f_{20}(\mathbf{x}) = g\left(z_1, z_2\right) + g\left(z_2, z_3\right) + \ldots + g\left(z_{D-1}, z_D\right) + g\left(z_D, z_1\right) + f_{20}*$

$z = \mathbf{M}_2 T_{asy}^{0.5}\left(\mathbf{M}_1(x - o)\right)$

8 Composite Functions

Eight composite functions are included in this suite, which merge properties of the sub-functions and maintains the continuity around the minima. A general form of a composite function f(x) is:

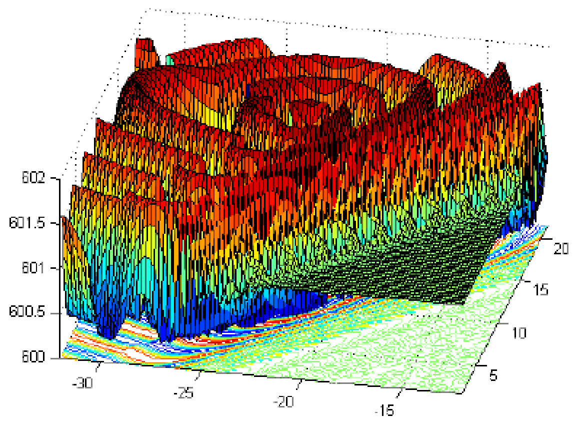


FIGURE 3.21: 3-D Map for UF-20 [1]

Functions	Unimodal Functions							
	DE		PSO		GWO		Proposed framework	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F1	9.28E+04	8.33E+03	8.21E+03	2.14E+03	1.74E+04	4.62E+03	**3.48E+03**	7.86E+02
F2	5.18E+09	5.61E+08	**1.67E+08**	5.30E+07	2.54E+08	7.55E+07	2.97E+08	5.72E+07
F3	2.12E+17	2.34E+17	**1.61E+12**	1.86E+12	1.87E+12	2.76E+12	2.52E+12	4.62E+12
F4	5.04E+05	4.06E+04	2.00E+05	2.45E+04	2.38E+05	3.06E+04	**1.58E+05**	1.46E+04
F5	4.03E+04	5.33E+03	3.06E+03	1.25E+03	1.34E+04	5.42E+03	**5.65E+02**	1.94E+02

TABLE 3.2: Results of Proposed framework and other algorithms on unimodal functions

$f(x) = \sum_{i=1}^{n} \{\omega_i * [\lambda_i g_i(x) + \text{ bias }_i]\} + f^*$

The exact functions and the 3-D map for each of the composite functions can be found in [1].

3.2.1 Comparison

We conducted this experiment at 100 dimensions, with 30 runs of each algorithm. The algorithms considered for comparison are the original GWO, PSO, and DE. For having a fair comparison, we considered 100 iterations in other algorithms. For PSO we used $c1$, $c2 = 2$, maximum velocity $= 6$, and w decreases from 0.9 to 0.6. For DE, mutation factor is set as 0.5 and crossover factor as 0.7.

The results are provided in Table 3.2-3.4. The **best value** of mean is highlighted in **bold**. Out of 28 functions, our proposed algorithm, performs better in 14 functions.

Functions	Multimodal Functions							
	DE		PSO		GWO		Proposed framework	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F6	1.29E+04	1.77E+03	9.92E+02	3.01E+02	1.60E+03	6.07E+02	**5.74E+02**	2.28E+02
F7	2.56E+05	3.02E+05	1.38E+03	2.28E+03	**-1.39E+02**	5.62E+02	4.96E+01	1.01E+03
F8	-6.79E+02	2.62E-02	-6.79E+02	2.42E-02	-6.79E+02	3.47E-02	**-6.79E+02**	2.93E-02
F9	-4.32E+02	2.53E+00	-4.62E+02	1.07E+01	**-4.86E+02**	7.55E+00	-4.77E+02	7.73E+00
F10	2.32E+04	1.91E+03	1.85E+03	6.08E+02	2.84E+03	4.70E+02	**1.76E+03**	3.32E+02
F11	1.62E+03	1.04E+02	1.05E+03	1.41E+02	**4.94E+02**	9.09E+01	5.78E+02	1.06E+02
F12	2.15E+03	1.30E+02	1.40E+03	1.26E+02	**7.85E+02**	1.56E+02	7.94E+02	1.00E+02
F13	2.29E+03	1.28E+02	1.80E+03	1.59E+02	1.12E+03	1.04E+02	**9.57E+02**	5.07E+01
F14	3.11E+04	6.56E+02	**1.92E+04**	1.21E+03	2.43E+04	5.90E+03	2.37E+04	1.92E+03
F15	3.31E+04	7.10E+02	**2.30E+04**	2.00E+03	2.78E+04	5.63E+03	2.56E+04	2.06E+03
F16	2.05E+02	3.51E-01	**2.04+02**	4.04E-01	2.05E+02	2.97E-01	**2.04E+02**	3.18E-01
F17	5.52E+03	3.76E+02	2.26E+03	1.97E+02	1.70E+03	1.61E+02	**1.48E+03**	7.32E+01
F18	5.60E+03	3.33E+02	2.55E+03	1.76E+02	2.02E+03	1.22E+02	**1.64E+03**	4.53E+01
F19	2.06E+06	6.05E+05	1.62E+03	5.68E+02	3.46E+04	2.43E+04	**7.33E+02**	7.03E+01
F20	**6.50E+02**	0.00E+00	**6.50E+02**	0.00E+00	**6.50E+02**	4.15E-14	**6.50E+02**	0.00E+00

Table 3.3: Results of Proposed framework and other algorithms on multimodal functions

Functions	Composite Functions							
	DE		PSO		GWO		Proposed framework	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
F21	1.34E+04	1.02E+03	3.31E+03	8.46E+02	6.05E+03	9.34E+02	**2.34E+03**	8.11E+02
F22	3.26E+04	7.82E+02	**2.67E+04**	1.82E+03	2.78E+04	4.43E+03	2.82E+04	1.85E+03
F23	3.55E+04	5.22E+02	2.99E+04	1.91E+03	**2.88E+04**	3.86E+03	2.97E+04	1.78E+03
F24	1.65E+03	6.54E+00	1.66E+03	3.30E+01	**1.53E+03**	2.16E+01	1.54E+03	2.08E+01
F25	1.79E+03	9.65E+00	1.98E+03	5.65E+01	**1.74E+03**	2.56E+01	1.77E+03	2.19E+01
F26	1.92E+03	5.76E+00	1.85E+03	1.91E+01	1.80E+03	1.54E+01	**1.77E+03**	1.12E+02
F27	6.00E+03	4.69E+01	5.46E+03	2.64E+02	**4.72E+03**	1.95E+02	4.85E+03	2.22E+02
F28	2.21E+04	1.42E+03	1.68E+04	1.18E+03	**1.04E+04**	1.21E+03	1.17E+04	1.54E+03

Table 3.4: Results of Proposed framework and other algorithms on composite functions

The results also show that the proposed framework outperforms GWO in 16 functions and ties in one function. We also achieved competitive results over PSO and DE. In detail, our algorithm outperforms in 3 out of 5 unimodal functions F1-F5, 10 out of 15 multimodal functions F6-F20, and 2 out of 8 composite functions F21-F28. It is interesting that all of the functions are able to achieve the same local minima in F20 - Expanded Scaffer's F6 Function.

Convergence graphs for the benchmark functions in Figures 3.22-3.25, compare the convergence of our proposed framework with the other state-of-the-art algorithms in the same objective space. From the figures, it is assuring that the proposed framework performs with a satisfactory convergence behaviour and converges faster than the other three algorithms in 21 out of 28 functions. This supports our assumption that a multi-population structure allows a better exploration and faster convergence.

3.2.2 Performance at Different Dimensions

To study the effect of changing the number of dimensions, we performed this test. Tables 3.5-3.7 shows how the performance of our algorithm changed as the number of dimension is changed from 50 to 70, and 100. It is expected that when the number of dimensions are increased, the performance of the algorithm reduces. The optima of the function does not change as we change the number of dimensions [1], but the optimization process becomes more tough due to involvement of many variables, therefore, in this, we make direct comparisons between the values.

An interesting observation is that the unimodal functions F2, F3, and composite functions F21 have a better result when the number of dimension is 70 as compared to 50. This is against our hypothesis of a reduction in the performance when dimensions is increased.

Another observation is that for the multimodal functions F8, F16, and F20 , there is not a significant difference when the number of dimension is shifted from 50 to 70,

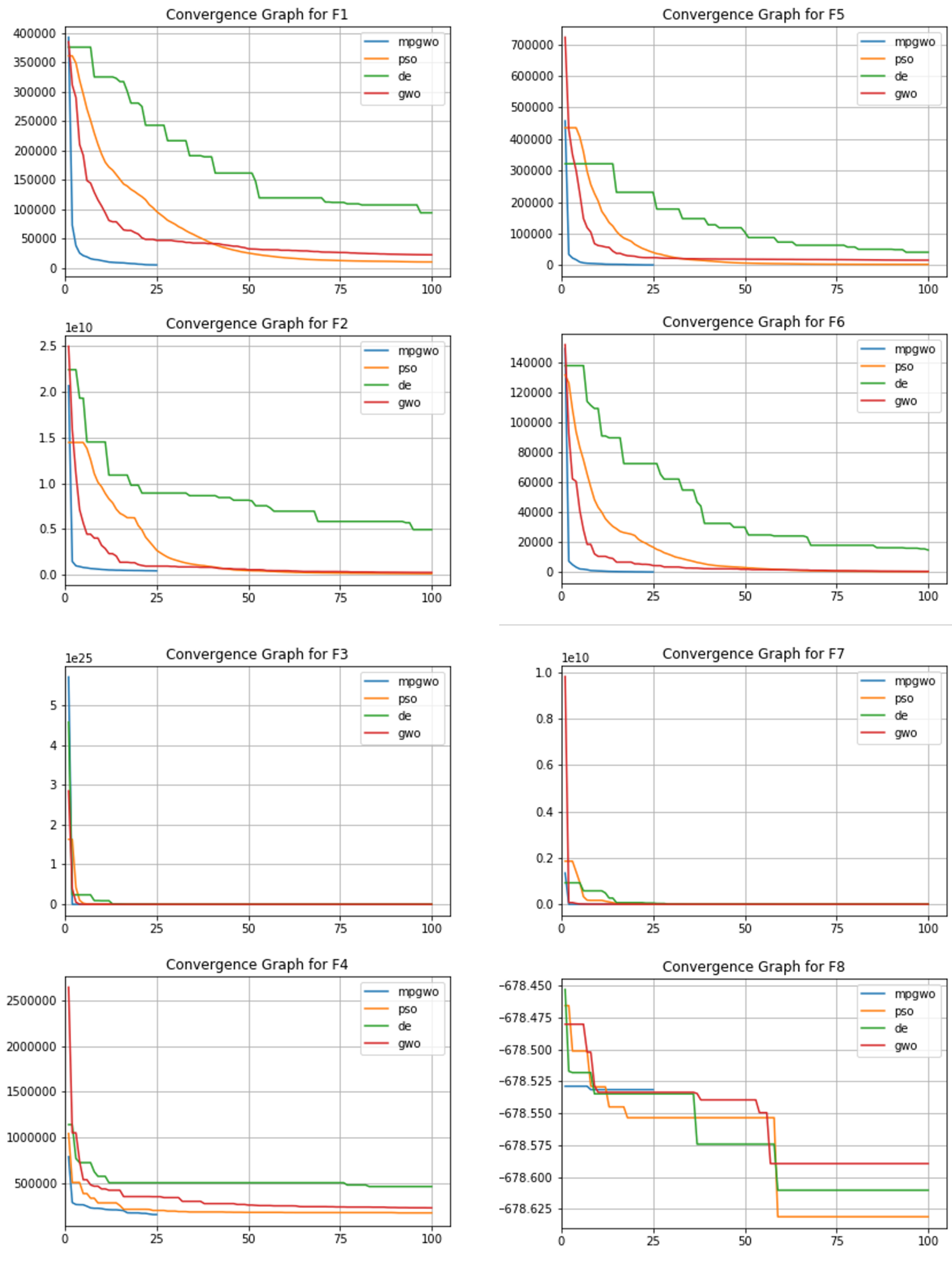


FIGURE 3.22: Convergence graphs for the functions F1-F8

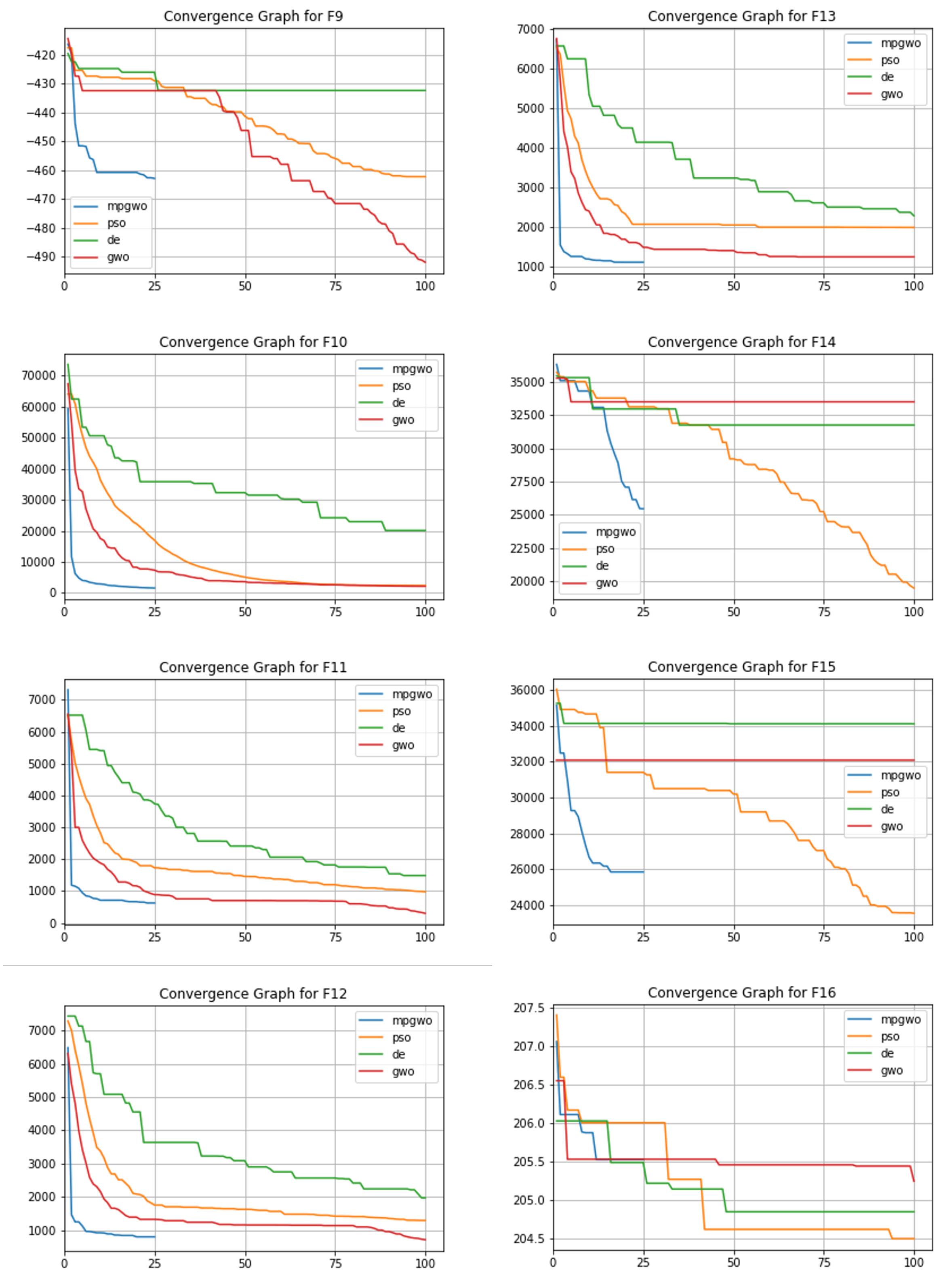


FIGURE 3.23: Convergence graphs for the functions F9-F16

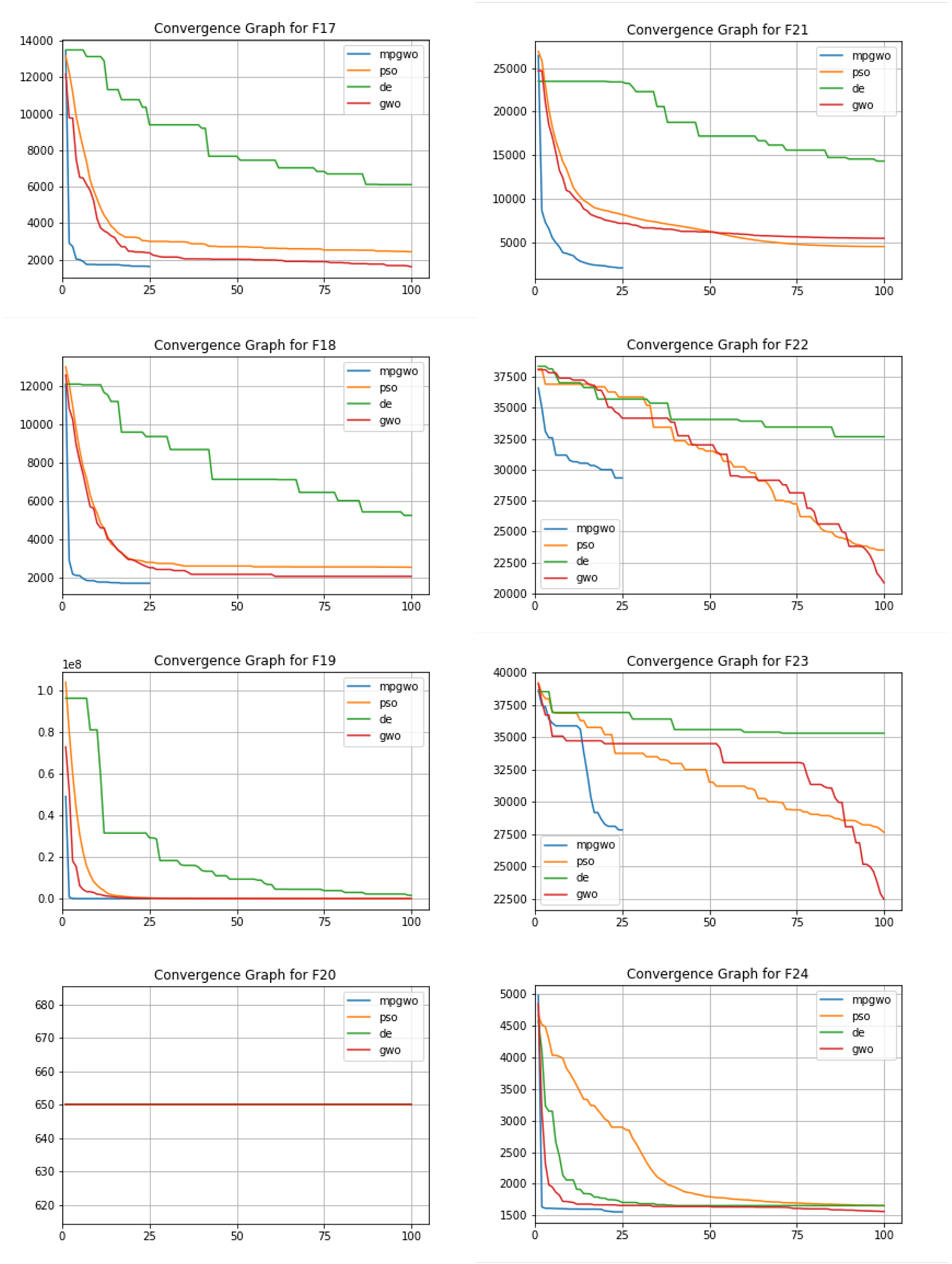


Figure 3.24: Convergence graphs for the functions F17-F24

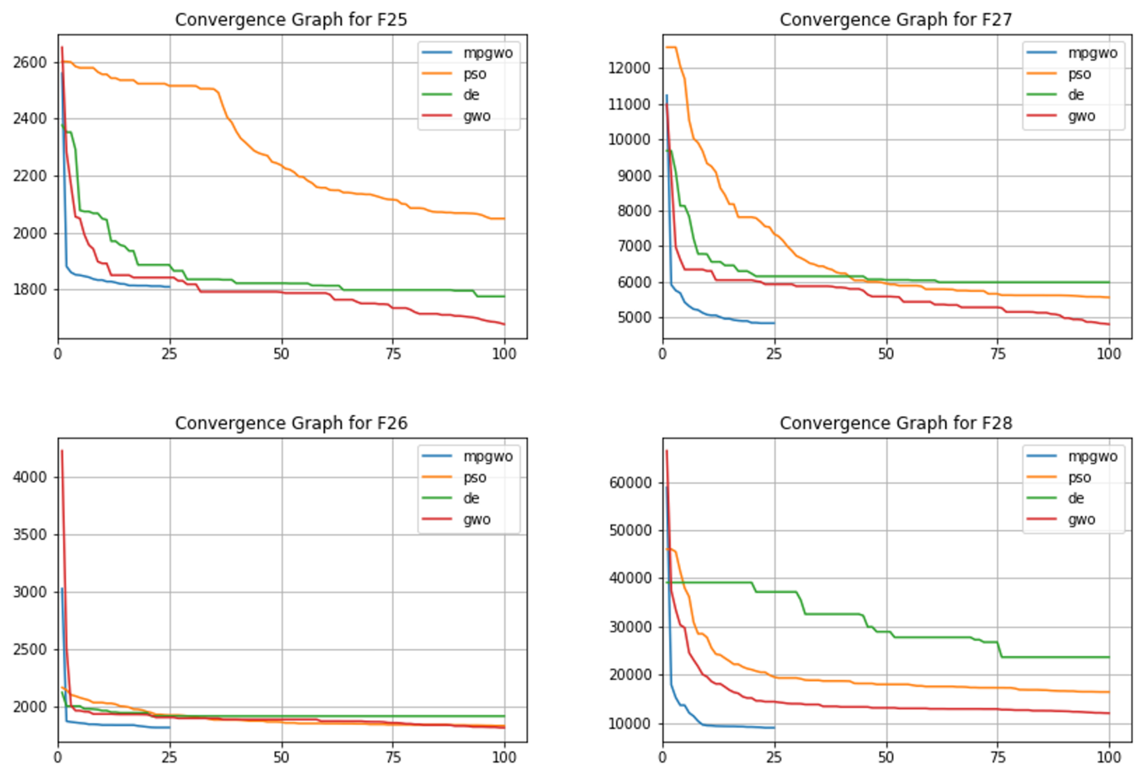


Figure 3.25: Convergence graphs for the functions F25-F28

and ultimately 100. Thus, there is a need to further study these patterns discovered, and analyse the benchmark functions and the process of optimization.

Unimodal Functions	Dim 50 Mean	Dim 70 Mean	Dim 100 Mean
F1	-8.01E+02	4.74E+02	3.48E+03
F2	6.77E+07	6.73E+07	2.97E+08
F3	3.10E+10	2.57E+10	2.52E+12
F4	6.01E+04	7.90E+04	1.58E+05
F5	-8.15E+02	-5.15E+01	5.65E+02

Table 3.5: Comparison of the Proposed Framework for different dimensions on unimodal functions.

3.2.3 Non-Parametric Statistical Tests

First, to prove the consistency of our proposed framework we performed Kruskal-Walli's test on the results of running the proposed method 30 times on the benchmark

Multimodal Functions	Dim 50	Dim 70	Dim 100
	Mean	Mean	Mean
F6	-6.57E+02	-3.46E+02	5.74E+02
F7	-7.04E+02	-6.79E+02	4.96E+01
F8	-6.79E+02	-6.79E+02	-6.79E+02
F9	-5.49E+02	-5.25E+02	-4.77E+02
F10	3.81E+01	4.43E+02	1.76E+03
F11	-8.19E+01	1.51E+02	5.78E+02
F12	7.07E+01	3.54E+02	7.94E+02
F13	2.12E+02	4.58E+02	9.57E+02
F14	9.21E+03	1.45E+04	2.37E+04
F15	1.14E+04	1.63E+04	2.56E+04
F16	2.04E+02	2.04E+02	2.04E+02
F17	7.70E+02	1.04E+03	1.48E+03
F18	8.93E+02	1.19E+03	1.64E+03
F19	5.49E+02	5.97E+02	7.33E+02
F20	6.22E+02	6.33E+02	6.50E+02

Table 3.6: Comparison of the Proposed Framework for different dimensions on multimodal functions.

Composite Functions	Dim 50	Dim 70	Dim 100
	Mean	Mean	Mean
F21	1.93E+03	1.26E+03	2.34E+03
F22	1.17E+04	1.73E+04	2.82E+04
F23	1.30E+04	1.85E+04	2.97E+04
F24	1.33E+03	1.39E+03	1.54E+03
F25	1.49E+03	1.56E+03	1.77E+03
F26	1.49E+03	1.52E+03	1.77E+03
F27	2.96E+03	3.56E+03	4.85E+03
F28	2.48E+03	5.24E+03	1.17E+04

Table 3.7: Comparison of the Proposed Framework for different dimensions on composite functions.

functions. The null hypothesis (H_0) states that all treatments come from the same distribution. This is tested at a significance level of 0.1. If the test suggests that null hypothesis is rejected, then that means one of the treatments are more significant than the rest. The distribution of the test statistics generated is approximated by Chi-square distribution. The results for this test came out as H = **0.1766**, and p-value = **0.99999**. From the results, p-value is > 0.1, therefore, the Kruskal-Walli's test failed to reject the null hypothesis H_0. Hence, we can state that our results after 30 independent runs came from the same normal distribution and our algorithm performed consistently.

Friedman Test Result	H=7.81	p=7.74E-10	
Wilcoxon Test Result			
	DE	PSO	GWO
Proposed	H = -2.35, p = 0.009	H = -1.12, p = 0.13	H = 0.76, p = 0.22

Table 3.8: Non-Parameteric Test Analysis

Next, we performed Friedman test to ascertain the statistical significance of the proposed algorithm. This test is a non-parametric test that provides a measure of the difference between multiple methods. The null hypothesis considered states that there is no significant difference between the results at a significance level of 0.1. After performing the test, we found the value of $p < 0.1$ (Table 3.8). Therefore, from this, we conclude that the null hypothesis is rejected and at least one set of significant results exists.

To do a pairwise comparison test, we performed the Wilcoxon test. The null hypothesis for this test is that the two sets of results have the same distribution. If the null hypothesis is rejected, then the p-value will be < 0.1, as the significance level of 0.1%. From the results in table 3.8, we understand that our algorithm provided a significant result only in one case.

Chapter 4

Proposed Model for Multi-Objective Optimization

This section will first explain the search process and the model architecture, along with the additional components such as elitism and mutation, followed by the evaluation done on IEEE CEC 2009 benchmark for multi-objective problems.

4.1 Model Architecture

The model is inspired by the hierarchy observed in grey wolves [13, 28], and mimics the leader structure with a dynamic multi-population design. A dynamic set of local leaders is maintained for each population, which is analogous to the leader selection observed in the traditional version of MOGWO [28]. A solution has more chances of being selected as a leader if it belongs to the archive's sparsely dense grid space. Broadly, our model, as shown in Figure 4.1, can be divided into five steps:

1. **Initialization of Population and Adaptive Archives**: First, a population of size N is initialized in a random uniform manner according to the bounds

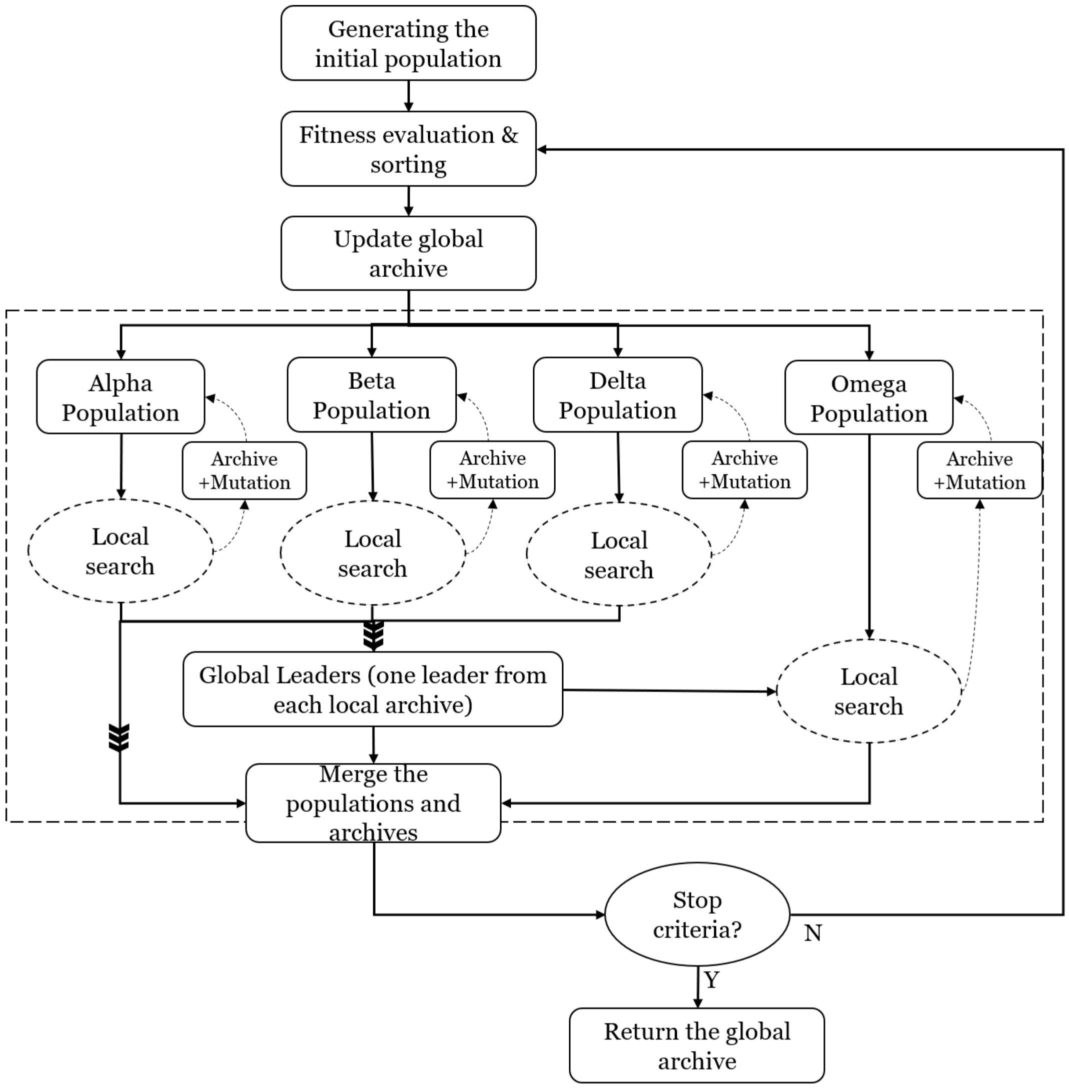


Figure 4.1: Proposed Framework Architecture for Multi-objective Optimization

specified by the function. The population P_t represents the population at time t with individuals $\{i_1, i_2, \ldots, i_N\}$. Hence, P_0 is the initial population with n random population.

Additionally, a global archive A_G and four local archives for each population - $A_\alpha, A_\beta, A_\delta, A_\omega$, are also initialized. The global archive is of a maximum capacity n_A, while the local archives are all of the same size - $n_A/4$. These adaptive archives are inspired from multi-objective particle swarm optimization (MOPSO) [26] and MOGWO [28]. A candidate individual may or may not be added based on the following conditions:-

1. If at least one member of the archive dominates the candidate individual; then the candidate solution is not added to the archive.

2. If the candidate individual dominates one or more archive members, then the dominated members are removed, and the incoming candidate individual is added.

3. If the candidate solution does not dominate any archive member and is not dominated by any member either, then two situations may arise. In the first situation, the archive still has space for one or more members, so the candidate solution is added to the archive. In the second situation, when the archive is full, one archive member is removed using a roulette wheel selection, where the probability of a member being removed is directly proportional to the density of the grid to which the member belongs.

This adaptive archive allows a diverse Pareto front since the non-dominated solutions are included based on individuals' grid density. Due to the nature of this archive, the selection of the leader is also performed based on the grid density. This is done by assigning a probability of selection to the archive members based on the density of the cell they belong to, as shown in Figure 2.3. This way, more leaders are selected from less dense cells, and hence the diversity of solution is improved. Another step done to improve the diversity of solutions is when the archive overflows. The solutions in crowded regions have more chance of being removed than from non-crowded regions.

Leader Selection: Probability of selection from i^{th} region $P(leader_i) \propto \frac{1}{N_i}$ where N_i is the number of obtained Pareto optimal solutions in the region.

Extra Element Removal: Probability of being selected for removal from i^{th} region $P(removal_i) \propto N_i$ where N_i is the number of obtained Pareto optimal solutions in the region1.

2. Fitness Evaluation: In this step, the individuals are evaluated based on the objective function. After the evaluation, the global archive is updated, and the new non-dominated solutions are added, if any are seen. After this, the population is divided into 4 fixed-size populations namely - Alpha population (A), Beta population (B), Delta population(Δ), and Omega population(Ω). As discussed earlier, this structure is analogous to the hierarchy observed in grey wolves [9, 13].

Therefore, the population is divided so that the individuals in Alpha are better than individuals in Beta, which are better than individuals in Delta, and ultimately Delta individuals are better than the individuals in Omega.

Since this is a multi-objective problem, individuals need to be ranked based on the problem's different objectives, in our case, bi-objective problems. Since the total size of the initial population is N, the size of Alpha, Beta, Delta, and Omega population is $N/4$.

Initially, the Alpha population is assigned $N/8$ top individuals according to each objective ($N/8 + N/8 = N/4$). The remaining individuals are then sorted again, and $N/8$ top individuals from each objective are assigned to Beta. This is repeated until the Omega population. In the end, we have four populations, each with $N/4$ individuals. Hence, $P_t = \mathrm{A}_t \cup \mathrm{B}_t \cup \Delta_t \cup \Omega_t$ and $\mathrm{A}_t \cap \mathrm{B}_t \cap \Delta_t \cap \Omega_t = \varnothing$

This way, the Alpha population plays a leading role in optimization, supported by Beta. Delta population is a subordinate to Beta, and finally, the Omega population contains the least fit individuals of each objective. This way, the Omega population submits to the top 3 leader populations.

By allowing this division of population dynamic and repeated at every iteration, it is possible to have individuals migrate from one population to another when the fitness values change over time.

3. Leader Populations Evolution: In this step, the leader populations - A, B, and Δ, are evolved. For the local search process, as done in this step and the next, we propose a local search algorithm. This local search algorithm is designed in such a way that it evolves a local population independently. After evolving the populations

Algorithm 3: Proposed Framework for Multi-Objective Optimization

```
Input  : $n$ : The total number of individuals in a population
         $N_t$ : Max. Number of Iteration
         $n_A$ : Maximum size of archive
Output: $A$ archive with non-dominated solutions
1  Initializing the population randomly between the bounds;
2  Elite ← {Ø} ;
3  i ← 1 ;
4  while $i <= N_t$ do
5      Sort and divide the populations based on the fitness values where each
         population gets $n/4$ individuals;
6      Run algorithm 4 in the top populations (Alpha, Beta and Delta
         populations);
7      Obtain the $\alpha$ leaders of each population and add them to the leaders set;
8      Elite ← Top 3 Leaders from archive;
9      Run algorithm 4 in Omega population with the obtained leaders;
10     Population ← population ∪ elite;
11     Merge the obtained local archives with the global archive;
12     while size(archive)>$n_A$ do
13         remove an element based on probability of removal;
14     end
15     for all individuals do
16         if rand <= 30% then
17             mutate random dimension;
18         end
19     end
20     i++;
21 end
22 Return archive ;
```

for a set number of iterations, it returns the evolved population, the local archive, and a leader selected from the archive. Since the populations are independent, this step can be done in parallel for each population and save time.

When the three populations, Alpha, Beta, and Delta, are evolved, they return three leaders from each population. This set of leaders is known as global leaders. Since each population co-evolves independently, not all of these leaders may be non-dominating to each other. Even though we divide the population so that the individuals in Alpha are better than the individuals in Beta and Delta, the leader obtained from Delta may dominate the leader obtained from Alpha. This is because each population searches in its own solution space and may have better fitness after their positions are updated.

4. Omega Population Evolution: The next step is to evolve the Omega population. As discussed earlier, the population structure is inspired by the hierarchy seen in grey wolves, in which pack follows the leader wolves [9, 13, 28]. To preserve this structure, we evolve the Omega population so that the population follows the leader populations; this is done by introducing the global leaders, obtained in the last step, in this population.

Our proposed local search algorithm can introduce the global leaders by taking an optimal parameter of the leaders. When the global leaders are provided, as in the Omega population, the global leaders are added to this population's local archive. If these leaders are non-dominated in the Omega population, they stay in the archive and guide the local search process.

5. Merging Populations and Archives: In this step, the four evolved populations obtained are merged, and the local archives are merged into the global archive. The non-dominated solutions that have been found by each local population are compared and stored in the global archive. The merged population is then mutated using a random probabilistic operator; this helps in escaping local optima, in some cases, and improve the exploration ability of the framework.

The algorithm for the proposed framework can be seen in Algorithm 3, which illustrates the dynamic co-evolution and the use of elite groups.

4.1.1 Local Multi-Objective Optimization Algorithm

The local optimization algorithm used for multi-objective optimization is explained in this part. The grey wolves' pack and behaviour inspired the proposed local search algorithm [28]. Additionally, it introduces a probabilistic mutation operator to improve the diversity of solutions, and for better convergence, an elite group of individuals is also added. This is done to improve the convergence for non-zero optimal values, which is considered a defect in GWO [11].

The algorithm can allow mutual information flow between populations by using the guide leaders' parameter, which are individuals from other populations that migrate to the omega population. These leaders can lead the omega population based on their fitness values compared to the population, similar to the guide leaders explained in Chapter 3. The algorithm for this local search is illustrated in Algorithm 4.

Algorithm 4: The Proposed Local Search for Multi-Objective Optimization

```
Input  : $n, n_A/4$ : Size of local population, and local archive
         $N_t$ : Number of local iteration
         $\vec{i_\alpha}$, $\vec{i_\beta}$, and $\vec{i_\delta}$ Guide leaders (optional)
Output: The last evolved set of population &
         $i_\alpha$ : local leader & $A_p$ Local archive
1  Initializing the local archive;
2  if leaders are given then
3      check for non domination and update archive;
4  end
5  Find the $\alpha$, $\beta$, and $\delta$ individuals based on probability of selection;
6  Elite ← {Ø};
7  i ← 1;
8  while $i <= N_t$ do
9      foreach individual ∈ Population do
10         Update individuals' Position using GWO equations;
11     end
12     Evaluate the fitness value of individuals;
13     Update local archive with non-dominated individuals;
14     while size(archive)>$n_A/4$ do
15         remove element based on probability of removal;
16     end
17     Update new $\alpha$, $\beta$, and $\delta$ ;
18     Elite ← $\alpha$;
19     Population ← population ∪ Elite;
20     i++;
21 end
22 Return the population, $\alpha$, and local archive
```

4.1.2 Additional Components

There are two additional components that we have incorporated in our framework-elite groups and mutation operators. These components are the same as the one introduced in Chapter 3. The multi-objective framework's difference is that the way the elite members are selected uses the same leader selection as seen in [26, 28]. As stated earlier, the probability of an element being selected from a region depends on the region's crowdedness. The goal is to select a leader from a less crowded region to improve the diversity of solutions. This has been explained in Section 2.2.

Additionally, the archives used are adaptive and have a maximum size, which is an input to the algorithm. Generally, the maximum archive size is the same as the size of the initial population. In our framework, we have one global archive, and each local population is assigned one archive.

4.2 Evaluation

In this section, we perform the evaluation of our algorithm under various experiments. First, we compare our model with the state of the art algorithms for multi-objective problems - NSGA II [22], and MOEAD [23]. We also compare our proposed framework with MOGWO [28]. The comparison is made in two settings; in the first setting, the comparison is made with 500 iterations of each state of the art algorithms to keep the evaluations fair; in the second setting, the computation time is kept similar. Then the performance of the proposed framework is studied under various settings.

These algorithms are run on an Intel Core i5-9300H processor with 8GB RAM, implemented on Python 3.5. The comparison is made by running the experiments ten independent times on each function, the average of which is used for the results.

The test environment used for the comparisons is the IEEE Congress on Evolutionary Computation's (CEC) 2009 test suite for unconstrained multi-objective problems

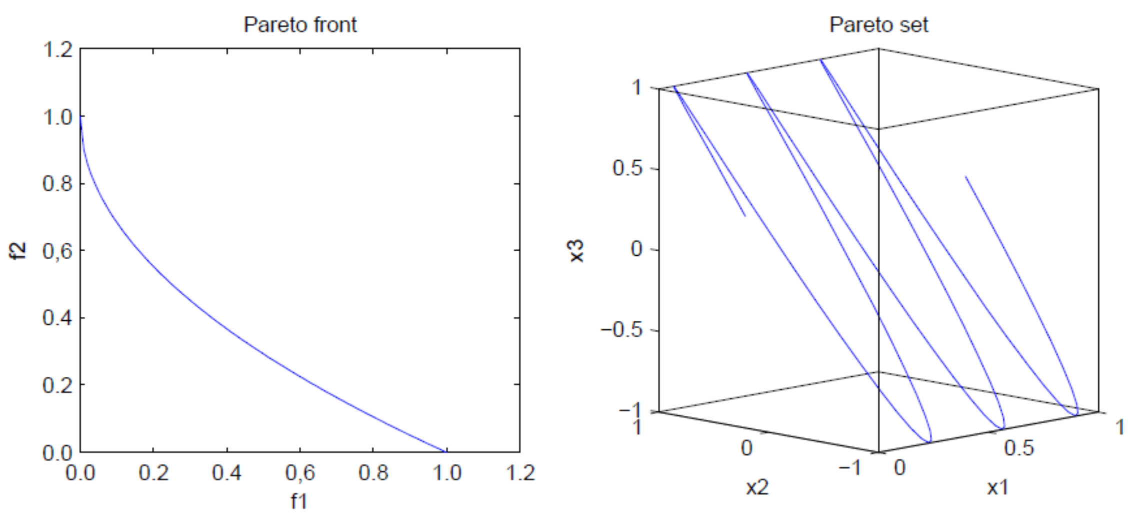


FIGURE 4.2: Illustration of Pareto front and Pareto Set for UF1 [2]

[2]. The suite consists of seven unconstrained problems that are bi-objective in nature. Each problem has its own search space and a different optimal Pareto front. The test environment suggested a dimension of 30 for the contest. The optimal Pareto fronts of the benchmarks are:

UF1, UF2, UF3- Continuous Pareto front represented by $f_2 = 1 - \sqrt{f_1}$.

UF4- It has a continuous Pareto front represented by the equation $f_2 = 1 - f_1^2$.

UF5- The Pareto front in this case has 2N+1 points, the points are represented as $(\frac{i}{2N}, 1 - \frac{i}{2N})$ for i = 1, 2, ..., 2N for CEC 2009 contest, N = 10.

UF6- The Pareto front consists of one isolated point, (0, 1), and N disconnected points $f_2 = 1 - f_1$, where $f_1 \in \bigcup_{i=1}^{N}[\frac{2i-1}{2N}, \frac{2i}{2N}]$.

UF7- The Pareto front in this case is continuous and represented by $f_2 = 1 - f_1$.

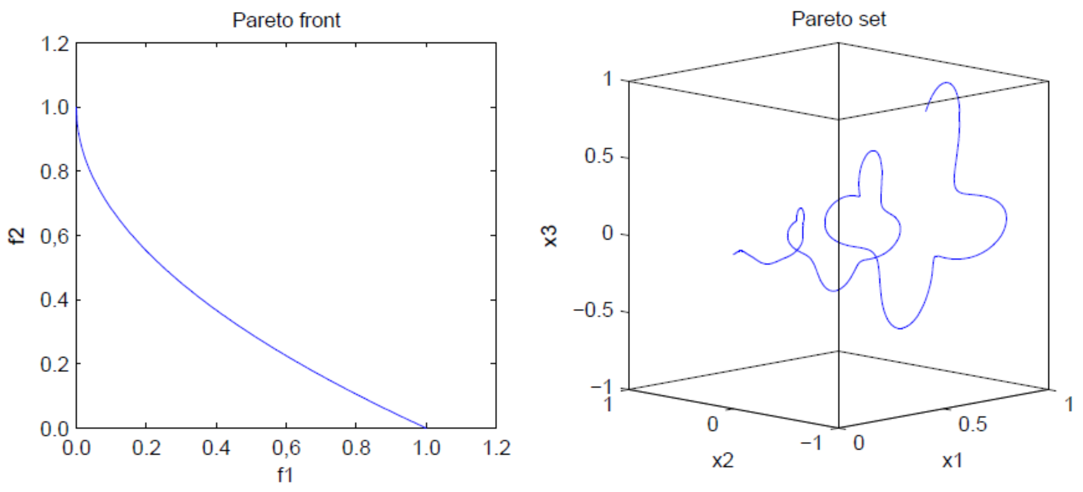


FIGURE 4.3: Illustration of Pareto front and Pareto Set for UF2 [2]

4.2.1 Performance Indicators

The performance indicators used for comparison are generational distance (GD)[33] and hypervolume [32].

For minimization problems, a greater hypervolume and a lower GD means better performance. These indicators are explained in Chapter 1.

4.2.2 Comparison under Fair Evaluation

This comparison was made to compare the overall performance of the algorithms. In order to allow a fair comparison, the number of iterations for the state of the art methods was considered relatively higher since, in our proposed framework, we have inner loops. The parameters considered are:-

1. Main iteration for the proposed framework: 50 with 15 local iterations
2. Iteration for NSGA-II, MOEAD, and MOGWO: 500
3. Population Size: 100

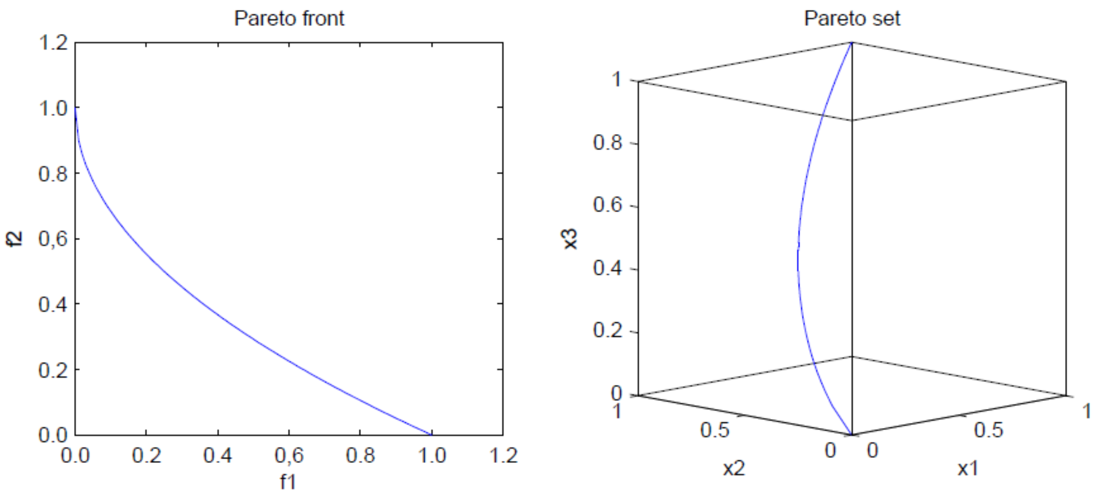


FIGURE 4.4: Illustration of Pareto front and Pareto Set for UF3 [2]

4. Maximum size of archive: 100

Table 4.1 compares the values of hypervolume, GD, and computation time for this setting. Additionally, the fronts obtained from each algorithm are also compared, in figures 4.9-4.15, from one of the ten runs for each function. The black solid line represents the actual Pareto front for each problem.

In UF1, similar fronts are observed in the case of MOEAD and NSGA-II; the GD values for them are significantly higher than the proposed framework and MOGWO. MOGWO has the best values of hypervolume and GD in this case. It is to be noted that though the proposed framework gave only 1.15% less hypervolume and 26% more GD, the time taken is almost one-third of the time taken by MOGWO. This emphasizes the time-optimality trade-off, which is again of importance. In Figure 4.9, it is evident that the solutions provided by the proposed framework are optimal, and it provides competitive performance.

In the second problem, H-MPGWO has better hypervolume than MOGWO only, and the best value for GD, which is achieved in less than one-third of the time. The value of GD for MOGWO is the same since it provides a very short region of the solution near the front, which in turn gives a better average distance from the actual

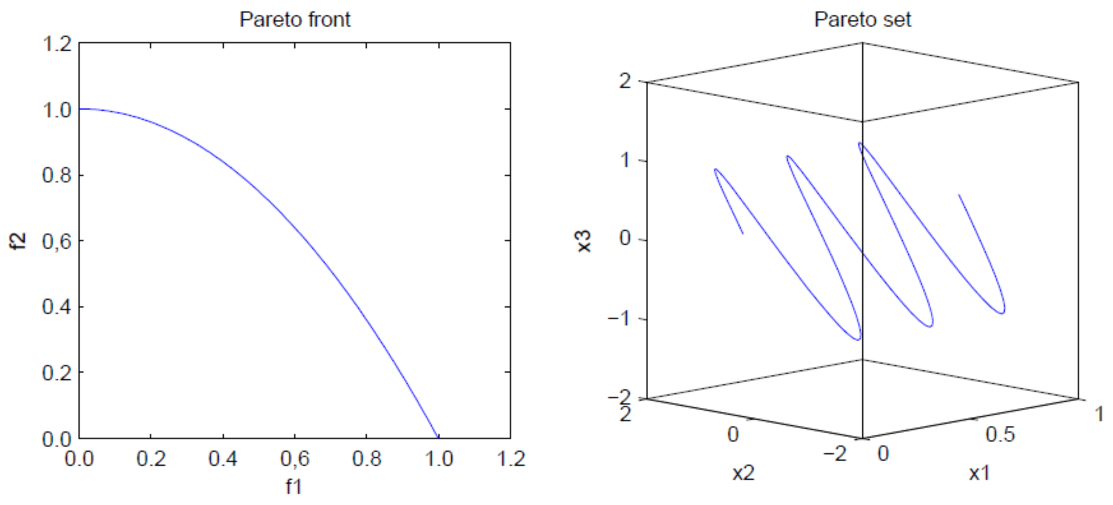


Figure 4.5: Illustration of Pareto front and Pareto Set for UF4 [2]

Pareto front. The solutions obtained from the proposed framework are much close to the actual Pareto front but are not as spread as NSGA-II and MOEAD. Thus, the proposed framework delivered the most optimal solutions, but not a wide variety of solutions, which can be observed in Figure 4.10.

In UF3, NSGA-II and MOEAD seem to have given a wide variety of solutions, but all of them are non-optimal when compared to the solutions obtained from MOGWO and the proposed framework. Though the proposed framework has a 6.4% decrease in the hypervolume than MOGWO, it still provides a notably better GD, which means that the average solution from the obtained set of solutions is closer to the actual Pareto front; this can be seen in Figure 4.11.

In UF4, MOEAD achieved the best hypervolume value but the least optimal solutions, which is evident in the figure 4.12 as well as the table 4.1. The proposed framework has hypervolume better than MOGWO only, and GD is better than MOGWO as well as MOEAD. Although, the GD of the proposed framework is 10.8% less than NSGA-II, it took significantly less time (62.5% decrease) as well. Figure 4.12 shows the obtained Pareto fronts for UF4, and it is evident that MOEAD provides a really well spread Pareto front.

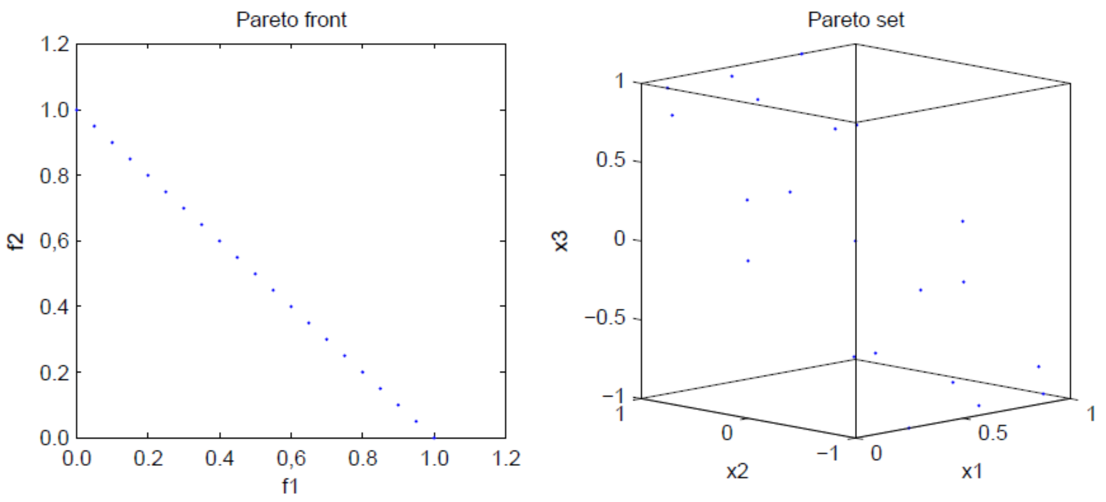


Figure 4.6: Illustration of Pareto front and Pareto Set for UF5 [2]

The value of GD is best for the proposed framework in UF5, with NSGA-II being at second place with a difference of almost 0.03. Figure 4.13 shows that the traditional MOGWO was able to get non-optimal solutions; on the other hand, the solutions from NSGA-II spread over a wide range of space and are optima. Hence it has the highest hypervolume for this function. For the proposed framework, the average obtained solution can be seen, in Figure 4.6, to be very close to the actual Pareto front.

The proposed framework performed better than both NSGA-II and MOEAD in UF6 and worse than MOGWO, where hypervolume is 0.7% less, and GD is 9.6% more than. The performance did not change significantly while it takes approximately twice the time in the case of MOGWO. In Figure 4.14, it can be seen that the traditional MOGWO has slightly better and more optimal solutions towards the bottom side of the front.

For the last problem, UF7, the proposed framework was able to achieve better values of hypervolume and GD than MOGWO and MOEAD, while NSGA-II outperforms the proposed framework for both metrics, by a margin of 3.04%, 26.28% for hypervolume and GD, respectively. In figure 4.15, it is evident that in UF7, MOEAD has the least optimal solutions, while NSGA-II has the widest front obtained.

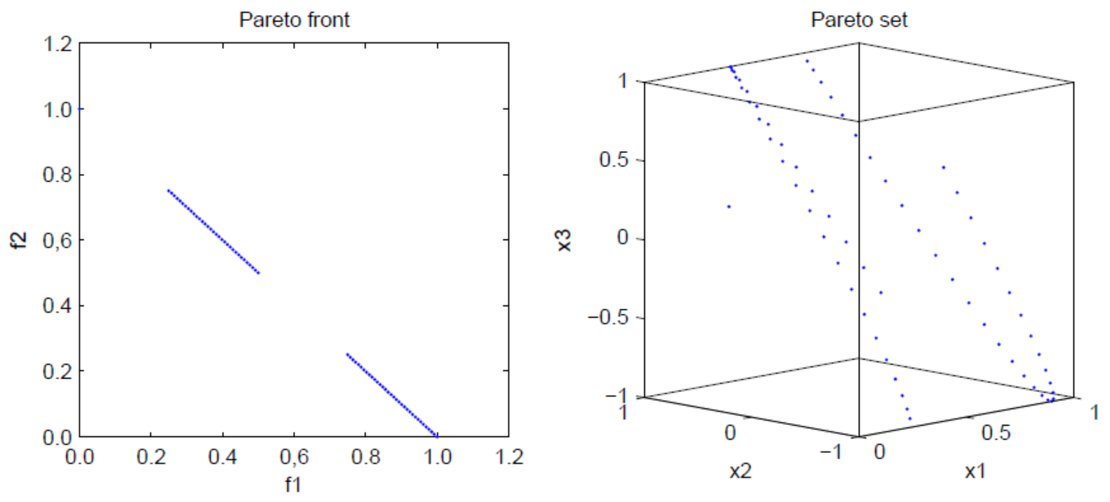


FIGURE 4.7: Illustration of Pareto front and Pareto Set for UF6 [2]

In this set of comparisons, the proposed framework only ran for 50 iterations in the outer loop, hence it took the least time in all tests. The following set of comparison was performed in order to have the time for each algorithm in the same range.

4.2.3 Comparison with Similar Computation Time

This comparison was done to evaluate performance of the algorithms when they are run for similar time scale. The parameters for this comparison are:

1. Main iteration for proposed framework: 50 with 15 local iterations
2. Iteration for NSGA-II, MOEAD: 100
3. Iteration for MOGWO: 200
4. Population Size: 100
5. Maximum size of archive: 100

Table 4.2 shows the comparison while keeping the processing time on a similar scale. Two metrics, as discussed earlier, are used to perform the comparison of the

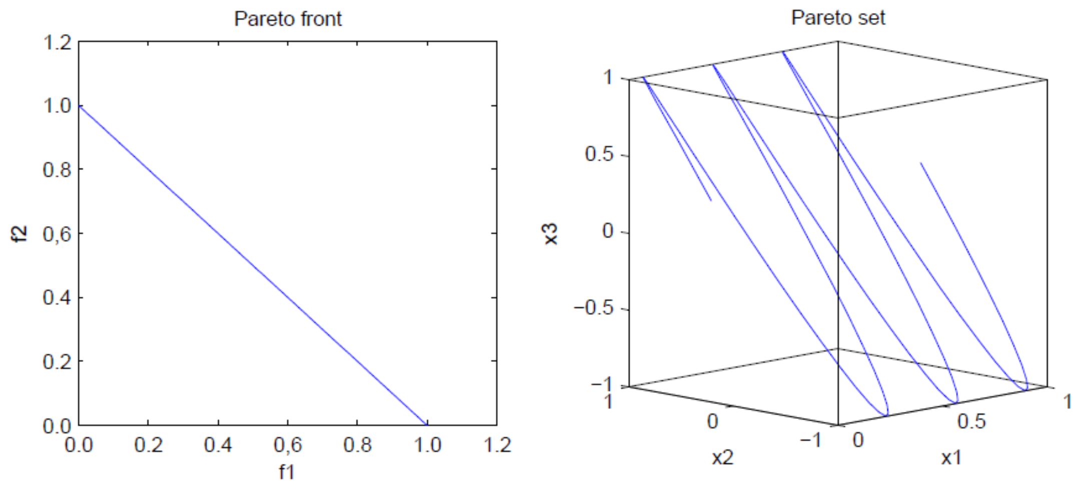


FIGURE 4.8: Illustration of Pareto front and Pareto Set for UF7 [2]

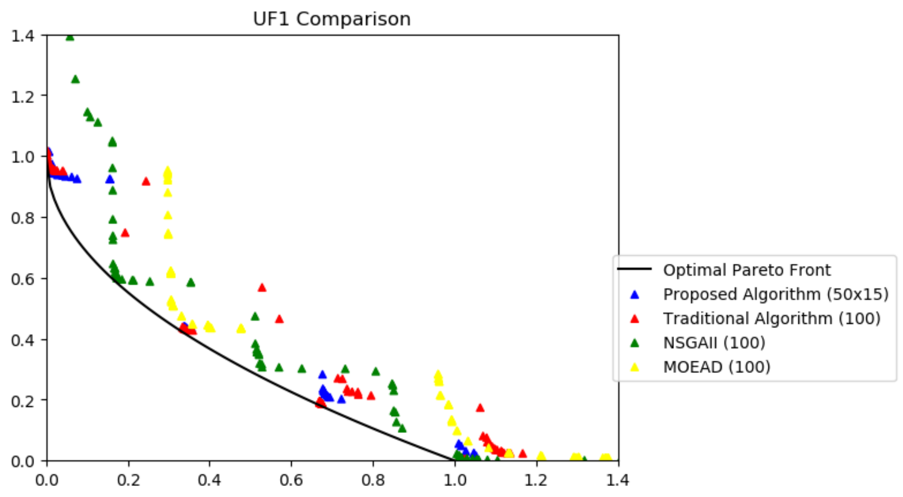


FIGURE 4.9: Comparison under Fair Evaluation-UF1

Pareto front based on its spread (hypervolume) and closeness to the actual front (GD). It is noted that since hypervolume depends on a reference point, the value of this metric changes from the previous table since the reference point is changed according to the performance of other candidate solutions. Figures 4.16-4.22 compares the obtained Pareto front for one of the ten runs for each algorithm to visualize and compare the fronts.

In UF1, the proposed framework provided four regions of the non-dominated solution, which are considerably close to the actual Pareto front, as illustrated in Figure

		H-MPGWO	MOGWO	NSGAII	MOEAD
UF1	Hypervolume	8.38	**8.48**	8.45	8.03
	GD	0.04	**0.03**	0.09	0.30
	Time (s)	**12.85**	35.07	164.95	287.08
UF2	Hypervolume	7.31	6.96	**8.48**	8.30
	GD	**0.01**	**0.01**	0.06	0.16
	Time (s)	**44.76**	160.01	125.31	345.46
UF3	Hypervolume	14.37	**15.36**	14.16	13.63
	GD	**0.18**	0.23	0.44	0.41
	Time (s)	**16.42**	39.74	117.69	383.85
UF4	Hypervolume	2.69	2.65	3.04	**3.11**
	GD	0.07	0.12	**0.06**	0.12
	Time (s)	**29.22**	127.72	80.36	295.63
UF5	Hypervolume	28.17	28.03	**31.67**	29.77
	GD	**0.89**	1.21	0.91	1.56
	Time (s)	**10.36**	21.27	136.26	268.17
UF6	Hypervolume	32.68	**32.92**	32.06	30.28
	GD	0.39	**0.35**	0.55	0.69
	Time (s)	**10.88**	21.11	146.33	240.32
UF7	Hypervolume	14.26	14.04	**14.71**	13.45
	GD	0.02	0.03	**0.01**	0.23
	Time (s)	**19.0497**	71.73	114.24	283.41

Table 4.1: Comparison between proposed framework(50 iterations) and state of the art methods ran for 500 iterations

4.16. At the same time, the solutions provided by NSGA-II and MOEAD are widely spread and comparatively farther from the actual Pareto front. This shows that if we only consider hypervolume, it is not fair since it is clear that the proposed framework has much more optimal solutions.

For UF2, NSGA-II and MOEAD achieved a wide range of solutions; that is why in table 4.2 the hypervolumes are considerably higher than the proposed framework. In the case of MOGWO, the solutions existed in only one small concentrated region, as seen in Figure 4.17; this defeats the purpose of having a Pareto front since there is shallow diversity in the solutions. But judging the performance just on the basis of hypervolume is not enough; in the figure, the proposed framework has a good spread of the solutions, as well as the solutions obtained, are much closer to the Pareto front than the ones obtained by NSGA-II and MOEAD.

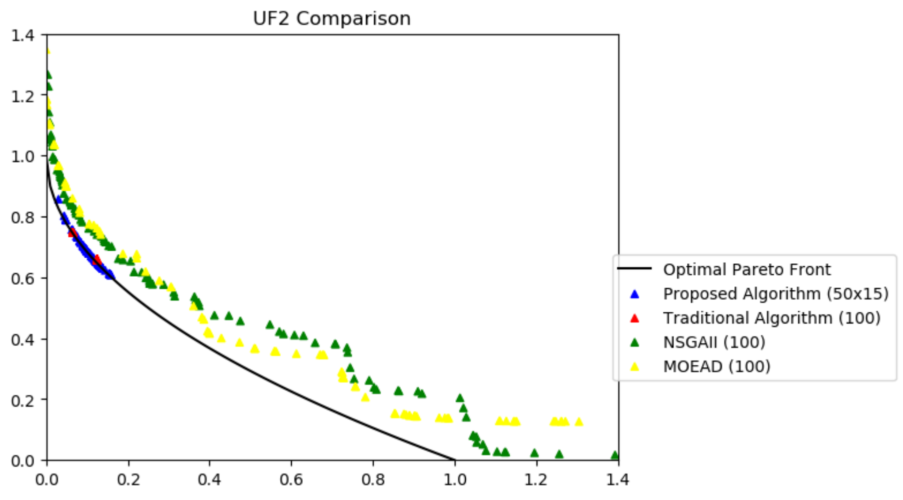


FIGURE 4.10: Comparison under Fair Evaluation-UF2

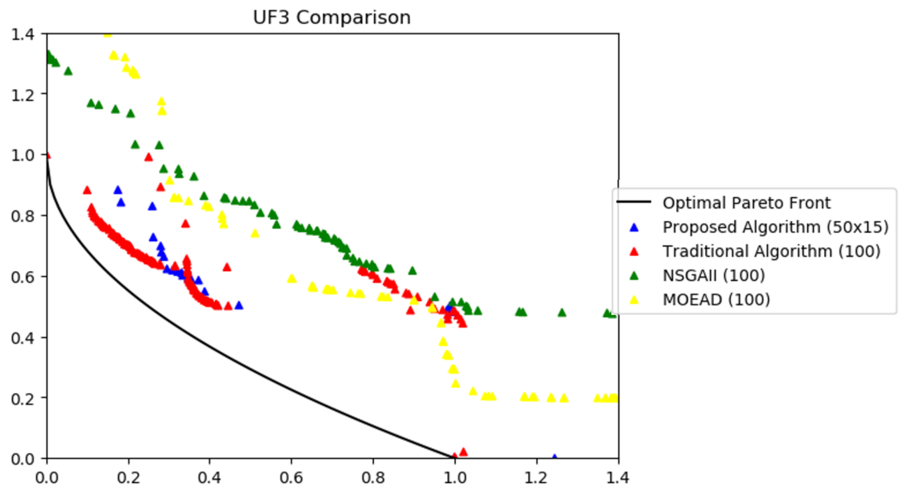


FIGURE 4.11: Comparison under Fair Evaluation-UF3

A similar trend could be seen in the plot for UF3, where the solutions obtained by the proposed framework completely dominate the solutions obtained by other approaches, despite not having the best hypervolume. Figure 4.18 shows that the spread of the solutions obtained by the proposed framework is competitive and not concentrated in one region, which may have defeated the purpose of having an optimal front.

In the fourth problem, the trend followed for the proposed framework, but for MOEAD were widespread but not of the best quality, In table 4.2, the GD of the proposed framework is the best, and hence the solutions obtained were much better,

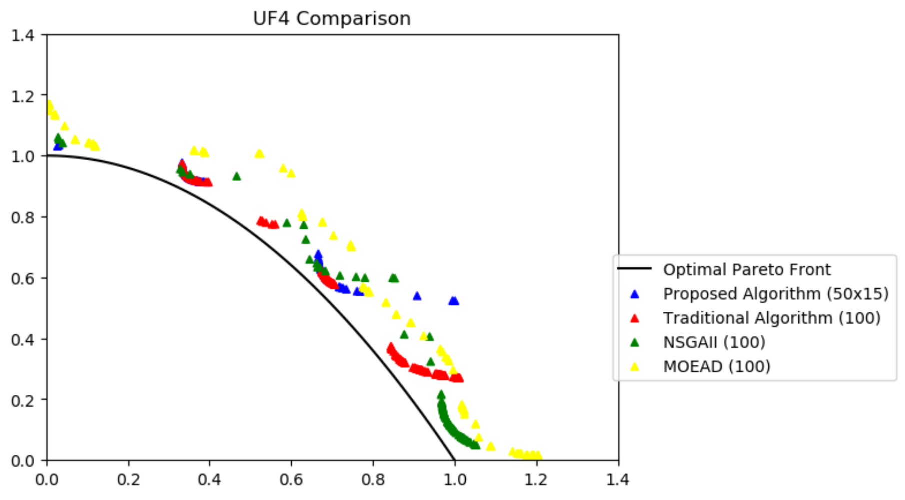


FIGURE 4.12: Comparison under Fair Evaluation-UF4

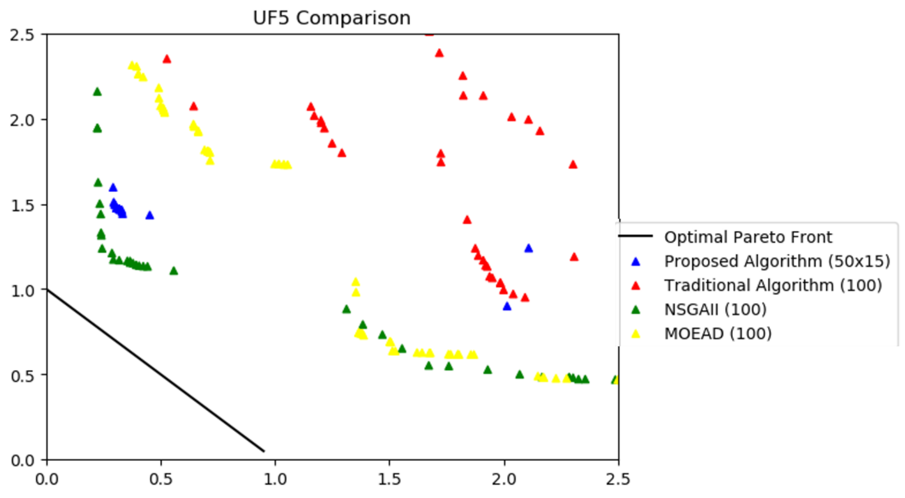


FIGURE 4.13: Comparison under Fair Evaluation-UF5

as reflected in Figure 4.19. NSGA-II and MOEAD obtained better hypervolume, but the GD is significantly worse than the proposed framework since the solutions are farther from the actual Pareto front.

In UF5 and UF6, the proposed framework had achieved the best values for hypervolume and GD compared to other algorithms. This is supported in the plots 4.20, 4.21 shown as well. The proposed framework solutions are the nearest to the Pareto front and more spread than the other algorithms. Figure 4.20, MOEAD solutions are not even visible in the scale shown and fall outside of it. While in Figure 4.22, the traditional MOGWO framework and the proposed framework seem to have a

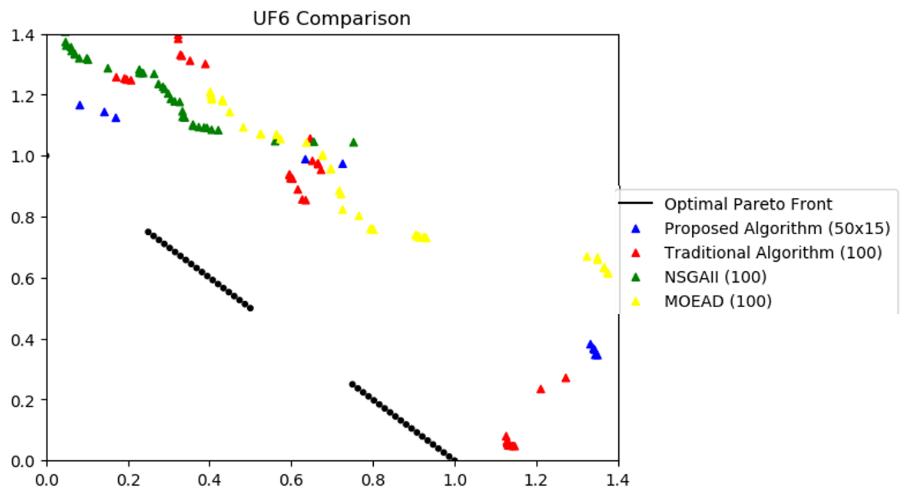


FIGURE 4.14: Comparison under Fair Evaluation-UF6

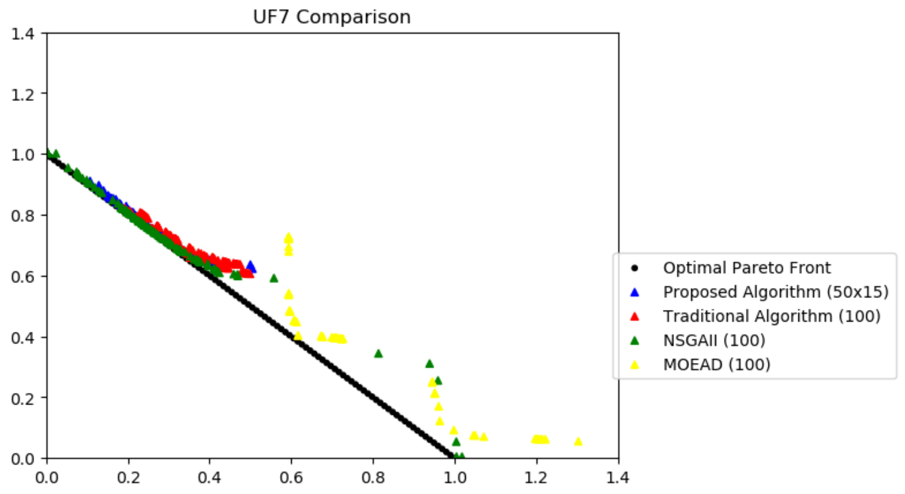


FIGURE 4.15: Comparison under Fair Evaluation-UF7

competitive performance, followed by NSGA slightly close to the actual Pareto front.

In UF7, the solutions received from MOEAD were comparatively farther from the Pareto front and hence had the worst GD value. The solutions provided by the proposed framework and MOGWO have a competitive comparison and are relatively close. Nevertheless, as seen in the table 4.2, the GD value of the proposed framework is almost half of the GD value of MOGWO.

Function	Metric	H-MPGWO	MOGWO	NSGAII	MOEAD
UF1	Hypervolume	**15.33**	15.25	15.01	13.54
	GD	0.04	**0.03**	0.35	0.68
UF2	Hypervolume	7.30	6.85	**8.01**	7.85
	GD	**0.01**	0.01	0.33	0.36
UF3	Hypervolume	23.07	**24.00**	21.28	19.71
	GD	**0.17**	0.30	0.77	0.77
UF4	Hypervolume	2.69	2.62	**3.09**	2.98
	GD	**0.07**	0.09	0.10	0.17
UF5	Hypervolume	**53.72**	50.50	49.28	43.75
	GD	**0.89**	1.34	2.29	3.13
UF6	Hypervolume	**218.02**	216.28	207.03	196.45
	GD	**0.39**	0.41	1.68	2.62
UF7	Hypervolume	14.26	**14.33**	15.04	12.83
	GD	**0.02**	0.03	0.29	0.73

Table 4.2: Comparison between proposed framework and state of the art methods; proposed framework is run for 50 iterations, MOGWO for 200 framework, and NSGA-II and MOEAD run for 100 iterations

4.2.4 Self-Comparison with Different Parameters

This comparison was performed to study the effects of various parameters on a 100 size population. The following settings were considered for this experiment:-

1. **S1**: The first setting is the baseline where the proposed framework was running for 50 iterations. This is the same setting we used in all our comparisons done prior.

2. **S2**: In this setting, the proposed framework runs for the same 50 iterations but without the elite component. This is done to show the effect of removing elitism.

3. **S3**: The third setting has 100 iterations and the same number of iterations in local populations (15).

4. **S4**: In the last setting, the proposed framework for 50 iterations with 30 inner iterations and in each local population.

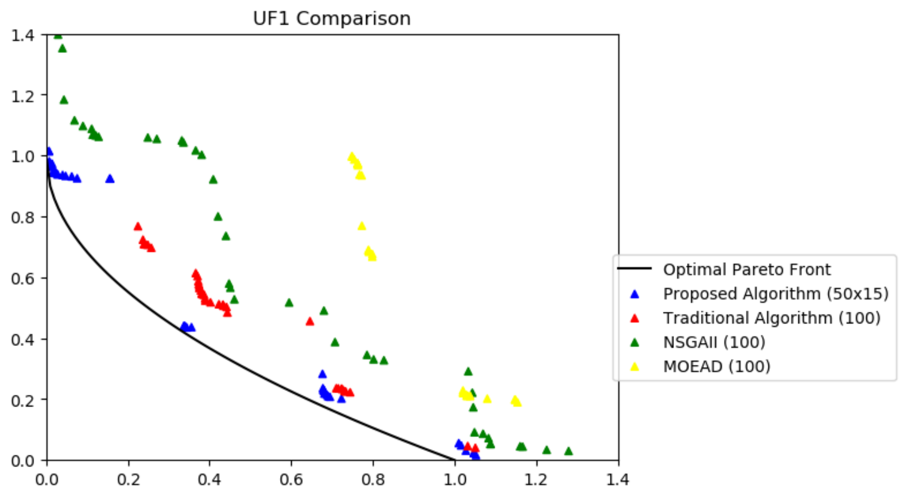


FIGURE 4.16: Comparison under Similar Time-UF1

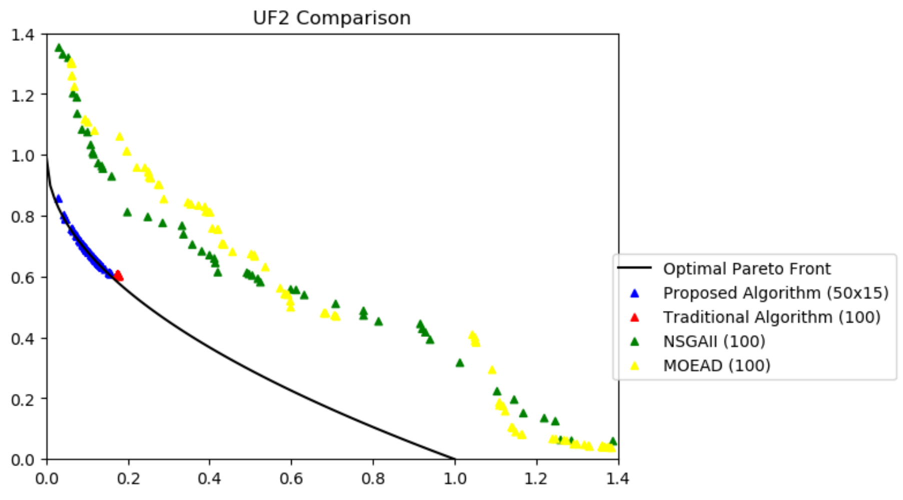


FIGURE 4.17: Comparison under Similar Time-UF2

The results are listed in Table 4.3, which shows the values of hypervolume and GD for each setting.

According to the results, the removal of the elite component in S2 increases the value of GD in all test cases except UF3 and UF5, where a decrease of 0.0041 and 0.0319, respectively, is observed. Hence, an elite component in the proposed framework allows a better convergence in most cases. Additionally, we see that the hypervolume does not change significantly in most cases. An increase or decrease can be because the removal of the elite group causes the population to explore instead of converging towards the optimal solutions, which may give a higher spread of solutions

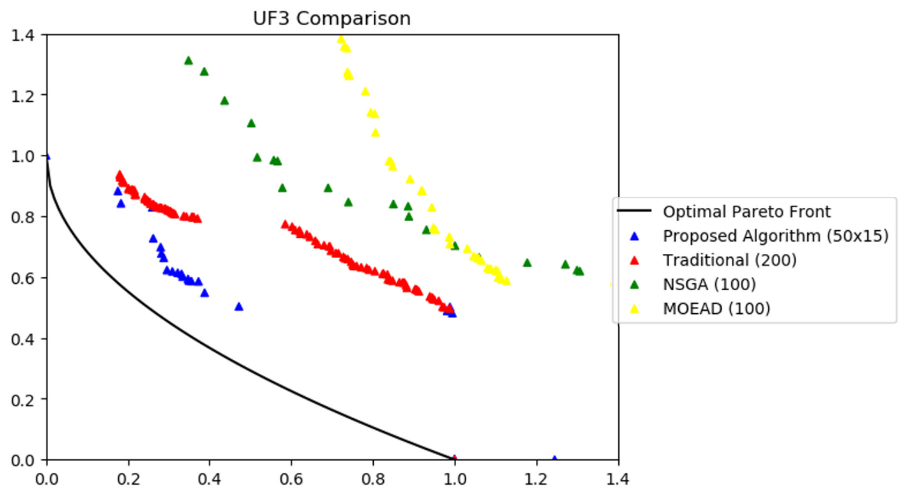


FIGURE 4.18: Comparison under Similar Time-UF3

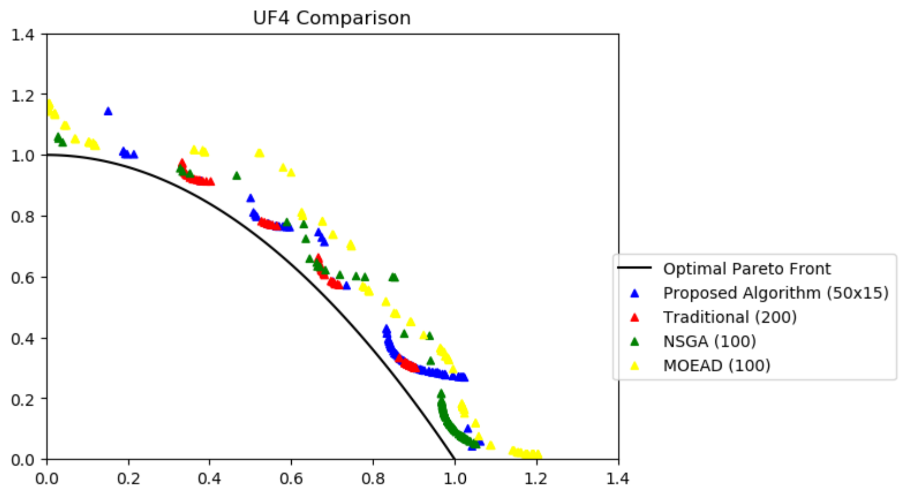


FIGURE 4.19: Comparison under Similar Time-UF4

but less optimal ones.

When increasing the number of iterations of the proposed framework, as done in S3, it is expected to obtain a better set of solutions, which can be seen as both hypervolume and GD have improved from S1 to S3. An exception can be seen in UF2, where even though the hypervolume improves, GD's value is also increased significantly, which means the average solution was less optimal in this case. This may be due to a significant number of solutions farther from the actual Pareto front than the closer ones.

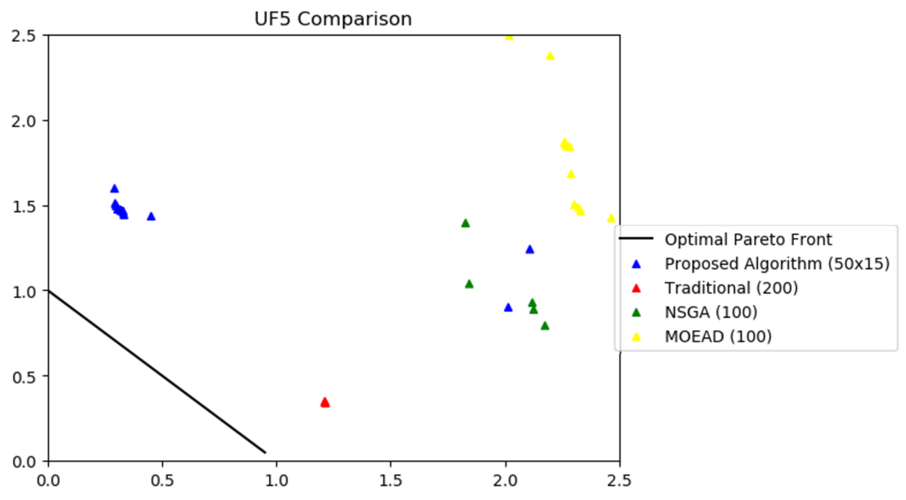


FIGURE 4.20: Comparison under Similar Time-UF5

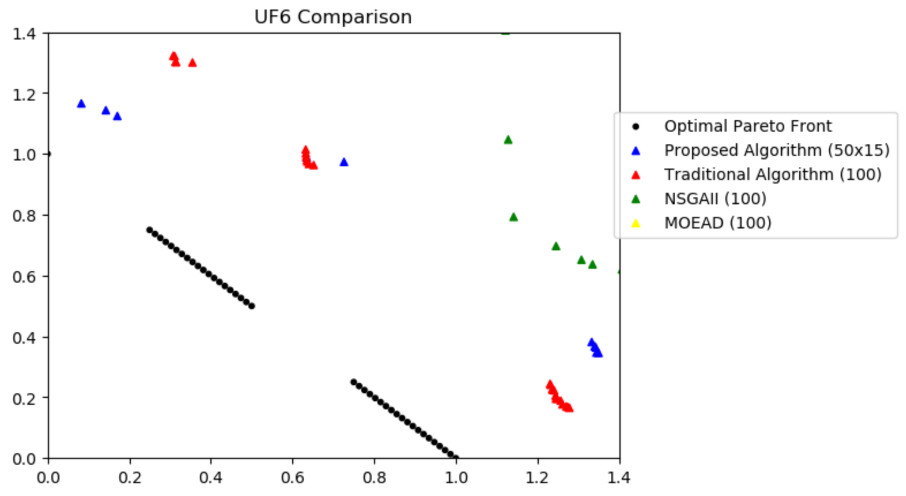


FIGURE 4.21: Comparison under Similar Time-UF6

A similar trend is seen from S1 to S4, where the local iterations in each population are increased. The exception is again UF2, where the hypervolume increases. However, GD's value worsens by a small margin, which can mean that though it provides a broader range of solutions, the average distance of these solutions from the actual Pareto front is also increased for this case. However, for the rest of the cases, it is seen that the obtained front is either more spread than those seen in S1, or the solutions are more closer to the obtained front and equally spread.

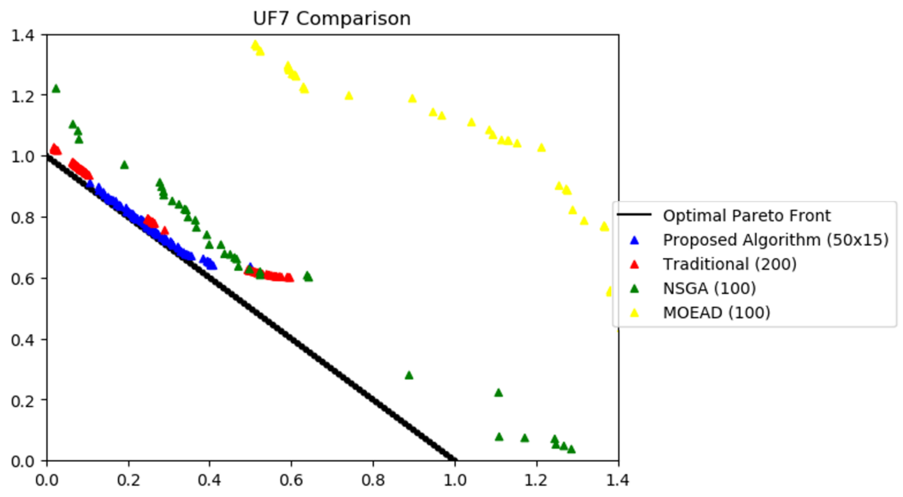


FIGURE 4.22: Comparison under Similar Time-UF7

Problem	Metric	S1	S2	S3	S4
UF1	Hypervolume	3.43	3.43	3.45	3.43
	GD	0.04	0.04	0.03	0.03
UF2	Hypervolume	7.31	7.46	7.36	7.32
	GD	0.01	0.01	0.02	0.01
UF3	Hypervolume	7.67	7.57	7.71	7.62
	GD	0.18	0.17	0.14	0.10
UF4	Hypervolume	2.69	2.84	2.79	2.77
	GD	0.07	0.07	0.06	0.06
UF5	Hypervolume	18.40	16.67	19.23	19.19
	GD	0.89	0.85	0.61	0.81
UF6	Hypervolume	2.31	2.12	2.30	2.34
	GD	0.39	0.46	0.39	0.36
UF7	Hypervolume	2.70	3.04	2.86	2.93
	GD	0.02	0.03	0.02	0.01

TABLE 4.3: Comparison for proposed framework at different settings

Chapter 5

Discussion

The results observed in Chapters 3 and 4 strongly indicate the superiority of the proposed frameworks over the traditional GWO. By introducing a co-operative multi-population structure, we can see an improvement in the overall performance because of the increase in flexibility and the independent local search performed by the four equal-sized populations. By using standard benchmarks provided by IEEE's Congress of Evolution Computation, the comparisons are made against not only the traditional grey wolf algorithms (GWO [9], and MOGWO [28]), but also state of the art approaches for single-objective and multi-objective problems.

We have also attempted to resolve and mitigate the poor performance caused by the inherent defects in GWO's evolution process [11]. This again supports our claim using the proposed framework, which has a co-evolving multi-population framework and is equipped with a mutation operator and elitism. These defects are covered for, and the performance is ultimately improved.

We also observe how individuals may have travelled from one population to another during the populations' regrouping. When we merge the populations, there may be an individual whose fitness may have changed drastically during the evolution and consequently were forced to change their population. For example, an individual in the Beta population became much more fit and ended up in the Alpha population.

An Alpha population individual may have strayed farther away from its position and is now a part of the Delta population.

Another way we migrate individuals externally is by using the guide/external leaders, as discussed. These leaders are the best individuals from Alpha, Beta, and Delta populations that take the spot of leaders in the Omega Population, depending on their comparative fitness values. This way, we allow a mutual-flow of information, an essential component of a co-operative multi-population structure.

Our proposed framework has some assumptions and limitations that we would like to discuss. The first being that we assume our problems to be static. We also assume that the problems are minimization problems, though there is no change in the framework in maximization problems. The framework has a limitation that it only caters to many-objective problems. The population division considers sorting the individuals based on their fitness and selecting a certain number of top individuals concerning each fitness function. Another limitation is that though parallelism can be implemented, it can not be utterly parallel since the omega population needs to wait for external leaders from the rest of the population. After careful analysis of the proposed algorithm, the proposed algorithm's time complexity is $O\left(M.N.log\left(N\right)+MN^2\right)$ for M objectives and N is the size of each population. Since we have limited maximum-sized archives, we do not expect the memory usage to be an issue for many individuals or objectives.

Chapter 6

Conclusion and Future Work

In this chapter, we will conclude our thesis and discuss future work that will be done. The first section concludes the thesis, discusses the results briefly and discusses whether the hypothesis was satisfied. The second section discusses the future work that will be done to cover the limitations and increase the work scope.

6.1 Conclusion and Future Work

To conclude our thesis, we introduced two novel multi-population frameworks for single and bi-objective optimization problems with a co-evolving structure and a hierarchy as observed in grey wolves. These frameworks consist of four populations - Alpha, Beta, Delta, and Omega; in the order of quality of solutions they contain. Each population performs an independent local search using a dedicated local search algorithm. Additionally, the search can learn from other populations using guide leaders, which are external individuals. To satisfy the pack structure, we designed the Alpha, Beta, and Delta populations to send a leader, which guides the search process performed in the Omega population. These populations search independently for some iterations, share their information by merging, and ultimately are regrouped into the four populations. This goes on until our defined stopping criteria is achieved.

The multi-objective problems are generally more complex than single-objective problems, which has been explained in Chapter 1. To handle this complexity in these problems, we incorporated archives in the framework for multi-objective problems. These archives are used to store Pareto optimal solutions that have been discovered throughout the search process. We introduced a global archive and local archive for each population, with fixed size and a density-based leader selection mechanism [26].

Our hypothesis has been tested by performing a comparison of various benchmarks that are provided by IEEE's CEC. The performance was compared with the traditional GWO and MOGWO and the state of the art methods for single and multi-objective problems. Various self-comparisons of the proposed framework were also performed at different parameters. The results support our claim of the significant improvement in performance accuracy and convergence rates, depicted by employing various performance metrics.

For our future works, we will aim to develop a more dynamic and versatile framework by incorporating an ensemble of mutation operators that will be used based on their reward values. This work will be carried forward to develop a framework for many-objective problems, with three or more objectives and dynamic optimization problems. We also target to incorporate constraints into our framework and evolve the population using a sophisticated constraint handling system. We will also target to have a dynamic hyper-parameter optimization, in order to have a dynamically self-optimizing framework.

Bibliography

[1] JJ Liang, BY Qu, PN Suganthan, and Alfredo G Hernández-Díaz. Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization. *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report*, 201212(34):281–295, 2013.

[2] Qingfu Zhang, Aimin Zhou, Shizheng Zhao, Ponnuthurai Nagaratnam Suganthan, Wudong Liu, and Santosh Tiwari. Multiobjective optimization test instances for the cec 2009 special session and competition. 2008.

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

[4] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, 1(1):67–82, 1997.

[5] Guohua Wu, Rammohan Mallipeddi, and Ponnuthurai Nagaratnam Suganthan. Ensemble strategies for population-based optimization algorithms–a survey. *Swarm and evolutionary computation*, 44:695–711, 2019.

[6] David H Wolpert and William G Macready. Coevolutionary free lunches. *IEEE Transactions on evolutionary computation*, 9(6):721–735, 2005.

[7] John Henry Holland et al. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence.* MIT press, 1992.

[8] James Kennedy and Russell Eberhart. Particle swarm optimization. In *Proceedings of ICNN'95-International Conference on Neural Networks*, volume 4, pages 1942–1948. IEEE, 1995.

[9] Seyedali Mirjalili, Seyed Mohammad Mirjalili, and Andrew Lewis. Grey wolf optimizer. *Advances in engineering software*, 69:46–61, 2014.

[10] Kenneth V Price. Differential evolution. In *Handbook of Optimization*, pages 187–214. Springer, 2013.

[11] Peifeng Niu, Songpeng Niu, Lingfang Chang, et al. The defect of the grey wolf optimization algorithm and its verification method. *Knowledge-Based Systems*, 171:37–43, 2019.

[12] Hui Xu, Xiang Liu, and Jun Su. An improved grey wolf optimizer algorithm integrated with cuckoo search. In *2017 9th IEEE international conference on intelligent data acquisition and advanced computing systems: technology and applications (IDAACS)*, volume 1, pages 490–493. IEEE, 2017.

[13] Cristian Muro, R Escobedo, L Spector, and RP Coppinger. Wolf-pack (canis lupus) hunting strategies emerge from simple rules in computational simulations. *Behavioural processes*, 88(3):192–197, 2011.

[14] Y. Diao, C. Li, S. Zeng, M. Mavrovouniotis, and S. Yang. Memory-based multi-population genetic learning for dynamic shortest path problems. In *2019 IEEE Congress on Evolutionary Computation (CEC)*, pages 2276–2283, 2019.

[15] Andrew William Hlynka and Ziad Kobti. Heritage-dynamic cultural algorithm for multi-population solutions. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 4398–4404. IEEE, 2016.

[16] Ben Niu, Yunlong Zhu, Xiaoxian He, and Henry Wu. Mcpso: A multi-swarm cooperative particle swarm optimizer. *Applied mathematics and computation*, 185(2):1050–1062, 2007.

[17] E. Emary, Hossam M. Zawbaa, and Aboul Ella Hassanien. Binary grey wolf optimization approaches for feature selection. *Neurocomputing*, 172: 371 – 381, 2016. ISSN 0925-2312. doi: https://doi.org/10.1016/j.neucom.2015.06.083. URL `http://www.sciencedirect.com/science/article/pii/S0925231215010504`.

[18] C. Han, M. Chen, L. Pan, and X. Chen. A community detection algorithm by utilizing grey wolf optimization. In *2017 9th International Conference on Modelling, Identification and Control (ICMIC)*, pages 567–572, 2017.

[19] Hamouda Chantar, Majdi Mafarja, Hamad Alsawalqah, Ali Asghar Heidari, Ibrahim Aljarah, and Hossam Faris. Feature selection using binary grey wolf optimizer with elite-based crossover for arabic text classification. *Neural Computing and Applications*, pages 1–20, 2019.

[20] İlker Gölcük and Fehmi Burcin Ozsoydan. Evolutionary and adaptive inheritance enhanced grey wolf optimization algorithm for binary domains. *Knowledge-Based Systems*, page 105586, 2020. ISSN 0950-7051. doi: https://doi.org/10.1016/j.knosys.2020.105586. URL `http://www.sciencedirect.com/science/article/pii/S0950705120300666`.

[21] Friedman Milton. A correction: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association. American Statistical Association*, 34(205):109, 1939.

[22] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and Tanaka Meyarivan. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: Nsga-ii. In *International conference on parallel problem solving from nature*, pages 849–858. Springer, 2000.

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

[24] N. Srinivas and Kalyanmoy Deb. Multiobjective function optimization using nondominated sorting genetic algorithms. 2, 06 2000.

[25] Yuan Yuan, Hua Xu, and Bo Wang. An improved nsga-iii procedure for evolutionary many-objective optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, pages 661–668, 2014.

[26] CA Coello Coello and M Salazar Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, pages 1051–1056. IEEE, 2002.

[27] Guangyuan Fu, Chao Wang, Daqiao Zhang, Jiufen Zhao, and Hongqiao Wang. A multiobjective particle swarm optimization algorithm based on multipopulation coevolution for weapon-target assignment. *Mathematical Problems in Engineering*, 2019, 2019.

[28] Seyedali Mirjalili, Shahrzad Saremi, Seyed Mohammad Mirjalili, and Leandro dos S Coelho. Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. *Expert Systems with Applications*, 47:106–119, 2016.

[29] Chao Lu, Liang Gao, Xinyu Li, and Shengqiang Xiao. A hybrid multi-objective grey wolf optimizer for dynamic scheduling in a real-world welding industry. *Engineering Applications of Artificial Intelligence*, 57:61–79, 2017.

[30] Sami Kahla, Youcef Soufi, Moussa Sedraoui, and Mohcene Bechouat. Maximum power point tracking of wind energy conversion system using multi-objective grey wolf optimization of fuzzy-sliding mode controller. *International Journal of Renewable Energy Research (IJRER)*, 7(2):926–936, 2017.

[31] Ali Behnood and Emadaldin Mohammadi Golafshani. Predicting the compressive strength of silica fume concrete using hybrid artificial neural network with multi-objective grey wolves. *Journal of Cleaner Production*, 202:54–64, 2018.

[32] David A Van Veldhuizen. Multiobjective evolutionary algorithms: classifications, analyses, and new innovations. Technical report, AIR FORCE INST OF TECH WRIGHT-PATTERSONAFB OH SCHOOL OF ENGINEERING, 1999.

[33] Eckart Zitzler and Lothar Thiele. Multiobjective optimization using evolutionary algorithms—a comparative case study. In *International conference on parallel problem solving from nature*, pages 292–301. Springer, 1998.

[34] Handing Wang, Shan He, and Xin Yao. Nadir point estimation for many-objective optimization problems based on emphasized critical regions. *Soft Computing*, 21(9):2283–2295, 2017.

Vita Auctoris

NAME:	Nimish Verma
PLACE OF BIRTH:	New Delhi, India
YEAR OF BIRTH:	1997
EDUCATION:	The Heritage School, Rohini, Delhi, India High School, CBSE, 2013-2015
	Guru Gobind Singh Indraprastha University, Delhi, India Bachelor's of Technology, Computer Science Engineering 2015-2019
	University of Windsor, Windsor ON, Canada Master's of Science, Computer Science 2019-2020