Objective Reduction in Many-Objective Optimization Problems

Arpi Sen Gupta

Submitted in partial fulfilment
of the requirements for the degree of

Master of Science

Faculty of Mathematics and Science
Brock University
St. Catharines, Ontario

Abstract

Many-objective optimization problems (MaOPs) are multi-objective optimization problems which have more than three objectives. MaOPs face significant challenges because of search efficiency, computational cost, decision making, and visualization. Many well-known multi-objective evolutionary algorithms do not scale well with an increasing number of objectives. The objective reduction can alleviate such difficulties — however, most research in objective reduction use non-dominated sorting or Pareto ranking. However, Pareto is effective in problems having less than four objectives. In this research, we use two approaches to objective reduction: random-based and linear coefficient-based. We use the sum of ranks instead of Pareto Ranking. When applied to many-objective problems, the sum of ranks has outperformed many other optimization approaches. We also use the age layered population structure (ALPS). We use ALPS in our approach to remove premature convergence and improve results. The performance of the proposed methods has been studied extensively on the famous benchmark problem DTLZ. The original GA and ALPS outperform the objective reduction algorithms in many test cases of DTLZ. Among all reduction algorithms, a linear coefficient based reduction algorithm provides better performance for some problems in this test suite. Random based reduction is not an appropriate strategy for reducing objectives.

Acknowledgements

At first, I will thank my God to help me keep my patience throughout the journey. I want to thank my supervisor, Prof. Brian Ross, for providing his excellent support, cooperation, and funding in my entire thesis and study. I am thankful to my supervisory committee for sharing their knowledge. I am also grateful to Anthony Awuley (ALPS implementation), Michael Gircys (sum of ranks implementation) and Kyle Harrison for their supports. I would like to thank the Computer Science Department at Brock University for providing excellent education and environment to achieve this goal. At last, I would like to thank my loving husband and kid for giving their continuous support and belief in me.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Problem Description

Many objective optimization problems (MaOPs) refer to optimization problems that contain four or more objectives. Recently, Maops are getting much attention as many real-world applications demand effective solutions that involve many criteria. A myriad of research is being conducted to effectively optimize MaOPs. Evolutionary algorithms (EAs) are a promising approach in solving MaOPs. EAs can deal with problems having different objectives residing in different search spaces, such as convex, nonconvex, multi-model, discontinuous, and degenerate space. However, it is evident that a large number of objectives affects the performance of EAs. It is also difficult to visualize solution trade-offs as well as high computational costs. Different ideas have been explored to overcome these obstacles [39, 30].

It is necessary to mention that all objectives in many MaOps are not equally important. If it is possible to divide the objectives set into a redundant set and an essential set then one can optimize the the essential objective set, since optimization will be simpler. In this regard, objective reduction is a possible way to deal with all difficulties that arise with MaOPs having a large number of objectives. Research [15] has explored appropriate methods for reducing objective set. Feature selection is one of the popular methods for objective reduction. Two methods of feature selection are filter and wrapper methods. These two methods help to find essential and redundant objective sets. Wrapper methods use error rates to score the subset, while filter methods use proxy measures to score the features . The backward selection algorithm (BSS) is a popular wrapper method, where objectives are reduced sequentially. For example, linear correlation coefficient determines the coefficient values between objectives and can be used to sequentially select objectives to remove. BSS is useful for large numbers of objectives.

The effectiveness of a MaOP also depends on which objective fitness strategy is used. Fitness strategies for MaOPs can be categorized into three main types: Pareto dominance based many-objective evolutionary algorithms (PDMOEAs), indicator-based many-objective evolutionary algorithms, and decomposition-based evolutionary algorithms [34]. Among these, PDMOEAs are using extensively, as they are based on the concept of non-dominated sorting of solutions. In non-dominated sorting, solutions are ranked as 1 when they are not dominated by any other solutions. The non-dominated solutions in the remaining set are ranked as 2, and the process continues until all solutions are ranked. This focuses on the essential attributes of many-objective optimization problems, where one solution cannot improve without worsening another one. Therefore, PDMOEAs are widely using as they can overcome some contradictory challenges of MaOPs. However, it is difficult to maintain the diversity of solutions that converge towards the true Pareto front. Also, PDMOEAs can only provide a diverse solution set when the number of objectives is not more than three. With a large number of objectives (four or more), the convergence ability of PDMOEAs decreases drastically, because large number of objectives causes exponential increase in the number of non-dominated solutions and the PDMOEAs rank all the population as rank 1. As a result, Pareto dominance flattens out, and no selection pressure is assigned by the fitness assignment process.

It is necessary to rank all the population effectively to get globally optimal solutions. Assuming the importance of a practical ranking approach, Bentley and Wakefield proposed the average ranking (sum of ranks or SR) as a fitness assignment technique [10]. SR favours individuals that are good performers in as many objectives as possible. In comparison with Pareto ranking, a solution can be non-dominated if just one objective is non-dominated. In SR, all of the objectives are ranked separately, and the sum of these ranks is assigned as the fitness. SR gives satisfactory results compared with Pareto dominance when the number of objectives increases. However, the solution derived by the SR converge to only a sub-area of the Pareto front and the solutions obtained are not diverse as Pareto solution [29]. Also, SR gives only one solution per run it is not be possible to compare some aspects of performance with Pareto-based approaches. To compare the result of PDMOEA and SR, we need solutions from many SR runs (one solution per run). However, there is a possibility that a significant number of SR solutions from different SR runs are attracted to the same sub-area of the Pareto front. In this situation solutions from different runs will not be varied as the Pareto solution set.

An EA named the age layered population structure (ALPS) genetic algorithm [27] focuses on reducing premature convergence of evaluation. ALPS has some specific attributes which make it significantly different from other EAs. Attracting local optima is one of the main reasons that EAs converge early. ALPS helps the EAs maintain diversity using age layers which restrict selection for reproduction. ALPS also generates new individuals in the bottom layer at regular intervals, which adds genetic diversity throughout the run. Using ALPS and SR, it may be possible to optimize solutions while improving the generated set of solutions. However, it is unlikely that ALPS can produce solutions as diverse as PDMOEAs.

1.2 Goals and Motivation

We know Pareto ranking cannot scale well when the number of objectives increases. Research shows that objective reduction is a useful technique for machine learning to handle many real-world applications [15, 39, 30]. To make EAs more effective for complex MaOPs, many objective reduction methods have been proposed. The primary goal of this thesis is to explore new strategies for MaOPs. We want to make EAs more effective by using the sum of ranks instead of Pareto ranking. We know that the SR can provide a better result for a large number of objectives. SR with ALPS can create a more optimmized solution for MaOPs. We also want to use different strategies of objective reduction using backward sequential selection algorithm (a feature subset selection algorithm) [6] . We will compare random and linear coefficient reduction techniques in our research. Moreover, we want to incorporate SR and ALPS with the above described objective reduction strategies. Our other goal is to study the DTLZ [22] benchmark problem set and compare our result for different objective reduction approaches.

1.3 Thesis Structure

At the end of this chapter we provide a table of all acronyms. We provide background knowledge in Chapter 2. Chapter 3 gives a literature review. Chapter 4 will introduce the experiments that we conducted, and in particular, ideas about objective reduction. Chapter 5 will provide details and discussion of experiments results, and statistical significance of results. Finally, we include our conclusions and future work in Chapter 6.

Please see Table 1.1 for acronyms used throughout the thesis.

Common acronyms	**Abbreviation**
GA	Genetic Algorithm
ALPS	Age Layered Population Structure
BSS	Backward Sequential Selection
NSGA-II	Non-dominated Sorting Genetic Algorithm -II
NSGA-III	Non-dominated Sorting Genetic Algorithm -III
DTLZ	Benchmark Problem Suite
PCC	Pearson Correlation Coefficient
PCA	Principle Component Analysis
SR	Sum of Ranks
IGD	Inverted General Distance
VRPTW	Vehicle Routing Problem With Time Windows
SBX	Simulated Binary Crossover

Table 1.1: Common Acronyms

Chapter 2

Background

2.1 Introduction

The chapter includes relevant background information on many-objective optimization and various fitness evaluation techniques for these problems. Overviews of the genetic algorithm (GA), age-layered population structure (ALPS) are provided. Lastly, this chapter discusses an idea about benchmark problems studied and performance measurements of many-objective optimization problems.

2.2 Many-objective Optimization Problems

Many-objective optimization problems (MaOPs) refer to the class of problems having four or more objectives with many decision variables and constraints. A general MaOPs can be mathematically formulated as

$$\begin{aligned} Minimize/Maximize\, f_m(\mathbf{x}),\ \ m = 1, 2, ..., M; \\ subject\, to\, g_j(\mathbf{x}) \geq 0,\ \ j = 1, 2, ..., J; \\ h_k(\mathbf{x}) = 0,\ \ k = 1, 2, ..., K; \\ x_i^{(L)} \leq x_i \leq x_i^{(U)},\ \ i = 1, 2, ..., n. \end{aligned} \tag{2.1}$$

where $\mathbf{x}$ is a decision vector, $g_i(\mathbf{x})$ and $h_k(\mathbf{x})$ are constraint functions [20]. This multi-objective minimization definition defines a lattice structure, and Pareto dominance is usually used to define this structure (see Section 2.3.2).

The main difficulty in MaOPs is finding the best solution set which optimizes all objectives concurrently. One of the simple methods of solving MaOPs is converting MaOPs problems into a series of single-objective optimization problems (SOO). This procedure uses a suitable way to optimize the scalarization function. A scalarization function is a function of the original objectives. However, this simple approach cannot eliminate the complexities in solving complex SOOs [36].

Another method based on the role of the decision maker (DM) can be adopted for solving MaOPs. The DM helps to select a solution using their experience, considering constraints and other perspectives of MaOPs [36]. Based on the DM role, the optimization methods of MaOPs can be categorized into two categories:

1. Generating method
2. Preference-based method

The generating method provides a solution set without the interference of DM, whereas the preference method uses references provided by the DM. The main three sub-categories of the generating method are no-preference method, posteriori methods with scalarization approach, and posteriori methods with a multi-objective approach. Among these, no reference methods do not require any input from the DM. An example of posteriori method with scalarization approach is the weighting method, which needs a solution from SOO. It is valid for the problem with few objectives, but it is difficult to select suitable values for weights [36].

The posteriori method with for MaOPs provides Pareto optimal solutions and more information for the DM. Based on the role of the DM, the preference-based process has been categorized into two parts: priori methods and interactive methods. Priori methods solve SOO problems which need a function formulated with original objectives and reference from the DM. Dependency on the DM can result in scarcity of knowledge and other obstacles. In an interactive approach, a solution can be provided by interacting with the DM while evaluating the MaOPs. The DM plays a promising role to find a computationally efficient solution by using these references in every iteration of the problem evaluation [36].

2.3 Fitness Evaluation Techniques For MaOPs

2.3.1 Weighted-Sum

The weighted sum approach converts multiple objective scores into a single fitness [10]. In this technique, some objectives get higher priority, which creates bias in the solution. The difficulty in determining the appropriate weights is another disadvantage of this approach.

For example, let **X** be a vector with $x_1,\ x_2,\ x_3, ...,\ x_n$ individuals, where n is the number of objectives. If we assign a weight $w_1,\ w_2,\ w_3, ...,\ w_n$ for each of the n objectives then the fitness score W(**X**) will be:

$$W(\mathbf{X}) = \sum_{i=1}^{m} w_m x_m \tag{2.2}$$

A solution with a minimum weighted sum is a more optimal solution. And a vector **X**$\leq$ **Y** if W(**X**) is less or equal to W(**Y**).

2.3.2 Pareto Ranking

Pareto ranking represents an important technique of MaOPs, where it is not possible to improve one objecive without making another objective worse. In MaOPs, sometimes one cannot optimize all objectives simultaneously. Pareto ranking provides non-dominated solutions which are known as the Pareto front and is defined as follows [2].

Pareto Dominance: Let $\mathbf{X}_i$ **and** $\mathbf{X}_j$ be two solutions vector where $\mathbf{X}_i$**,** $\mathbf{X}_j \in \mathcal{F}$ and $|\mathbf{X}_i|$ =$|\mathbf{X}_j|$ = M. We can say that $\mathbf{X}_i$ dominates $\mathbf{X}_j$ if and only if:

$$\forall_m \in \{1,2,..,M\} : f_m(\mathbf{X_i}) \leq f_m(\mathbf{X_j}) \wedge \exists_m \in \{1,2,..,M\}\, f_m(\mathbf{X_i}) < f_m(\mathbf{X_j})\ [i \neq j] \tag{2.3}$$

In other words a solution is dominated by another solution if it is better in at least one objective and not worse in the other objectives, where f_m is the objective function [24].

Using Pareto dominance, we next define Pareto ranking. First, we arrange the solutions in non-dominated order, and give a rank of 1 to the first non-dominated set. Then the solutions of first non-dominated set are removed, and the next set of non-dominated is given a rank of 2. The process continues until all solutions are ranked. Pareto ranking can indicate Pareto non-dominated solutions, where every objective gets the same priority. However, with four or more objectives, it is increasing likely that all solutions become non-dominated and have a rank of 1. Therefore, Pareto ranking cannot work well with a large number of objectives as selective pressure vanishes [28].

Two more definitions are as follows. The Pareto front is the solution set currently being chosen that is Pareto non-dominated. The true Pareto front is the set of optimal non-dominated solutions in the problem being studied.

2.3.3 Normalized Sum Of Ranks

An alternative strategy for many-objective optimization was introduced by Bentley and Wakefield [5]. Average ranking or sum of ranks (SR) outperforms Pareto ranking when a large number of objectives is involved. SR does not give a set of non-dominated solutions like Pareto ranking, but instead, gives one solution at a time. The mechanism of SR calculation as follows

1. Compute the raw score of all objectives values for all individuals.
2. Rank objectives based on minimization (or maximization) of objectives values.
3. Makes a summation of each individual's rank vector according to formula 2.4 (below).

If we consider a solution x with the set of ranks (R_{x1}, R_{x2}, R_{x3}, ..., R_{xm}), where R_{xj} is the rank of x for jth objective and m is the number of objectives, then the sum of rank is calculated as :

$$SR(x) = \sum_{j=1}^{M} R_{xj} \tag{2.4}$$

In SR, unfair distribution of scores can happen in (2.4). The reason is that some objectives have more ranks than others and they can create bias in the sum. For example, if objective A has more ranks (less repetition of objective scores) than objective B, then B's contribution in the sum is dwarfed by A's score. To overcome this, it is necessary to normalize objective scores. In normalization, we find the highest rank of each objective, and divide each rank of that objective by this value. This normalizes each objective score between 0 and 1. It thus scales all ranks to the same range, and removes unfair distribution caused by a different magnitude ranges of ranks. After normalization, a summation of the normalized objective values for each individual is calculated and used for fitness.

In Table 2.1, we consider 3 objectives for the sum of ranks. First, we compute the fitness per objective and rank all individuals in the population. We calculate the sum of ranks and perform a re-rank. We see that, objective B has more repetitive values than other two objectives. Because of this imbalance, B's contribution will be overwhelmed by the other objectives. To overcome this situation, we normalize the objectives scores in Table 2.2 and make a summation of them. We rank the individuals based on their normalized sum of ranks. The solutions with rank 1 will be considered as better solution in whole population.

Individual	Obj(A)	Obj(B)	Obj(c)	R(A)	R(B)	R(C)	SR	Re-rank
1	1	9	5	2	1	2	5	1
2	2	100	4	3	2	1	6	2
3	10	9	9	4	1	4	9	3
4	16	100	8	5	2	3	10	4
5	16	9	500	5	1	5	11	5
6	0	1000	1000	1	3	6	10	4

Table 2.1: Sum of Ranks [33]

Individual	NR(A)	NR(B)	NR(C)	NSR	Re-rank
1	0.4	0.333	0.333	1.066	1
2	0.6	0.6667	0.1667	1.4337	2
3	0.8	0.333	0.6667	1.7997	3
4	1	0.6667	0.5	2.1667	4
5	1	0.333	0.833	2.166	4
6	0.2	1	1	2.2	5

Table 2.2: Normalized Sum of Ranks [33]

2.4 Genetic Algorithm

The genetic algorithm (GA) is an evolutionary algorithm inspired by the Darwin's theory of natural selection. It was invented in 1960 by John Holland [26]. Using this algorithm, fittest individuals are selected to make offspring for future generations based on natural selection. GA is popular for solving search and optimization problems. It uses biologically inspired idea such as selection, crossover, and mutation to provide high-quality solutions.

Figure 2.1 shows the entire process of the GA. The GA starts with the creation of an initial population, which represents potential solutions. Based on the problem definition, the fitness of every individual is calculated to identify the relative potential strengths of each individual.

Fitness proportional selection operators such as tournament selection or roulette-wheel selection are used to determine the parents for creating offspring using crossover and mutation operator. New populations are updated with new offspring. The procedure will continue until an ideal solution is found, or the maximum number of generations is reached.

Some more details of the GA are as follows:

Representation and initial population: Initial population consists of randomly generated integers, floating-point or binary strings called chromosomes. It is

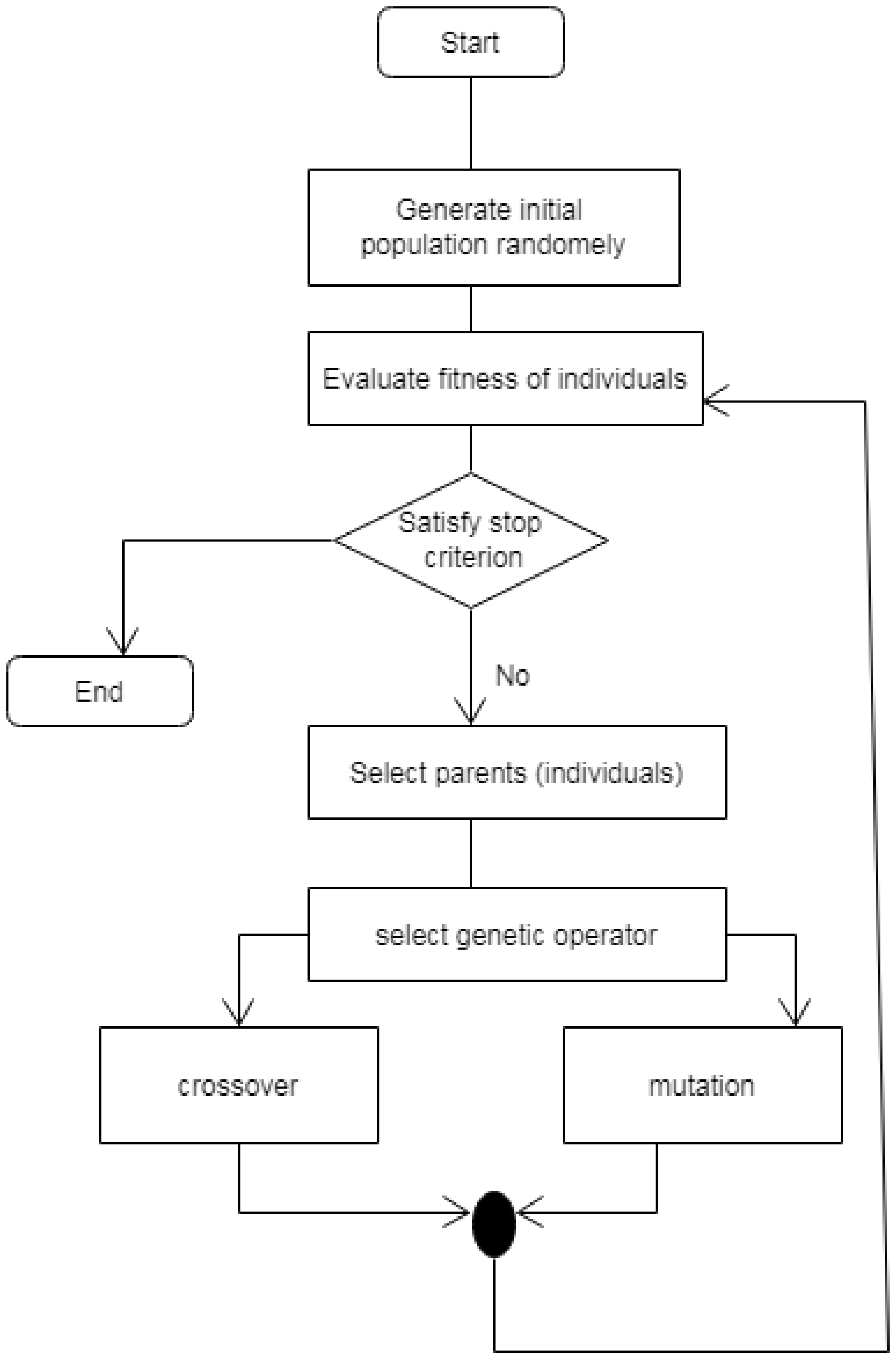
Start
Generate initial population randomely
Evaluate fitness of individuals
Satisfy stop criterion
End
No
Select parents (individuals)
select genetic operator
crossover
mutation

Figure 2.1: Work Flow of Genetic Algorithm

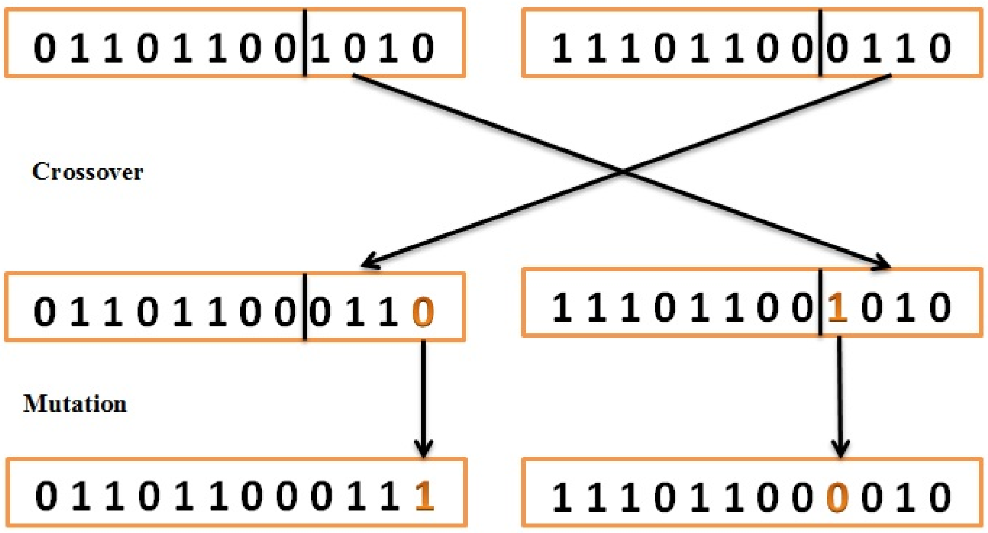

Figure 2.2: Crossover and Mutation

necessary to represent each chromosome based on the aspect of the problem being optimized.

Fitness Based Selection Operator: The fitness based selection operator called tournament selection is used to select an individual based on fitness to produce offspring. It selects K (from user parameter) random individuals and keeps the best 2 for crossover.

Reproduction Operator: Crossover and mutation are used to generate new offspring from selected parents. In crossover, two individuals pair to recombine their genes to create new offspring. Genes from parent chromosomes are mated using random crossover points. For example, in Figure. 2.2, random slices of the chromosomes are exchanged between the parents. By pairing genes, two new individuals are created. Mutation introduces randomness in the search space. In mutation, one offspring is created by flipping a random bit in the parent .

Algorithm 1 shows the pseudo-code for a simple GA. In this algorithm, the initial population is created with random individuals. Fitness of all individuals is computed by the fitness function. The fittest (elite) individuals in population, are transferred to the next generation. After that, the selection procedure is applied to select parents from the current generation to make offspring using crossover and mutation. Finally the existing population is replaced by the new population. The GA will continue until the maximum generation is reached or a solution is found.

Algorithm 1 Pseudocode for GA

```
1: procedure GA
2:     Randomly generate individual to make population
3:     while until number of generation or cannot find ideal solution do
4:         Evaluate chromosome fitness
5:         Select elite individual
6:         Select parent for reproduction
7:         Do crossover and mutation to generate offspring
8:         replace old population with new population
9:     end while
10: end procedure
```

2.5 Age-Layered Population Structure

Age layered population structure (ALPS) was invented by Hornby [27] to reduce premature convergence in meta-heuristic search using an evolutionary algorithm. The main differences between ALPS with the traditional genetic algorithm are (a) it periodically generates new individuals in the population and (b) it segregates individuals in different layers according to their age. Using the age of an individual, ALPS restricts how individuals compete and breed with other individuals. Hornby applied ALPS in many problems, and results showed that it performed well compared to GA.

A randomly generated individual starts their age from age 0, as their genetic material has just been introduced into the population. An individual's age is counted by how long it has been used as a parent in the run. The individuals who came from reproduction, such as mutation and crossover, take the age of their oldest parent plus one. The individuals who are used as a parent to create offspring get one added to their previous age, as their genetic material has been used to generate offspring.

The ALPS scheme separates individuals into age layers (see Table 2.3). These values are multiplied by an age gap parameter to define the maximum age of individuals in each layer. For example, given a polynomial aging scheme in Table 2.3 with an age gap of 7 and 6 layers, the maximum ages for the layers will be 7, 14, 28, 63, 112. There is no age limit for top layer. A corresponding layer opens when the preceding layer reaches its maximum age. For example, layer 1 will open in age 8 as layer 0 reaches its maximum age. Individuals move into the next layer when they reach maximum age of the layer. ALPS periodically generates random individuals

Age-Scheme	Layer					
	0	1	2	3	4	5
Linear	1	2	3	4	6	7
Fibonacci	1	2	3	5	13	21
Polynomial (n^2)	1	2	4	9	16	25
Exponential (2^n)	1	2	4	8	16	32

Table 2.3: Age Scheme for ALPS [27]

in the lowest layer, which directs the search into new directions, and reduces the probability of premature convergence. The individuals in a higher layer always can be replaced with individuals from the lower layer. An individual can stay in the top layer indefinitely [27].

Another important characteristic of ALPS is that individuals only compete and mate with individuals from the same layer or the preceding lower layer. As a result, an individual with high fitness cannot dominate the entire population. This prevents ALPS converging prematurely.

Algorithm 2 describes the ALPS algorithm [9]. Like the GA, the initial population is generated randomly. If generation is zero or layer 0 is in initialize mode, a new population will be generated in layer 0. The population is evaluated for every layer. If the individuals of layers are involved in reproduction, age of offspring and parents will be updated. If individuals reach the maximum age of the layer they move to next layer. The process continues until the ideal solution found or maximum number of generation is reached.

Algorithm 2 Pseudocode for ALPS

```
1: procedure ALPS( )
2:     Read parameter file
3:     SetLayer(number of layers, age gap, aging scheme)
4:     while number of generation or !termination criterion met do
5:         if generation==0 or layer0 is in initialize mode then
6:             Randomly generate individual in layer 0
7:         else
8:             EvaluatePopulation()
9:         end if
10:    end while
11: end procedure
```

```
procedure EvaluatePopulation( )
    if (layer0) then
        Select individual from current layer
    else
        Select individual from current layer,current layer-1
    end if
    Do crossover
    Do mutation
    Update parents(Individual) age
    Set offspring(Individual) age
    if (individual.age>layer.agelimit) then
        move to upper layer
    end if
end procedure
```

2.6 Non-dominated sorting genetic algorithm (NSGA-II)

In the non-dominated sorting approach, solutions are divided into the different non-dominated sets, and given a different label in each non-dominated set [28]. First, solutions are searched based on the non-dominated method (one objective cannot be improved without worsening other objectives) and this solution are defined set by label “first non-dominated set” and removed from the solution set. Non-dominated sorting is continued to find the second non-dominated set, and labeled as “second non-dominated set”. Both non-dominated sets are removed from the solution set, and non-dominated sorting continues until all solutions are labeled and so fort.

NSGA-II is one of the popular non-dominated sorting GA which eliminates three difficulties of normal GA [18]. NSGA-II uses many sophisticated mechanisms to provide solutions that are well spread and closer to the Pareto front. In this research, they improved computational complexity, remove non-elitism approach, and the dependence of sharing parameter. In this algorithm, the crowding distance metric has been used as a diversity maintenance operator. Crowding distance of a particular solution is calculated by taking the average distance of its neighbouring solutions. NSGA-II also includes elitism, which helps to improve the GA’s performance. The main difference with a regular GA is that, NSGA-II uses elitism to create a 2N-size population with the current population N and its best individuals. After comparing solutions, it keeps the best N solutions for future generations. Solutions are analyzed using tournament selection using non-dominated sorting and crowding distance between solutions.

2.7 Objective Reduction

The challenges for MaOPs increases exponentially with the growth of the number of objectives. It also involves high computation costs to optimize all the objectives together. Also, the visualization of high dimensional solution spaces become impossible. Objective reduction can be a useful solution in this situation. Objective reduction strategies focus on redundant and essential objective sets to make the objective set smaller and easier to optimize. The essential objective set is capable of making the reduced Pareto front similar to the Pareto front of the original objective set. That way, it will retain all of the characteristics of the original Pareto front. This way, by reducing the redundant set, it is possible to eliminate many of the difficulties of MaOPs.

2.7.1 Wrapper Method vs. Filter Method

Feature selection is one of the simpler methods for objective reduction, that helps with data visualization, and improves performance. There are two methods considered for feature selection: filter method and the wrapper method [37]. The difference between these two methods is as follows. The wrapper methods depends on the optimization algorithm for selecting features and uses the algorithm to decide which objectives to reduce. methods consider all combinations of features, calculate their values and select the best solution from these combinations. Filter methods use general characteristics of the training data to select features to reduce. Wrapper methods can provide better performance, but are more computationally expensive than filter methods. Filter methods are useful for reducing features from extensive feature sets. In filter methods, features are ranked based on the selection criteria, and features are removed based on having low rank. The main disadvantage of filter methods is that they ignore the interaction between objectives and the optimization algorithm [37].

2.7.2 Backward Sequential Selection

Backward sequential selection (BSS) [6] is a popular method for feature selection (later, we reduce the objective set in our runs using the BSS strategy). In this method, an initial essential objective set starts with all objectives. Then redundant objectives are removed sequentially. The final essential set will result after removing the desired number of redundant objectives. This method is effective when the objective sets are large. The main difficulty of BSS is that it cannot evaluate the objective after it has been discarded, as it is a “greedy” reduction method.

2.7.3 Pearson Correlation Coefficient

Pearson correlation coefficient (PCC) is also known as linear correlation coefficient. It calculates a linear correlation between two variables X and Y. The value of PCC lies between -1 and 1. The value 1 represents positive correlation, 0 means no correlation, and -1 represents a negative correlation. The PCC for a sample of data can be calculated using the following equation:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{2.5}$$

where n is the sample size, x_i, y_i are sample point indexd with i and $\bar{x}$,$\bar{y}$ are sample mean [3].

2.8 DTLZ Benchmark Suite

To learn about the performance of an optimization algorithm, it is necessary to choose a test problem. According to [22], the main characteristics of a test problem should be:

1. Scalability of any number of objectives and decision variables.
2. Test problems must have difficulties similar to real-world problems.
3. Test problems should satisfy two main goals of many-objective optimization such as convergence near the true Pareto optimal front, and diversity of solutions. Test problems also should introduce hindrance to get widely distributed Pareto optimal solutions.
4. The availability of decision variables is another important factor of many-objective optimization test problems. They should have an exact Pareto optimal solution with exact shape and location in decision space.

The DTLZ problem suite was introduced by Dev et al.[22]. We use the DTLZ1-DTLZ7 benchmark test suite, which represents different shapes and complexities, and

Name	Problem	Properties
DTLZ1	$f_1 = \frac{1}{2}\prod_{i=1}^{M-1} x_i(1+g(\mathbf{x}_M))$ $f_{m=2:M-1} = \frac{1}{2}\left(\prod_{i=1}^{M-m} x_i\right)(1-x_{M-m+1})(1+g(\mathbf{x}_M))$ $f_M = \frac{1}{2}(1-x_1)(1+g(\mathbf{x}_M))$ $g(\mathbf{x_M}) = 100\left[\lvert x_M\rvert + \sum_{x_i\in\mathbf{x_M}}((x_i-0.5)^2 - cos(20\pi(x_i-0.5)))\right]$	Linear
DTLZ2	$f_1 = \prod_{i=1}^{M-1} cos(x_i\pi/2)(1+g(\mathbf{x}_M))$ $f_{m=2:M-1} = \prod_{i=1}^{M-1} cos(x_i\pi/2)sin(y_M - m + 1\pi/2)(1+g(\mathbf{x}_M))$ $f_M = sin(1+g(\mathbf{x}_M))$ $g(\mathbf{x_M}) = \sum_{x_i\in\mathbf{X}_M}(x_i-0.5)^2$	Concave
DTLZ3	Similar to DTLZ2 except the g function is replaced by g function from DTLZ1	Concave Multimodal
DTLZ4	Similar to DTLZ2 except all $x_i \in x$ are replaced by x_i^α where α>0	Concave
DTLZ5	Similar to DTLZ2, except all $x_2, ..., x_M - 1 \in \mathbf{X}$ are replaced by $\dfrac{1+2g(r)x_i}{4(1+g(r))}$ where $g(r) = g(\mathbf{x}_M)$ from DTLZ2	Degenerate
DTLZ6	Similar to DTLZ5 except g is replaced by $g = \sum_{x_i\in\mathbf{X}_M} x_i^{0.1}$	Degenerate
DTLZ7	$f_{m=1:M-1} = x_m$ $f_M = (1+g(\mathbf{x}_M))M - \sum_{i=1}^{M-1}[\frac{f_i}{i+g}(1+sin(3\pi f_i))]$ $g(\mathbf{x}_M) = 1 + \frac{9}{\lvert\mathbf{X}_M\rvert}\sum_{x_i\in\mathbf{x}_M} x_i$	Mixed Disconnected

Table 2.4: Summary of DTLZ benchmark problem [8]

introduces practical environments and challenges seen in the real-world problems. It is also widely used in MaOPs research.

Table 2.4 summarizes the DTLZ problem suite and the properties of the Pareto fronts. The values of x$_i$ are restricted to [0,1].

2.9 Performance Measurements

2.9.1 Hypervolume

The hypervolume indicator is one of the popular performance indicators to compare the performance of many-objective optimization algorithms. Hypervolume is the n-dimensional space in the objective space contained by n-solution points generated from an optimization algorithm. Hypervolume metrics provides single scale measurement by measuring the size of the space covered by the solution set. The values of hypervolume indicate how a solution set spreads in solution space, along with the distance of solutions set to the Pareto optimal front. By containing the entire Pareto front in the solution set, an optimization algorithm can provide a maximum hypervolume result. Because of this property, hypervolume can give better results for large solution sets than small solution sets. Hypervolume is measured based on reference points, which can be the worst possible points in the solution set. A disadvantage is that, hypervolume computation is NP-hard, and therefore computationally expensive with increasing numbers of objectives. Because of this disadvantage, it is not possible to calculate exact hypervolume metrics when processing large objective sets [14].

We use a fast method called WFG algorithm for calculating hypervolume measurement which is included in ECJ [1]. In the WFG algorithm, the hypervolume metric is calculated based on the sum of exclusive hypervolumes [38]. This WFG algorithm uses domination to keep the solution set small for efficiency sake.

2.9.2 Inverted General Distance

Inverted general distance (IGD) is another widely used performance measurement technique in many-objective optimization research. The advantage of IGD is that it represents the quality of the approximated Pareto front by considering how solutions spread and converge towards the true Pareto front.

The IGD computes the minimum distance between the objective vectors of Pareto front and the objective vectors of the approximate front, and finds the average with respect to the true Pareto front. If the POF* is the approximated Pareto front and the POF is the true Pareto front, then the IGD will be

$$IGD = \frac{\sqrt{\sum_{i=1}^{n} d_i^2}}{|POF|} \tag{2.6}$$

where the |POF| is the number solutions in the true Pareto front, and d$_i$ is the minimum Euclidean distance between the solution of POF* and the member of the true Pareto front. To get a close convergence towards the true Pareto front, IGD should be minimum. As the IGD needs to compute the minimum distance for every solution of approximate Pareto front, it becomes computationally expensive for a large number of approximate Pareto solutions [31]. Moreover the IGD is not Pareto compliant and the result of IGD changes with the size of reference front [11].

Chapter 3

Literature Review

3.1 Many Objective Optimization

Research on many-objective optimization (MaO) is gaining popularity because of real-world applications and challenges to optimize their solutions. Several algorithms have been developed to find the optimal solutions for MaOPs. Among these, NSGA-II (non dominated sorting genetic algorithm) [18], NSGA-III [17] and SPEA2 [40] (strength Pareto evolutionary algorithm) are famous optimization algorithms. Among these NSGA-II maintains a crowding comparison procedure with tournament selection which enables NSGA-II to make diverse and converged solutions for a large number of objectives.

Bentley and Wakefeld [10] first introduced six multi-objective ranking methods for genetic algorithms. The main goal was to show how the GA can converge to acceptable solutions. They examined separate objectives with unequal effective ranges. Multi-objective ranking methods can be categorized into two parts:

1. **Range dependent:** The fitness vectors of solution vectors change when an effective range of objective vectors changes.

2. **Range independent:** The fitness vectors of solution vectors do not change when an effective range of objective vectors changes.

Effective range is defined by the minimum and maximum value of possible objective values. The valid range should be the same for all objectives to treat all solutions equally by the GA. The sum of weighted objectives (weighted sum) is an example of a range dependent method. Here, a predefined weight is required to make useful domains, and the weight may change if the objective range change. Non-dominated sorting is considered an independent range method because it does not need objective weighting. Bentley and Wakefeld worked in both range-dependent and range-independent methods. They also proved that objectives discussed in the range-dependent multi-objective ranking methods can dominate others, which results in weak solutions. Much research has since been done using range dependent and independent range methods in many-objective optimization.

Corne and Knowles [16] emphasize that average ranking (sum of ranks) performs better than many other ranking methods when the number of objectives increases. For example, problems with Pareto ranking is that a significant portion of the population is non-dominated, and the selection pressure disappears. They introduce a coverage relationship similar to dominance relations, where a solution vector s_i covers other solution vector s_j when s_i is not worse than s_j on any objectives. They described the favour relation for assigning relative rank over the population. In the favour relation, we can consider favour s_i over s_j in which number of objective values i (i <n) of s_i are better than on the same objectives of s_j, and j (j<i) objectives of s_j are better than same objectives of s_i.

Kukkonen [28] also introduced an alternative of Pareto dominance relation named ranking dominance for MaOPs. Pareto dominance cannot distinguish solutions with a large number of objectives. Ranking dominance performs advanced search in this situation. But in ranking dominance, individual objectives deteriorate, which hinders converging to Pareto front. For this reason, a ranking dominance relation has been proposed where diversity and convergence will be maintained.

Recently, Palakonda, and Mallipeddi [34] proposed a new algorithm named " Pareto dominance based algorithms with the ranking method ". In this algorithm, two main goals of the evolutionary algorithms such as diversity and convergence are achieved.

3.2 ALPS

Using ALPS (see section 2.5) on MaOPs is a new concept.Very little work has been proposed based on this idea.

Biswas [13] and Opoku-Amankwaah [33] have used ALPS with multi-objective optimization problems. They study the multi-depot vehicle routing problem with time windows (VRPTW) in their research. Biswas used Pareto ranking and first introduced multi-objective optimization for VRPTW. In this work, a new crossover named best route crossover is used with ALPS. Opoku-Amankwaah implemented ALPS with three inter-layer strategies, and used SR . Biswas and Opoku-Amankwaah both got competitive and better results in many test cases. But in both cases, they worked on only two objectives of the vehicle routing problem, which is not a MaOPs.

ALPS has been used with the SPEA2 algorithm in the Portfolio optimization problem [7]. A non-linear multi-factor model has been generated using these algorithms. SPEA2 with ALPS provided excellent results in many experiments. They showed that the ALPS algorithm with a large age gap can produce better results. Large gaps create more age layers, which are able to increase diversity on the population.

NSGA-II has been used to develop a new system, MOJITO [32], which optimizes analog circuit topology with multi-objective and multi-topology sizing. In this work, NSGA-II is used in every layer of ALPS to optimize two objectives.

3.3 Objective Reduction

Purshouse and Fleming first introduced the possibility of objective reduction in the field of MOEAs [35]. They discuss in detail various relationships between single objectives. In this work, relation between single objectives has been divided into three categories: conflict, harmony, and independence between objectives. When improvement of one objective deteriorates the performance of others, the relation is referred as conflicting objectives. The opposite relation is harmony. In the independence relation, the global optimization problem can be divided into sub-problems to solve separately. They also focused on the effect of objectives on evolutionary multi-objective optimization [35].

Deb and Saxena worked with Pareto-optimal solutions through dimensionality reduction [21]. They proposed a new MO procedure for solving large-objective problems by focusing on finding representative sets of Pareto optimal to minimize the objective set. They use principle component analysis (PCA) based NSGA-II to eliminate redundant sets. This was the first attempt to solve problem with upto 50 objectives. Also, the algorithm successfully generated actual combinations of objectives to get the true Pareto front.

An important contribution in MaOPs is explored by Brockhoff and Zitzler [15]. They tried to cover all of the theoretical foundations of objective reduction and specific algorithms for those reductions. They observed how adding and omitting objectives impacts the optimizations. They proposed a logical foundation for objective reduction, as well as two algorithms for reducing the number of objectives systematically. Based on this research, lots of research has emerged to find more practical approach for MaOPs.

Yuan et al. [39] introduced the idea of objective reduction idea with feature selection. They proposed three many-objective formulations that maintain the dominance structure for the solutions set. Three error measures (η, δ, γ) were used. The first two formulations are based on the Pareto dominance structure, and the third one utilizes the correlation between objectives. They also introduce objective reduction algorithms with NSGA-II for each error formulation, where a good tradeoff of solutions can be achieved. They optimize problems based on the number of objectives selected and the error ratio.

A summary of these and other papers is in Table 3.1. We categorized the table based on the performance measures and test problem suites.

Paper Name	**DTLZ**	**IGD**	**HV**
Finding Acceptable Pareto-Optimal Solutions using Multiobjective Genetic Algorithms [16]	NO	NO	NO
Techniques for Highly Multiobjective Optimisation: Some Nondominated Points are Better than Others [10]	NO	NO	NO
Ranking-Dominance and Many-Objective Optimization [28]	YES	NO	NO
Pareto Dominance-Based Algorithms With Ranking Methods for Many-Objective Optimization [34]	YES	NO	YES
Using Age Layered Population Structure for the Multi-Depot Vehicle Routing Problem [13, 33]	NO	NO	NO
Ranking Methods for Many-Objective Optimization [24]	YES	NO	NO
Effects of Combining ALPS and SPEA2 Genetic Programming algorithms in a Portfolio Optimization Problem [7]	NO	NO	NO
Objective Reduction in Evolutionary Multiobjective Optimization: Theory and Applications [15]	YES	NO	YES
Objective Reduction in many objective optimization-Evolutionary multiobjective approaches and complrehensive analysis [39]	YES	YES	YES
Objective reduction for many-objective optimization problems using objective subspace extraction [30]	YES	NO	NO
Searching for Pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems [21]	YES	NO	NO
Simultaneous Multi-Topology Multi-Objective Sizing Across Thousands of Analog Circuit Topologies [32]	NO	NO	NO
A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II [18]	YES	NO	NO

Table 3.1: Summary of Literature Review

Chapter 4

System Design

4.1 Introduction

This chapter provides implementation details of our experiments and proposed methodology. We will explain the system design concerning the GA and ALPS. The methods of objective reduction will also be discussed. At the end of this chapter, we will describe parameter settings for the DTLZ benchmarks for all of the experiments.

4.2 Many Objective Optimization Without Objective Reduction

We divided our research into two parts based on objective reduction methods. In the first part, we implemented the DTLZ test suite (DTLZ1-DTLZ7) [22] using GA and ALPS. We maintain consistent parameter settings in both algorithms to remove bias in the results. Later, we perform these experiments with a simple DTLZ implementation as many-objective optimization problems, and we compare our results with an other algorithm as well as with our own algorithms. We used ECJ [1] to implement all of the experiments. Also, we used Awuley's ALPS [9] and Gircys's SR implementation [25] in our experiments.

4.2.1 Chromosome Representation and Initial Population

We used the decision variables from the DTLZ test suite to build a chromosome for our algorithms. The value of the decision variables are real numbers and the ranges are within 0.0 and 1.0. For example, decision variables in DTLZ1(3,8) will be

(0.456,0.894,0.729,0.235,0.569,0.468,0.145,0.367). The initial population is generated randomly. Maximum generations and population size will vary as required.

4.2.2 Fitness Evaluation

We used tournament selection to select individuals for reproduction (see Section 2.4).

4.2.3 Fitness-Proportional Selection

The fitness-proportional selection is used to select individuals for reproduction in order to make offspring for the next generation. We used tournament selection, which selects parent individuals based on their ranking. In tournament selection, a user-specified number of individuals are chosen randomly. Among these, two individuals with the best fitness are selected to generate offspring for the next generation. For mutation, one most fit individual is used.

4.2.4 Crossover and Mutation

We used simulated binary crossover (SBX) [12] and polynomial mutation [19] in GA and ALPS. The binary-coded GAs are very effective for discrete search. But our optimization problems need continuous search spaces. The search power of other real-coded crossover operators are not adequate in this search space. SBX is popular for real-coded GA as it has similar search power of single-point crossover of binary-coded GA. SBX with real coded GA can eliminate the precision difficulties and fixed mapping problem [12]. Polynomial mutation [19] is also designed for real-coded GA.

Following [22], the SBX recombination operator (η_c=15) and polynomial mutation operator (η_m=20) have been used in all experiments. These two operators are recommended for the DTLZ test suite in [22]. The crossover probability is 0.9, and mutation probability is 1/D where D the number of decision variables in the DTLZ test problem considered.

4.3 Many Objective Optimization With Objective Reduction

We divided our objective reduction parts into two parts. To reduce objectives for our many-objective optimization problems, we used objective masks. We generate an objective mask for every individual. A mask has K bits for K objectives. A mask bit of 1 indicates that the corresponding objective will be considered in the ranking method, and 0 indicates the opposite. The definition of masks is described below.

4.3.1 Individual-based Masking

We implemented objective masks for both GA and ALPS. Individual-based masking means we set a mask for every individual. In order to make the mask, we first initialize an array with zero which will be used to hold bitmask values for an individual. After that, we reassign the value 1 in random positions of this array. This masking technique is thus called random masking. We say a mask bit is on for an objective when the value of this corresponding position is 1. For instance, if we have four objectives, and we assign bitmask [0,0,1,1] for an individual, then we can say that objectives 3 and 4 are used for fitness for this individual. It is important to note that, in random masking, we always maintain one essential constraint. Bits are set on for exactly half of the objectives, i.e. if the number of the objectives is four, then we must have 2 bits on in the mask. This is necessary so that the sum of ranks will be naturally balanced all the time, otherwise, unbalanced SR scores can arise. For example, if an individual has only one bit on (bit=1) it has to compete with other individuals that have more than many bits on. It is easier to optimize 1 objective than many . Moreover, it is also possible that an individual contains a mask with no bits turn on. In this regard, the SR will not score them. Using this strategy, we assume that the population will be evenly scored. We also use a parameter named mask duration, which indicates how long or how many generations an individual will continue using this mask. If we use the value 5 for mask duration, then every five generations a new mask will be set for every individual. We use values of 1,5 and 10 for the mask-duration. During crossover and mutation, offspring will inherit a mask from one of their parents.

4.3.2 Layer-based Masking

We also implement a layer-based masking strategy for ALPS. In layer-based masking, we generate the same mask value for all individuals of the same layer. We also use the mask duration similar to that in individual-based masking, to reset the masking values at regular intervals for the entire ALPS layer. We adopted two techniques to set the mask value for every layer.

1. **Random Masking:** We generate the mask values in a similar manner to individual-based masking. The main difference is that, in individual masking,

every individual has its own mask value, where in layer masking, every individual in the layer shares the same mask.

2. **Correlation Coefficient Masking:** We use statistical linear correlation and backward selection algorithm [6] to implement this approach. First, we initialize an array to generate a mask for each layer, and set the value 1 in every position. As usual, every position represents an objective, and initially, every bitmask is turned on. Then we calculate the linear correlation between two objectives using the Pearson Correlation Coefficient formula. If we have four objectives A, B, C, D, we make a table with the coefficient values of AB, AC, AD, BC, BD, and CD (Note that AB=BA). Then, we find the highest correlation value from the table and take one objective randomly from that entry pair. We turn the bit mask off for that objective and delete all entries associated with that objective from the coefficient table. The process will continue till a preassigned number of objectives remain (user parameter). We set the mask this way for all individuals of a specific layer.

Chapter 5

Experiments

5.1 Introduction

In this chapter, we include all experiments using regular GA, ALPS, and objective reduction. This chapter provides overall experimental details and analyzes the performance of the algorithms. In the first experiment, we compare our normalized sum of rank-based GA with other works. We will introduce ALPS in many-objective optimization problem in the next experiment and compare the result with regular GA.

5.2 General Setup

In every experiment, we maintain the same parameter settings for all DTLZ test suite [22] (see Table 5.1). DTLZ1-DTLZ7 from the DTLZ test suites are used to evaluate all algorithms. We select different test problems to observe the different shapes and locations of the optimal solution in the solution space. In each problem set, we considered 4, 6, 8, and 10 objectives, respectively. The decision variable depends on the number of objectives (M) and parameter K where K is 5, 10 and 20. It is calculated by $M - 1 + K$. We used SBX crossover and polynomial mutation. The mutation probability is 1/D, and the crossover probability is 0.9 for all experiments. The maximum generation number is 700 and population size is 1200 for all of the experiments. Tournament selection with size 4 has been used.

The parameter settings for ALPS are included in Table 5.2 . We used polynomial ageing scheme where age gap is 5. The number of layers for ALPS is 5. The layer-replacement strategy is ReverseTournamentWorst. The population size for each layer is 240 (240*5= 1200 generations in one evaluation).

To calculate the hypervolume and the IGD matrix, we took 30 solutions from 30 runs and used the 30 solutions as the solution set for hypervolume and IGD calculation. For the IGD matrix, we calculate the result based on the true Pareto front provided in Jmetal. We can calculate the IGD matrix for DTLZ1-DTLZ4 and DTLZ7 as Jmetal has provided Pareto front only these test suites with 4,6 and 8 objectives. As IGD calculates the distance between approximate front to true front, it evaluates an algorithm as a "best" when this algorithm provides small IGD values. For the hypervolume matrix, we normalize all of the objective values [23] and set reference point as $(1,1,1,..,1)_N$. Hypervolume result is best when it is high. Bold face will indicate better result for all cases.

5.3 Result NSGA-II, GA, ALPS and Normalized Sum Of Rank

In this part, we compared our result using GA and ALPS with NSGA-II. We ran the NSGA-II in Jmetal [2]. However, the IGD result is not Pareto compliant, and it depends on the size of the Pareto Front. The population size is 1200, and the generation size is 700 for both of the algorithms.

5.3.1 Result Analysis

IGD

Table 5.3 shows the IGD result between NSGA-II, GA, and ALPS. From these Tables, we can see that NSGA-II outperforms GA and ALPS in every case except DTLZ2(8,17). The reason is that, NSGA-II uses sophisticated non-dominated sorting and crowding distance sorting approach which makes diverse and converged solution. Also, we know that SR provides only one solution per run, whereas non-dominated sorting approach provides a set of solutions. Original GA with SR cannot provide a diverse solution like NSGA-II. Also, the fitness evaluation technique of these two algorithms is different. As IGD focuses on convergence as well as diversity, it is apparent that we cannot get good result with GA and ALPS, that will outperform NSGA-II. Therefore, in the rest of the thesis, we will compare our algorithms with each other. In Figure 5.1, we generated a graph using GA for exemplary data. In this graph, we put the average objective scores of the best individual from 30 runs in

Problem	**K**	**Objectives(M)**	**Decision Variable(D)**	**Mutation Prob.(1/D)**
DTLZ1	5	4	8	0.125
		6	10	0.1
		8	12	0.08
		10	14	0.07
DTLZ2	10	4	13	0.076
		6	15	0.066
		8	17	0.058
		10	19	0.0526
DTLZ3	10	4	13	0.076
		6	15	0.066
		8	17	0.058
		10	19	0.0526
DTLZ4	10	4	13	0.076
		6	15	0.066
		8	17	0.058
		10	19	0.0526
DTLZ5	10	4	13	0.076
		6	15	0.066
		8	17	0.058
		10	19	0.0526
DTLZ6	10	4	13	0.076
		6	15	0.066
		8	17	0.058
		10	19	0.0526
DTLZ7	20	4	23	0.04
		6	25	0.04
		8	27	0.037
		10	29	0.034

Table 5.1: Parameter settings for DTLZ

GA Parameter	
Generation	700
Population	1200
mutation-probability	1/D
crossover-prabability	0.9
crossover-type	SBX
crossover-distribution-index	20
mutation-type	polynomial
mutation-distribution-index	20
Selection	Tournament
Size	4
Jobs	30
Fitness	Sum of rank
Diversity Penalty	10
ALPS Parameter	
Age-gap	5
number-of-layers	5
Population size	240
selection-pressure	0.8
Tournament-size	4
Aging-scheme	Polynomial
Layer-replacement	ReverseTournamentWorst

Table 5.2: Parameter settings for GA and ALPS

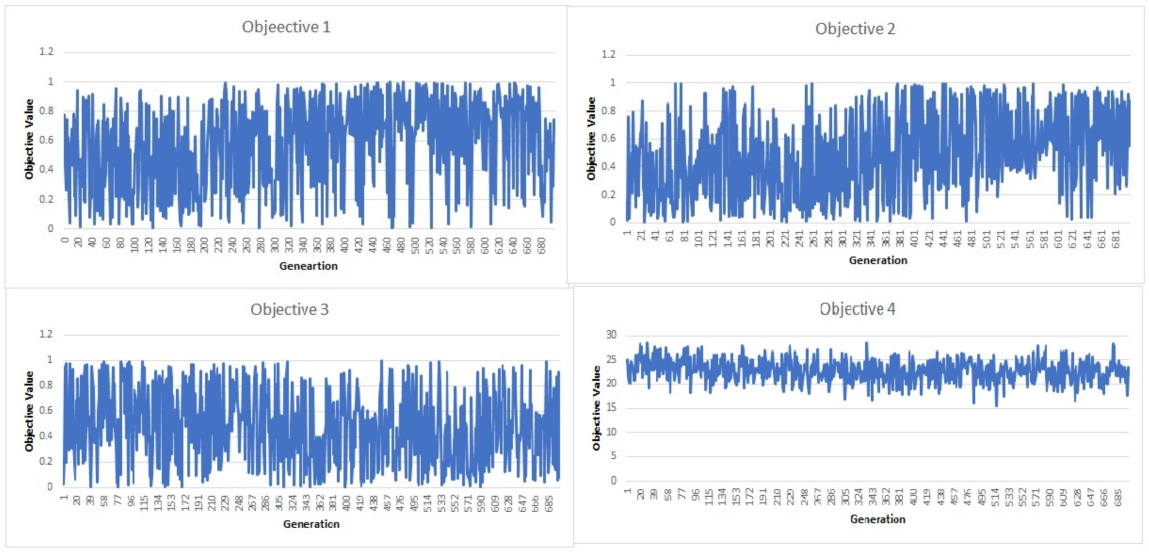

Figure 5.1: Best individual of GA for DTLZ1

y-axis and generation in x-axis. In this figure, we see that the GA is trying to get the best solutions throughout the generation. Although the information is not useful, as there exist noise due to different runs jumping around several space.

5.4 Experiment for DTLZ with ALPS and GA

In this experiment, we optimize the DTLZ test suite with ALPS. Our main goal will be comparing the IGD and the hypervolume result with GA to observe the performance of ALPS in regards to MaOPs. We know that many-objective evolutionary algorithms (MOEAs) can be attracted by local optima, which can lead to early convergence in many EAs. ALPS can reduce this premature convergence by generating new genes through the new individuals. By introducing new individuals in the bottom layer ALPS restricts premature convergence. As a result, every time the new search space is extended, search can be able to focus on the global optima.

5.4.1 Result Analysis

Hypervolume

The hypervolume result represents the performance of the algorithms based on the diversity of solutions. Table 5.4 represents the comparison results between GA and

Problem	**Algorithm**	**Result**		
		M=4	**M=6**	**M=8**
DTLZ1	NSGAII	**0.001942952**	**0.493794442**	**0.206775984**
	GA	5.274900632	6.611777213	1.948641322
	ALPS	2.76089036	1.964830765	1.489876904
DTLZ2	NSGAII	**0.005059919**	**0.03454788**	0.060221254
	GA	0.050318577	0.067223691	**0.049765828**
	ALPS	0.067740799	0.060183795	0.05378555
DTLZ3	NSGAII	**0.005137123**	**7.896551534**	**11.010406**
	GA	36.95092159	38.50124834	35.1878284
	ALPS	35.07309141	26.6088674	34.32926854
DTLZ4	NSGAII	**0.004973564**	**0.023082208**	**0.06730297**
	GA	0.084007912	0.08304716	0.068599965
	ALPS	0.105053564	0.095198168	0.07789963
DTLZ7	NSGAII	**0.004933684**	**0.03900**	**0.047500682**
	GA	0.924303479	1.263734703	1.454404778
	ALPS	1.004190122	1.534017005	1.568327064

Table 5.3: IGD result between NSGA-II, GA, and ALPS

ALPS. Twenty-one test cases were considered for optimization. Among 21 test cases, ALPS outperformed GA for 6 test cases. These test cases are DTLZ2(6,15), DTLZ4(8,15), DTLZ5(6,15), DTLZ5(8,17), DTLZ6(4,13), DTLZ6(6,15). GA made better result in other test cases. ALPS performed well with a large number of objectives. So, ALPS can make solutions which are near the Pareto front even for a large number of objectives. As the hypervolume result represents the diversity of solution set, it is evident that in each run GA can make solutions in different sub-area of the Pareto front.

IGD

The IGD matrix represents the performance of an algorithm according to the diversity of a solution set and the convergence towards Pareto front. Table 5.5 shows the IGD values of DTLZ test for GA and ALPS. From this table, it can be observed that ALPS and GA exceed each other in many test instances. ALPS outperforms GA in DTLZ1 for all objectives. We know that this search space contains $(11^K - 1)$ local optima [22] where EAs can be attracted by these fronts. From the IGD result, it is evident that ALPS has significantly smaller IGD result than GA, which represents excellent diversity as well as the convergence of ALPS. ALPS also outperforms GA in DTLZ2 and DTLZ3 with 6 objectives.

Problem	**Algorithm**	**Result**			
		M=4	**M=6**	**M=8**	**M=10**
DTLZ1	GA	21 **0.7960**	**0.8191**	**0.7761**	**0.7961**
	ALPS	0.7355	0.6688	0.7119	0.6265
DTLZ2	GA	**0.6278**	0.3974	**0.4613**	**0.4258**
	ALPS	0.4115	**0.4094**	0.4108	0.2908
DTLZ3	GA	**0.6457**	**0.6572**	**0.7283**	**0.7250**
	ALPS	0.4206	0.3408	0.3421	0.1858
DTLZ4	GA	**0.7652**	**0.6014**	**0.5333**	0.4723
	ALPS	0.2994	0.5291	0.4248	**0.5815**
DTLZ5	GA	**0.4535**	0.3542	**0.1766**	0.1723
	ALPS	0.2155	**0.3547**	0.1296	**0.2762**
DTLZ6	GA	0.2604	0.2455	**0.1634**	**0.1479**
	ALPS	**0.3597**	**0.2805**	0.0943	0.0905
DTLZ7	GA	**0.4791**	**0.1794**	**0.1438**	**0.1163**
	ALPS	0.0805	0.0137	0.0149	0.0006

Table 5.4: Hypervolume result between GA and ALPS

Furthermore, GA provides a better result than ALPS in DTLZ7 and DTLZ4. DTLZ7 has a disconnected Pareto front. MOEAs has to maintain stable and diverse subpopulation to perform well in this test problem. Based on the IGD result, we can say that the GA maintained distributed subsolutions through the runs. Moreover, DTLZ4 introduces a dense solutions where GA provides a better result from ALPS. It can be said that GA makes a diverse solutions set with 30 solutions where all the solutions are not in the same subarea of Pareto front.

In a nutshell, ALPS made better result in IGD where GA was better for ALPS. So, we can say, ALPS can make solutions more converged to the Pareto front, but GA is useful in making the diverse solutions.

5.5 Experiment With Objective Reduction in GA

In this experiment, we implemented objective reduction using random masking in the GA.

We compare GA with three variations of random masking, named GA objective reduction K1 (GAORK1), GA objective reduction K5 (GAORK5), and GA objective reduction K10 (GAORK10). Here K is a predefined parameter that denotes a time period when generating new random mask is regenerated. K=1 means a new objective mask will be created in every generation while K=10 indicates all individuals will

Problem	**Algorithm**	**Result**		
		M=4	**M=6**	**M=8**
DTLZ1	GA	5.2749	6.6117	1.9486
	ALPS	**2.7608**	**1.9648**	**1.4898**
DTLZ2	GA	**0.0503**	0.0672	**0.0497**
	ALPS	0.0677	**0.0601**	0.0537
DTLZ3	GA	36.9509	38.5012	35.1878
	ALPS	**35.0730**	**26.6088**	**34.3292**
DTLZ4	GA	**0.0840**	**0.0830**	**0.0685**
	ALPS	0.1050	0.09519	0.0778
DTLZ7	GA	**0.9243**	**1.2637**	**1.4544**
	ALPS	1.0042	1.5340	1.5683

Table 5.5: IGD result between GA and ALPS

maintain their specific objective mask for ten generations at an extent.

5.5.1 Hypervolume

Table 5.6 shows the hypervolume result for this experiment. For this comparison, we used DTLZ1-DTLZ7 test cases. From this result, it is observed that GA outperformed most of the objective reduction GA. Only GAORK1 and GAORK10 beat the GA in two test instances which are DTLZ5(4,13) and DTLZ6(4,13). We can say that random-based objective reduction cannot make a diverse set of solution like original GA. To keep the same solutions as the original Pareto front , the algorithm must generate essential objectives which must contain the characteristics of the original front. As we cannot determine the essential objective set, the result is focusing on only the same sub-area all the time.

5.5.2 IGD

Table 5.7 represents the IGD results of this experiment. It is observed that GA with random masking cannot outperform regular GA in any instances. It is hard to say the specific reason behind this result, but we can speculate that because random masking is not following any particular rules to reduce the objective set, it cannot determine which objectives are essential or redundant. If the mask discards essential objectives, then it is hard to perform well using objective reduction. The reverse thing can happen if random masking selects actual redundant objectives which means we can get good result. But overall random mask was not found to be effective.

Problem	**Algorithm**	**Result**			
		M=4	**M=6**	**M=8**	**M=10**
DTLZ1	GA	**0.7960**	**0.8191**	**0.7761**	**0.7961**
	GAORK1	0.4227	0.3263	0.1166	0.0359
	GAORK5	0.7837	0.4506	0.3857	0.1124
	GAORK10	0.5896	0.4178	0.3477	0.0854
DTLZ2	GA	**0.6278**	**0.3974**	**0.4613**	**0.4258**
	GAORK1	0.2526	0.0415	0.0062	0.0023
	GAORK5	0.2529	0.2320	0.0468	0.0076
	GAORK10	0.2939	0.1911	0.0570	0.0099
DTLZ3	GA	**0.6457**	**0.6572**	**0.7283**	**0.7250**
	GAORK1	0.3499	0.0234	0.0224	0.0025
	GAORK5	0.3697	0.2060	0.1840	0.0059
	GAORK10	0.3518	0.3064	0.0525	0.0172
DTLZ4	GA	**0.7652**	**0.6014**	**0.5333**	**0.4723**
	GAORK1	0.2271	0.1828	0.0368	0.0279
	GAORK5	0.3108	0.2836	0.1134	0.0434
	GAORK10	0.3363	0.2866	0.1100	0.0377
DTLZ5	GA	0.4535	**0.3542**	**0.1766**	**0.1723**
	GAORK1	**0.5723**	0.0642	0.0248	0.0067
	GAORK5	0.2842	0.1464	0.0734	0.0357
	GAORK10	0.2847	0.1832	0.0943	0.0111
DTLZ6	GA	0.2604	**0.2455**	**0.1634**	**0.1479**
	GAORK1	0.2004	0.0352	0.0117	0.0035
	GAORK5	0.2327	0.1908	0.0568	0.0085
	GAORK10	**0.2894**	0.1943	0.0633	0.0097
DTLZ7	GA	**0.4791**	**0.1794**	**0.1438**	**0.1163**
	GAORK1	0.0518	3.10E-35	1.28E-05	3.39E-07
	GAORK5	0.0042	8.07E-30	2.31E-05	2.51E-07
	GAORK10	0.0001	5.46E-55	7.94E-06	6.10E-07

Table 5.6: Hypervolume result for objective reduction in GA

Problem	Algorithm	Result		
		M=4	M=6	M=8
DTLZ1	GA	**5.2750**	**6.6111**	**1.9486**
	GAORK1	18.4562	12.9656	8.9800
	GAORK5	14.2864	13.2906	8.7931
	GAORK10	21.3774	13.1975	8.1152
DTLZ2	GA	**0.0503**	**0.0672**	**0.0497**
	GAORK1	0.1849	0.1809	0.1295
	GAORK5	0.1667	0.1829	0.1865
	GAORK10	0.1751	0.1865	0.1304
DTLZ3	GA	**36.9509**	**38.5012**	**35.1878**
	GAORK1	130.2412	116.9390	91.6698
	GAORK5	111.6325	110.5309	78.2435
	GAORK10	117.8958	97.3544	81.8913
DTLZ4	GA	**0.0840**	**0.0830**	**0.0685**
	GAORK1	0.1779	0.1568	0.1307
	GAORK5	0.1668	0.1591	0.1290
	GAORK10	0.1684	0.1532	0.1285
DTLZ7	GA	**0.9243**	**1.2637**	**1.4544**
	GAORK1	2.1100	3.4253	3.5415
	GAORK5	2.3573	3.4814	3.5466
	GAORK10	2.3480	3.5956	3.5921

Table 5.7: IGD result for objective reduction in GA

In summary , GA with random reduction is not appropriate strategy for objective reduction.

5.6 Experiment With Random Objective Reduction in ALPS

In this experiment, we implemented two techniques with ALPS for random objective reduction. First, we added individual-based random masking for the objective reduction in ALPS. We next performed layer-based random masking. Indeed, random masking was not satisfactory in the GA. But we want to do random objective reduction in ALPS , in case it could benefit form it. We know that ALPS introduces new individuals at the regular intervals, which can create the possibility to make good solutions. When generating new individuals, ALPS creates random masks for the new individuals. As a result, the probability of getting an appropriate masking may increase. Therefore, our main intention is to see whether random masking in ALPS can outperform random masking with GA.

We use a strategy like random masking in GA, where all masks have the same number of bits set to "on". In individual-based masking, we have three algorithms named ALPS Objective Reduction Random K1 (ALPSORK1), ALPS Objective Reduction Random K5 (ALPSORK5), and ALPS Objective Reduction Random K10 (ALPSORK10).

In individual masking, a different objective mask is generated for every solution where a specific objective mask is created per layer in layer masking. In layer masking, we executed random masking and coefficient masking. For layer-based random masking we implemented 3 algorithms named ALPS Objective Reduction Layer-wise Random K1 (ALPSORLRK1), ALPS Objective Reduction Layer-wise Random K5 (ALPSORLRK5), ALPS Objective Reduction Layer-wise Random K10 (ALPSORLRK10).

5.6.1 Hypervolume

Table 5.8 and 5.9 show the hypervolume result for this experiment. In Table 5.8, we compared the original ALPS with the individual-based random objective reduction in ALPS. Here we can see that individual-based random objective reduction outperformed original ALPS in many cases. ALPSORK1 performed well in DTLZ3(10,19). ALPSORK5 outperformed ALPS in DTLZ2(4,13), DTLZ3(4,13), DTLZ3(10,17), DTLZ5(4,13) and DTLZ(10,19). Also, ALPSORK10 exceed ALPS for all in DTLZ4 except DTLZ4(10,19). Here, it is evident that random objective reduction algorithms with ALPS performed well in the hypervolume result. We can speculate that by generating random mask in regular interval ALPS is generating diversity in solution space. In Table 5.9, we compare individual-based random objective reduction with layer-based random objective reduction. According to this result, individual-based random masking and layer-based random masking outperformed each other in many cases. Layer-based random reduction performed well in DTLZ1(6,15), DTLZ3(4,13), DTLZ3(8,15) , DTLZ4(4,13), DTLZ4(8,15). From DTLZ1-DTLZ4, layer-based random masking outperformed individual-based random masking for 4 and 8 objectives where individual-based masking performed well for 6 and 10 objectives. In DTLZ6, individual-based random reduction outperformed layer-based random reduction in all cases where layer-based random reduction beat individual-based random reduction for all cases of DTLZ7.

Problem	**Algorithm**	**Result**			
		M=4	**M=6**	**M=8**	**M=10**
DTLZ1	ALPS	0.7355	**0.6688**	**0.7119**	**0.6265**
	ALPSORK1	0.5699	0.3230	0.4272	0.2912
	ALPSORK5	**0.8852**	0.6629	0.5403	0.3658
	ALPSORK10	0.6634	0.6514	0.4038	0.3401
DTLZ2	ALPS	0.4115	**0.4094**	**0.4108**	**0.2908**
	ALPSORK1	0.4039	0.1111	0.0748	0.0415
	ALPSORK5	0.4012	0.1743	0.1636	0.1818
	ALPSORK10	**0.4339**	0.1985	0.0730	0.1294
DTLZ3	ALPS	0.4206	**0.3408**	**0.3421**	0.1858
	ALPSORK1	0.3790	0.3213	0.0629	**0.3130**
	ALPSORK5	**0.5500**	0.3307	0.1365	0.0558
	ALPSORK10	0.5042	0.2571	0.0978	0.0761
DTLZ4	ALPS	0.2994	0.5291	0.4248	**0.5815**
	ALPSORK1	0.4370	0.4958	0.3615	0.4517
	ALPSORK5	0.5101	0.4704	0.3958	0.3895
	ALPSORK10	**0.5676**	**0.5854**	**0.5566**	0.5018
DTLZ5	ALPS	0.2155	**0.3547**	0.1296	0.2762
	ALPSORK1	0.6160	0.2673	0.1719	0.0493
	ALPSORK5	**0.7109**	0.2934	0.2287	**0.3218**
	ALPSORK10	0.4043	0.3017	**0.3467**	0.1249
DTLZ6	ALPS	**0.3597**	**0.2805**	**0.0943**	**0.0905**
	ALPSORK1	0.3077	0.0860	0.0358	0.0073
	ALPSORK5	0.3241	0.1926	0.0532	0.0131
	ALPSORK10	0.2790	0.1365	0.0548	0.0163
DTLZ7	ALPS	0.0805	**0.0137**	**0.0149**	0.0006
	ALPSORK1	0.0142	0.0002	0.0006	0.0003
	ALPSORK5	0.0115	0.0009	0.0042	**0.0016**
	ALPSORK10	**0.0186**	0.0005	0.0016	9.73E-05

Table 5.8: Hypervolume result between ALPS and ALPS with random objective reduction

Problem	Algorithm	Result			
		M=4	M=6	M=8	M=10
DTLZ1	ALPSORK1	0.5699	0.323	0.4272	0.2912
	ALPSORK5	**0.8852**	0.6629	**0.5403**	**0.3658**
	ALPSORK10	0.6634	0.6514	0.4038	0.3401
	ALPSORLRK1	0.6238	**0.6831**	0.4446	0.1886
	ALPSORLRK5	0.8785	0.3423	0.391	0.1826
	ALPSORLRK10	0.5708	0.5789	0.3084	0.2462
DTLZ2	ALPSORK1	0.4039	0.1111	0.0748	0.0415
	ALPSORK5	0.4012	0.1743	**0.1636**	**0.1818**
	ALPSORK10	**0.4339**	**0.1985**	0.073	0.1294
	ALPSORLRK1	0.477	0.1131	0.075	0.0327
	ALPSORLRK5	0.3828	0.1313	0.071	0.0352
	ALPSORLRK10	0.1832	0.1483	0.0763	0.0295
DTLZ3	ALPSORK1	0.379	0.3213	0.0629	**0.3130**
	ALPSORK5	0.5500	**0.3307**	0.1365	0.0558
	ALPSORK10	0.5042	0.2571	0.0978	0.0761
	ALPSORLRK1	0.5886	0.2433	**0.2033**	0.0528
	ALPSORLRK5	**0.6004**	0.1714	0.1901	0.0767
	ALPSORLRK10	0.3957	0.1596	0.1382	0.0817
DTLZ4	ALPSORK1	0.4370	0.4958	0.3615	0.4517
	ALPSORK5	0.5101	0.4704	0.3958	0.3895
	ALPSORK10	**0.5676**	0.5854	**0.5566**	0.5018
	ALPSORLRK1	0.4562	**0.5909**	0.3651	**0.6034**
	ALPSORLRK5	0.4167	0.3688	0.5053	0.4784
	ALPSORLRK10	0.5081	0.4780	0.456	0.4034
DTLZ5	ALPSORK1	0.6160	0.2673	0.1719	0.0493
	ALPSORK5	**0.7109**	0.2934	0.2287	**0.3218**
	ALPSORK10	0.4043	**0.3017**	**0.3467**	0.1249
	ALPSORLRK1	0.3344	0.2677	0.0972	0.1302
	ALPSORLRK5	0.4625	0.2136	0.1514	0.2199
	ALPSORLRK10	0.383	0.2407	0.1342	0.0960
DTLZ6	ALPSORK1	0.3077	0.0860	0.0358	0.0073
	ALPSORK5	**0.3241**	**0.1926**	0.0532	0.0131
	ALPSORK10	0.279	0.1365	**0.0548**	0.0163
	ALPSORLRK1	0.2467	0.1031	0.0246	**0.0346**
	ALPSORLRK5	0.1990	0.1199	0.0264	0.0259
	ALPSORLRK10	0.2195	0.1081	0.0263	0.0138
DTLZ7	ALPSORK1	0.0142	0.0002	0.0006	0.0003
	ALPSORK5	0.0115	0.0009	0.0042	0.0016
	ALPSORK10	0.0186	0.0005	0.0016	9.73E-05
	ALPSORLRK1	**0.2569**	0.0144	**0.0597**	**0.0500**
	ALPSORLRK5	0.0859	0.0184	0.0281	0.0057
	ALPSORLRK10	0.1155	**0.0358**	0.0304	0.0088

Table 5.9: Hypervolume result between ALPS individual-based random reduction and layer-based random reduction

5.6.2 IGD

Table 5.10, 5.11 represents the IGD values of this experiment. In Table 5.10, we compare the result of the algorithm based on individual-based masking with normal ALPS. We can see that individual-based masking performs well in DTLZ4 and DTLZ7. In other instances, ALPS beats objective reduction ALPS. It is necessary to say that, in our previous experiment, ALPS cannot make improve over GA in DTLZ4 and DTLZ7. But the objective reduction algorithm made a better results in these two instances. For DTLZ7, random masking with ALPS outperforms GA . However, random masking cannot make a significant impact on GA but can cause considerable positive impact with ALPS. The main reason is that, considering a different sets of objectives at regular interval. That means if an essential objective set is discarded, it can be recovered by introducing new objective sets, which increases the possibility of better performance. In Table 5.11, we make a comparison between individual-based random objective reduction for ALPS and layer-based random objective reduction for ALPS . Layer-based random reduction performed well for large number of objectives. Also, layer-based reduction beat individual based reduction at least in one instance of every test problem.

5.7 Experiment With Coefficient-Based Objective Reduction in ALPS

In coefficient masking, we calculated the linear coefficient values between objectives and discarded those objectives with high coefficient values. We set our coefficient parameter as 2. This value indicated that we reduce two objectives using coefficient masking for every problem instances. For layer-based comefficient masking we implemented 3 algorithms named ALPS Objective Reduction Layer-based Pearson Coefficient K1 (ALPSORLPK1), ALPS Objective Reduction Layer-based Pearson Coefficient K5 (ALPSORLPK5) and ALPS Objective Reduction Layer-based Pearson Coefficient K10 (ALPSORLPK10).

5.7.1 Hypervolume

In this part, we compare the hypervolume result of layer-based random reduction with the layer-based coefficient reduction in Table 5.12. From this result, we can see

Problem	**Algorithm**	**Result**		
		M=4	**M=6**	**M=8**
DTLZ1	ALPS	**2.7609**	**1.9648**	**1.4899**
	ALPSORK1	10.5791	8.1653	5.7017
	ALPSORK5	5.8032	6.5576	3.1547
	ALPSORK10	11.6713	5.2989	4.8269
DTLZ2	ALPS	**0.0677**	**0.0602**	**0.0538**
	ALPSORK1	0.1063	0.1259	0.0798
	ALPSORK5	0.0984	0.1097	0.0611
	ALPSORK10	0.0945	0.1045	0.0724
DTLZ3	ALPS	**35.0731**	**26.6089**	**34.3293**
	ALPSORK1	55.9097	56.8601	57.2904
	ALPSORK5	58.6247	56.7505	36.7820
	ALPSORK10	65.2850	67.6739	50.6708
DTLZ4	ALPS	0.1051	0.0952	**0.0779**
	ALPSORK1	0.1149	0.1045	0.0878
	ALPSORK5	**0.0962**	0.0927	0.0788
	ALPSORK10	0.0965	**0.0883**	0.0791
DTLZ7	ALPS	1.0042	1.5340	1.5683
	ALPSORK1	0.8802	**1.2032**	**1.3236**
	ALPSORK5	0.8927	1.4150	1.4059
	ALPSORK10	**0.8063**	1.3106	1.3463

Table 5.10: IGD result between ALPS and ALPS with random objective reduction

Problem	**Algorithm**	**Result**		
		M=4	**M=6**	**M=8**
DTLZ1	ALPSORK1	10.5791	8.1653	5.7017
	ALPSORK5	5.8032	6.5576	**3.1547**
	ALPSORK10	11.6713	5.2989	4.8269
	ALPSORLRK1	8.9764	**4.5455**	5.5277
	ALPSORLRK5	**4.2700**	6.4494	4.4131
	ALPSORLRK10	9.8046	4.6311	5.1464
DTLZ2	ALPSORK1	0.1063	0.1259	0.0798
	ALPSORK5	0.0984	0.1097	**0.0611**
	ALPSORK10	**0.0945**	0.1045	0.0724
	ALPSORLRK1	0.1029	**0.1030**	0.0876
	ALPSORLRK5	0.1101	0.1038	0.0854
	ALPSORLRK10	0.1230	0.1101	0.0868
DTLZ3	ALPSORK1	55.9097	56.8601	57.2904
	ALPSORK5	58.6247	56.7505	**36.7820**
	ALPSORK10	65.2850	67.6739	50.6708
	ALPSORLRK1	**49.4902**	**52.2869**	41.7958
	ALPSORLRK5	62.5383	71.1562	46.3227
	ALPSORLRK10	56.5755	61.8125	49.0568
DTLZ4	ALPSORK1	0.1149	0.1045	0.0878
	ALPSORK5	**0.0962**	0.0927	0.0788
	ALPSORK10	0.0965	0.0883	0.0791
	ALPSORLRK1	0.1020	**0.0881**	0.0812
	ALPSORLRK5	0.1129	0.0982	**0.0753**
	ALPSORLRK10	0.1060	0.0914	0.0816
DTLZ7	ALPSORK1	0.8802	**1.2032**	1.3236
	ALPSORK5	0.8927	1.4150	1.4059
	ALPSORK10	0.8063	1.3106	1.3463
	ALPSORLRK1	0.7835	1.1769	1.2899
	ALPSORLRK5	**0.7398**	1.2775	1.1325
	ALPSORLRK10	0.8109	1.1587	**1.0886**

Table 5.11: IGD result between individual-based random reduction and layer-based random reduction of ALPS

that ALPS layer-based coefficient reduction performed well with the large number of objectives for all DTLZ test suite except DTLZ4. The layer-based random reduction also outperformed coefficient masking in many cases. But, in DTLZ4, DTLZ6, and DTLZ7, layer-based random reduction defeated layer-based coefficient in every case except for 6 objectives. Layer-based coefficient reduction performed well for DTLZ1, DTLZ2, and DTLZ5. In DTLZ3 both performed well in two cases.

5.7.2 IGD

In Table 5.13, we compare all layer-based objective reduction algorithms. We can see that layer-based coefficient reduction provides good result in many instances of DTLZ test suite, especially for large objectives. Layer-based random reduction outperforms coefficient masking in only DTLZ1 (4 obj.), DTLZ3 (4 obj.) and DTLZ7 (4 and 6 objectives).

The main goal of objective reduction algorithms is reducing redundant objective set and also maintaining diversity and convergence of solution set. In this regards, coefficient-based reduction overcame both of the challenges for the IGD and the hypervolume measurements. Moreover, we only reduced 2 coefficients here. If we increased the number of reduction, it may provide even better results.

5.8 Statistical Significance

When we need to compare algorithms it is preferable to use statistical significance to get a firmer idea of algorithm performance. Because of the nature of SR solutions, it is very difficult to do statistical significance with IGD and hypervolume measures. To do the statistical significance, we need many IGD or hypervolume results. However, we make a solution set with 30 solutions from 30 runs (one solution from one run) to calculate a single IGD or hypervolume result. In this situation, it will take a significant amount of runs and time to complete a larger number of IGD or hypervolume results.

Because of this, we propose an idea for using statistical significance. We calculated the minimum distance from each solution (30 total) to the true Pareto front and used these 30 distances for statistical significance calculations. It is important to note that this calculation is different from the IGD and the hypervolume. It is possible that an algorithm which performed well in IGD and hypervolume cannot perform well with our distance calculation. and hypervolume performance matrix focus on diversity as well as convergence towards the Pareto front. But in our approach, we are focusing

Problem	**Algorithm**	**Result**			
		M=4	**M=6**	**M=8**	**M=10**
DTLZ1	ALPSORLRK1	0.6238	**0.6831**	0.4446	0.1886
	ALPSORLRK5	**0.8785**	0.3423	0.3910	0.1826
	ALPSORLRK10	0.5708	0.5789	0.3084	0.2462
	ALPSORLPK1	0.4166	0.5684	0.1761	**0.3839**
	ALPSORLPK5	0.6035	0.3626	0.2289	0.0844
	ALPSORLPK10	0.6560	0.4706	**0.4753**	0.1955
DTLZ2	ALPSORLRK1	**0.4770**	0.1131	0.0750	0.0327
	ALPSORLRK5	0.3828	0.1313	0.0710	0.0352
	ALPSORLRK10	0.1832	0.1483	0.0763	0.0295
	ALPSORLPK1	0.3971	0.1293	**0.1008**	0.0129
	ALPSORLPK5	0.2482	**0.1867**	0.0465	**0.0389**
	ALPSORLPK10	0.3348	0.1830	0.0492	0.0142
DTLZ3	ALPSORLRK1	0.5886	0.2433	**0.2033**	0.0528
	ALPSORLRK5	0.6004	0.1714	0.1901	**0.0767**
	ALPSORLRK10	0.3957	0.1596	0.1382	0.0817
	ALPSORLPK1	0.3860	0.2530	0.1573	0.0528
	ALPSORLPK5	**0.8015**	0.2966	0.0819	0.0758
	ALPSORLPK10	0.5292	**0.3604**	0.1844	0.0161
DTLZ4	ALPSORLRK1	0.4562	**0.5909**	0.3651	**0.6034**
	ALPSORLRK5	0.4167	0.3688	**0.5053**	0.4784
	ALPSORLRK10	0.5081	0.4780	0.4560	0.4034
	ALPSORLPK1	0.5379	0.4107	0.4197	0.2828
	ALPSORLPK5	**0.5657**	0.4051	0.4436	0.5796
	ALPSORLPK10	0.4208	0.4221	0.4040	0.4171
DTLZ5	ALPSORLRK1	0.3344	0.2677	0.0972	0.1302
	ALPSORLRK5	**0.4625**	0.2136	0.1514	0.2199
	ALPSORLRK10	0.3830	0.2407	0.1342	0.0960
	ALPSORLPK1	0.4042	0.2500	**0.1925**	**0.3242**
	ALPSORLPK5	0.4548	**0.3130**	0.1878	0.0996
	ALPSORLPK10	0.3918	0.1871	0.1396	0.1291
DTLZ6	ALPSORLRK1	**0.2467**	0.1031	0.0186	**0.0346**
	ALPSORLRK5	0.1990	**0.1199**	0.0264	0.0259
	ALPSORLRK10	0.2195	0.1081	0.0263	0.0268
	ALPSORLPK1	0.1948	0.0887	0.0242	0.0074
	ALPSORLPK5	0.2059	0.1100	0.0194	0.0104
	ALPSORLPK10	0.2001	0.0701	**0.0629**	0.0045
DTLZ7	ALPSORLRK1	0.2569	0.0144	**0.0597**	**0.0500**
	ALPSORLRK5	**0.0859**	0.0184	0.0281	0.0057
	ALPSORLRK10	0.1155	0.0358	0.0304	0.0088
	ALPSORLPK1	0.2765	0.0285	0.0121	0.0146
	ALPSORLPK5	0.0527	**0.0483**	0.0266	0.0083
	ALPSORLPK10	0.1579	0.0126	0.0232	0.0218

Table 5.12: Hypervolume result between ALPS layer-based random reduction and layer-based coefficient reduction

Problem	**Algorithm**	**Result(M=4)**	**Result(M=6)**	**Result(M=8)**
DTLZ1	ALPSORLRK1	8.9764	4.5455	5.5277
	ALPSORLRK5	**4.2700**	6.4494	4.4131
	ALPSORLRK10	9.8046	4.6311	5.1464
	ALPSORLPK1	8.5425	4.2070	6.0596
	ALPSORLPK5	7.8320	4.1786	4.7203
	ALPSORLPK10	9.9927	**3.6505**	**3.4633**
DTLZ2	ALPSORLRK1	0.1029	0.1030	0.0876
	ALPSORLRK5	0.1101	0.1038	0.0854
	ALPSORLRK10	0.1230	0.1101	0.0868
	ALPSORLPK1	0.1139	0.1086	**0.0739**
	ALPSORLPK5	0.1119	**0.1015**	0.0846
	ALPSORLPK10	**0.1026**	0.1054	0.0822
DTLZ3	ALPSORLRK1	**49.4902**	52.2869	41.7958
	ALPSORLRK5	62.5383	71.1562	46.3227
	ALPSORLRK10	56.5755	61.8125	49.0568
	ALPSORLPK1	72.6242	55.6170	49.7999
	ALPSORLPK5	50.3784	51.5576	52.8551
	ALPSORLPK10	58.0014	**49.1427**	**40.6074**
DTLZ4	ALPSORLRK1	0.1020	**0.0881**	0.0812
	ALPSORLRK5	0.1129	0.0982	0.0753
	ALPSORLRK10	0.1060	0.0914	0.0816
	ALPSORLPK1	0.1095	0.0960	**0.0744**
	ALPSORLPK5	**0.0894**	0.0928	0.0828
	ALPSORLPK10	0.1044	0.0977	0.0789
DTLZ7	ALPSORLRK1	0.7835	1.1769	1.2899
	ALPSORLRK5	**0.7398**	1.2775	1.1325
	ALPSORLRK10	0.8109	**1.1587**	**1.0886**
	ALPSORLPK1	0.8551	1.2408	1.1998
	ALPSORLPK5	0.8481	1.2131	1.1800
	ALPSORLPK10	0.7625	1.1640	1.1675

Table 5.13: IGD result between layer-based random reduction with layer-based coefficient reduction in ALPS

Algorithm	**Results(M=4)**	**Results(M=6)**	**Results(M=7)**
GA	198.1704	202.6239	**185.2771**
ALPS	**82.8640**	**106.1653**	**107.0111**
GAORK1	346.4852	269.3821	227.9081
GAORK5	354.5016	279.0873	251.6517
GAORK10	383.1681	289.1909	238.5397
ALPSORK1	238.7649	197.2436	173.3853
ALPSORK5	252.4335	200.5174	**161.0399**
ALPSORK10	245.9198	177.1890	159.5583
ALPSORLRK1	213.1332	187.8548	180.0100
ALPSORLRK5	224.9485	187.6209	**173.5455**
ALPSORLRK10	217.2850	174.7711	162.8595
ALPSORLPK1	215.4489	186.2649	173.3645
ALPSORLPK5	228.0512	185.7157	166.4767
ALPSORLPK10	221.1316	181.9990	166.8288

Table 5.14: Statistical Significance of DTLZ1

on only convergence towards the Pareto front, and this makes a significant difference in performance consideration of these algorithms. However, our primary goal was to find out is there any statistical significance or not, which this technique permits. We used the Wilcoxon Mann Whitney rank sum test test with 0.05 confidence level to do the statistical significance for these algorithms [4].

5.8.1 DTLZ1

In DTLZ1 (Table 5.14) , we see that ALPS has the least mean distance in all of the test instances, where GAORK1, GAORK5, and GAORK10 cannot perform well in many cases. ALPSORLRK10 made significant results for 6 objectives and ALPSORK5, ALPSORK10 got small means for 6 and 8 objectives. ALPSORLRK1 works well for 4 objectives. Also, ALPS, GA, ALPSORK10 and ALPSORLRK5 were significantly better for 8 objectives.

5.8.2 DTLZ2

In DTLZ2 (Table 5.15), we see that GA and ALPS were significantly better in all of the test instances . In DTLZ2, no objective reduction algorithm performed well except ALPSORK5 and ALPSORK10, where ALPSORK5 is performing well for 4 objectives and ALPSORK10 is for 4 and 6 objectives.

Algorithm	**Results(M=4)**	**Results(M=6)**	**Results(M=8)**
GA	**0.8237**	**0.9664**	**0.8916**
ALPS	**1.1714**	**1.0642**	**1.1985**
GAORK1	2.4406	2.3977	2.4365
GAORK5	2.3540	2.3756	2.4187
GAORK10	2.4242	2.3823	2.4175
ALPSORK1	1.62 21	1.7690	1.7701
ALPSORK5	1.6121	**1.4575**	1.4437
ALPSORK10	**1.5401**	**1.5221**	1.4948
ALPSORLRK1	1.8076	1.7494	1.7798
ALPSORLRK5	1.8591	1.8911	1.8531
ALPSORLRK10	1.9348	1.8703	1.7705
ALPSORLPK1	1.9049	1.7946	1.8540
ALPSORLPK5	1.8953	1.8948	1.8819
ALPSORLPK10	1.8785	1.8146	1.8579

Table 5.15: Statistical Significance of DTLZ2

5.8.3 DTLZ3

Like DTLZ2, ALPS got the minimum distance for all of the test cases (see Table 5.16). The main difference with DTLZ2 is, ALPSORK5 got the least mean among all of the objective reduction algorithms.

5.8.4 DTLZ4

For DTLZ4 (Table 5.17), ALPSORK10, ALPSORLRK1, and ALPSORLRk10 got minimum mean 3 for 6 and 8 objectives. Like other instance, GA and ALPS outperformed all of the algorithms.

5.8.5 DTLZ7

In DTLZ7 (Table 5.18), all of the objective reduction algorithms statistically perform well, specially all layer-based objective reduction algorithms. Individual-based reduction performed well only for 4 objectives. GA and ALPS cannot perform well in this test suite.

5.8.6 Summary of Statistical Difference

Table 5.19 shows the overall result of all algorithms based on statistical significance. Again, it is necessary to remember that the algorithm that works well in IGD or

Algorithm	**Results(M=4)**	**Results(M=6)**	**Results(M=8)**
GA	1120.38	1101.49	**1013.56**
ALPS	**831.63**	**827.58**	**882.77**
GAORK1	2046.40	2050.87	2028.20
GAORK5	1907.48	1955.34	1983.13
GAORK10	2032.72	1964.64	1988.24
ALPSORK1	1443.35	1409.77	1438.16
ALPSORK5	1365.68	1349.94	1260.97
ALPSORK10	1460.44	1401.42	1372.63
ALPSORLRK1	1409.07	1405.43	1415.04
ALPSORLRK5	1419.09	1485.61	1425.96
ALPSORLRK10	1413.24	1458.09	1433.87
ALPSORLPK1	1540.57	1440.29	1464.42
ALPSORLPK5	1513.29	1515.39	1578.57
ALPSORLPK10	1493.53	1451.31	1436.78

Table 5.16: Statistical Significance of DTLZ3

Algorithm	**Results(M=4)**	**Results(M=6)**	**Results(M=7)**
GA	**0.8504**	**0.9294**	**1.0461**
ALPS	**1.0487**	**1.1154**	**1.2283**
GAORK1	2.4200	2.3736	2.4162
GAORK5	2.3278	2.3715	2.3973
GAORK10	2.3416	2.3610	2.3854
ALPSORK1	1.6288	1.6239	1.6128
ALPSORK5	1.4911	1.4285	1.4788
ALPSORK10	1.4066	1.4791	1.4704
ALPSORLRK1	1.5710	1.5031	1.4688
ALPSORLRK5	1.6672	1.5467	1.5150
ALPSORLRK10	1.5746	1.4198	1.5353
ALPSORLPK1	1.6023	1.5148	1.5209
ALPSORLPK5	1.6153	1.5165	1.6177
ALPSORLPK10	1.6311	1.5222	1.4762

Table 5.17: Statistical Significance of DTLZ4

Algorithm	Results(M=4)	Results(M=6)	Results(M=7)
GA	16.1133	25.0659	32.6283
ALPS	16.9646	24.6028	34.4722
GAORK1	29.3621	47.3527	63.2068
GAORK5	32.6001	48.9782	64.1198
GAORK10	32.6155	49.6901	65.0988
ALPSORK1	**12.7930**	20.9122	27.3136
ALPSORK5	13.2474	20.7571	26.5202
ALPSORK10	13.5960	20.5256	26.3222
ALPSORLRK1	**12.9228**	**19.6684**	24.2324
ALPSORLRK5	**11.4299**	**17.6387**	22.2964
ALPSORLRK10	**11.6656**	**17.6767**	22.7714
ALPSORLPK1	**11.8510**	**17.6386**	23.0869
ALPSORLPK5	**11.5574**	**17.4697**	22.1732
ALPSORLPK10	**11.7429**	**17.3821**	22.7586

Table 5.18: Statistical Significance of DTLZ7

hypervolume may not work well in statistical difference. In this approach, an algorithm works well when it has a solution with minimum distance from the Pareto front. From Table 5.19, we see that ALPS and GA did better in 12 and 7 instances. ALPSORK5 and ALPSORK10 objective reduction algorithms performed well for 2 test cases . All layer-based algorithm performed well specially layer-based random reduction ALPSORLRK5 performed well for 7 test cases.

5.9 Summary

In summary, GA and ALPS without objective reduction performed well in many cases where layer-based objective reduction algorithms made better performance among all the reduction algorithms. Table 5.20- Table 5.23 are representing the overall summary of all of the experiments. In Table 5.20 and 5.22, considering all of the experiments, we included how many times an algorithm was better in hypervolume and IGD measurement. From Table 5.21 and 5.23, we can know that which algorithm is better for which test cases using hypervolume and IGD result. In hypervolume (Table 5.20) measurement, GA defeated all of the algorithms. ALPSORK5 and ALPSORk10 were best for two instances. All layer-based reduction methods beat other algorithms for 1 time. According to IGD (Table 5.22), GA and ALPS exceed different algorithms in 6 cases. ALPSORK5, ALPSORLRK5, and ALPSORLPK10 made the best result for one instance. Among all objective reduction algorithms, individual based-random

Algorithm	**Better In**
GA	7
ALPS	12
GAORK1	0
GAORK5	0
ALPSORK1	0
ALPSORK5	2
ALPSORK10	2
ALPSORLRK1	3
ALPSORLRK5	7
ALPSORLRK10	3
ALPSORLPK1	3
ALPSORLPK5	3
ALPSORLPK10	3

Table 5.19: Comparison between all algorithms (Statistical Significance)

reduction, and layer-based coefficient reduction performed better. Considering hypervolume and IGD measurement, objective reduction with GA cannot find good results for these test cases, whereas ALPS objective reduction outperforms others in many situations. Also, it is apparent that among all reduction algorithms, layer-based coefficient reduction, and individual-based random reduction in ALPS reduction performed well in both IGD and hypervolume measurements. From Figure 5.2 and 5.3, it is obvious why GA objective reduction could not be performed well in any test cases. From Figure 5.3, we see that the average score of objectives F1 became steady for long time and the objective score of F4 is noisy while the average score of objectives is changing over the generation in layer-based coefficient masking (Figure 5.2). In the layer-based coefficient masking, algorithms are trying to get a good solution which will be more near to the optimal solution.

Algorithm	**Better In**
GA	17
ALPS	3
GAORK1	0
GAORK5	0
GAORK10	0
ALPSORK1	0
ALPSORK5	2
ALPSORK10	2
ALPSORLRK1	1
ALPSORLRK5	1
ALPSORLRK10	0
ALPSORLPK1	1
ALPSORLPK5	0
ALPSORLPK10	1

Table 5.20: Comparison between all algorithms (Hypervolume)

Problem	**Better Algorithm**			
	M=4	**M=6**	**M=8**	**M=10**
DTLZ1	ALPSORK5	GA	GA	GA
DTLZ2	GA	ALPS	GA	GA
DTLZ3	ALPSORLPK5	GA	GA	GA
DTLZ4	GA	GA	ALPSORK10	ALPSORLRK1
DTLZ5	ALPSORK5	ALPS	ALPSORK10	ALPSORLPK1
DTLZ6	ALPS	ALPS	GA	GA
DTLZ7	GA	GA	GA	GA

Table 5.21: Comparison between all algorithms (hypervolume)

Algorithm	**Better In**
GA	6
ALPS	6
GAORK1	0
GAORK5	0
GAORK10	0
ALPSORK1	0
ALPSORK5	1
ALPSORK10	0
ALPSORLRK1	0
ALPSORLRK5	1
ALPSORLRK10	0
ALPSORLPK1	0
ALPSORLPK5	0
ALPSORLPK10	1

Table 5.22: Comparison between all algorithms (IGD)

Problem	**Better Algorithm**		
	M=4	**M=6**	**M=8**
DTLZ1	ALPS	ALPS	ALPS
DTLZ2	GA	GA	GA
DTLZ3	ALPS	ALPS	ALPS
DTLZ4	GA	GA	GA
DTLZ7	ALPSORK1	ALPSORLRK5	ALPSORLPK10

Table 5.23: Comparison between all algorithms (IGD)

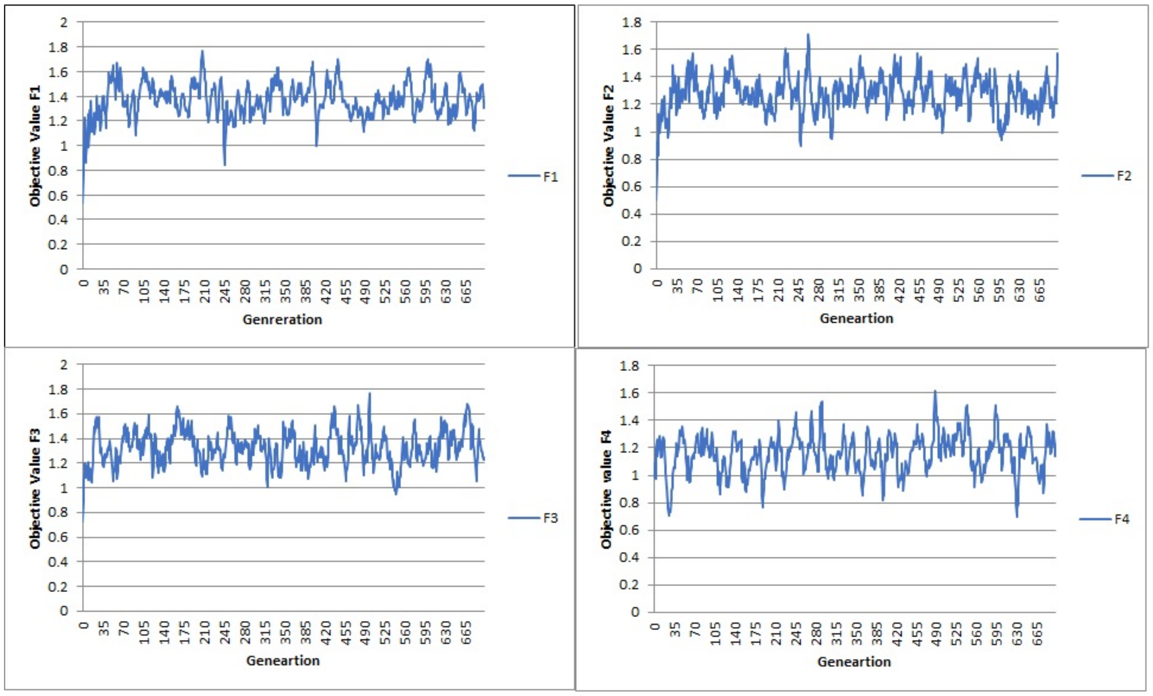

Figure 5.2: Average score for layer-based coefficient objective reduction of ALPS (DTLZ2)

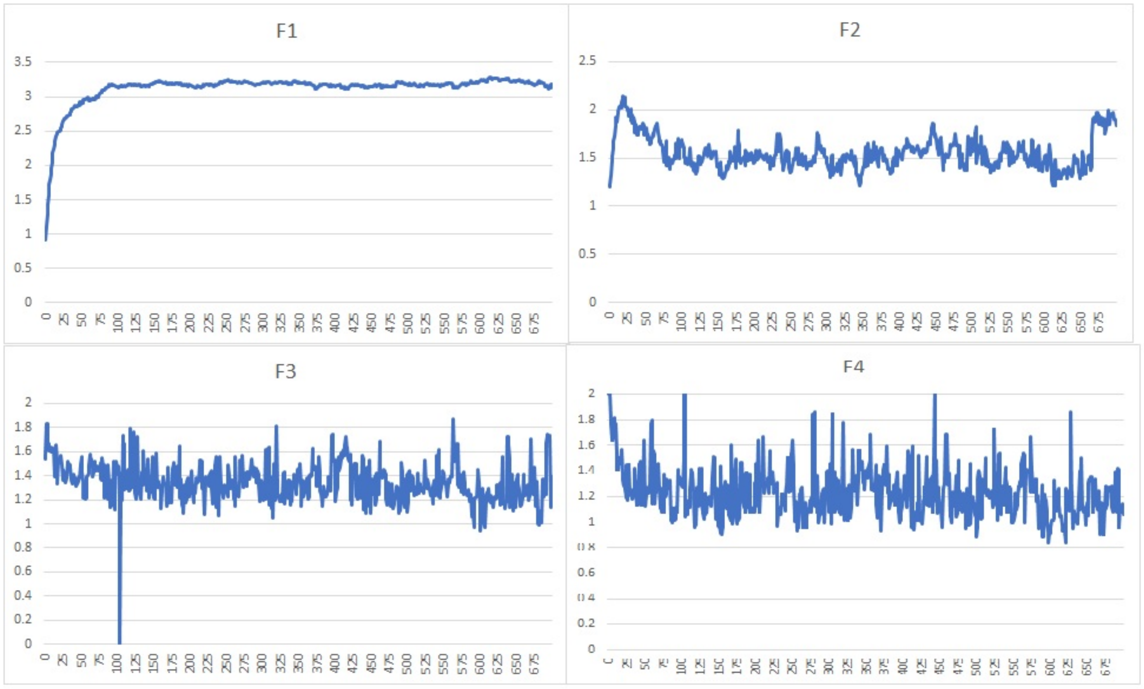

Figure 5.3: Average score for individual-based random objective reduction of GA (DTLZ2)

Chapter 6

Conclusion

6.1 Summary

A rich diversity of optimized solutions is considered one of the challenging tasks for many MOEAs. The complexities involved to get such solutions increases with the growth of the number of objectives. We conducted this thesis based on two primary purposes. The first one is to introduce an algorithm using ALPS and SR. The second goal is to explore diiferent approaches of objective reduction with ALPS and SR. To reach our goals, we did many benchmark experiments to observe and analyze the results.

In the first part, we did experiments with the original GA and ALPS on the DTLZ test suite using SR. We found that GA and ALPS both did better in many problems. However, ALPS provided better results in the IGD performance while GA has better hypervolume measurements. In IGD, we considered 15 test instances. ALPS performed well in 7 instances where GA was good for 8 instances. Using the IGD results, we can say that ALPS gives solutions that are more converged to the true Pareto front. For the hypervolume result, we used 21 instances for comparing results. GA was better for 16 test cases while ALPS did better for 5 test cases. From this result, it is evident that the solutions from the GA are more diverse than ALPS, and they are not in the same sub-area of Pareto front as in ALPS.

Next, we implemented random objective reduction and correlation-based objective reduction. We used random reduction for GA and ALPS, and correlation-based reduction in ALPS. We made two variations for random-based reduction: individual-wise random masking, and layer-wise random masking. In correlation-based objective reduction, we used Pearson correlation to calculate the correlation between two objectives, and we omitted two highly correlated objectives using BSS.

Result showed that GA with random-based reduction could not provide better results in any problem set. Moreover, in most cases of objective reduction ALPS with random-based reduction outperform GA with random reduction. On the other hand, in general, correlation-based masking worked well for many test cases. The reason is that correlation-based masking can determine redundant objectives, which is important in objective reduction. In conclusion, GA and ALPS without objective reduction were competitive in many problems, while correlation-based objective reduction is preferable for other problems. Random reduction is not a recommended strategy.

6.2 Future Work

A myriad of new work can be done to extend our research.

Because of time limitations, we could not adequately compare our result with others research.
Initially, we tried to compare our results with other research papers. However, most other research uses Pareto dominance. As SR and Pareto dominance are two different approaches and generate different kinds of solutions, it is hard to compare them. In spite of that, we wanted to compare the hypervolume results of GA with other papers. But most papers used a mean result of the 30 hypervolume results. This is easy with Pareto dominance since each run produces multiple non-dominated solutions. However, as we get one solution from one SR run, we need many runs to make 30 hypervolume results for our research. We did compare our IGD results with NSGA-II runs. However, the results are not easily compared as both used different approaches and we did not have access to their details of these experiments. Because of this limitation, we focused on comparing our algorithms with each other.

Work to find an appropriate means for comparing the SR with Pareto dominance should be done. A recommended approach is to re-run other algorithms (NSGA-II, SPEA2, etc.) and pay consideration to issues such as hypervolume with statistical significance. Besides, when measuring hypervolume, SR cannot provide a good result as the solutions are not diverse like Pareto based EAs. Future work can be conducted to increase the hypervolume result for SR by introducing diversity strategies. Although ALPS outperformed objective reduction algorithms, it is not competitive with other MOEAs like NSGA-III and SPEA2. This is because, these algorithms use sophisticated diversity mechanisms. ALPS can be modified to get results that are closer to the Pareto front. Correlation-based objective reduction can be implemented with many variations. In our correlation-based approach, we reduced only two objectives for every problem set. Further reduction of objectives should be explored. We know that correlation reduction gave the best performance amongst all of the reduction algorithms we tried. By discarding additional objectives, it may outperform regular GA and ALPS. Comparisons of it with other popular reduction strategies such as Principle Component Analysis (PCA) [5] should also be undertaken, as PCA may produce even better results. Finally, introducing new approaches in random objective reduction may bring improved performance. If it is possible to find redundant objectives using random masking, a satisfactory result may be seen for many MOEAs.

Bibliography

[1] Ecj. `https://cs.gmu.edu/~eclab/projects/ecj/`. Accessed: 2018-06-30.

[2] Jmetal. `http://jmetal.sourceforge.net/`. Accessed: 2019-07-30.

[3] Pearson correlation coefficient. `https://en.wikipedia.org/wiki/Pearson_correlation_coefficient`. Accessed: 2019-05-30.

[4] Willcoxon and Mann-Whitney u test. `http:http://sphweb.bumc.bu.edu/otlt/mph-modules/bs/bs704_nonparametric/BS704_Nonparametric4.html`. Accessed: 2019-07-30.

[5] Hervé Abdi and Lynne J. Williams. Principal component analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 2(4):433–459, July 2010.

[6] David W. Aha and Richard L. Bankert. A comparative evaluation of sequential feature selection algorithms. In *"Learning from Data: Artificial Intelligence and Statistics V"*, pages 199–206. Springer New York, 1995.

[7] Khalid A. Al-Abid. Effects of combining ALPS and SPEA2 genetic programming algorithms in a portfolio optimization problem. M.Sc IS, University College London, 2007.

[8] J. A. Andrews and J. R. Devine. Armature design for coaxial induction launchers. *IEEE Transactions on Magnetics*, 27(1):639–643, Jan 1991.

[9] Anthony Awuley. Feature selection and classification using age layered population structure genetic programming. *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 2417–2426, 2016.

[10] Peter J Bentley and Jonathan P Wakefield. Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms. In *Soft computing in engineering design and manufacturing*, pages 231–240. Springer, 1998.

[11] Leonardo C. T. Bezerra, Manuel López-Ibáñez, and Thomas Stützle. An empirical assessment of the properties of inverted generational distance on multi- and many-objective optimization. In Heike Trautmann, Günter Rudolph, Kathrin Klamroth, Oliver Schütze, Margaret Wiecek, Yaochu Jin, and Christian Grimme, editors, *Evolutionary Multi-Criterion Optimization*, pages 31–45. Springer International Publishing, 2017.

[12] Ram Bhusan Agrawal, Kalyanmoy Deb, and Ram Bhushan Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9, 06, 2000.

[13] Sanjib Biswas. Multi-objective genetic algorithms for multi-depot VRP with time windows. M.Sc Thesis, Brock University, 2017.

[14] Lucas Bradstreet. *The hypervolume indicator for multi-objective optimisation: calculation and use.* Phd. Thesis, University of Western Australia, 2011.

[15] Dimo Brockhoff and Eckart Zitzler. Objective reduction in evolutionary multiobjective optimization: Theory and applications. *Evolutionary computation*, 17(2):135–166, 2009.

[16] David W Corne and Joshua D Knowles. Techniques for highly multiobjective optimisation: some nondominated points are better than others. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 773–780. ACM, 2007.

[17] K. Deb and H. Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, Part I: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):577–601, Aug 2014.

[18] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, April 2002.

[19] Kalyanmoy Deb and Debayan Deb. Analysing mutation schemes for real-parameter genetic algorithms. *Int. J. Artif. Intell. Soft Comput.*, 4(1):1–28, February 2014.

[20] Kalyanmoy Deb and Deb Kalyanmoy. *Multi-Objective Optimization Using Evolutionary Algorithms.* John Wiley & Sons, Inc., New York, NY, USA, 2001.

[21] Kalyanmoy Deb and D Saxena. Searching for pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems. In *Proceedings of the World Congress on Computational Intelligence (WCCI-2006)*, pages 3352–3360, 2006.

[22] Kalyanmoy Deb, Lothar Thiele, Marco Laumanns, and Eckart Zitzler. Scalable multi-objective optimization test problems. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02*, volume 1, pages 825–830. IEEE, 2002.

[23] Carlos M Fonseca, Joshua D Knowles, Lothar Thiele, and Eckart Zitzler. A tutorial on the performance assessment of stochastic multiobjective optimizers. In *Third International Conference on Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 216, page 240, 2005.

[24] Mario Garza-Fabre, Gregorio Toscano Pulido, and Carlos A Coello Coello. Ranking methods for many-objective optimization. In *Mexican international conference on artificial intelligence*, pages 633–645. Springer, 2009.

[25] Michael Gircy. Image evolution using 2d power spectra. M.Sc Thesis, Brock University, 2018.

[26] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control and Artificial Intelligence.* MIT Press, Cambridge, MA, USA, 1992.

[27] Gregory S Hornby. Alps: the age-layered population structure for reducing the problem of premature convergence. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, pages 815–822. ACM, 2006.

[28] Saku Kukkonen and Jouni Lampinen. Ranking-dominance and many-objective optimization. In *2007 IEEE Congress on Evolutionary Computation*, pages 3983–3990. IEEE, 2007.

[29] Miqing Li, Shengxiang Yang, Xiaohui Liu, and Ruimin Shen. A comparative study on evolutionary algorithms for many-objective optimization. In Robin C. Purshouse, Peter J. Fleming, Carlos M. Fonseca, Salvatore Greco, and Jane Shaw, editors, *Evolutionary Multi-Criterion Optimization*, pages 261–275. Springer Berlin Heidelberg, 2013.

[30] Naili Luo, Xia Li, and Qiuzhen Lin. Objective reduction for many-objective optimization problems using objective subspace extraction. *Soft Computing*, 22(4):1159–1173, 2018.

[31] Justin Maltese, Beatrice M Ombuki-Berman, and Andries P Engelbrecht. A scalability study of many-objective optimization algorithms. *IEEE Transactions on Evolutionary Computation*, 22(1):79–96, 2018.

[32] Trent McConaghy, Pieter Palmers, Georges Gielen, and Michiel Steyaert. Simultaneous multi-topology multi-objective sizing across thousands of analog circuit topologies. In *2007 44th ACM/IEEE Design Automation Conference*, pages 944–947. IEEE, 2007.

[33] Audrey Opoku-Amankwaah. An age layered population structure genetic algorithm for the multi-depot vehicle problem. M.Sc Thesis, Brock University, 2017.

[34] Vikas Palakonda and Rammohan Mallipeddi. Pareto dominance-based algorithms with ranking methods for many-objective optimization. *IEEE Access*, 5:11043–11053, 2017.

[35] Robin C Purshouse and Peter J Fleming. Conflict, harmony, and independence: Relationships in evolutionary multi-criterion optimisation. In *International Conference on Evolutionary Multi-Criterion Optimization*, pages 16–30. Springer, 2003.

[36] Gade Pandu Rangaiah. *Multi-objective optimization: techniques and applications in chemical engineering*, volume 1. World Scientific, 2009.

[37] Noelia Sánchez-Maroño, Amparo Alonso-Betanzos, and María Tombilla-Sanromán. Filter methods for feature selection – a comparative study. In Hujun Yin, Peter Tino, Emilio Corchado, Will Byrne, and Xin Yao, editors, *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, pages 178–187, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg.

[38] L. While, P. Hingston, L. Barone, and S. Huband. A faster algorithm for calculating hypervolume. *IEEE Transactions on Evolutionary Computation*, 10(1):29–38, Feb 2006.

[39] Yuan Yuan, Yew-Soon Ong, Abhishek Gupta, and Hua Xu. Objective reduction in many-objective optimization: evolutionary multiobjective approaches

and comprehensive analysis. *IEEE Transactions on Evolutionary Computation*, 22(2):189–210, 2018.

[40] Eckart Zitzler, Marco Laumanns, and Lothar Thiele. SPEA2: Improving the strength pareto evolutionary algorithm. volume 103. Eidgenössische Technische Hochschule Zürich (ETH), Institut für Technische , 2001.