EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION VIA DIFFERENTIAL EVOLUTION

CHONG JIN KIAT

NATIONAL UNIVERSITY OF SINGAPORE

2015

EVOLUTIONARY MULTI-OBJECTIVE OPTIMIZATION VIA DIFFERENTIAL EVOLUTION

CHONG JIN KIAT

(M.Sc. , NUS)

A THESIS SUBMITTED

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

NATIONAL UNIVERSITY OF SINGAPORE

2015

DECLARATION

I hereby declare that the thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis

This thesis has also not been submitted for any degree in any university previously

_____

Chong Jin Kiat

14 August 2015

Acknowledgments

First of all, I would like to express my sincere and utmost gratitude to my supervisor, Associate Professor Tan Kay Chen, for his invaluable support, professional guidance and encouragement throughout my Ph.D. studies in the National University of Singapore. It has been an extremely tough journey for me as a part-time Ph.D. candidate, and his patience, understanding and graciousness have given me continuous motivation in my research work.

I would also like to extend my greatest thanks to my lab mates for the sharing of their experiences and the help rendered to me from time to time, especially to Vui Ann, Sen Bong, Qiu Xin and Arrchana who have given me a lot of useful information and valuable inputs throughout my years of research. I would also like to thank my supervisors at the Defence Science and Technology Agency for their kind support and understanding which allows me to manage my work together with my studies.

Finally, I would like to express my deep appreciation to my loved ones who have stood by me in all these years of my research, especially to my beautiful wife Michelle for her love, care and support and to my lovely daughter Sarah for giving me the joy during the difficulties encountered in my course of studies. Last but not least, I would like to give my greatest thanks to God for giving me the wisdom and strength to complete my studies. Without Him in my life, this journey will never be possible.

Contents

Summary

Multi-objective optimization is an area that deals with decision making based on multiple criteria, and is concerned with mathematical optimization problems that involves two or more conflicting objectives that need to be optimized simultaneously. It has been extensively applied in many fields of engineering, logistics, economics, bioinformatics, finance and any many other applications in real-life. Differential evolution is a simple but powerful evolutionary optimization algorithm with many successful applications. The primary aim of this thesis is to develop a novel opposition-based self-adaptive differential evolution algorithm in the context of multi-objective optimization and to implement the algorithm to solve both theoretical and real-life application problems with vastly different characteristics and representation schemes.

First of all, a novel opposition-based self-adaptive hybridized differential evolution algorithm (OSADE) is devised. This is achieved by incorporating opposition-based learning into a self-adaptive mechanism for the control parameters (mutation factor and crossover rate) of Differential Evolution, and is then hybridized with the multi-objective evolutionary gradient search (MO-EGS) which acts as a form of local search. OSADE is applied on a comprehensive suite of test benchmark problems and compared with several state-of-the-art evolutionary algorithms. The experimental results indicate that OSADE is able to achieve superior optimization performance for the problems tested on. In addition, OSADE is also able to achieve good scalability in terms of both the number of decision variables and objective functions.

OSADE is also extended to handle many-objectives optimization whereby its opposition-based self-adaptive mutation scheme is joined with an opposition-based local mutation scheme, and one of the schemes will be selected according to a linear decreasing probability rule. The formulated novel mutation scheme is then incorporated into a grid-based framework whereby grid concept is used to determine the mutual relationship of individuals in a grid environment, and the fitness of the individuals in the population are distinguished by a set of grid criteria.

The resultant grid-based DE variant is termed as GrDE and is tested on a set of many-objective problems from the DTLZ and WFG test suites. Experimental results demonstrated the proposed novel algorithm GrDE is as competitive as the other algorithms under comparison in terms of achieving a well-approximated and well-distributed solution set for the many-objective problems in this thesis.

Finally, a study of the extension of OSADE to solve a multi-objective multiple travelling salesmen problem (MmTSP) is conducted. For this study, the opposition-based self-adaptive differential evolution operator found in the original OSADE is being implemented in a decomposition-based framework instead of the domination-based framework as used in the original OSADE. As the DE operator is originally intended for continuous optimization, it cannot be directly applied for the MmTSP which is a combinatorial problem. As such, a heuristic rule known as the Smallest Position Value (SPV) rule is applied in the algorithm to convert the solutions in real-number representation as found by the DE operation into chromosomes of integer-value representation so that the evaluation of the solutions according to the MmTSP can be achieved. In order to enhance the algorithm, the DE operator is then hybridized with the multi-point evolutionary gradient search (EGS) to act as a form of local search. The simulation results reveal that the proposed resultant algorithm D-OSADE is able to generate a set of diverse solutions with good proximity results.

Lists of publications

The publications that were published, under revision, and in the process of submission during the course of my research are listed as follows.

Journals

1. J. K. Chong and K. C. Tan, "A Novel Multi-objective Memetic Algorithm based on Opposition-based Self-adaptive Differential Evolution", *Memetic Computing*, accepted.

2. J. K. Chong and K. C. Tan, "An Investigation of a Grid-based Differential Evolution Algorithm for Many-objective Optimization", to be submitted.

3. J. K. Chong and K. C. Tan, "An Opposition-based Self-adaptive Differential Evolution Algorithm with Decomposition for Solving the Multi-objective Multiple Traveling Salesman Problem", to be submitted.

4. V. A. Shim, K. C. Tan, J. Y. Chia, and J. K. Chong, "Evolutionary Algorithms for Solving Multi-objective Travelling Salesman Problem", *Flexible Services and Manufacturing Journal*, vol. 23, no. 2, pp. 207-241, 2011.

Conferences

1. J. K. Chong and K. C. Tan, "An Opposition-based Self-adaptive Hybridized Differential Evolution Algorithm for Multi-objective Optimization (OSADE)", *in Proceedings of the 18th Asia Pacific Symposium on Intelligent and Evolutionary Systems*, pp. 447-461, 2014.

List of Tables

List of Figures

Chapter 1

Introduction

In our daily lives, we are constantly dealing with multiple criteria decision making in many different areas. In fact, many real-world problems in a wide spectrum of applications, like engineering, finance, logistics, bioinformatics and many others, involve the difficult task of optimizing several conflicting objectives simultaneously. For example, an investment manager will need to consider both the maximization of returns and the minimization of risk when he makes an investment decision. This is the case of risk-return tradeoff, and an investment can render higher returns only if it is subject to higher risk, and therefore the two objectives are conflicting. Such a type of problem is commonly known as a multi-objective optimization problem (MOOP). MOOP is not a trivial optimization problem because no single solution is optimal for all the objectives in the problem. As such, the search methods catered for MOOP must be able to find a set of alternative solutions that is representative of the tradeoff between the different conflicting objectives. In addition, the presence of complex, non-linear, non-differential or even high dimensional search space could result in additional difficulties when solving the MOOPs. Due to these challenges, it is seen that most deterministic methods face difficulties when dealing with MOOPs, and hence unable to obtain reasonable solutions within limited computational resources. In addressing these issues, the use of stochastic search techniques are seen to be a better alternative technique over deterministic methods in dealing with MOOPs [1, 2].

As seen from literature, the use of evolutionary algorithms (EAs) are shown to be effective in solving basic MOOPs. EAs are population-based approaches that draw inspiration from biological evolution [3, 4], and they are usually stochastic by nature and comprise several general characteristics. First and foremost, EAs are able to sample multiple candidate solutions in a single simulation run. Another important feature in EAs is the adoption of the idea of the survival-of-the-fittest to allow candidate solutions of better fitness to be retained after they have been found. Lastly, EAs utilize several stochastic mechanism inspired from biological evolution which include reproduction, mutation, recombination and selection to explore the search space. With these characteristics, we are able to find the successful use of EAs in a wide range of application problems. Some examples of EAs being used in applications include the optimization of jobs scheduling in large-scale grid applications [5], computational optics [6], telecommunications [7], and economic power dispatch [8], just to name a few.

Over the decades, several different EAs that have been developed to handle MOOPs, and examples include genetic algorithms, evolutionary programming, evolutionary strategies, and particle swarm optimization, just to name a few. However, there are certain limitations associated with EAs as gathered from the understanding from literature. Firstly, there is no guarantee that optimal solutions can be found by EAs in a finite amount of time. The use of EAs also involves certain control parameters that are required to be tuned for optimal performance, and this may involve a tedious process of trial and error for the parameter tuning before they can be used for the handling of the problems in an effective and efficient way. Besides these, EAs may also suffer from other weaknesses like loss of diversity, slow convergence and stagnation of population, and these weaknesses will become even more prominent in the event if the EAs are being presented with multi-modal problems which possess several local and global optima. In multi-modal problems which may be seen in some practical applications, EAs may also take a considerable amount of time to locate the global optima or may even get trapped in a local optima. Hence this will create difficulties for researchers in terms of balancing both solution

accuracy and convergence rate for such type of problems in multi-objective optimization.

Differential evolution (DE) is a simple and efficient population-based EA that has been reported in several studies on its high robustness, fast convergence speed and good solution quality, which makes it a very popular EA in the evolutionary computation community. Due to such strengths seen in DE, the use of this EA may offer an answer to some, but not all, of the aforementioned limitations. In order to overcome these limitations, several improvement works have been performed on DE so as to allow it to be able to meet the the aim of balancing solution accuracy and convergence rate while maintaining diversity in the population for different types of MOOPs. In order to achieve fast convergence speed and effective global search capability concurrently, researchers have explored the use of hybridizing different EAs for MOOPs as well as formulating memetic algorithms which integrates local search in EAs, and these efforts can also be found in DE [9, 10]. Researchers have also introduced the use of adaptive or self-adaptive DE variants [11–14] that eliminates the need to undergo the tedious trial-and-error process of tuning the control parameters in DE to an optimal setting for the problems tested on. However, there is still a lot of room for improvement in the aspect of improving DE as a whole, and this thesis aims to explore feasible ways of enhancing the basic DE algorithm to handle a wide range of MOOPs.

1.1 Multi-objective Optimization

Multi-objective (MO) optimization is an area of multiple criteria decision making, and is concerned with mathematical optimization problems that involve two or more objectives that are required to be optimized simultaneously. The principles behind MO optimization has been extensively over the years, and MO optimization is also widely applied in many applications fields that include engineering, finance, logistics, and bioinformatics, just to name a few. This section presents the basic fundamentals and principles behind MO optimization.

1.1.1 Basic Concepts

A multi-objective optimization problem (MOOP) is a non-trivial and complicated problem which involves the simultaneous optimization of several contradicting objectives in order to satisfy problem constraints. A MOOP consists of a set of objective functions that can either be maximized or minimized subject to a set of constraints that limit the set of possible outcome which is also known as the solution space. Without any loss of generality, an MOOP can be mathematically formulated in the minimization case as follows:

Minimize:

$$\mathbf{f}(\mathbf{x}) = (f_1(\mathbf{x}), f_2(\mathbf{x}), ..., f_m(\mathbf{x})) \tag{1.1}$$

subject to:

$$\mathbf{g}(\mathbf{x}) \leq 0$$

$$\mathbf{h}(\mathbf{x}) = 0$$

where $\mathbf{f}(\mathbf{x})$ is the set of objective functions, $\mathbf{f}(\mathbf{x}) \in \mathbb{R}^m, \mathbb{R}^m$ is the objective space, m is the number of objective functions, $\mathbf{x} = (x_1, x_2, ..., x_n)$ is the decision vector, $\mathbf{x} \in \mathbb{R}^n$, $\mathbb{R}^n$ is the decision space, n is the number of decision variables, $\mathbf{g}$ is the set of inequality constraints, and $\mathbf{h}$ is the set of equality constraints.

In an MOOP, it is different from a single-objective optimization problem as there is no single optimal solution but the optimal solutions exist as a set of non-dominated solutions which is representative of the tradeoff between the multiple conflicting objectives. This set of non-dominated solutions is also known as the Pareto optimal set whereby the solutions in this set cannot be improved in any objective without causing degradation in at least one of the objectives. As such, the role of fitness assignment to every solution for the MOOP is of high importance as this will promote the survival of fitter and less crowded solutions to the next generation.

1.1.2 Pareto Optimality and Pareto Dominance

The concepts of Pareto optimality and Pareto dominance are fundamental in MO optimization, with Pareto dominance forming the basis of solution quality.

Let $\mathbf{v} = (v_1, ..., v_n)$ and $\mathbf{w} = (w_1, ..., w_n)$ represent two decision vectors of solutions that consist of n decision variables. Through the definition of Pareto dominance, there can be three possible relationships in the minimization case between two solutions [15–17] as follows:

1. Strong dominance: $\mathbf{v}$ strongly dominates $\mathbf{w}$ ($\mathbf{v} \prec \mathbf{w}$) if and only if

$$f_i(\mathbf{v}) < f_i(\mathbf{w}) \; \forall i \in \{1, 2, ..., m\} \tag{1.2}$$

2. Weak dominance: $\mathbf{v}$ weakly dominates $\mathbf{w}$ ($\mathbf{v} \preceq \mathbf{w}$) if and only if

$$f_i(\mathbf{v}) \leq f_i(\mathbf{w}) \text{ for } i \in \{1, 2, ..., m\} \text{ and } \exists \; f_i(\mathbf{v}) < f_i(\mathbf{w}) \text{ for at least one } i \tag{1.3}$$

3. Incomparable: $\mathbf{v}$ is incomparable with $\mathbf{w}$ ($\mathbf{v} \sim \mathbf{w}$) if and only if

$$\exists i \in \{1, 2, ..., m\} : f_i(\mathbf{v}) > f_i(\mathbf{w}) \text{ and } \exists j \in \{1, 2, ..., m\} : f_j(\mathbf{v}) < f_j(\mathbf{w}) \tag{1.4}$$

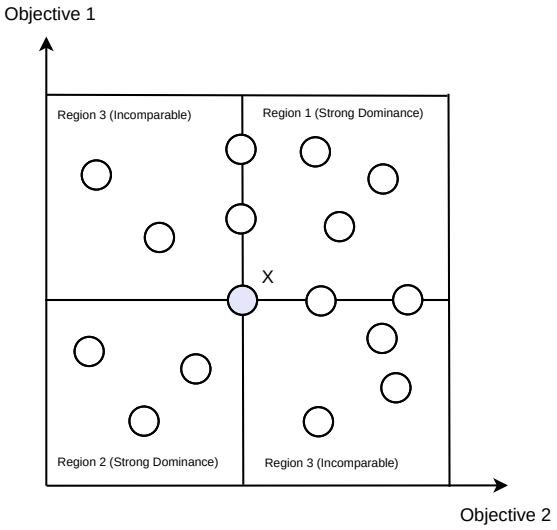

Figure 1.1: Illustration of Pareto dominance

In Figure 1.1, we use a two-objective minimization example to illustrate and further explain the dominance relationships between solutions. Let **X** be the reference solution in this example,

and the different dominance relationships are represented in three different regions. Solution **X** is said to strongly dominate the solutions that are resided in Region 1 because it is better in both objectives. However, the solution **X** is strongly dominated by solutions that are located in Region 2 since these solutions have values in both the objectives when compared to solution **X**. There are some solutions that are located on the boundaries of Region 1 and Region 3, and they possess the same objective values in one of the objectives as solution **X**, but solution **X** has a lower value in another objective. As such, solution **X** is said to weakly dominate these solutions. Lastly, the solutions that are found in Region 3 are superior in one of the objectives, but inferior in the other objective when compared to solution **X**. Hence these solutions are said to be incomparable to solution **X**.

In the context of MO optimization, a feasible decision vector $\mathbf{x}^* \in \mathbb{R}^n$ and its corresponding outcome, which can be represented as $\mathbf{f}(\mathbf{x}^*) \in \mathbb{R}^{\mathbf{m}}$, is Pareto optimal if there does not exist another decision vector that dominates it. The set of all the Pareto optimal decision vectors is called the Pareto optimal set (PS) and the corresponding outcome, which is also known as the objective vectors, is often called the Pareto optimal front (PF) [3]. An example of the Pareto optimal front of an MOOP with two objectives is as illustrated in Figure 1.2. For simplicity, the terms 'weakly dominate' and 'strongly dominate' will both be referred to as 'dominate' for the rest of the thesis. In the figure, solutions B and C dominate solution E, and the solutions A, B, C and D are non-dominated to each other. The set of non-dominated solutions (A, B, C, and D) forms the Pareto optimal front.

1.1.3 Multi-objective Optimization (MO) Goals

Real-world MO optimization problems usually have complex objective functions and constraints, and the information regarding the Pareto optimal front and its tradeoff is usually limited or not known *a priori*. In the absence of any clear preference on the part of the decision-maker, the

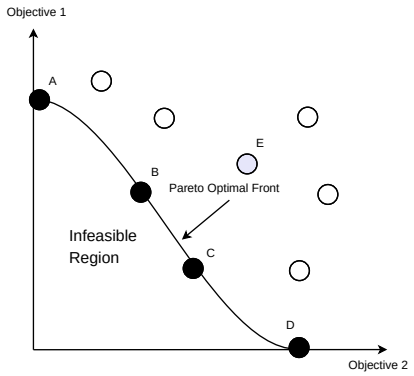

Figure 1.2: Illustration of Pareto optimal front

ultimate goal of MO optimization is to discover the entire tradeoff. However, by definition, this set of objective vectors is potentially an infinite set, and it is likely not possible to find the entire tradeoff. Moreover, the presence of too many alternatives could also be too overwhelming for the decision-making process. As such, it could be more practical to settle for the discovery of as many non-dominated solutions as possible with limited computational resources. More precisely, the aim of MO optimization is to find an approximate set of solutions that is as close to the Pareto optimal front as possible. The approximate set should satisfy the following optimization goals [15, 18].

1. Minimize the distance between the obtained solutions and the Pareto optimal front.

2. Obtain a good distribution of the obtained solutions along the Pareto optimal front.

3. Maximise the spread of the obtained solutions along the Pareto optimal front.

The first goal of convergence or proximity is the first and foremost consideration of all optimization problems, whereas the other two optimization goals of maximizing diversity and spread of solutions are entirely unique to MO optimization. The second goal which is related to diversity defines how well is the coverage of the obtained solutions in the optimal space, while the third goal which is related to spacing defines how evenly the obtained solutions are being distributed in the optimal space. The rationale of finding a diverse and evenly distributed

approximate set of solutions close to the Pareto optimal front is to provide the decision-maker with sufficient information about the tradeoffs between the multiple solutions before the final decision is made. It should also be noted that the goals of convergence and diversity are somewhat conflicting in nature, and this explains why multi-objective optimization is much more difficult than single-objective optimization.

1.1.4 Different Approaches for Multi-objective Optimization

In the past few decades, we are able to see many approaches for multi-objective optimization frameworks that are being proposed to solve MOOPs. An approach for multi-objective optimization simply refers to the way how an optimization algorithm adopts for the handling of multiple conflicting objectives. Here, an overview of the general classification of the different approaches for multi-objective optimization is presented as follows:

1. **Preference-based Approach:** The fundamental concept behind this approach is the aggregation of the multiple conflicting objectives in an MOOP into a single-objective optimization problem or to use preference knowledge of the problems so that the focus can be on certain objectives in the problem to be optimized. EAs that are commonly used for the solving of single-objective optimization problems can then be applied to solve the aggregated function. An example of this approach can be found in [19]. However, the preference-based approach has two major drawbacks. First, it is only able to obtain only one approximate optimal solution in a simulation run. Secondly, the approach requires the specification of a weight vector or the preference information of decision-makers for aggregation purposes.

2. **Domination-based Approach:** This approach allows the simultaneous optimization of all objectives in order to solve an MOOP. In this approach, fitness assignment to every solution is an imperative feature for the assurance of the survival of the fittest to the next generation. The concept of Pareto dominance is applied in this approach to compare and define the quality of every solution with regards to the entire solution set. As this approach is able to generate a

set of tradeoff solutions in an effective manner, it is highly popular among researchers in the field of evolutionary multi-objective optimization. However, the drawback of this approach is the weakening of the selection pressure when the number of objectives increases in the optimization problem. In addition, this approach must also be complemented with a diversity preservation scheme so that the goal of achieving a diverse set of solutions can be attained together with the goal of convergence.

3. **Decomposition-based Approach:** In this approach, an MOOP is being decomposed into several subproblems where each subproblem is formed using an aggregation-based technique, and all the subproblems will then be optimized concurrently. For this approach, the fitness of a solution will depend purely on the value of the aggregated objective. Hence the issue of the weakening of selection pressure encountered for domination-based approach in higher dimensional space will not be experienced in decomposition-based approach. Besides this, the approach does not need a diversity preservation scheme to be included as required in the domination-based approach. In the decomposition-based approach, the proper choice of uniformly distributed weight vectors can lead to a diverse set of solutions being produced if the subproblems are optimized well. The advantages seen in this approach has led to an increasing attention by the research community in recent years.

1.2 Evolutionary Algorithms in Multi-objective Optimization

From literature, many optimization techniques have been used to handle MOOPs. However, the use of evolutionary algorithms (EAs) is one of the most popular approach for solving MOOPs. As such, the emphasis of this thesis is on the implementation of EAs, and in particularly the use of Differential Evolution (DE) for the solving of MOOPs. In this section, the basic concept of a typical MOEA and its general flow in multi-objective optimization will be presented. In addition, a brief description of DE and its general flow will also be mentioned in the section as well.

1.2.1 Multi-objective Evolutionary Algorithms (MOEAs)

Multi-objective Evolutionary Algorithms (MOEAs) [3] are computing paradigms that utilize mechanisms inspired by biological evolution in driving the search towards optimal solutions. The success of this computing paradigm is mainly attributed to the underlying concepts of the survival-of-the-fittest and genetic recombination as found within MOEAs. MOEAs have gained their reputation as suitable tools or optimization due to two main reasons. First of all, MOEAs do not require any background knowledge of the problem when performing optimization. Secondly, MOEAs also do not require any gradient or directional information when they explore the search space. MOEAs generally involve heuristic genetic searches which allow them to perform searches in any fitness landscape in an efficient manner.

The basic idea of a typical MOEA is presented in Figure 1.3 . First, a set of solutions is randomly generated to form an initial population. The population will undergo evolution and a set of fitter solutions will be preserved over the evolutionary process. During the evolutionary process, the solutions will first be evaluated to obtain their objective values. This is followed by a fitness assignment process for all the solutions as their objective values cannot be directly translated as their fitness level. This is due to the fact that MOOPs involve multiple conflicting objectives that have to be simultaneously optimized. A simple way for the assignment of fitness to solutions is by the aggregation technique as performed in both preference-based and decomposition-based approaches for multi-objective optimization. Another method will be to determine the domination relationships between solutions as seen in domination-based approach. Next a selection process is followed in order to identify fitter solutions to become parent solutions to undergo genetic operations. A detailed description of the selection operators can be found in [20, 21]. These parent solutions will then mate among themselves through crossover operation for reproduction of offspring. In the crossover operation, two solutions are randomly chosen from the parent population which is also known as the mating pool. Exchanges of the alleles of the solutions will be done so that the offspring will inherit the characteristics from both the parent solutions. One of the factors

driving the successes of MOEAs in the exploration of search space lies in the proper implementation of the crossover operators like the single-point crossover or the multi-point crossover [22]. Besides the crossover operation, some of the generated offspring will also undergo mutation by means of the random perturbation of some alleles. The aim of the use of mutation operators like swap mutation, bit-flip mutation and polynomial mutation is to prevent the search from getting stuck in some local optima. After the genetic operations, both the generated offspring as well as the parent solutions will be stored in an archive. The fundamental principle of the evolutionary process which is the concept of the survival-of-the-fittest [23] follows. Through this concept, fitter solutions between the parent solutions and the offspring as determined by their fitness level will be selected into a new population that will undergo evolution in the next generation. The process is iterated until a stopping criterion is met. In MOEAs, the output solutions from the evolutionary process are multiple tradeoff solutions instead of a single best solution as found in typical EAs. It is also to be highlighted that MOEAs are required to maintain diversity of the promising solutions throughout the evolutionary process and this poses another level of challenge to them as well.

Begin
Initialization: Randomly initialize a population of *N* solutions
Evaluation: Evaluate all the solutions in the population
Do While ("Stopping Criterion is not satisfied")
Fitness assignment: Assign fitness to the population according to a fitness assignment scheme
Selection: Select a set of parent solutions
Variation: Perform crossover and mutation to the parent solutions to create offspring
Evaluation: Evaluate all offspring and store the offspring in an archive
Archiving: Store all parents and offspring in an archive
Elitism: Perform fitness assignment on all solutions in archive and select *N* best solutions to form a new population for the next generation
End Do
Output: Output the final set of solutions
End

Figure 1.3: Pseudo-code of a typical MOEA

1.2.2 Differential Evolution in Multi-objective Optimization

Differential evolution (DE) [24] is arguably one of the most powerful stochastic population-based algorithm for continuous global optimization in current practice. The basic operation of DE is very similar to most standard EAs. However, what makes DE different from traditional EAs is that the DE algorithm and its variants perturb the population individuals of the current generation with the scaled differences of randomly selected and distinct population individuals. Hence DE does not need any predefined probability distribution as it uses self-referential mutation for the generation of offspring, and adapts the step length intrinsically along the evolutionary process. The basic process flow of a typical DE algorithm is as illustrated in 1.4.

Begin
Initialization: Randomly initialize a population
Do While ("Stopping Criterion is not satisfied")
Evaluation: Calculate the fitness of each solution in the population
Mutation: Generate a mutant vector for each solution (target vector) in the population using a differential mutation strategy.
Crossover: Perform crossover between the mutant vector and target vector to form a trial vector.
Selection: Compare trial vector with target vector using the greedy criterion to determine whether the trial vector will become a member of the next generation. Replace the target vector with the trial vector if the latter has a better fitness.
End Do
End

Figure 1.4: Pseudo-code of a typical DE algorithm

Due to its simplicity and efficiency, it attracted a great amount of attention from researchers and engineers since its inception in 1995 [25], and this in turn led to the introduction of many DE variants for the improvement of the original DE algorithm for use in both theoretical and application problems [26, 27]. Due to the success of DE in single-objective optimization, the implementation of DE for multi-objective optimization has also been gaining research interest and attention from the research community [28, 29]. DE is also suitable to be integrated in any frameworks catered for multi-objective optimization due to the fact that a typical DE algorithm shares a common algorithmic flow as a typical MOEA. Thus, the fitness assignment approach

and the diversity preservation mechanism in the aforementioned approaches for multi-objective optimization can be directly employed by DE.

1.3 Opposition-Based Learning (OBL)

In the field of machine intelligence, there are several methods of handling challenging hyper-dimensional problems, and these include search and optimization techniques like genetic algorithms, connectionist approaches like neural networks, and feedback-oriented algorithms like reinforcement agents. Opposition-Based Learning (OBL) is a novel technique that originated from Tizhoosh [30] in 2005, and since then, this technique has emerged into a rapidly growing research area with several works [31–35] being proposed for the studying of both theoretical models and technical methods to handle different complex problems. The main underlying idea of Opposition-Based Learning in optimization lies in the concurrent consideration of an estimate and its corresponding opposite estimate that is closer to the global optimum, and this is done with the aim of arriving at a better candidate solution.

1.4 Self-adaptation in Evolutionary Algorithms

For the implementation of evolutionary algorithms (EAs), users not only have to decide on the appropriate encoding scheme and the evolutionary operators, but also need to determine the suitable parameter settings that are able to lead to the success of the algorithms. In order to achieve this, trial-and-error parameter and operator tuning is involved and this process is often time-consuming and this may lead to high computational costs being incurred. As such, researchers have looked into different approaches for the adaptation of parameters and operators in EAs [36–38]. From literature [39], there are three distinct categories of parameter adaptation techniques: deterministic, adaptive and self-adaptive control rules. For the deterministic method, the parameter values are modified according to certain predetermined logic and does not

consider any form of feedback from the search process. Adaptive rules integrate some form of feedback from the search process to guide in the parameter adaptation. As for the self-adaptive approach, the parameters are directly encoded into the population individuals, which will then undergo evolution together with the individuals. Hence self-adaptation will allow an evolution strategy in an evolutionary algorithm to adapt itself to any general class of optimization problems by reconfiguring the involved parameters accordingly without any user interaction. Self-adaptive rules mainly utilize the feedback from the search process such as the fitness values of individuals to help in the updating of parameters. Through this, parameter values that are involved with the fitter individuals will also have a higher chance of survival. In summary, self-adaptation has been found to be highly advantageous for the modifying parameter values in an automatic way throughout the evolutionary search process [40, 41].

1.5 Research Gaps and Goals/Scope

In this section, the main motivations for the work performed in this thesis as well as the goals of the research are described.

1.5.1 Research Gaps

Over the years, many new algorithms have been developed for MOOPs but their optimization performance in complex MOOPs still calls for improvements. This creates research gaps for the study on MOOPs which can be summarized as follows:

1. EAs have been extensively used in the solving of MOOPs. However, EAs suffer from several limitations like slow convergence, loss of diversity, stagnation of population, and often requires the need for a tedious trial-and-error process to determine optimal settings for the control parameters within them. Moreover, there is no assurance that optimal solutions can be found by the EAs as well. This calls for a need to explore the development of better algorithms that are able to overcome such limitations when handling MOOPs.

2. In real-world application problems, many of them are subject to more than three objectives and these problems are often referred to as many-objective problems. Many EAs are usually able to work well on problems with two or three objectives, but their performance deteriorate as the number of objectives increases. Hence there is a growing need to look into better approaches for EAs to handle this class of problems. Moreover, there are not many studies on the use of DE in the solving of many-objective problem and hence this leads to a motivation to explore this aspect.

3. Several attempts have been carried out to study the performance of different EAs in solving permutation-based problems like the multiple traveling salesman problem (mTSP). However, there are not many studies on the solving of multi-objective multiple traveling salesman problem. Furthermore, none has studied the performance of a self-adaptive differential evolution under the decomposition-based approach for multi-objective optimization.

1.5.2 Research Goals/Scope

The main aim of this study is to propose a DE algorithm that can solve a variety of MOOPs in an effective and efficient manner. The specific goals of this research are:

1. To develop an algorithm for MOOPs using an opposition-based self-adaptive differential evolution variant hybridized with a local search (OSADE).

2. To study the optimization performance of OSADE developed under the domination-based approach in a wide range of MOOPs.

3. To extend the use of OSADE in many-objective optimization problems.

4. To extend OSADE into a decomposition-based approach for multi-objective optimization and adapt it for solving a permutation-based problem like the multi-objective multiple travelling salesman problem (MmTSP).

The proposed algorithm, OSADE, employs the incorporation of opposition-based learning into a self-adaptive mechanism for differential evolution. The use of opposition-based learning is proposed as it can increase the probability of finding a near-optimal setting for the DE control parameters during the self-adaption process in the algorithm. With near-optimal settings being derived during the evolutionary process, better solutions can also be generated throughout the evolutionary process as well. In addition, hybridization of the novel opposition-based DE operator with the multi-objective evolutionary gradient search (MO-EGS) is done with the aim of improving the exploitation abilities of the overall algorithm. The extension of OSADE in many-objective optimization may provide an alternative approach on the use of DE in handling many-objective problems. For this, another novel DE mutation scheme is formulated by the synergy of the original opposition-based self-adaptive mutation scheme from OSADE with a local mutation scheme that is also incorporated with opposition-based learning. This mutation scheme is then incorporated into a grid-based framework to arrive at a novel DE variant called GrDE which displays promising results in terms of achieving a well-approximated and well-distributed set of solutions for the many-objective problems that it was tested on. Lastly, OSADE is extended to handle a multi-objective multiple traveling salesman problem (MmTSP) whereby its mutation scheme is being incorporated into the decomposition-based approach for multi-objective optimization. The search behaviour of the algorithm is further enhanced by hybridizing it with the multi-point evolutionary gradient search (EGS) which acts as a form of local search. As DE is originally intended for continuous optimization, this study also presents an avenue for DE to be used for permutation-based combinatorial optimization problems by incorporating a heuristic rule that allows the conversion of parameters with float values into permutation-based parameters required in the problem. The focus of this thesis is on the implementation of DE variants to handle MOOPs. However, there are many variances of MOOPs, and it would not be possible to consider all of them. Thus, the MOOPs considered in this thesis will be limited to a list of mentioned benchmark global continuous test problems in Section 2.8 and a permutation-based

combinatorial optimization problem. Other problems like combinatorial binary MOOPs, constrained MOOPs, and other real-world application problems will not be included in the scope of this thesis.

1.6 Contributions

In this thesis, an opposition-based self-adaptive differential evolution algorithm for solving a variety of multi-objective optimization problems has been devised. This algorithm is also extended to handle many-objective optimization problems and the multi-objective multiple traveling salesman problem which is a permutation-based combinatorial optimization problem. The design, implementation, results and analysis of the proposed algorithms in this study are also provided in detail. The itemized contributions of this thesis are listed below:

1. A novel memetic algorithm termed as OSADE is proposed by hybridizing a newly formulated opposition-based self-adaptive differential evolution variant with the multi-objective evolutionary gradient search as a form of local search. This marks the potential in the synergy between a self-adaptive DE and local search for solving multi-objective optimization problems. This contribution is realized in Chapter 3.

2. The potential of OSADE is further explored by extending it handle many-objective optimization. This study investigates the formulation of a novel mutation scheme which is then integrated into a grid-based framework to formulate a new grid-based differential evolution variant which displays promising performance in solving many-objective problems. This contribution is realized in chapter 4.

3. A first attempt to adapt a self-adaptive DE algorithm for solving the multi-objective multiple traveling salesman problem (MmTSP) has been carried out in this thesis. For this, OSADE is extended by integrating its mutation scheme into the decomposition-based approach for multi-objective optimization, and a heuristic rule is applied to allow DE to be used in this

combinatorial optimization problem. In this approach, the resultant decomposition-based differential evolution variant is hybridized with the multi-point evolutionary gradient search so as to allow further exploitation of the solutions found. Through this study, an adaptation approach for DE, which is originally catered for continuous optimization, is also suggested for the solving of permutation-based combinatorial optimization problems. This contribution is realized in chapter 5.

1.7 Organization of the Thesis

The potential of using the differential evolution for solving MOOPs gives the primary motivation for the research studies presented in this thesis. In order to achieve the aforementioned aims, an opposition-based self-adaptive differential evolution algorithm and algorithms that are extended from the former have been devised. The details of the proposed algorithms and their implementation to solve multiple MOOPs with different difficulties and problem nature are then presented.

The organization of the remaining chapters of this thesis is as follows. Chapter 2 starts by giving an overview of some state-of-the-art MOEAs under the different approaches for multi-objective optimization. This will be followed by a review of the origins of the differential evolution (DE) algorithm, and extensive overview of basics, development and applications of DE. Some other popular MOEAs that are either widely considered in the thesis or commonly used in evolutionary computation will also be introduced. The chapter also gives a description of the various performance metrics that are used to provide a quantitative analysis of the simulation results in the thesis. This will then be followed by a description of the different test problems that are used for comparison of performance between the proposed algorithm and the other state-of-the-art algorithms considered.

Chapter 3 presents the opposition-based self-adaptive differential evolution (OSADE) that involves the domination-based approach for multi-objective optimization. Opposition-based

learning is incorporated in a self-adaptive mechanism that allows the control parameters of differential evolution to be adapted according to the evolutionary process. The multi-objective evolutionary gradient search is then hybridized with the resultant differential evolution variant to act as a form of local search for OSADE. The performance of this novel algorithm is tested using a wide suite of benchmark test problems with a scalable number of decision variables and objective functions that pose different challenges for the algorithm. The results on the comparison of the algorithm with other state-of-the-art algorithms will be presented in detail.

Chapter 4 describes a novel grid-based differential evolution algorithm that is based on the OSADE mentioned in Chapter 3. For the implementation, opposition-based learning is being incorporated into a local mutation scheme, and this scheme is then joined with the original opposition-based self-adaptive mutation scheme from the OSADE using a linear decreasing probability rule. The resultant mutation scheme is then integrated into a grid-based framework to form a novel grid-based differential evolution variant called GrDE. The proposed algorithm will be compared with several state-of-the-art evolutionary algorithms that were previously tested on many-objective problems as seen in literature. For this study, the comparison results, sensitivity analysis and computational time analysis are being presented.

Chapter 5 studies the optimization performance of a decomposition-based OSADE (D-OSADE) in solving a multi-objective multiple salesmen traveling problem (MmTSP). Unlike the previous implementations where the test problems are in real-number representation, the permutation-based representation is studied here. For this study, the mutation scheme from OSADE is being incorporated into the decomposition-based approach for multi-objective optimization, and then hybridized with multi-point evolutionary gradient search to enhance the search behaviour. As DE is used for the generation of offspring containing parameters with float values, hence a heuristic rule will be applied for the conversion of the float values into permutation-based ones that are suitable for the evaluation of the MmTSP.

Finally, Chapter 6 presents the conclusions of this thesis and discusses future work.

Chapter 2

Literature Review

MOEAs represent a class of stochastic optimization algorithms for multi-objective optimization that emulate the process of natural evolution which consists of operations inspired by biological evolution processes including reproduction, mutation, recombination, natural selection and survival of the fittest. Over the years, many studies have been carried out to look into the algorithmic issues of MOEAs, and this chapter will first provide a short overview of evolutionary computation before looking into some of the state-of-the-art MOEAs under the different frameworks of multi-objective optimization. This will be followed by a comprehensive review of Differential Evolution which will cover the origin, basics, development and applications of DE. A review of some state-of-the-art MOEAs that are widely considered in this thesis will also be highlighted in detail. Besides these, a description of the performance metrics that are used for performance assessment of all the algorithms in this study will be provided. Finally, the different test problems that are used to test the efficiency of the algorithms in the study will be presented.

2.1 Evolutionary Computation

More than a few decades ago, a number of researchers proposed ideas of mimicking the underlying mechanisms found in biological evolution for the development of algorithms to handle optimization problems. The ideas eventually became known as Evolutionary Computation, which

can be regarded as a branch of computational intelligence that can be characterized by the type of algorithms that is involved. The algorithms are collectively referred to as evolutionary algorithms which are population-based meta-heuristic optimization algorithms. These algorithms are based on the concept of simulating the evolution of population individuals via processes of selection and reproduction. These processes are dependent on the fitness levels of the individuals as defined by an environment. In summary, evolutionary computation is based on an iterative progress such as the evolution of a population, and the selection of individuals in the population is done via a random search conducted under simultaneous processing and guided by processes based on biological mechanisms of evolution.

2.2 Multi-objective Evolutionary Algorithms

In this section, a review of three general classifications of the approaches used in multi-objective optimization is presented. Some state-of-the-art algorithms that were developed based on these frameworks will also be highlighted.

1. Preference-based Approach

In the preference-based approach [3, 42–45], the underlying concept is to perform aggregation of the multiple conflicting objectives in MOOPs using classical methods into a single-objective optimization problem, or to use preference knowledge of the problems so that the focus will just be on certain objectives to be optimized. A fundamental technique commonly used in this approach is the weighted sum method which basically involves the multiplication of every conflicting objective in the problem by a user specified weight value, and then combining them through summation into a single objective to be optimized.

The main issue that lies with the preference-based approach is that it is only able to obtain a single solution in one simulation run. Hence the goal of achieving a set of tradeoff solutions of good proximity and diversity for multi-objective optimization cannot be achieved with this approach with a single simulation run. Besides this, this approach also requires preference

knowledge on the MOOPs, in terms of appropriate weight values or predefined values from users, to be furnished to the optimizers. Due to such concerns, the emphasis placed on the research in this approach revolves on how to employ preference information effectively when performing optimization.

2. Domination-based Approach

The main working principles behind the domination-based approach lies in the assignment of a fitness value to every solution in the population for the optimization of all objectives in an MOOP. This concept was first initiated by Goldberg [1], but there were no concrete works from him to support this concept in terms of its suitability in handling MOOPs. Instead, Fonseca and Fleming [46] proposed the multi-objective genetic algorithm (MOGA), which is considered the first formal MOEA based on the domination-based approach for MOOPs. In this algorithm, the fitness of a solution is first determined according to the number of solutions that dominates it, and it will then be used to determine the rank of the solutions. Next the fitness of the solutions under the same rank will be shared using a fitness sharing mechanism. Through these two steps, MOGA is able to maintain a set of non-dominated solutions in a single simulation run.

Srinivas and Deb [47] proposed the non-dominated sorting genetic algorithm (NSGA) which is somewhat similar to the MOGA but uses a newly proposed ranking and sharing mechanism. In NSGA, the solutions are basically ranked according to the level of domination, and the solutions belong to a previous rank will be ignored during the ranking procedure at a particular rank level. At the same time, diversity of the solutions is being maintained via a fitness sharing mechanism, and a stochastic remainder selection scheme will be followed which gives a higher probability of selecting solutions from a lower front.

However, the element of elitism is absent in both MOGA and NSGA. Moreover, these algorithms also requires a sharing parameter to be specified. In addition, the NSGA also incurs higher computational complexity when compared to other MOEAs. To overcome these limitations, elitism mechanisms such as an external archive of solutions are being proposed and

included in MOEAs with the aim of maintaining a good distribution of optimal solutions. With the concept of elitism in mind, the strength Pareto evolutionary algorithm (SPEA) was proposed by Zitzler and Thiele [48], whereby an external archive or population is used to maintain a set of non-dominated solutions discovered during the evolutionary process. If the predefined size of the external population is reached, the crowded solutions as determined by a clustering algorithm will also be discarded. The SPEA2 was later proposed by Zitzler *et al.* [49] with improved versions of fitness assignment, archiving and diversity preservation mechanism with the aim of enhancing the SPEA.

On the other hand, Deb *et al.* came up with the NSGA-II [50] which is an improved version of the NSGA. In NSGA-II, it preserves a set of archived solutions at the start of the evolutionary process whereby N parent solutions and N child solutions are being stored in an archive. Non-dominated sorting will then be applied on the entire archive of $2N$ solutions to classify them into different domination ranks based on their domination relationship with each other. Due to the fact that only the best N solutions are to be selected as parents for the next generation, hence a parameter called the crowding distance will be calculated for every solution in the archive. The crowding distance is basically a proximity measure between a solution and its neighbours, and the less crowded solutions will be favoured and selected as parents. As contrasted with the NSGA, NSGA-II comes with lower computational complexity, incorporates elitism, and does not require the specification of a sharing parameter. With these good features, NSGA-II has established itself as one of the most reputable and widely used MOEA that is used to solve many real-world problems. In addition, NSGA-II also serves as a baseline algorithm for many MOEAs. However, NSGA-II displays poor performance in MOOPs with more than three objectives. This is due to the fact that many solutions are non-dominated to each other in a higher objective space. Hence the ranking and crowding mechanisms in NSGA-II will not be suitable in identifying the superior solutions efficiently for many-objective problems.

Over the past decade, the dominance-based approach has become of the most focused re-

search areas in multi-objective optimization. This is because this approach allows a set of tradeoff solutions to be generated in a single simulation run which is useful and appropriate for the case of multi-objective optimization. Besides this, diversity preservation of solutions is viable by considering the distribution of the solutions in a regulated population. However, the primary limitation of the dominance-based approach is that its selection pressure decreases when the number of objectives in a problem is being increased. As such, this approach is mainly suitable for handling MOOPs with two or three objective functions. Moreover, some of the diversity preservation schemes in the MOEAs that are based on this approach are also dependent on certain sharing parameter values which needs to be predefined *a priori* before the optimization process.

3. Decomposition-based Approach

The decomposition-based approach operates by decomposing an MOOP into multiple subproblems and these subproblems will then be optimized simultaneously. In most of the algorithms that are based on this approach, the use of Pareto dominance concept is absent or might only be applied to a certain extent.

An early adoption of the decomposition-based approach can be found in the multi-objective genetic local search algorithm (MOGLS) proposed by Ishibuchi *et al.* [51]. In MOGLS, aggregation of the multiple objectives of an MOOP is first performed using a weighted sum technique. A pair of parent solutions is then chosen during the selection phase to undergo crossover and mutation so as to generate an offspring. Optimization will be performed on the offspring using local search where the fitness function of the offspring is represented by an aggregated weighted-sum function. In every local search operation, a new vector with weight information will be randomly generated. Besides this, an external archive is also being used to store the non-dominated solutions. At the end of every local search operation, an offspring which is non-dominated when compared to any parent or archived solutions will be inserted into the archive. Any solutions in the archive that are dominated by the newly added solutions will also be discarded. For MOGLS, Pareto dominance is partially applied here for the comparison of the offspring with the parent

and archived solutions.

The MOEA based on decomposition (MOEA/D) proposed by Zhang and Li [52] is regarded as a state-of-the-art MOEA based on the decomposition-based approach for multi-objective optimization. In this algorithm, an MOOP is first decomposed into several subproblems whereby each subproblem is constructed by using a weighted sum or Tchebycheff method. Hence each subproblem is treated as a scalar objective optimization problem. A set of uniformly distributed weight vectors will be generated for the purpose of aggregation. The neighbourhood relationship between the subproblems will be decided with the help of the values of the Euclidean distances between the weight vectors. For the optimization of a subproblem, only its neighbouring solutions will be taken into consideration, and this will be ensured by performing genetic operations between a subproblem and its neighbouring solutions. Updating will be done to that subproblem and its neighbouring solutions with the best solution in terms of aggregated fitness value. It is also to be noted that there is no dedicated population pool for archiving and elitism in MOEA/D, and there is no specific elitism mechanism placed in the algorithm. However, elitism is being applied indirectly when updating is done to the nearest neighbours of a subproblem after the comparison of their fitness values to that of the newly generated solution for the subproblem. With this, MOEA/D provides a remedy for the issue of weakened selection pressure experienced by the domination-based MOEAs when dealing with many-objective problems. Besides this, there is also no requirement for a diversity preservation scheme to be specified for MOEA/D as there is already preserved diversity in the predefined weight vectors. With these advantages, MOEA/D gains its popularity in the research community for the handling of MOOPs.

Due to the good performance seen in MOEA/D, an improved version known as the MOEA/D-DE was proposed in [53]. The key difference between MOEA/D and MOEA/D-DE is that the differential evolution (DE) operator is being used instead of GA as the search heuristic in the latter. As the DE operator involves the generation of an offspring with three distinct parent solutions, a wider range of offspring can be produced. With the DE operator being used in

MOEA/D-DE, the exploration ability of the algorithm can be enhanced. This in turn leads to a more effective exploration of the search, which will be particularly useful when dealing with problems with complicated Pareto sets. Another difference between the original MOEA/D and the MOEA/D-DE lies in the maximal number of subproblems that are allowed to be replaced by an offspring. In the original MOEA/D, the maximal number allowed can be as much as the neighbourhood size (T). However, this may cause the diversity among the population individuals to be reduced significantly if an offspring is of higher quality than its parent and the neighbours of the parent as the offspring will replace all of them. In order to overcome this shortcoming in the original MOEA/D, the maximal number of subproblems that are allowed to be replaced by an offspring will be bounded to a parameter n_T where n_T is set to a much smaller value than T.

Ever since the successful implementation of both MOEA/D and MOEA/D-DE, more initiatives have been put in by the research community to either improve the original MOEA/D or to apply it in real-world problems. In a new version known as the MOEA/D with dynamic resource allocation (MOEA/D-DRA), different computational efforts are allocated to different subproblems. This algorithm was tested on CEC'09 unconstrained MOOPs, and has been named as the best performing MOEA in the CEC'09 competition for unconstrained multi-objective optimization. In another effort by Nebro and Durillo [54], a study on the parallelization of MOEA/D was conducted. As for real-world problems, MOEA/D has been employed in flowshop scheduling problems [55], and in applications related to the optimization of passive vehicle suspension [56], antenna arrays synthesis [57], and sensor network routing problems [58], just to name a few.

2.3 History of Differential Evolution

Differential evolution (DE) was introduced by Kenneth Price and Rainer Storn in 1996, and it traces its origin from attempts to solve the Chebyshev polynomial fitting problem with the use of genetic annealing [59]. Genetic annealing is a population-based, combinatorial optimization algorithm that is based on genetic algorithms and simulated annealing techniques. It basically

introduces the concept of evolution into the annealing process, and implements a thermodynamic annealing criterion via thresholds. However genetic annealing is unable to solve the Chebyshev problem well despite its successful application in the solving of several combinatorial tasks. As such, Price converted the bit-string encoding and logical operations in genetic annealing into floating-point representation and arithmetic operations respectively, and this resulted in the creation of the differential mutation operator. Storn also proposed the creation of separate parent and child populations in order to adapt to the parallel machine architectures. In addition, Price and Storn eliminated the annealing criterion as well. Through these, the Differential Evolution algorithm was formed, and Price and Storn were able to solve the Chebyshev problem effectively and efficiently with it.

2.4 Fundamentals of Differential Evolution

The aim of Differential Evolution is to evolve a population of *NP D*-dimensional parameter vectors representing the candidate solutions or individuals at G generation which are denoted by $\mathbf{X_{i,G}} = (x^1_{i,G}, ..., x^D_{i,G}), i = 1, 2, ..., NP$ towards the global optima. The initial population is generated by uniformly randomizing individuals within the search space which is bounded by a predefined set of minimum and maximum parameter bounds denoted by $\mathbf{X_{min}} = (x^1_{min}, ..., x^D_{min})$ and $\mathbf{X_{max}} = (x^1_{max}, ..., x^D_{max})$. The following equation shows how the initialization is performed for the j^{th} parameter of the i^{th} individual at generation G = 0.

$$x^j_{i,0} = x^j_{min} + rand_j(0,1) \times (x^j_{max} - x^j_{min}) \quad j = 1, 2, ..., D \tag{2.1}$$

where $rand_j(0,1)$ represents a uniformly distributed random real number between 0 and 1.

2.4.1 Mutation operation of Differential Evolution

After the initialization stage, mutation will be performed to produce a mutant vector $\mathbf{V_{i,G}}$ for every individual $\mathbf{X_{i,G}}$, which is also referred to as the target vector, in the current population. At

every generation G, a mutant vector $V_{i,G}$ can be generated for the target vector using a mutation strategy. Below are five different mutation strategies that are commonly used in implementations involving the DE algorithm.

$$DE/rand/1: \quad V^j_{i,G} = X^j_{r1,G} + F \cdot (X^j_{r2} - X^j_{r3}) \tag{2.2}$$

$$DE/best/1: \quad V^j_{i,G} = X^j_{best} + F \cdot (X^j_{r1} - X^j_{r2}) \tag{2.3}$$

$$DE/rand-to-best/1: \quad V^j_{i,G} = X^j_i + F \cdot (X^j_{best} - X^j_i) + F \cdot (X^j_{r1} - X^j_{r2}) \tag{2.4}$$

$$DE/best/2: \quad V^j_{i,G} = X^j_{best} + F \cdot (X^j_{r1} - X^j_{r2}) + F \cdot (X^j_{r3} - X^j_{r4}) \tag{2.5}$$

$$DE/rand/2: \quad V^j_{i,G} = X^j_{r1} + F \cdot (X^j_{r2} - X^j_{r3}) + F \cdot (X^j_{r3} - X^j_{r5}) \tag{2.6}$$

In the above mutation strategies, the indices $r_1, r_2, r_3, r_4, and r_5$ are mutually exclusive integers that are randomly generated in [1, NP] where NP is the number of individuals in the current population, and the indices must also be different from the index i, and $j = (1, 2, ..., D)$ where D is the number of decision variables. The generation of the indices will be done for every mutant vector. F represents a control parameter which is known as the scaling factor. It is a real and constant value which is usually between 0 and 2, and is used to control the amplification of the difference vectors in the mutation strategies. $\mathbf{X_{best,G}}$ is the individual with the best fitness value in the population at generation G.

2.4.2 Crossover operation of Differential Evolution

After the mutation stage, crossover will be applied between a target vector $\mathbf{X_{i,G}}$ and its corresponding mutant vector $\mathbf{V_{i,G}}$ so as to produce a trial vector $\mathbf{U}_{i,G} = (\mathbf{u^1_{i,G}}, \mathbf{u^2_{i,G}}, ..., \mathbf{u^D_{i,G}})$. The

main purpose of crossover operation is to increase the diversity of the perturbed parameter vectors. In the most fundamental version of the DE algorithm, the binomial crossover is used which is demonstrated as follows:

$$u_{i,G}^{j} = \begin{cases} v_{i,G}^{j} & \text{if } (rand_j[0,1] \leq CR) \; or \; (j = j_{rand}) \\ x_{i,G}^{j} & \text{otherwise} \end{cases} \tag{2.7}$$

where $j = \{1, 2, ..., D\}$ and D is the total number of decision variables. CR is the crossover rate which is a constant specified by the user and is within the range [0, 1]. This control parameter helps in determining the number of parameter values that are copied from the mutant vector to the trial vector. j_{rand} is an integer randomly chosen from the range [1, D] and is used to ensure that the trial vector will get at least one parameter from the mutant vector and hence the trial vector will definitely be different from the target vector. In the binomial crossover operation, the jth parameter of the mutant vector $\mathbf{V_{i,G}}$ will be copied to the corresponding element of the trial vector $\mathbf{U_{i,G}}$ if $rand_j[0,1] \leq CR$ or if $j = j_rand$ where $rand_j[0,1]$ is a random real number between 0 and 1. Otherwise the jth parameter will be copied from the corresponding target vector $\mathbf{X_{i,G}}$. Besides the binomial crossover, there is another form of crossover operator known as the exponential crossover. For this type of crossover, the parameters of the trial vector $\mathbf{U_{i,G}}$ are obtained from the corresponding mutant vector $\mathbf{V_{i,G}}$ from a randomly chosen parameter index until the first occurrence of $rand_j(0,1) > CR$. The remaining parameters of the trial vector $\mathbf{U_{i,G}}$ will be copied from the corresponding target vector $\mathbf{X_{i,G}}$.

2.4.3 Selection operation of Differential Evolution

The final stage of the Differential Evolution algorithm is the selection operation. All the parameter values of the trial vector $\mathbf{U_{i,G}}$ will be checked to ensure that they are within a specified set of lower and upper bounds. If any of the parameter values exceed the bounds, then it will be set to a randomly generated value within the pre-specified range. Next, the trial vectors will be evalu-

ated according to the optimization problem to obtain the objective function values, and they will then undergo the selection operation. For this, the objective function value of every trial vector $f(\mathbf{U_{i,G}})$ will be compared to that of its corresponding target vector $f(\mathbf{X_{i,G}})$ in the current population. In this comparison, minimization of the problem is assumed. If the objective function value of the trial vector is lesser or equal to that of its corresponding target vector, the trial vector will replace the target vector and enters the population for the next generation. Otherwise, the trial vector will be ignored and the target vector will be retained in the population for the next generation. The selection operation is demonstrated as follows:

$$X_{i,G+1} = \begin{cases} U_{i,G}, & \text{if } f(U_{i,G}) \leq f(X_{i,G}) \\ X_{i,G}, & \text{otherwise} \end{cases} \tag{2.8}$$

The above-mentioned steps of mutation, crossover and selection operations will be repeated until the predefined termination criterion is met.

2.5 Development of Differential Evolution

Due to the promising results achieved by Differential Evolution, researchers are motivated to exploit the potential of this algorithm. As such, the researchers are constantly working on the development of improved variants of the DE algorithm for the purpose of enhancing its optimization performance. In this section, an overview of the contemporary research efforts is being presented.

2.5.1 Control Parameters in Differential Evolution

There are mainly three control parameters in Differential Evolution whereby the setting of their values will have a strong influence on the optimization performance of the algorithm. As such, researchers have provided certain suggestions on ways to choose the values for these control

parameters. Firstly, Storn and Price [24] suggested that the population size NP for DE should be set between $5D$ and $10D$ where D is the dimensionality of the problem. For the scaling factor F, they suggested an initial value of 0.5, and a range of [0.4, 1] for F to be effective. As for the crossover rate CR, a first reasonable value can be 0.1. However if the problem is near unimodal or if fast convergence is desired, then CR can adopt a larger value like 0.9. In the event of premature convergence, the increase of either the mutation factor F or the population size NP could be helpful.

In [60], different parameter settings for DE were experimented on the Sphere, Rosenbrock and Rastrigin functions. From the experimental results, it was demonstrated that both the searching capability and convergence speed of DE are very sensitive to the control parameter settings. The authors suggested that the population size NP should be between $3D$ and $8D$, and F and CR to be 0.3 and 0.9 respectively.

In another study [61], it was suggested that F can be set within the range of [0.4, 0.95] with 0.9 as a good initial choice. As for the setting of CR, it would depend on the type of problem to be handled. If the problem is a separable one, CR could be set to a lower value between [0, 0.2], and if the problem is non-separable and multi-modal, then a larger value of 0.9 would be more appropriate for CR. However, the characteristics of real-life problems are usually unknown, and hence the authors concluded that it would be difficult for an appropriate value for CR to be chosen beforehand.

From these, it can be seen that there are different conclusions drawn on the rules for the manual setting of the control parameter values in DE. This may be attributed to the differences in the type of benchmark problems chosen for the studies, the type of trial vector generation strategies used, or even the combinations of the parameter values used. Due to the conflicting conclusions, researchers become uncertain on the appropriate guideline for the setting of parameters in DE, especially when applying the algorithm to real-life problems.

In order to address this issue, researchers have proposed different techniques to eliminate

the need for manual tuning of the DE parameters. In [62], Das *et al.* proposed two different methods to relieve the need for the setting of the scaling factor F. The first was a random variation method for F within the range of [0.5, 1], and the second was to linearly reduce F with increasing generation count from a maximum to a minimum value. Many researchers also turned to the adaptation of the control parameters F and CR. For example, the Fuzzy Adaptive Differential Evolution (FADE) [11] was introduced by Liu and Lampinen which uses fuzzy logic controllers to adapt the search parameters for the mutation and crossover operations. From the experimental results on a set of standard test functions, it is seen that the FADE algorithm outperformed the conventional DE algorithm on higher dimensional problems. In [63], Zaharie proposed a parameter adaptation method for DE based on the idea of controlling the population diversity. The concept was then extended by Zaharie and Petcu [64] who developed an adaptive Pareto DE algorithm for multi-objective optimization and further explored a parallel implementation for it. Another effort by Abbass [65] saw the self-adaptation of the crossover rate for DE for multi-objective optimization, whereby the crossover rate is being encoded into every individual so as to allow it to be evolved together with the decision variables of the individual. As for the scaling factor F, it is generated for each variable using a Gaussian distribution $N(0, 1)$.

Following these efforts, more works that focus on the adaptation of control parameters in DE were reported. In [66], Omran *et al.* proposed a self-adapted scaling factor F which is quite similar to the adaptation of crossover rate CR in [65]. This approach is known as the SDE whereby the generation of the crossover rate CR is done for every individual using a normal distribution $N(0.5, 0.15)$. It was seen that this approach fared better than other versions of DE when compared on a set of four benchmark problems. Adaptation of the population size in DE was also proposed in a study by Teo who developed the Differential Evolution with Self-Adapting Populations (DESAP) algorithm [14] based on the self-adaptive Pareto DE by Abbass mentioned above. Another innovative approach known as the jDE was introduced by Brest *et al.* [67] whereby both the control parameters F and CR are being encoded as extended variables

in every individual, and these parameters are being adjusted by two new parameters $\tau 1$ and $\tau 2$. In jDE, a set of F values was allocated to the individuals in the population. A random number $rand_1$ was then uniformly generated in the range of [0, 1]. If $rand_1 < \tau 1$, then F will be preset to a new random value, otherwise it will remain unchanged. For the case of CR, the adaptation was similar to how it was being performed for F except that another random uniformly generated number $rand_2$ was being compared to $\tau 2$ instead.

2.5.2 Strategies For Trial Vector Generation in Differential Evolution

An important aspect in Differential Evolution algorithm lies in the trial vector generation strategy it uses. Besides the strategies provided by Storn [24] for Differential Evolution, researchers have proposed several different strategies to improve the performance of the conventional DE. In the conventional DE, the base vector in the trial vector generation is just a randomly selected population vector without any consideration of the fitness of the vector. In order to guide the trial vector generation in a better manner, Price [68] suggested that a base vector can be formed by combining a target vector with a randomly selected vector which has a lower or equal objective function value to that of the target vector. In another study, Fan and Lampinen proposed the Trigonometric Mutation DE (TDE) algorithm [69] which involves a novel trigonometric mutation scheme that is used alongside with the mutation scheme from the original DE. The choice of the mutation scheme to be used is decided stochastically by an additional control paramter Mt. It is seen that this DE variant is able to outperform the conventional DE algorithm when compared on certain benchmark and real-world problems.

A detailed empirical study was conducted by Mezura-Montes *et al.* [27] to compare eight different DE trial vector generation strategies so as to identify the suitable strategies for different optimization problems according to their characteristics. The first five strategies are namely rand/1/bin, rand/1/exp, best/1/bin, best/1/exp, and current-to-rand/1/bin [70], and these strategies involve discrete recombination. The next two strategies involve arithmetic recombination and

they are current-to-rand/1 and current-to-best/1 [71]. The last strategy is the rand/2/dir [72] which incorporates objective function information. A total of 13 scalable test functions were chosen from [73], and they can be classified under four different categories which are namely uni-modal and separable, uni-modal and non-separable, multi-modal and separable, and multi-modal and non-separable. From the experimental results obtained, certain conclusions can be drawn from this study. Firstly, it was found that the use of binomial crossover always lead to better performance over exponential crossover. For both uni-modal and separable, and uni-modal and non-separable problems, best/1/bin strategy turned out to be the most competitive one. However for the case of multi-modal and separable problems, both rand/1/bin and rand/2/dir were as competitive as best/1/bin. Lastly, for the case of multi-modal and non-separable problems, both rand/1/bin and rand/2/dir performed much better than best/1/bin.

2.5.3 Hybridization involving Differential Evolution

Hybridization is considered to be another way of improving the performance of the DE algorithm. Zhang and Xie [74] introduced the DEPSO which is a hybrid between the PSO and the classical DE, and it is shown to outperform both PSO and DE when tested on a set of benchmark functions. In another research effort by Das *et al.* [75], a novel scheme was proposed to improve the performance of particle swarm optimization (PSO) by a vector differential operator borrowed from DE. Sun *et al.* [76] proposed the hybridization of DE with Estimation of Distribution Algorithm (EDA) to formulate the DE/EDA algorithm which utilizes the local information obtained by the DE mutation and the global information retrieved from a population of solutions by EDA modeling.

2.5.4 Multi-modal problems involving Differential Evolution

As seen in some real-world optimization problems, there is often a need to locate not just one optimum but rather an entire set of global or local optimal. Such problems are also known

as multi-modal optimization problems, and DE has also been extended to handle this class of problems as well. For this, DE has been modified to establish its capability in deciding the number of subpopulations required in a multi-modal problem, and to determine the minimal spanning distance between individuals of each subpopulation. In another approach knowen as the 'island model', it was adopted in the multi-resolution multi-population DE by Zaharie [77] and in the MultiDE by Hendershot [78] to handle multi-modal problems. DE was also extended with a crowding scheme to formulate the crowding DE that is able to track and maintain multiple optima as seen in another piece of work by Thomsen [79], and it is observed that this proposed algorithm outperformed the well-known fitness sharing scheme that penalizes similar candidate solutions.

2.5.5 Applications involving Differential Evolution

Since the inception of DE, it has been applied by scientists and engineers in a wide range of optimization problems that include engineering design for digital filters [80] and gas circuit breakers [81], neural networks and fuzzy systems [82, 83], scheduling tasks in terms of plant scheduling [84] and heterogenous multiprocessor scheduling [85], image processing [86, 87] and even in bioinformatics whereby DE is being used to find predictive genes subsets in microarray data [88]. These examples demonstrate the usefulness of DE in not just theoretical but also practical applications as well.

2.6 Frequently Considered MOEAs

In this section, the details of four state-of-the-art MOEAs are presented. These evolutionary algorithms are either frequently considered in this thesis or commonly used in the area of evolutionary computation.

2.6.1 Non-dominated Sorting Genetic Algorithm II (NSGA-II)

NSGA-II [50] is one of the most well-known domination-based algorithms for multi-objective optimization. As the fitness assignment operators of NSGA-II are being used by some of the proposed algorithms in this thesis, a detailed description of the operation in NSGA-II will be highlighted in this section. NSGA-II is also one of the algorithms that will be compared with the proposed algorithms in this thesis. The process flow of NSGA-II is presented in Figure 2.1.

Begin
1. **Initialization:** At generation $g = 0$, randomly generate N solutions as the initial population, $Pop(g)$
2. **Evaluation:** Evaluate all solutions in $Pop(g)$

Do While ("Stopping Criterion is not satisfied")
3. **Fitness assignment:** Perform Pareto ranking and crowding distance to the population $Pop(g)$

For i=0 to N
4. **Selection:** Select parent solutions to perform genetic operations using binary tournament selection
5. **Crossover:** Perform crossover to the selected parent solutions to generate an offspring
6. **Mutation:** Perform mutation to the offspring

End for
7. **Evaluation:** Evaluate all offspring and store the offspring in an archive A
8. **Archiving:** Combine parent and offspring solutions $Pop(g) \cup A$
9. **Elitism:** Perform Pareto ranking and crowding distance to $Pop(g) \cup A$. Select the best N solutions to form new population $Pop(g+1)$. $g = g + 1$

End Do
10. **Output:** Output the final set of solutions $Pop(g)$

End

Figure 2.1: Pseudo-code of NSGA-II

The algorithm begins by the random generation of N solutions for an initial population $Pop(g = 0)$. Evaluation of all the solutions in $Pop(g)$ is then performed to obtain their objective values. This is followed by the fitness assignment to all the solutions in the population according to Pareto ranking and crowding distance. Through the use of binary tournament selection, two chromosomes are then randomly selected into tournament, and the fitter one that has a lower rank or a greater crowding distance will be selected. The selection process will pick a pair of parent solutions, and crossover is then performed on the chosen parent solutions to generate an offspring. The offspring will then undergo mutation whereby random perturbation will be applied to it. The selection, crossover and mutation operations will be repeated N times in order for N

offspring to be generated. Evaluation of the offspring is then performed and they are then stored in a temporary archive A. The parent solutions and the offspring will then be merged to form a combined population $Pop(g) \cup A$ which is of $2N$ in size. The combined population is sorted again according to non-domination, and elitism is applied whereby only the best N solutions with lower Pareto ranks or greater crowding distances are selected to form a new population $Pop(g+1)$ for the next generation. The same procedure will be iterated over generations until the predefined stopping criterion is met.

The concept of the use of Pareto ranking and crowding distance operator in NSGA-II is as explained here. A solution X is considered to be fitter than another solution Y if and only if all the objective function values of solution X are lesser than those of solution Y (for the case of minimization problems). However, the solutions X and Y are incomparable if and only if the objective function values of solution X are not all smaller than those of solution Y. With the application of this concept, solutions in the population are first ranked accordingly to their rank of domination. Solutions which are not dominated by any other solutions will be given Rank 1 (lowest rank) while the solutions that are solely dominated by the solutions of the lowest rank will be given Rank 2, and so forth. As such, this would mean that the solutions with lower ranks are considered to be fitter than those with higher ranks. For illustration purposes, this ranking mechanism is displayed in Figure 2.2. With the use of natural selection, the solutions with lower ranks will have a better chance of being selected over those solutions with higher ranks. However, if there is a tie in the ranks between two solutions, then the one with a greater crowding distance is preferred. Figure 2.3 provides an illustration of how the crowding distance parameter is being measured for a solution X. The calculation is performed by summing over its two nearest neighbour, which means that the crowding distance is a total summation from $D1$ to $D4$. As for the solutions located at the boundary positions of the front (solutions Y and Z in Figure 2.3), their crowding distances will be set to infinity so as to increase the chance of these solutions surviving a tournament selection indefinitely. Through this procedure, the algorithm

will have a better ability to produce a set of diverse solutions.

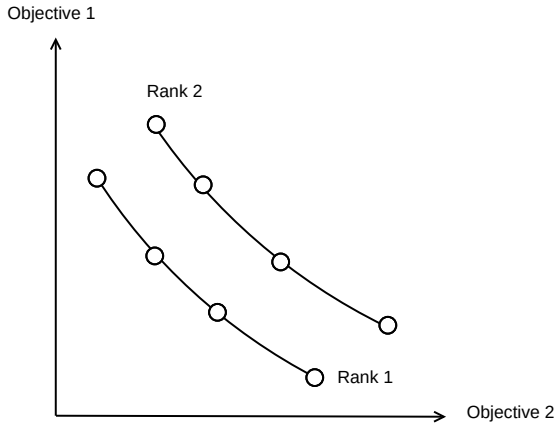

Figure 2.2: Pareto-based ranking

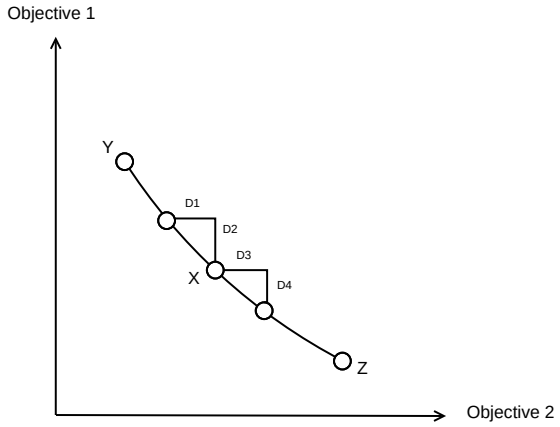

Figure 2.3: Crowding distance measurement

In NSGA-II, the genetic operations depend on whether the variable representation is in terms of binary number or real number. For the case of binary-number representation, single point crossover is used. The degree of crossover is controlled by a crossover probability (p_c), and it works by randomly dividing a chromosome into two parts, which is then followed by an exchange of the parts between two parent solutions. As for the mutation operation, bit-flip mutation is used which is basically performed by a random flipping of an allele according to a mutation probability (p_m).

If the variable representation is in real number, simulated-binary crossover (SBX) will be

implemented instead. As for the mutation operation, the polynomial mutation operator will be applied to the offspring. In the SBX operation, two parent solutions (x_1 and x_2) are selected from the population and a random real number u between [0, 1] is generated. The creation of the offspring is as follows:

$$\bar{x} = 0.5[(x_1 + x_2) - \bar{\beta}|x_2 - x_1|] \tag{2.9}$$

$$\bar{x} = 0.5[(x_1 + x_2) + \bar{\beta}|x_2 - x_1|] \tag{2.10}$$

$$\int_0^{\bar{\beta}} \rho(\beta) d\beta = u$$

$$\rho(\beta) = \begin{cases} 0.5(\varphi + 1)\beta^{\varphi} & \text{if } \beta \leq 1 \\ 0.5(\varphi + 1)\frac{1}{\beta^{\varphi+2}} & \text{otherwise} \end{cases}$$

where β is the spread factor which follows a polynomial probability distribution $\varphi(\beta)$ and φ is the distribution index which defines the probability for the creation of child solutions that are distant from or near to the parent solutions. As for the polynomial mutation operation, an offspring is produced in the following way:

$$\bar{x} = \begin{cases} x_1 + \sigma(x_L - x_U) & \text{if } u < p_m \\ x_1 & \text{otherwise} \end{cases} \tag{2.11}$$

$$\sigma = \begin{cases} (2u)^{1/(\varphi+1)} & \text{if } u < 0.5 \\ 1 - (2 - 2u)^{1/(\varphi+1)} & \text{otherwise} \end{cases} \tag{2.12}$$

where x_L and x_U are the lower and upper bounds for the decision variable x respectively. φ is the distribution index and and u is a random real number between [0, 1].

2.6.2 The Strength Pareto Evolutionary Algorithm 2 (SPEA2)

The SPEA2 [49] is an improved version of the Strength Pareto Evolutionary Algorithm (SPEA), and it is regarded as one of the most important multi-objective evolutionary algorithm that utilizes the elitism approach. The SPEA [48] has displayed good performance when compared with other multi-objective evolutionary algorithms, and this explains why it is used as a reference for comparison in several works or in different applications [89, 90]. In SPEA2, it differs from its predecessor in terms of having a fine-grained fitness assignment strategy, and the clustering technique utilised in the predecessor is replaced by another truncation method that prevents boundary solutions from being discarded during environmental selection. The process flow of SPEA2 is presented in Figure 2.4.

Begin
1. Initialization: At generation g = 0, randomly generate N solutions as the initial population, *Pop(g)*, and create an empty archive, *A*
Do While ("Stopping Criterion is not satisfied")
2. Archiving: Copy all the non-dominated individuals in *Pop(g)* to *A*
3. Fitness Assignment: Perform fitness assignment according to a fine-grained Fitness assignment strategy to the combined population made up of *Pop(g)* and *A*
4. Environmental Selection: Copy al the non-dominated individuals from the combined population to the archive for the next generation. Perform truncation to the new archive
For i=0 to N
5. Mating Selection: Select parent solutions to perform genetic operations using binary tournament selection
6. Crossover: Perform crossover to the selected parent solutions to generate offspring
7. Mutation: Perform mutation to the offspring
End for
g = g + 1
End Do
8. Output: Output the final set of solutions *Pop(g)*
End

Figure 2.4: Pseudo-code of SPEA2

Pertaining to the fitness assignment of SPEA2, a fine-grained fitness assignment strategy which factors in density information is used. Every individual i in the population $Pop(g)$ and the archive A is given a strength value $S(i)$, where $S(i)$ represents the number of solutions it dominates, and can be defined as follows:

$$S(i) = |\{j \mid j \in Pop(g) + A \wedge i \succ j\}| \tag{2.13}$$

where $|\bullet|$ denotes the cardinality of a set, $+$ represents the multiset union and the symbol $\succ$ refers to the Pareto dominance relation. With the S values, the raw fitness $R(i)$ of an individual i is computed as follows:

$$R(i) = \sum_{j \in Pop(g)+A, j \succ i} S(j) \tag{2.14}$$

This means that the raw fitness of an individual i in SPEA2 is calculated on the basis of the strength value of solutions in both the population and archive which dominate it. This is different from SPEA where only the archive members are considered for the computation of the raw fitness. It is to be highlighted that the raw fitness is to be minimized, which implies that a value of $R(i) = 0$ (for the case of a non-dominated individual) is favoured over a higher $R(i)$ value (for the case where the individual i is dominated by many individuals).

It is to be mentioned that this raw fitness assignment scheme may not be effective in situations when most of the individuals do not dominate each other. Therefore, the authors of SPEA2 included additional density information to differentiate individuals possessing similar raw fitness values. For this, the distances of every individual i to all the individuals j in the population and archive is measured in the objective space and stored in a list. Sorting will be performed to the elements in the list in terms of increasing order, and the kth element in the list will give a distance σ_i^k. For this, k is set to be the square root of the sample size [91], and thus $k = \sqrt{N + \bar{N}}$ where N and $\bar{N}$ are the population size and archive size respectively. The density estimate $D(i)$ for individual i will be taken as the inverse of distance to the kth nearest neighbour and this is represented by:

$$D(i) = \frac{1}{\sigma_i^k + 2} \tag{2.15}$$

The fitness of individual i represented by $F(i)$ will be given by $F(i) = R(i) + D(i)$. As for the archive updating process in SPEA2, it is different from SPEA in two ways. First of all, the

number of individuals in the archive is maintained at the same size. Secondly, the truncation method utilized in SPEA2 will prevent the boundary solutions from being eliminated. Hence for the environmental selection stage in SPEA2, all the non-dominated solutions from the archive and the current population will be firstly be updated to the archive A_{g+1} of the next generation as follows:

$$A_{g+1} = \{i \mid i \in Pop(g) + A \wedge F(i) < 1\} \tag{2.16}$$

The environmental selection is completed if the non-dominated front is of the same size as the archive. If this does not happen, there will be two situations whereby the archive is too small or too big. For the first case, the best $\bar{N} - |A_{g+1}|$ dominated individuals in the previous population and archive will be placed into the new archive for the next generation. This is performed by sorting the multiset $Pop(g) + A$ according to the fitness values, and following by the placement of the first $\bar{N} - |A_{g+1}|$ individuals i with $F(i) \geq 1$ from the resultant ordered list to the new archive A_{g+1}. However if the size of the current set of non-dominated individuals exceeds the size of the archive, truncation is performed which will iteratively discard individuals from A_{g+1} until the size of the archive is reached. For every iterative step in the truncation process, the individual which has the lowest distance to another individual is discarded. In the event if there are several individuals with the same lowest distance, the second smallest distance will then be considered and so forth.

2.6.3 MOEA with Decomposition (MOEA/D)

The MOEA/D algorithm [52] is widely used in this thesis as an algorithm for comparison, and its pseudo-code is as presented in Figure 2.5.

At the start of the algorithm, a set of uniformly distributed weight vectors $\boldsymbol{\lambda}^1$,...,$\boldsymbol{\lambda}^N$ is first generated where N is the predefined size of subproblems. Next, the Euclidean distance between all weight vectors is computed. For every weight vector i, Q closest neighbouring solutions

($B(i) = \{i_1, ..., i_Q\}, i \in [1, N]$) to it in terms of Euclidean distance will be identified. N solutions are then randomly generated to form the initial population $Pop(g = 0)$. Evaluation of all the solutions in $Pop(g)$ is then performed so as to obtain their objective values FV. The best objective values of the population will then be used as an initial reference point ($\mathbf{z}^*$) for the aggregation function that is used in the algorithm.

Begin

1. Initialization

a) Generate a set of uniformly distributed weight vectors ($\boldsymbol{\lambda}^i, ..., \boldsymbol{\lambda}^N$)

b) Calculate the Euclidean distance among the weight vectors. Determine the Q neighboring solutions ($B(i) = \{i_1, ..., i_Q\}, i \in [1, N]$) for each weight vectors according to the shortest Euclidean distance.

c) At generation $g = 0$, randomly generate N solutions to be the initial population, $Pop(g = 0)$. Evaluate the solution and set $FV^i = \boldsymbol{f}(\boldsymbol{x}^i), i \in 1, ..., N$

d) Initialize reference point of the Tchebycheff approach ($\boldsymbol{z}^*$) by setting the value of $\boldsymbol{z}^*$ to be the lowest objective values of the solutions

Do while ("Stopping Criterion is not satisfied")

For $\boldsymbol{i} = \boldsymbol{1}: \boldsymbol{N}$

2. Selection: Randomly select two solutions from $B(i)$

3. Crossover: Perform crossover to the parent solutions to generate an offspring

4. Mutation: Perform mutation to the generated offspring

5. Evaluation: Evaluate the generated offspring ($\boldsymbol{y}$) to obtain the corresponding objective values, $\boldsymbol{f}(\boldsymbol{y})$

6. Update of $\boldsymbol{z}^*$: For $j = 1, ..., m$, if $z_j^* > f_j(\boldsymbol{y})$, then set $z_j^* = f_j(\boldsymbol{y})$

7. Fitness assignment: Assign fitness (g^{te}) to each solution using Tchebycheff approach

8. Update Solution: For $j \in B(i)$, if $g^{te}(\boldsymbol{y}|\boldsymbol{\lambda}^j, \boldsymbol{z}^*) \leq g^{te}(\boldsymbol{x}^j|\boldsymbol{\lambda}^j, \boldsymbol{z}^*)$, then set $\boldsymbol{x}^j = \boldsymbol{y}$ and $FV^j = \boldsymbol{f}(\boldsymbol{y})$

End For

End Do

End

Figure 2.5: Pseudo-code of MOEA/D

The Tchebycheff approach is an aggregation technique that can be used in MOEA/D for the decomposition of an MOOP into N scalar optimization subproblems, and the aggregation function according to this approach will be presented here for the subproblem j as follows:

$$g^{te}(\mathbf{x}|\boldsymbol{\lambda}^j, \mathbf{z}^*) = \max_{1 \leq i \leq m} \{\lambda_i^j | f_i(\mathbf{x}) - z_i^*|\} \tag{2.17}$$

where m is the number of objective functions and f_i is the objective value for the i^{th} objective function. In the evolutionary process, the aim of the MOEA/D is to minimize all N subproblems

simultaneously in a single simulation run.

After the initiation stage, the evolutionary process begins. For each subproblem i, two neighbouring solutions will be randomly selected from $B(i)$. The SBX crossover operator is then applied on the selected parents to create an offspring. The offspring will then undergo polynomial mutation. Evaluation of the generated offspring $\mathbf{y}$ will be followed in order to obtain the corresponding objective value $\mathbf{f}(\mathbf{y})$. The reference point $\mathbf{z}^*$ will then be updated if $z_j^* > f_j(\mathbf{y})$. The fitness (g^{te}) of the aggregated scalar function will also be calculated for all subproblems and the offspring as well. Lastly, updating of solution will be done in the following manner. For every neighbouring solution j for a subproblem i where $j \in B(i)$, it will be replaced by the offspring $\mathbf{y}$ if the fitness of the offspring $\mathbf{y}$ is better than that of the neighbouring solution j. The above-mentioned procedures will be iterated until a stopping criterion is met.

2.6.4 Multi-objective Evolutionary Gradient Search (MO-EGS)

The multi-objective evolutionary gradient search (MO-EGS) is widely used as a local search technique in some of the proposed algorithms in this thesis. Evolutionary gradient search (EGS) [92] is a hybrid of both gradient search and evolutionary strategies which encompasses the merits of their features. It has been shown that the hybridization of an evolutionary algorithm with gradient search is able to improve the optimization performance of the search algorithm [93]. The key concept in EGS involves a robust evolutionary process that exploits gradient information of the trajectory of solutions and utilizes this information to arrive at movements in the search space that will lead to the generation of good solutions. The two main steps in EGS are 1) estimation of gradient, and 2) updating of solution with a steepest descent method. Iteration of these two main steps is performed until a predefined stopping criterion is attained. For the MO-EGS [94], EGS was extended to cater towards MO optimization by using an external archive to store all the non-dominated solutions. A recurrent truncation process will be applied to eliminate the most crowded archive members based on a niche count if the archive size reaches a predefined value.

In this way, elitism is achievable and MO-EGS will be able to find a set of uniformly distributed and diverse solutions closer to the true Pareto front.

Begin

1. Input: Define the initial step size σ_0. $t = 1$

Do while ("Stopping criterion is not achieved")

For $i = 1$ **to** X (Number of parent solutions)

2. Initial Solution: Select a solution x_i from the selection pool

3. Reproduction: Randomly generate N local neighbours s_j by perturbing x_i using normal mutation $N(0, \sigma^2)$

4. Evaluation: Compute the objective function values F(x_i) of x_i

5. Archiving: Update the non-dominated solutions in an archive

6. Direction: Estimate the global gradient vector as follows:

$$\vec{g} = \frac{\sum_{i=1}^{N}[F(s_j) - F(x_i)](s_j - x_i)}{\left\|\sum_{i=1}^{N}[F(s_j) - F(x_i)](s_j - x_i)\right\|}$$

7. Offspring generation: Produce an offspring q as follows:

$$q = x_i - \sigma_t \vec{g}$$

8. Update parameter: Update the mutation step size σ_{t+1} as follows:

$$\sigma_{t+1} = \begin{cases} \sigma_t \varepsilon & if\ F(q) < F(x_i) \\ \sigma_t / \varepsilon & otherwise \end{cases}$$

where $\varepsilon = 1.8$

9. Update solution: Update the parent solution x_i as follows:

$$x_i = q\ if\ F(q) < F(x_i)$$

10. Output: Output x_i

End for i

End Do

End

Figure 2.6: Pseudo-code of MO-EGS

The procedural steps of the MO-EGS are presented in Figure 2.6. The algorithm starts by defining the initial step size σ_0. A step size is required in MO-EGS to control the mutation strength to be applied for the generation of the local neighbors and the offspring. Upon the selection of an initial parent solution, N local neighbours will be randomly generated by perturbing this solution using normal mutation with zero mean and σ^2 variance. Next the global gradient direction is to be estimated from the local neighbours according to the formula in step 6. An offspring q will then be created in step 7, and this is followed by updating the mutation step size σ with the control of a factor ϵ which is recommended to be 1.8 according to [94]. The solution will then be updated in step 9 and the process is iterated until the stopping criterion is achieved.

2.7 Performance Metrics

In the context of multi-objective optimization, the goal is to find a set of tradeoff solutions that are able to meet the common goals in multi-objective optimization which are namely proximity, diversity and distribution of solutions. In order to conduct quantitative measurement of the generated solution set, it is important to identify a set of suitable performance metrics for this. There has been an increasing concerns on the choice of performance metrics, and comparative studies have been done by many researchers [18, 95–99] where a suite of unary performance metrics pertinent to the goals of proximity, diversity and distribution has been used. From here, four performance metrics or indicators that are widely used for the quantitative comparison between algorithms on the different goals of multi-objective optimization are being selected and applied in this thesis. The performance metrics are as follows:

1. **Generational Distance (GD):** This indicator is a representative metric which provides a quantitative measurement for the proximity goal of multi-objective optimization.

2. **Inverted Generational Distance (IGD):** This indicator is a representative metric which provides a quantitative measurement for the proximity, diversity and distribution goals of multi-objective optimization.

3. **Hausdorff Distance (HD):** This indicator is a representative metric which provides a quantitative measurement for the proximity, diversity and distribution goals of multi-objective optimization.

4. **Hypervolume (HV):** This indicator is a representative metric which provides a quantitative measurement of the for the proximity, diversity and distribution goals of multi-objective optimization.

The mathematical description of the performance metrics is provided in Appendix A.

2.8 Test Problems

In order to examine the optimization performance of an MOEA, it is important to employ the use of benchmark test problems which are able to test the ability and efficiency and identify any potential pitfalls of an MOEA. In order to achieve this, the test problems should possess different characteristics and difficulties. From literature, there is a large range of test problems designed for multi-objective optimization [100–103], and a total of 93 state-of-the-art test problems will be used in Chapter 3 and Chapter 4 of this thesis. These test problems include Zitzler-Deb-Thiele's (ZDT) Test Problems [104], Deb-Thiele-Laumanns-Zitzler's (DTLZ) Test Problems [105], CEC'09 (UF) Test Problems [106], and Walking Fish Group (WFG) Test Problems [107]. There is a total of 5 ZDT problems and 10 UF problems. For the case of DTLZ and WFG test problems, they are scalable in terms of the number of objective functions. A total of 42 DTLZ test instances and 36 WFG test instances with varying number of objective functions within these two test suites will be considered in this thesis. The details of these test problems in terms of their mathematical formulation, characteristics, and difficulty levels can be found in Appendix B.

2.9 Summary

In this chapter, a detailed literature review on both MOEAs and differential evolution has been conducted. First, a discussion on the different state-of-the-art MOEAs under the three main approaches of multi-objective optimization (preference-based approach, domination-based approach, and decomposition-based approach) was covered. This was followed by a comprehensive review of the background details of Differential Evolution. This chapter also provided a detailed description of four different state-of-the-art MOEAs that are widely considered in this thesis. Lastly, the performance metrics and benchmark test problems that are used in the thesis were also presented.

Chapter 3

A Novel Opposition-based Self-adaptive Differential Evolution (OSADE)

Under the framework of evolutionary paradigms, many evolutionary algorithms have been designed for handling multi-objective optimization problems. Each of the different algorithms may display exceptionally good performance in certain optimization problems, but none of them can be completely superior over one another. As such, different evolutionary algorithms are being synthesized to complement each other in view of their strengths and the limitations inherent in them. In this chapter, a novel memetic algorithm known as the Opposition-based Self-adaptive Differential Evolution algorithm (OSADE) is proposed by incorporating opposition-based learning into a self-adaptive mechanism for the control parameters of differential evolution, and then hybridizing it with the multi-objective evolutionary gradient search (MO-EGS) as a form of local search. Experimental studies are conducted to compare the optimization performance of OSADE with some state-of-the-art evolutionary algorithms.

3.1 Chapter Objectives

The main objective of this chapter is to present the development of a novel memetic algorithm termed as OSADE, and to compare the optimization performance of OSADE under a comprehensive suite of unconstrained continuous multi-objective optimization test problems with several state-of-the-art evolutionary algorithms. Through this chapter, the potential of synthesizing an opposition-based self-adaptive differential evolution with local search for multi-objective optimization is being investigated.

3.2 Introduction

Multi-objective Evolutionary Algorithms (MOEAs) [3] are defined as a broad class of population-based stochastic optimization techniques that draws inspiration from biological evolution to solve multi-objective optimization problems (MOOPs) [4,108–110]. With the reputation of being powerful global optimization tools, MOEAs achieve remarkable results when they are applied to several practical optimization problems that involve multiple non-commensurable with competing criteria relating to the design specifications and constraints. However, MOEAs in general have weaknesses like loss of diversity, slow convergence, stagnation of population etc. For the case of multi-modal problems that possess several local and global optima, such issues associated with MOEAs will become even more prominent. Also, MOEAs may take considerable amount of time to locate the local optimal. With these overall concerns associated with MOEAs, several research efforts were initiated to develop optimization techniques to address these concerns here. In the development of these optimization techniques which comprise gradient techniques and stochastic algorithms, researchers are generally concerned with convergence rate and solution accuracy.

For general multi-modal problems, gradient techniques have the tendency to be trapped in local minima rather than reaching a global solution. This is not desirable as the ability to find

the global minimum is usually considered of higher importance compared to convergence speed. On the other hand, stochastic algorithms like evolutionary algorithms, simulated annealing (SA) or even particle swarm optimization (PSO) have better effectiveness in finding the global minimum as compared to gradient techniques. However, stochastic algorithms may have slower convergence rate which may be of a hindrance when it comes to practical applications. This means that convergence speed and robustness of optimization algorithms may not be achievable simultaneously.

Differential evolution (DE) [71] is an efficient evolutionary algorithm known for its simplicity and ease of use, and it is also recognized as one of the most powerful stochastic real-parameter optimization algorithms in current use. In the operation of DE, it follows similar computational steps as seen in most traditional EAs. However, the distinguishing difference between this state-of-the-art algorithm and the other EAs is that DE involves the perturbation of the current generation individuals with the scaled differences of other randomly selected and distinct population members. Therefore, there is no separate probability distribution for generating new offsprings. As compared to most standard evolutionary algorithms, DE requires fewer parameters, and has a faster convergence rate in most cases together with stronger global convergence ability and robustness. Due to the multiple criteria nature of most real-world problems, DE has also been extended to handle multi-objective optimization [111–113]. Despite the strengths witnessed in DE, it is inevitable that there are still certain drawbacks in this evolutionary algorithm when placed under certain problems, applications or environment. For example, DE may suffer from stagnation whereby the algorithm does not progress in finding new solutions. It may also experience premature convergence especially in multi-modal problems, as there is possibility of DE being trapped in local optima [114]. Like other evolutionary algorithms, the performance of DE may also deteriorate in higher dimensional search space in terms of increasing decision variables or with many objectives [115]. In addition, DE is known to possess strong exploration capability which helps to locate the region of global optimum, but it is slow when it comes to

the exploitation of the solutions [116]. As such, these leads to continuous efforts by researchers in enhancing DE to overcome the weaknesses described above, and this is witnessed by several important works [69, 117] on the enhancement of DE with the aim of formulating variants with improved performance in terms of robustness and faster convergence, as well as the ability in maintaining diversity.

In order to achieve fast convergence and effective global search capability concurrently, researchers have turned to the use of memetic algorithms, which can be viewed as a combination of two or more methods like evolutionary algorithms or with other specialized methods like local search. From literature review, a wide range of works on memetic algorithms [118–121] that focus on improving convergence can be found. These algorithms were demonstrated to have their strengths in the problems that they were tested on. Even for practical engineering problems, they have also proven their efficiency and reliability in various types of applications [122–124] like aerodynamic design, inverse problems and power systems. There were also attempts of hybridizing DE with other evolutionary algorithms like Particle Swarm Optimization and Evolutionary Programming [9, 125, 126] so as to improve the convergence speed and robustness of it.

The performance of DE is also highly attributed to the setting of its control parameters which are namely the mutation factor, the crossover rate and the population size. In many cases, parameter tuning via trial-and-error is involved for the setting of the control parameters which calls for several tedious experimental runs in order to determine the optimized parameter values for the DE algorithm so as to ensure its success. However, this is time-consuming and hence requiring more computational cost. Moreover, a fixed parameter setting may not be appropriate for every type of problem or even for the different evolutionary stages of the optimization process for solving a problem. As such, researchers introduced the use of different adaptive or self-adaptive mechanisms in several DE variants [11–14] to dynamically update the control parameters without requiring any preceding knowledge of the relationship between the settings of the control parameters and the properties of the optimization problems given. Such an approach can improve the

convergence rate for a given optimization problem if the control parameters are adapted to appropriate values at the different stages of the evolutionary process. From the reported results, these adaptive and self-adaptive DE variants are able to demonstrate improvement in the convergence performance over the classic DE in terms of speed and reliability when compared over several test benchmark problems.

This chapter attempts to address the issues of DE as discussed above by exploring a memetic algorithm for multi-objective optimization. This chapter also presents a more in-depth study of the algorithm by using it to handle a comprehensive set of 38 test benchmark MOOPs, and studying its performance in terms of scalable problems. The algorithm termed as OSADE is a hybridization of a novel Opposition-based Self-adaptive Differential Evolution variant with the Multi-objective Evolutionary Gradient Search (MO-EGS), and the fundamental idea of this hybridization is based on the assumption that combining the two algorithms may complement the limitations of each optimizer while maintaining their strengths. The use of the novel DE variant not only contributes to stronger global search, but also eliminates the need to manually tune the DE parameters to their optimized values. On the other hand, the use of MO-EGS as a form of local search is able to enhance the overall exploitation ability of OSADE.

The remainder of this chapter is organized as follows. Section 3.3 presents the existing work that implement the hybridization of DE with other evolutionary algorithms, as well as some of the self-adaptive DE algorithms that are found in literature. Section 3.4.3 presents a detailed description of the proposed algorithm OSADE. The test problems and the other algorithms that are used for comparison as well as the implementation are outlined in Section 3.5. Section 3.7 presents the experimental results and discussions, and a summary is provided in Section 3.8.

3.3 Related Works

This section reviews some past works that involve the hybridization of DE with other evolutionary algorithms, as well as some of the self-adaptive DE algorithms that are found in literature.

Hybridization can be defined as the integration of the best features of two or more algorithms in order to arrive at a new algorithm that could possibly perform better than the original parent algorithms when compared over certain benchmark test problems or even application-specific ones. The DEPSO [74] is a hybrid of high popularity which basically switches between the particle swarm optimization algorithm (PSO) and DE at odd and even iterations respectively, and it achieved better convergence over its ancestor algorithms when tested on certain constrained optimization problems. Sun *et al.* proposed a new hybrid algorithm DE/EDA [76] that combines DE with the Estimation of Distribution Algorithm (EDA) whereby the technique utilizes a probability model for the determination of promising search regions so that emphasis can be placed on these areas for the search process. Based on the experimental results, this DE/EDA algorithm is seen to perform better than both DE and the EDA. In another approach by Das *et al.* [127], the selection strategy of the basic DE was modified by adopting the concepts of simulated annealing (SA) so that the probability of accepting the inferior solutions can be varied during evolution.

Works involving the integration of local search methods with DE can also be readily found. Local search algorithms mainly involve the exploration of the vicinity of a candidate solution in the search space until a locally optimal point is located or when a certain time period has passed. In order to improve the performance of classical DE, Noman and Iba [116] combined an adaptive local search with the simplex-based crossover scheme (SPX) originally proposed by Tsutsui *et al.* [128] for real-coded genetic algorithms, and incorporated the resultant crossover-based local search with DE to give forth the DEahcSPX which demonstrated better performance over classical DE in terms of convergence speed over a set of benchmark problems.

In self-adaptive DE algorithms, the main control parameters are dynamically adapted to the characteristic of the fitness landscapes of different optimization problems during the evolutionary search process. As such, the trial and error method of tuning the parameters is not required. A novel approach known as the SaDE [129] encompasses the self-adaptation of trial vector generation strategies and their associated control parameters according to the past experi-

ence of the generation of promising solutions. For the control parameters, the mutation factors F are randomly generated at every generation according to a normal distribution. In this way, both small and large values of F can be generated which assists in local and global search abilities respectively during the entire evolutionary process. For the crossover probabilities CR, they are generated at every five generations according to a normal distribution. Brest *et al.* [67] proposed the jDE that is based on the classic DE/rand/1/bin. In this approach, the individuals are encoded with the values of control parameters F and CR which exist in the individuals as extended variables. The values of F and CR are initialized at the start, and they are updated at every generation according to a uniform distribution. This approach is done in the belief that better parameter values of F and CR may lead to better individuals that will likely survive and in turn produces better offspring, and hence propagating better parameter values of F and CR to the next generation.

There are also some works on self-adaptive DE variants catered for multi-objective optimization. MOSaDE [130] was extended from SaDE mentioned earlier to solve problems in the multi-objective domain. This algorithm was evaluated on a set of 19 benchmark problems and satisfactory performance was obtained for most of the problems. The MOSaDE was later enhanced with objective-wise learning strategies (called OW-MOSaDE) to tackle problems with multiple conflicting objectives [131]. In MOSaDE, one set of parameters is learnt for all objectives, but in OW-MOSaDE, one set of parameters is learnt for each objective. This results in better performance of the OW-MOSaDE over the original MOSaDE. In another piece of work by Zamuda *et al.* [132], a self-adaptive DE algorithm (DEMOwSA) based on DEMO [28] was proposed whereby the adaptation of the F and CR parameters are derived from evolutionary strategies. The algorithm was tested on 19 test functions and managed to achieve good results on the test suite.

3.4 Opposition-based Self-Adaptive Differential Evolution (OSADE)

OSADE incorporates opposition-based learning into a self-adaptive mechanism for the DE control parameters (mutation factor and the crossover probability), and is then hybridized with the MO-EGS which acts as a form of local search to enhance the exploitation abilities of the overall algorithm. The algorithm follows the mutation and crossover strategies from the classical DE commonly known as the DE/rand/bin/1. As OSADE is designed to solve multi-objective optimization problems, there is no single best solution but rather a set of Pareto-optimal solutions. As such, the non-dominated sorting, ranking and elitism techniques as found in NSGA-II are incorporated into OSADE for the comparison of the quality of the solutions found so as to obtain the Pareto optimal solutions.

3.4.1 Background

In order to improve convergence and eliminate the need to determine optimized control parameter values for differential evolution, the approach in the self-adaptive mechanism in DEMOwSA [132] is being extended for this study. This approach traces its principles to a self-adaptive mechanism found in evolutionary strategies whereby the mutation factor F and the crossover rate CR of differential evolution are being encoded in every individual of the population. This means that there will be two additional variables in every individual, and they will also undergo the evolutionary process for their values to be adjusted appropriately in a self-adaptive manner for every generation.

Inspiration was drawn from another approach [35] to enhance the above-mentioned self-adaptation method so as to achieve the aim of finding near-optimal values for the DE control parameters during the evolutionary process. This approach mentioned in [35] uses the concept of Opposite Number which is based on Opposition-Based Learning [30]. As seen from several evolutionary optimization methods, the use of random guesses or estimates are frequently used when we are looking for a solution to a problem. At the same time, the evolutionary optimization

methods will also work towards the search for the optimal solution and this incurs computational time which is related to the distance between the estimated solutions and the optimal one. However, the guess might be far from the exact solution as it may be based on past experience or could be totally random. Hence, in order to accelerate convergence towards the optimal solution, the opposite number of the estimated solution will be also checked as there will be 50% chance that the estimated solution will be further from the optimal one compared to the opposite solution according to probability theory.

The concept of the Opposite Number in Opposition-Based Learning is employed in this study and its definition is as follows:

$$\bar{x} = y + z - x \tag{3.1}$$

where $x \in [y, z]$ is a real number, and $\bar{x}$ represents the Opposite Number. In higher dimensionality, the definition of Opposite Number can be extended to the Opposite Point [30]. Let there be a Point $P = (x_1, x_2, ...x_D)$ in the D-dimensional space where $x_1, x_2, ...x_D \in R$ where R represents all real numbers, and $x_i \in [y_i, z_i] \forall (1, 2, ..., D)$. The Opposite Point $\bar{P} =$ will then be defined as follows:

$$\bar{x_i} = y_i + z_i - x_i \tag{3.2}$$

If Opposition-Based Learning is applied to multi-objective optimization, the point P can represent a candidate solution in the D-dimensional space, and with $\bar{P}$ as its Opposite Point. Both the points will be evaluated concurrently to determine the fitter one. If $F(\bar{P}) \geq F(P)$ where $F(\bullet)$ represents a fitness function that measures the fitness of candidate solutions, then the Opposite Point $\bar{P}$ can replace P; otherwise P will still be the chosen candidate solution.

3.4.2 Novel Opposition-based Self-adaptive Differential Evolution operator

This section provides a description of the proposed novel opposition-based self-adaptive differential evolution operator that is used in OSADE. Firstly at the parent generation G, for every target vector x_i^G, three other vectors x_{r1}^G, x_{r2}^G and x_{r3}^G are randomly selected whereby $r1 \neq r2 \neq r3 \neq i$, and $r1, r2, r3 \in \{1, 2, ..., NP\}$ where NP is the size of the population. In OSADE, every individual is encoded with values of the DE control parameters F and CR that exist as extended variables. These encoded values, which are initialized as zero at the start of every run, are needed in the opposition-based self-adaptive mechanism for the control parameters.

The key difference between the self-adaptive mechanism in DEMOwSA and the one proposed here for OSADE is that opposition-based learning is being applied in the updating of the mutation factor. It is to be highlighted that the adaptation of DE parameters in OSADE utilizes weighted averaging of the encoded parameter values from four different individuals instead of simple averaging as used in DEMOwSA. The current individual and three randomly selected individuals are compared based on their Pareto ranks and/or niche counts to determine the best individual which will be awarded the highest weightage of 0.4 for its contribution to the averaging of the parameter values. Individuals with better Pareto ranks are preferred as they are nearer to the optimal Pareto front, and the selection of individuals with lower niche counts promotes diversity [133]. Hence this approach allows fitter individuals to give a higher contribution towards the adaptation of the parameter values so as to achieve near-optimal DE control parameter values. The weight factors assigned to the subsequent ranked individuals will then be decremented by 0.1. The computation of the current average values of F and CR for the parent generation G which are denoted by $\langle F_G \rangle_i$ and $\langle CR_G \rangle_i$ are as follows:

$$\langle F_G \rangle_i = \frac{\omega_1 \times F_{i,G} + \omega_2 \times F_{r1,G} + \omega_3 \times F_{r2,G} + \omega_4 \times F_{r3,G}}{\omega_1 + \omega_2 + \omega_3 + \omega_4} \tag{3.3}$$

$$\langle CR_G \rangle_i = \frac{\omega_1 \times CR_{i,G} + \omega_2 \times CR_{r1,G} + \omega_3 \times CR_{r2,G} + \omega_4 \times CR_{r3,G}}{\omega_1 + \omega_2 + \omega_3 + \omega_4} \tag{3.4}$$

where ω_1, ω_2, ω_3 and ω_4 are the weight factors assigned to the four different individuals x_i^G, x_{r1}^G, x_{r2}^G, and x_{r3}^G .

With the current average values of F and CR for the parent generation, the values of the F and CR for the child generation $G+1$ which are denoted by $\bar{F}_{i,G+1}$ and $\bar{CR}_{i,G+1}$ are then adapted according to the following formulae as follows:

$$\bar{F}_{i,G+1} = \langle F_G \rangle_i \times e^{\tau N(0,1)} \tag{3.5}$$

$$\bar{CR}_{i,G+1} = \langle CR_G \rangle_i \times e^{\tau N(0,1)} \tag{3.6}$$

where $\tau = \frac{1}{8 \times \sqrt{2D}}$, D is the number of decision variables in the problem, and $N(0,1)$ represents a randomly generated number under Gaussian distribution. Both $\bar{F}_{i,G+1}$ and $\bar{CR}_{i,G+1}$ are adapted between the a predefined set of lower and upper bounds for the control parameters. Next, the Opposite Number [35] of the mutation factor F is generated as follows:

$$\bar{F\_opp}_{i,G+1} = F_{upper} + F_{lower} - \bar{F}_{i,G+1} \tag{3.7}$$

whereby F_{upper} and F_{lower} are the upper and lower bounds for the parameter F. Two different mutant vectors are then generated using the value of F and its Opposite Number, and they are compared according to Pareto dominance whereby the non-dominated one will be the offspring. If both the mutant vectors are non-dominated to each other, one of them will be randomly selected to enter the child population.

Lastly, the encoded values of both F and CR in the individual $x_{i,G+1}$ are updated. The updating of F will depend on the outcome of the selection of the mutant vector. If the selected mutant vector is the one created using $\bar{F}_{i,G+1}$ as generated by equation 3.5, then the encoded value of F in $x_{i,G+1}$ will be updated with this F value; otherwise the encoded value will be updated with the Opposite Number of $\bar{F}_{i,G+1}$. However for the case of encoded value of CR in $x_{i,G+1}$, it will always be updated with the value of $\bar{CR}_{i,G+1}$ as generated by equation 3.6.

3.4.3 Algorithmic Framework

The algorithm starts by the random generation of an initial candidate population which is then evaluated and fitness assignment is conducted for all the solutions. OSADE is designed to solve multi-objective optimization problems where there is no single best solution but rather a set of Pareto-optimal solutions. As such, the non-dominated sorting, ranking and elitism techniques as found in NSGA-II are incorporated into OSADE for the comparison of the quality of the solutions found so as to obtain the Pareto optimal solutions. Binary tournament selection is then performed to choose promising solutions for reproduction. In the reproduction stage, the novel opposition-based self-adaptive differential evolution operator is used for the generation of offspring. Through reproduction, N child solutions are produced and archived. M solutions are then selected from archive to undergo local search (MO-EGS), and the solutions generated will be added to the archive. The updated archive will become the child population of the generation, and will be combined with the parent population. Elitism is performed to select the parent population for the next generation. The process is iterated until the stopping criterion is met. The pseudo code of OSADE is as explained in Figure 3.1.

Begin
1. Initialization: At generation g = 0, randomly generate N solutions as the initial population, *Pop (g)*
2. Evaluation: Evaluate all the solutions in *Pop (g)*
Do while ("Stopping criterion is not met")
3. Fitness Assignment: Assign fitness to every individual in *Pop(g)*) using Pareto ranking and crowding distance.
4. Selection: Select *N* parent solutions using binary tournament selection.
5. Reproduction: Use the novel opposition-based self-adaptive Differential Evolution (DE) operator to create *N* offspring
6. Evaluation: Evaluate all offspring and store all offspring in a fixed size temporary archive *A*
7. Selection: Select *M* individuals via binary tournament selection from *A* for the local search
8. Local Search: Perform the MO-EGS on the *M* individuals to generate offspring. Add the offspring to *A* if they are not dominated by any of the archived members. If the predetermined archive size of *A* is reached, eliminate the most crowded archive member.
9. **Archiving**: Combine the parents and offspring to form a merged population *Pop (g)* $\cup$ *A*
10. Elitism: Perform Pareto ranking and crowding distance calculation for *Pop1 (g)* $\cup$ *A*. Select the best *N* solutions to form a new population *Pop (g+1)*. g = g + 1
End Do
11. Ouput: Output the final set of solutions *Pop (g)*
End

Figure 3.1: Pseudo-code of OSADE

3.5 Problem Description and Implementation

A total of 38 benchmark test problems from Section 2.8 were chosen to test the optimization performance of the proposed algorithm OSADE in terms of convergence to the true Pareto front as well as the ability in maintaining a set of diverse solutions. The test problems used included ZDT problems, DTLZ problems, UF problems, and WFG problems. For the test problems, they may possess two, three or five objective functions, and have a scalable number of decision variables. These problems were chosen because they cover different characteristics of multi-objective optimization, namely convex Pareto front, non-convex Pareto front, discrete Pareto front, multi-modality and non-uniformity of solution distribution. The presence of these characteristics will be able to pose challenges to an evolutionary multi-objective optimization algorithm.

Five state-of-the-art algorithms, namely NSGA-II, MOEA/D-SBX, NSDE, MOEA/D-DE and the MO-EGS were chosen for performance comparison with OSADE. NSGA-II [50] is a popular algorithm in evolutionary multi-objective optimization as it has the ability to achieve promising solutions for most of the MOOPs. This algorithm uses Pareto ranking and crowding distance as fitness assignment operators, binary tournament selection, uniform crossover, bit-

flip mutation, and parent-offspring archiving. The non-dominated sorting differential evolution (NSDE) [134, 135] is an extension of the basic differential evolution to cater towards multi-objective optimization. It adopts the non-dominated sorting, ranking and elitism techniques found in NSGA-II, but the main difference between them is that the NSDE uses the differential evolution mutation operator instead of the SBX operator.

For the MOEA/D-SBX [52] and MOEA/D-DE [53], they are evolutionary algorithms that decompose any given MOP into a number of single-objective sub-problems. Every sub-problem is optimized simultaneously during the evolutionary search process. For the decomposition of the MOOP, the Tchebycheff approach [42] is used in these two algorithms for this study. The difference between MOEA/D-SBX and MOEA/D-DE lies in their genetic operators whereby the SBX crossover operator is employed together with polynomial mutation for MOEA/D-SBX while MOEA/D-DE uses the DE/rand/1 crossover with polynomial mutation.

Lastly for the MO-EGS [94], it is extended from the Evolutionary Gradient Search (EGS) [92], and it basically combines the strong features from gradient search and evolutionary strategies. The details of its implementation can be found in Section 2.6.4.

In order to draw a fair comparison of different optimization algorithms, performance metrics that are relevant and applicable to the goals of MO optimization of proximity and distribution should be utilized. In our study, Inverted Generational Distance (IGD) [136] and Hausdorff Distance [137] were used as the performance metrics for the assessment of the optimization performance achieved by the algorithms. IGD is a unary indicator whereby the distance of every solution in the optimal Pareto front to the obtained Pareto front is being calculated. HD computes the distance between the optimal Pareto front and the generated solution set, and takes the maximum value between the modified generational distance and inverted generational distance. With this, both proximity and diversity issues will be taken into consideration. For these two metrics, a lower value indicates better performance.

A comparative study of OSADE, NSGA-II, MOEA/D-SBX, NSDE, MOEA/D-DE and

MO-EGS was carried out to examine their optimization performance in the test problems described earlier. All algorithms were implemented in C++ and ran on an Intel Core i3, 2.8 GHz personal computer. For OSADE, its control parameters F and CR will be self-adapted in the evolutionary process, and therefore their values were not fixed. However, the upper and lower bounds for the F and CR control parameters as well as their initial values would need to be defined. For the other algorithms under comparison, their parameter settings used in this study followed the ones used in their original studies. The overall experimental and parameter settings are as summarized in Table 3.1.

Table 3.1: Parameter settings

Parameter	Settings
Population size (for all algorithms except MOEA/D-SBX and MOEA/D-DE	100 for problems with 2 objectives 300 for problems with 3 objectives 500 for problems with 5 objectives 1500 for problems with 7 objectives
Population size (MOEA/D-SBX and MOEA/D-DE	100 for problems with 2 objectives 300 for problems with 3 objectives 495 for problems with 5 objectives 1716 for problems with 7 objectives
Stopping criterion	50000 evaluations for problems with 2 objectives 150000 evaluations for problems with 3 objectives 250000 evaluations for problems with 5 objectives 750000 evaluations for problems with 7 objectives
Number of decision variables for ZDT problems	300 for ZDT1, ZDT2 and ZDT3, and 100 for ZDT4 and ZDT6
Number of decision variables for DTLZ problems	12 for DTLZ1 and DTLZ3 120 for all the other DTLZ problems
Number of decision variables for UF problems	30
Number of decision variables for WFG problems	30
Number of independent runs	30
Mutation rate	1/n (where n denotes the number of decision variables
Crossover rate for NSGA-II	0.8
Crossover rate for NSDE	0.8
Mutation scale factor for NSDE	0.5
Distribution index in SBX	20
Distribution index in polynomial mutation	20
Neighbourhood size for MOEA/D algorithms	20
Number of local neighbours for MO-EGS	10
Initial encoded F and CR values in OSADE	0
Lower Bound of F and CR in OSADE	0.1
Upper Bound of F and CR in OSADE	0.9

3.6 Investigation on Proposed Self-Adaptive Mechanism

As this chapter involves the proposal of an opposition-based self-adaptive mechanism for the DE algorithm, an initial study was conducted to investigate the effect of how the proposed self-adaptive mechanism is able to lead to an improvement in the optimization performance of differ-

ential evolution. For this investigation, the proposed opposition-based self-adaptive mechanism was incorporated into a non-dominated sorting differential evolution algorithm and was then compared with a conventional non-dominated sorting differential evolution (termed as normal DE) using three selected test problems. The three problems used were namely UF5, WFG4, and DTLZ2 with 3 objectives, and the number of decision variables used in the problems was set according to the settings found in Table 3.1. In order to perform both convergence and diversity analysis on the performance between the two algorithms, the mean GD and IGD evolution plots of the algorithms for the three test problems are being plotted in Figure 3.2-3.4.

For the UF5 problem, it is observed that the DE with the proposed self-adaptive mechanism has a slightly faster initial convergence rate when compared to the normal DE, and the advantage of the use of the self-adaptive mechanism is seen at about 10,000 fitness evaluations where the DE with the self-adaptive mechanism achieves a much faster convergence rate and better convergence performance over normal DE with consistently lower GD values attained till the end of the evolutionary process. The IGD values obtained by the DE with the self-adaptive mechanism are also lower than the values obtained by normal DE which suggests better diversity of the solution set as well. For the WFG4 problem, the DE with the proposed self-adaptive mechanism achieves significantly better convergence rate over the normal DE as seen from the GD evolution plot. Moreover, there is also clear advantage on the use of the mechanism as seen by the lower GD and IGD values obtained by the DE with the self-adaptive mechanism over normal DE throughout the entire evolutionary process. Lastly for the DTLZ2 problem, there is obvious superiority by the DE with the self-adaptive mechanism over the normal DE as seen by a much faster convergence rate and much lower GD and IGD values obtained at the end of the evolution. In summary, it can be deduced that the use of the proposed opposition-based self-adaptive mechanism in DE is able to lead to faster convergence rate as well as stronger ability in achieving better convergence and diversity in the solution sets found by DE incorporated with the mechanism. This suggests that performance improvement can be achieved if the proposed mechanism is being used in DE

algorithm when handling the benchmark test problems in this study.

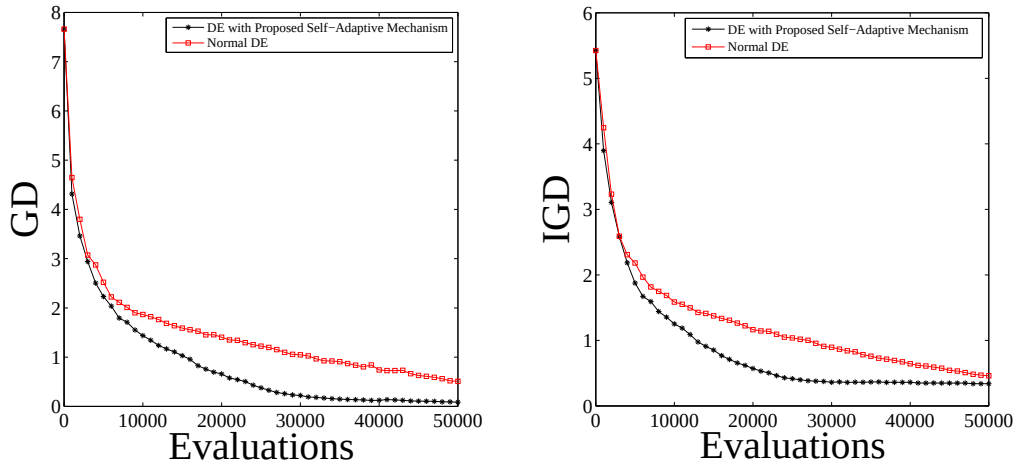

Figure 3.2: GD/IGD evolution plots by DE with and without proposed self-adaptive mechanism for UF5

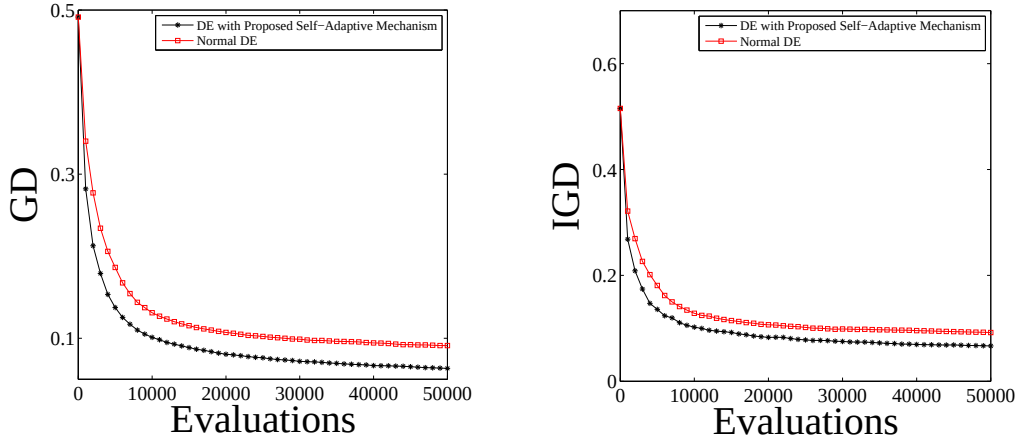

Figure 3.3: GD/IGD evolution plots by DE with and without proposed self-adaptive mechanism for WFG4

3.7 Simulation Results and Discussions

Comparative studies were conducted for the performance evaluation of the six algorithms under a comprehensive suite of benchmark test functions. Simulation results in terms of the measurement of the average values of the Inverted generational distance (IGD) and the Hausdorff distance (HD) over 30 simulation runs are presented in Tables 3.2-3.6. The parentheses beside the test problems indicate the number of objectives (M) and decision variables (D) for the problems.

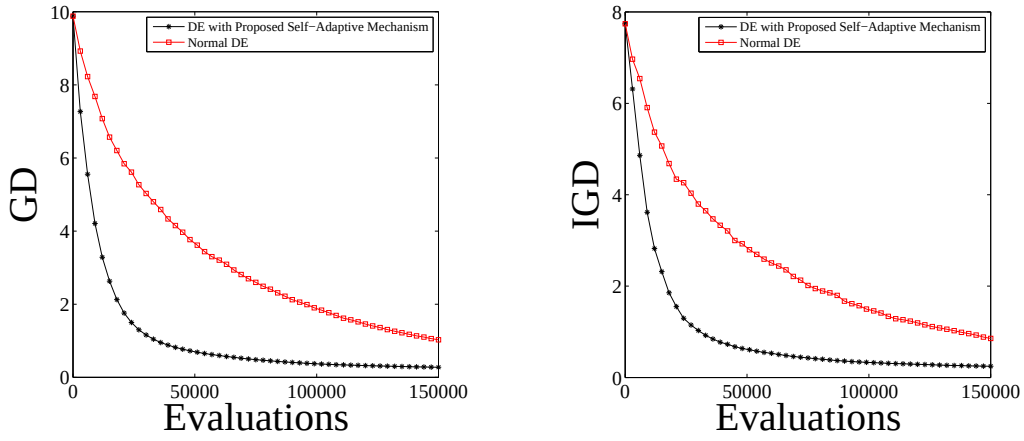

Figure 3.4: GD/IGD evolution plots by DE with and without proposed self-adaptive mechanism for DTLZ2

The best entries in terms of mean values are also marked in boldface. The Wilcoxon ranksum test was also conducted at the 5% significance level to determine whether the best performing algorithm differ from the results of competitors in a statistically significant way. The entries which are significantly different from the best entries will also be indicated by the symbol †.

3.7.1 Comparative studies for ZDT problems

ZDT test problems [104] are a set of simple bi-objective optimization problems that are scalable in the number of decision variables, and have different characteristics in the Pareto optimal front such as convexity, concavity, discontinuity, local optimality and non-uniformity. As most evolutionary algorithms are able to solve the ZDT problems without difficulties, the number of decision variables was set to ten times its original settings in this study. This would then pose greater challenges to the algorithms due to a larger search space. The results indicate that OSADE has the overall best performance. A notable achievement by OSADE is the ability for it to reach convergence for ZDT4 while all the other algorithms are unable to do so for this problem. ZDT4 problem is an extremely multi-modal problem with the presence of many local optima, and therefore it is likely that the other algorithms encountered difficulties by getting trapped in the local optima. The overall good performance achieved by OSADE may be attributed to the

Table 3.2: Results obtained by the algorithms for ZDT problems

Problem(M,D)	Algorithm	IGD	HD (with p=2)
ZDT1(2,300)	NSGA-II-SBX	0.1977±0.0295†	0.0206±0.0032†
	MOEA/D-SBX	0.2994±0.0318†	0.0302±0.0032†
	NSDE	1.2669±0.1233†	0.2978±0.0601†
	MOEA/D-DE	1.0660±0.0353†	0.1075±0.0037†
	MO-EGS	0.0967±0.0906†	0.3010±0.3644†
	OSADE	**0.0041±0.0001**	**0.0008±0.0001**
ZDT2(2,300)	NSGA-II-SBX	0.3763±0.0562†	0.0386±0.0058†
	MOEA/D-SBX	0.4685±0.1384†	0.0555±0.0187†
	NSDE	2.5621±0.2239†	0.3165±0.0628†
	MOEA/D-DE	1.8994±0.1689†	0.1938±0.0175†
	MO-EGS	0.1067±0.1108†	0.3810±0.6440†
	OSADE	**0.0044±0.0001**	**0.0006±0.0001**
ZDT3(2,300)	NSGA-II-SBX	0.1566±0.0204†	0.0194±0.0035†
	MOEA/D-SBX	0.3095±0.0395†	0.0314±0.0036†
	NSDE	0.8378±0.1102†	0.2150±0.0284†
	MOEA/D-DE	0.9092±0.0439†	0.1059±0.0050†
	MO-EGS	0.1141±0.1015†	0.1939±0.3093†
	OSADE	**0.0047±0.0004**	**0.0014±0.0001**
ZDT4(2,100)	NSGA-II-SBX	25.317±2.0645†	2.7240±0.2120†
	MOEA/D-SBX	29.050±3.6618†	2.9541±0.3694†
	NSDE	27.684±6.2070†	4.0101±0.8528†
	MOEA/D-DE	35.385±8.4533†	3.5936±0.8524†
	MO-EGS	0.6790±0.4208†	0.3555±0.3335†
	OSADE	**0.0053±0.0002**	**0.0009±0.0001**
ZDT6(2,100)	NSGA-II-SBX	0.9193±0.0558†	0.0935±0.0056†
	MOEA/D-SBX	0.6167±0.0391†	0.0631±0.0039†
	NSDE	5.5319±0.2267†	0.5563±0.0228†
	MOEA/D-DE	3.7095±0.2538†	0.3733±0.0255†
	MO-EGS	**0.0024±0.0001**	0.0083±0.0185†
	OSADE	0.0026±0.0003	**0.0025±0.0002**

complementary effects of the opposition-based self-adaptive DE and the MO-EGS as a form of local search.

3.7.2 Comparative studies for DTLZ problems

The suite of DTLZ problems created by Deb *et al.* [138] is scalable to any number of objectives and decision variables. Hence for this study, the DTLZ problems consisted of either three or five objective functions. The number of decision variables in DTLZ1 and DTLZ3 was set to 12 as they are highly multi-modal and hence more difficult problems. For the other DTLZ problems, they are generally easier to solve and hence the number of decision variables was set to 120 instead. For the case of DTLZ problems with three objective functions, OSADE achieves competitive performance when compared to the other algorithms in this study. From the simulation results, OSADE either achieves the lowest IGD and HD values, or comes close to the best values, for most of the DTLZ problems. However for the case of DTLZ5 and DTLZ6, OSADE did not fare

Table 3.3: Results obtained by the algorithms for DTLZ problems (3-objectives)

Problem(M,D)	Algorithm	IGD	HD (with p=2)
DTLZ1(3,12)	NSGA-II-SBX	**0.0006±0.0001**	0.0019±0.0011†
	MOEA/D-SBX	0.0008±0.0001†	**0.0013±0.0001**
	NSDE	0.0444±0.0667†	4.5010±0.7898†
	MOEA/D-DE	0.0069±0.0018†	0.1569±0.4036†
	MO-EGS	19.238±9.5019†	2.2134±1.2701†
	OSADE	**0.0006±0.0001**	**0.0013±0.0001**
DTLZ2(3,120)	NSGA-II-SBX	0.0019±0.0001†	0.0036±0.0001†
	MOEA/D-SBX	**0.0015±0.0001**	**0.0029±0.0001**
	NSDE	0.0719±0.0100†	0.4964±0.0639†
	MOEA/D-DE	0.0025±0.0001†	0.0117±0.0105†
	MO-EGS	3.4349±0.7449†	0.2398±0.0473†
	OSADE	**0.0015±0.0001**	0.0453±0.0570†
DTLZ3(3,12)	NSGA-II-SBX	0.0017±0.0001†	0.0038±0.0007†
	MOEA/D-SBX	**0.0015±0.0001**	**0.0029±0.0001**
	NSDE	0.1478±0.1141†	12.172±4.0629†
	MOEA/D-DE	0.0023±0.0023	0.0260±0.0345†
	MO-EGS	41.071±52.124†	3.3025±3.4263†
	OSADE	**0.0015±0.0001**	0.0038±0.0012†
DTLZ4(3,120)	NSGA-II-SBX	**0.0022±0.0001**	**0.0043±0.0002**
	MOEA/D-SBX	0.0081±0.0068†	0.0132±0.0104†
	NSDE	0.4348±0.0365†	1.0623±0.0241†
	MOEA/D-DE	0.4417±0.0270†	0.7655±0.0427†
	MO-EGS	27.214±0.2098†	1.6295±0.0125†
	OSADE	0.0024±0.0003	0.0321±0.0041†
DTLZ5(3,120)	NSGA-II-SBX	**0.0009±0.0001**	0.0015±0.0001
	MOEA/D-SBX	**0.0009±0.0001**	**0.0014±0.0001**
	NSDE	0.0239±0.0056†	0.1343±0.0452†
	MOEA/D-DE	0.0010±0.0001	0.0022±0.0001†
	MO-EGS	3.9459±0.9802†	0.6006±0.0266†
	OSADE	0.0598±0.0021†	1.2112±0.0121†
DTLZ6(3,120)	NSGA-II-SBX	1.6099±0.0352†	2.8740±0.0571†
	MOEA/D-SBX	0.7098±0.0349†	1.2720±0.0654†
	NSDE	1.9219±0.1212†	4.0637±0.1495†
	MOEA/D-DE	0.1062±0.0562†	**0.2465±0.1000**
	MO-EGS	0.3675±0.0516†	4.3460±0.2491†
	OSADE	**0.0049±0.0009**	5.0802±0.0190†
DTLZ7(3,120)	NSGA-II-SBX	**0.0037±0.0001**	0.0081±0.0001†
	MOEA/D-SBX	0.0065±0.0001†	0.0079±0.0001
	NSDE	0.0911±0.0161†	0.6551±0.1314†
	MOEA/D-DE	0.0151±0.0020†	0.1441±0.0345†
	MO-EGS	0.1341±0.0218†	**0.0074±0.0007**
	OSADE	0.0041±0.0001†	0.0076±0.0002

so well when compared to the other algorithms. For DTLZ5, it is observed that the algorithms that incorporate the use of SBX operator yield better results when compared to the algorithms with the DE operator. This suggests that the use of DE operator may not be that powerful in tackling problems with degenerate Pareto front.

In DTLZ problems with five objective functions, the results demonstrate that OSADE achieves the overall best performance for DTLZ1, DTLZ3 and DTLZ7 when pitted against all the algorithms under comparison. For DTLZ2, OSADE achieves the lowest IGD value but not for the HD metric. For DTLZ4, DTLZ5 and DTLZ6, it is observed that the decomposition-based algorithms generally displayed better performance in terms of better IGD and HD values than the rest

Table 3.4: Results obtained by the algorithms for DTLZ problems (5-objectives)

Problem(M,D)	Algorithm	IGD	HD (with p=2)
DTLZ1(5,12)	NSGA-II-SBX	1.6105±0.9697†	8.6632±2.7045†
	MOEA/D-SBX	0.0196±0.0003†	0.0196±0.0003†
	NSDE	4.4756±1.5371†	12.266±0.9618†
	MOEA/D-DE	18.770±11.729†	22.209±11.740†
	MO-EGS	12.706±1.6501†	1.4095±0.2941†
	OSADE	**0.0028±0.0001**	**0.0028±0.0001**
DTLZ2(5,120)	NSGA-II-SBX	0.0692±0.0044†	0.0704±0.0058†
	MOEA/D-SBX	0.0538±0.0023†	**0.0538±0.0023**
	NSDE	0.2714±0.2210†	1.0752±0.0594†
	MOEA/D-DE	0.0556±0.0004†	0.0556±0.0004†
	MO-EGS	5.3918±0.4946†	0.4062±0.0483†
	OSADE	**0.0125±0.0003**	0.1181±0.0310†
DTLZ3(5,12)	NSGA-II-SBX	5.4124±1.3374†	21.852±1.5717†
	MOEA/D-SBX	0.0493±0.0023†	**0.0493±0.0023**
	NSDE	15.625±2.2059†	32.773±0.7238†
	MOEA/D-DE	69.095±26.876†	71.656±26.245†
	MO-EGS	43.464±32.848†	5.5862±2.1677†
	OSADE	**0.0122±0.0005**	0.2206±0.2750†
DTLZ4(5,120)	NSGA-II-SBX	0.3921±0.0623†	0.3921±0.0623†
	MOEA/D-SBX	**0.0567±0.0049**	**0.0567±0.0049**
	NSDE	1.4805±0.0437†	1.4805±0.0437†
	MOEA/D-DE	0.6201±0.0498†	0.6201±0.0498†
	MO-EGS	26.983±0.2091†	1.9432±0.0143†
	OSADE	1.3938±0.0160†	1.3938±0.0160†
DTLZ5(5,120)	NSGA-II-SBX	0.2509±0.0312†	0.4495±0.0135†
	MOEA/D-SBX	0.0193±0.0004	0.0193±0.0004
	NSDE	0.0482±0.0393†	0.9584±0.0169†
	MOEA/D-DE	**0.0190±0.0001**	**0.0190±0.0001**
	MO-EGS	4.3806±0.8635†	0.6802±0.2133†
	OSADE	0.1590±0.0062†	1.0276±0.0221†
DTLZ6(5,120)	NSGA-II-SBX	6.0927±0.0425†	6.0927±0.0425†
	MOEA/D-SBX	0.9352±0.0267†	0.9352±0.0267†
	NSDE	5.06490±0.447†	5.1273±0.3514†
	MOEA/D-DE	0.6051±0.1533†	**0.6079±0.1488**
	MO-EGS	0.3873±0.0965†	4.1574±0.0704†
	OSADE	**0.0532±0.0025**	4.2545±0.1387†
DTLZ7(5,120)	NSGA-II-SBX	0.0284±0.0014†	0.0456±0.0104†
	MOEA/D-SBX	0.1227±0.0001†	0.1227±0.0001†
	NSDE	0.1298±0.0405†	0.9822±0.0173†
	MOEA/D-DE	0.2897±0.0296†	0.2897±0.0296†
	MO-EGS	0.3540±0.0999†	0.0849±0.1359†
	OSADE	**0.0199±0.0001**	**0.0199±0.0001**

for these three problems. This demonstrates the better ability of the decomposition-based algorithms in solving many-objective problems over domination-based algorithms. This is attributed to the fact that decomposition-based algorithms allows better selection of promising solutions by the use of aggregated fitness values. For the other domination-based algorithms in this study, the domination behaviour between solutions is required to be determined before deciding on which are the superior solutions. However, as the number of objective functions increases, the domination behaviour will be weakened. Hence it will be harder for domination-based algorithms to select the better solutions in the higher objective space.

It is observed that OSADE generally displays better performance for DTLZ1, DTLZ3 and

DTLZ7, and competitive performance for DTLZ2, with both three and five objective functions. DTLZ1 and DTLZ3 are highly multi-modal problems, and the success of OSADE in handling these problems well is likely attributed to the strong exploratory capabilities inherent in its DE operator which allows the algorithm to escape from local optimal. As for DTLZ2, the local search phase in OSADE helps in producing adequate selection pressure towards the large spherical Pareto front in the high-dimensional objective domain. The good performance shown by OSADE for DTLZ7 could also be attributed by the strong exploratory nature of its DE operator complemented by the local search component as this helps the algorithm in discovering the distributed sub-populations in all the disconnected Pareto-optimal regions. In addition, the archival method used in OSADE is also effective in maintaining the solutions found in the disconnected Pareto-optimal regions. As such, these factors may explain why OSADE is able to handle the DTLZ7 problem well.

3.7.3 Comparative studies for UF problems

UF problems are a set of difficult Multi-objective optimization problems (MOOPs) with complicated shapes of Pareto-sets (PS) proposed for the CEC 2009 competition [106]. In these problems, the number of decision variables is scalable but it shall be retained as 30 for all UF problems. For this test suite, UF1 to UF7 are test instances with two objective functions while UF8 to UF10 consist of three objective functions. In general, UF test instances possess arbitrary prescribed Pareto set (PS) shapes, which could be useful for the study of how multi-objective evolutionary algorithms deals with complicated PS shapes with varying nature of the Pareto front (PF). From the IGD and HD results, we are able to witness OSADE achieving the overall best performance for this suite of test problems.

Figures 3.5-3.7 display the Pareto fronts of UF1 generated from the different algorithms. The solutions plotted are the non-dominated solutions obtained from all 30 simulation runs. From the comparison of the different evolved Pareto fronts, it is clear that OSADE is able to maintain

Table 3.5: Results obtained by the algorithms for UF problems

Problem(M,D)	Algorithm	IGD	HD (with p=2)
UF1(2,30)	NSGA-II-SBX	0.1200±0.0245†	0.0461±0.0085†
	MOEA/D-SBX	0.1257±0.0497†	0.0511±0.0214†
	NSDE	0.0523±0.0128†	0.0244±0.0186†
	MOEA/D-DE	0.0488±0.0289	0.0224±0.0154†
	MO-EGS	0.1588±0.0956†	0.0599±0.0295†
	OSADE	**0.0404±0.0051**	**0.0169±0.0025**
UF2(2,30)	NSGA-II-SBX	0.0478±0.0102†	0.0214±0.0062†
	MOEA/D-SBX	0.0574±0.0296†	0.0303±0.0161†
	NSDE	0.0450±0.0056†	0.0179±0.0026†
	MOEA/D-DE	0.0342±0.0236	0.0170±0.0128†
	MO-EGS	0.0510±0.0045†	0.0197±0.0018†
	OSADE	**0.0204±0.0012**	**0.0082±0.0005**
UF3(2,30)	NSGA-II-SBX	0.2370±0.0403†	0.0897±0.0137†
	MOEA/D-SBX	0.3092±0.0532†	0.1162±0.0198†
	NSDE	0.1387±0.0092†	0.0471±0.0092
	MOEA/D-DE	**0.0744±0.0138**	**0.0300±0.0138**
	MO-EGS	0.1748±0.0099†	0.1107±0.0188†
	OSADE	0.2087±0.0101†	0.0698±0.0035†
UF4(2,30)	NSGA-II-SBX	0.0538±0.0022†	0.0173±0.0009†
	MOEA/D-SBX	0.0566±0.0047†	0.0187±0.0019†
	NSDE	0.0730±0.0078†	0.0235±0.0024†
	MOEA/D-DE	0.0824±0.0078†	0.0264±0.0025†
	MO-EGS	0.1495±0.0076†	0.0477±0.0025†
	OSADE	**0.0408±0.0012**	**0.0130±0.0001**
UF5(2,30)	NSGA-II-SBX	0.3047±0.1036†	0.0261±0.0111†
	MOEA/D-SBX	0.4359±0.0993†	0.0288±0.0093†
	NSDE	0.8773±0.1742†	0.1310±0.0350†
	MOEA/D-DE	0.6670±0.1384†	0.0649±0.0228†
	MO-EGS	1.3705±0.2850†	0.2439±0.0424†
	OSADE	**0.1494±0.0119**	**0.0132±0.0001**
UF6(2,30)	NSGA-II-SBX	0.1462±0.0465†	0.0559±0.0181†
	MOEA/D-SBX	0.1744±0.0548†	0.0681±0.0197†
	NSDE	0.0442±0.0088†	0.0170±0.0043†
	MOEA/D-DE	0.0465±0.0291	0.0232±0.0136†
	MO-EGS	0.0698±0.0091†	0.0339±0.0041†
	OSADE	**0.0240±0.0043**	**0.0121±0.0002**
UF7(2,30)	NSGA-II-SBX	0.1683±0.1340†	0.0712±0.0528†
	MOEA/D-SBX	0.3224±0.1380†	0.1326±0.0514†
	NSDE	0.0329±0.0090†	0.0183±0.0088†
	MOEA/D-DE	0.0234±0.0063	0.0123±0.0043†
	MO-EGS	0.0723±0.0040†	0.0278±0.0015†
	OSADE	**0.0178±0.0020**	**0.0083±0.0012**
UF8(3,30)	NSGA-II-SBX	0.2203±0.0047†	0.1828±0.0076†
	MOEA/D-SBX	0.1607±0.0379†	0.0793±0.0216†
	NSDE	0.1478±0.0143†	0.0607±0.0198†
	MOEA/D-DE	**0.0937±0.0082**	0.0967±0.0539†
	MO-EGS	0.1168±0.0173	0.0517±0.0079†
	OSADE	0.1098±0.0028	**0.0464±0.0011**
UF9(3,30)	NSGA-II-SBX	0.1710±0.0423†	0.0756±0.0172†
	MOEA/D-SBX	0.1220±0.0566†	0.0548±0.0290†
	NSDE	0.1822±0.0671†	0.0706±0.0332†
	MOEA/D-DE	0.1058±0.0485†	0.1585±0.0534†
	MO-EGS	0.1995±0.0622†	0.0434±0.0115†
	OSADE	**0.0805±0.0022**	**0.0296±0.0010**
UF10(3,30)	NSGA-II-SBX	0.3274±0.0596	0.2872±0.2685†
	MOEA/D-SBX	0.3257±0.1810	**0.1326±0.0565**
	NSDE	2.4853±0.2086†	0.8295±0.0696†
	MOEA/D-DE	0.6108±0.0047†	0.2138±0.0218†
	MO-EGS	5.1364±0.7926†	1.8028±0.1555†
	OSADE	**0.3197±0.0456**	0.5281±0.2944†

a good diverse set of solutions when compared to the other algorithms. In terms of convergence, OSADE has an evolved Pareto front which is closer to the Pareto optimal front as compared to the other algorithms.

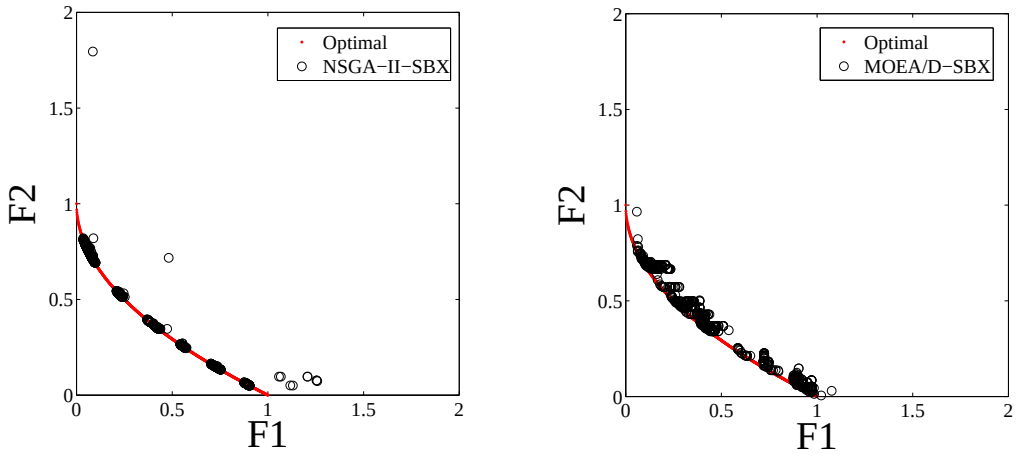

Figure 3.5: Pareto front of UF1 generated from NSGA-II-SBX and MOEA/D-SBX

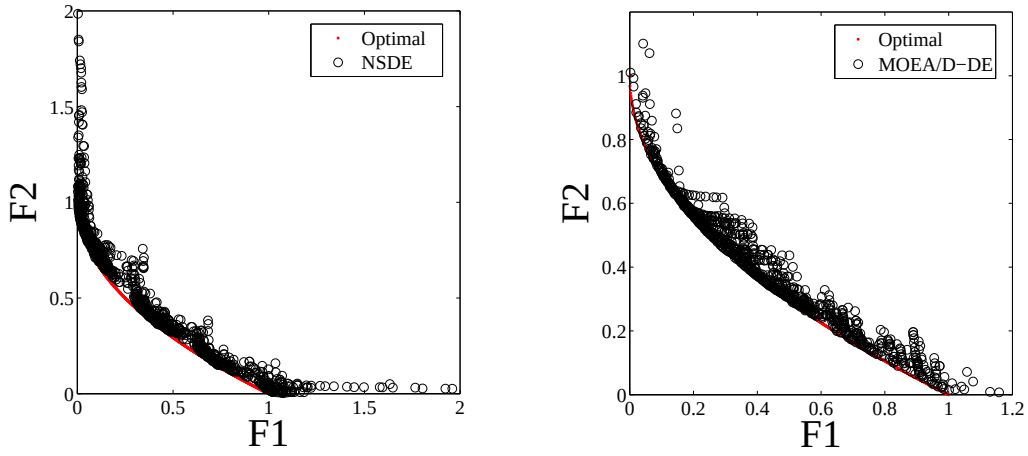

Figure 3.6: Pareto front of UF1 generated from NSDE and MOEA/D-DE

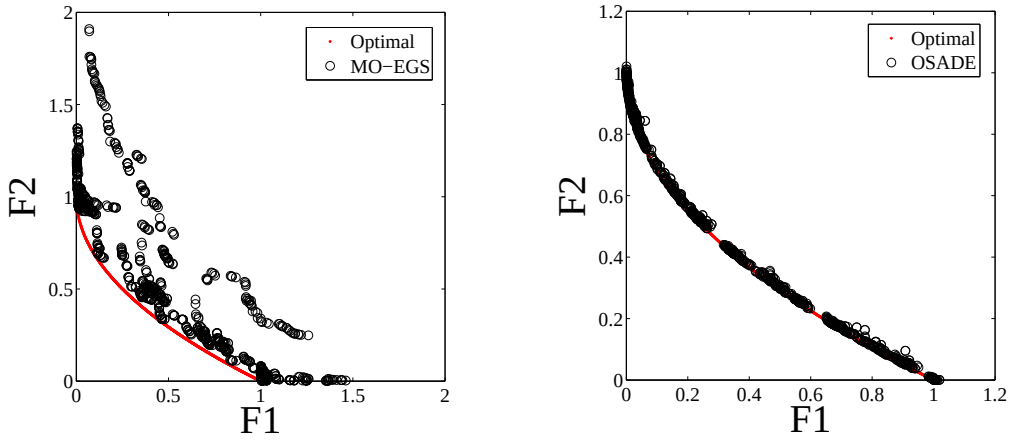

Figure 3.7: Pareto front of UF1 generated from MO-EGS and OSADE

In order to explore further on the convergence performance of all the algorithms used in this study, the evolutionary trajectories of the average HD values by all the algorithms for the UF1 problem are being plotted on Figures 3.8. From the plot, it is observed that OSADE achieves faster convergence as compared to the other algorithms whereby it takes about 10,000 fitness evaluations to find the approximate Pareto front. Moreover, OSADE also achieves the best convergence performance when compared to the other algorithms.

In order to solve UF problems well, algorithms need to be able to generate solution sets of higher diversity in order to explore the search space effectively, especially during the early stages of the search due to the presence of complicated Pareto sets in these problems. It has also been indicated that algorithms like the MOEA/D-SBX may not be suitable in dealing with the test instances in the UF test suite as the population in MOEA/D-SBX may lose diversity and the SBX operator in MOEAs have the shortcomings of producing inferior solutions [53]. As OSADE is inherently a DE variant, hence it possesses strong exploratory capability that is able to produce a set of diverse solutions, and this makes it suitable and effective for solving UF problems.

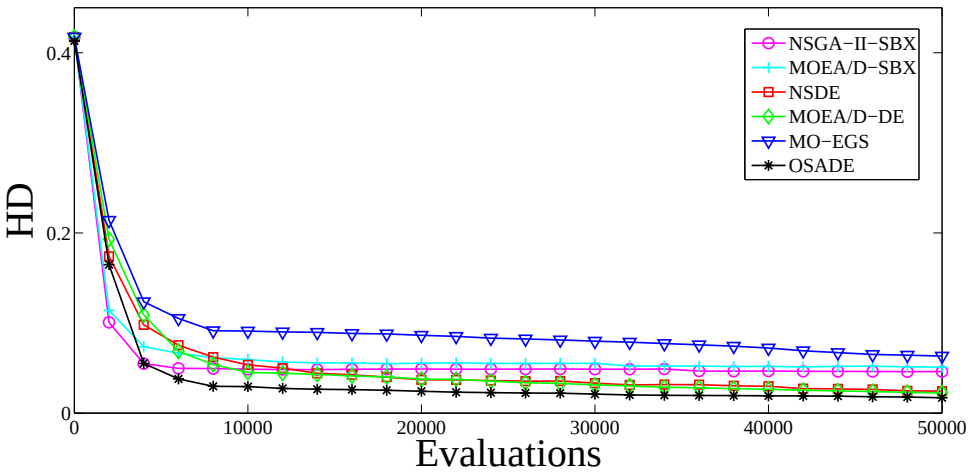

Figure 3.8: Evolutionary trajectories of mean HD by all the algorithms for UF1

3.7.4 Comparative studies for WFG problems

The final set of test problems used in this study is also a set of difficult MOOPs known as the WFG problems [107] that involve various transformation types. WFG problems are also scalable in terms of the number of objectives and the decision variables. In this study, two objectives and 30 decision variables were applied for the WFG problems. The decision vector used in the WFG problems consists of two position-related parameters and 28 distance-related parameters. In general, WFG problems are mainly subjected to bias, shift and reduction transformations. Bias transformations will impact the search process by biasing the fitness landscape which leads to substantially more solutions in some regions of the search space. As for shift transformations, the location of the optima can be set subject to skewing by bias transformations. This allows the avoiding of extremal or medial parameters that has optimal values at zero. Reduction transformations allow the incorporation of secondary parameters which can create dependencies between the position and distance-related parameters. If the reduction is done before the shift transformation, the objectives of the problem can be made effectively non-separable. As such, the WFG test suite provides a wide range of challenging problems with characteristics that are not available in other test suites. As seen from the simulation results, OSADE is able to generate a set of solutions with the best IGD and HD values in four WFG problems which include WFG1, WFG5, WFG6 and WFG8. Due to the bias or shift transformations, most of the solutions in WFG problems are located in a certain section of the search space. As such, the use of local search in OSADE can help in the determination of favourable search direction towards the optima, and this enhances overall convergence ability of OSADE.

Figures 3.9-3.11 display the Pareto fronts of WFG1 generated from the different algorithms. It could be seen that all the algorithms were unable to converge to global optimality for this problem. However, it is observed that the evolved Pareto front by OSADE is the nearest to the Pareto optimal front as compared to all the other algorithms. This demonstrates that OSADE obtains better performance in terms of both proximity and diversity when compared to the other

Table 3.6: Results obtained by the algorithms for WFG problems

Problem(M,D)	Algorithm	IGD	HD (with p=2)
WFG1(2,30)	NSGA-II-SBX	1.3881±0.0889†	0.3344±0.0257†
	MOEA/D-SBX	1.1814±0.1142†	0.2740±0.0314†
	NSDE	1.2669±0.0118†	0.2836±0.0026†
	MOEA/D-DE	1.2308±0.0033†	0.2755±0.0007†
	MO-EGS	1.2441±0.0048†	0.2793±0.0012†
	OSADE	**0.6882±0.0195**	**0.1542±0.0044**
WFG2(2,30)	NSGA-II-SBX	0.1017±0.0652†	0.0454±0.0335†
	MOEA/D-SBX	0.1567±0.0374†	0.0663±0.0184†
	NSDE	**0.0306±0.0100**	**0.0076±0.0022**
	MOEA/D-DE	0.0500±0.0064†	0.0153±0.0021†
	MO-EGS	0.3351±0.0549†	0.0834±0.0179†
	OSADE	0.0339±0.0003	0.0083±0.0001
WFG3(2,30)	NSGA-II-SBX	0.0250±0.0014†	0.0062±0.0004
	MOEA/D-SBX	0.0299±0.0033†	0.0073±0.0008†
	NSDE	**0.0245±0.0011**	**0.0062±0.0003**
	MOEA/D-DE	0.0292±0.0026†	0.0071±0.0006†
	MO-EGS	0.1833±0.0392†	0.0416±0.0088†
	OSADE	0.0338±0.0008†	0.0077±0.0002†
WFG4(2,30)	NSGA-II-SBX	0.0184±0.0008†	0.0047±0.0003†
	MOEA/D-SBX	**0.0155±0.0005**	**0.0039±0.0001**
	NSDE	0.0994±0.0036†	0.0223±0.0038†
	MOEA/D-DE	0.0791±0.0103†	0.0185±0.0024†
	MO-EGS	0.1585±0.0113†	0.0357±0.0026†
	OSADE	0.0186±0.0004†	0.0077±0.0006†
WFG5(2,30)	NSGA-II-SBX	0.0671±0.0010†	0.0152±0.0003†
	MOEA/D-SBX	0.0665±0.0007†	0.0154±0.0002†
	NSDE	0.0733±0.0011†	0.0168±0.0002†
	MOEA/D-DE	0.0673±0.0002†	0.0156±0.0004†
	MO-EGS	0.0794±0.0102†	0.0183±0.0024†
	OSADE	**0.0608±0.0014**	**0.0138±0.0003**
WFG6(2,30)	NSGA-II-SBX	0.0482±0.0041†	0.0109±0.0009†
	MOEA/D-SBX	0.0447±0.0070†	0.0101±0.0016†
	NSDE	0.0819±0.0204†	0.0185±0.0045†
	MOEA/D-DE	0.0818±0.0182†	0.0184±0.0041†
	MO-EGS	0.1110±0.0096†	0.0255±0.0024†
	OSADE	**0.0354±0.0041**	**0.0081±0.0009**
WFG7(2,30)	NSGA-II-SBX	0.0172±0.0007†	0.0046±0.0007†
	MOEA/D-SBX	**0.0141±0.0001**	**0.0037±0.0001**
	NSDE	0.0332±0.0018†	0.0077±0.0018†
	MOEA/D-DE	0.0180±0.0008†	0.0044±0.0008†
	MO-EGS	0.1083±0.0117†	0.0246±0.0027†
	OSADE	0.0159±0.0003†	0.0044±0.0001†
WFG8(2,30)	NSGA-II-SBX	0.0802±0.0038†	0.0185±0.0009†
	MOEA/D-SBX	0.0766±0.0054†	0.0177±0.0012
	NSDE	0.1272±0.0121†	0.0288±0.0027†
	MOEA/D-DE	0.1118±0.0112†	0.0254±0.0026†
	MO-EGS	0.2172±0.0095†	0.0490±0.0023†
	OSADE	**0.0705±0.0007**	**0.0164±0.0002**†
WFG9(2,30)	NSGA-II-SBX	0.0200±0.0019†	0.0053±0.0005†
	MOEA/D-SBX	**0.0177±0.0013**	**0.0048±0.0002**
	NSDE	0.0335±0.0008†	0.0081±0.0008†
	MOEA/D-DE	0.0348±0.0194†	0.0084±0.0193†
	MO-EGS	0.1594±0.0263†	0.0366±0.0059†
	OSADE	0.0194±0.0003†	0.0053±0.0001†

algorithms for this problem.

The evolutionary trajectories of the mean HD values by all the algorithms for the WFG1 problem are being plotted in Figure 3.12. From the plot, it is observed that the initial convergence in OSADE is much faster than all the other algorithms for this problem, and it may also take more than 50,000 evaluations for OSADE to fully converge. Despite this, its convergence performance

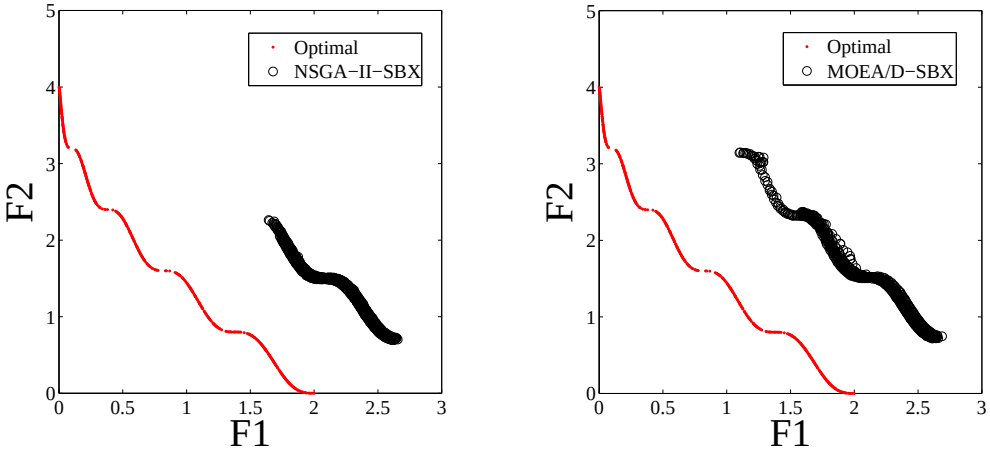

Figure 3.9: Pareto front of WFG1 generated from NSGA-II-SBX and MOEA/D-SBX

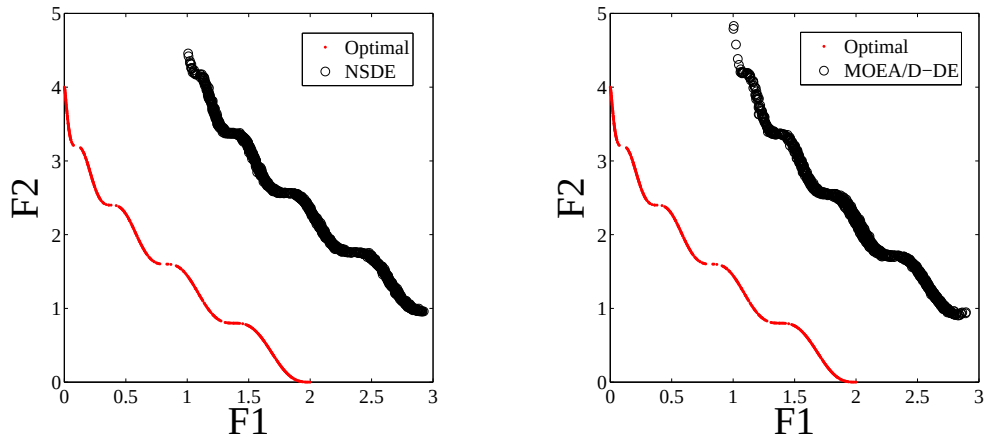

Figure 3.10: Pareto front of WFG1 generated from NSDE and MOEA/D-DE

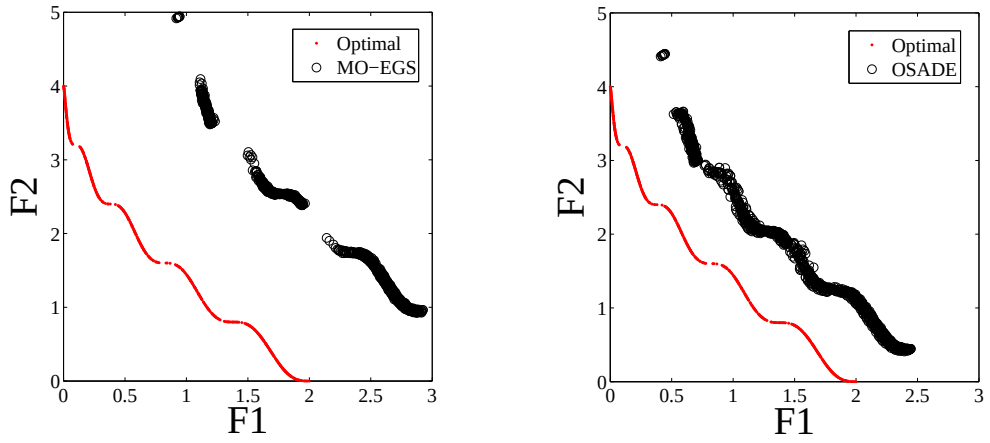

Figure 3.11: Pareto front of WFG1 generated from MO-EGS and OSADE

is still superior over all the other algorithms for this problem. The fast convergence rate achieved by OSADE suggests that the opposition-based self-adaptive mechanism for the DE parameters (F and CR) can lead to their optimal values being found more effectively and efficiently as the evolutionary process progresses. With the constant updating of the DE parameters to their near-optimal values during the evolutionary process, fitter solutions can be produced by the OSADE algorithm for the test problems. With fitter solutions going into the next generations, this will in turn propagate near-optimal DE parameter values for the subsequent generations as well. In this way, faster and better overall convergence is then achievable by the OSADE.

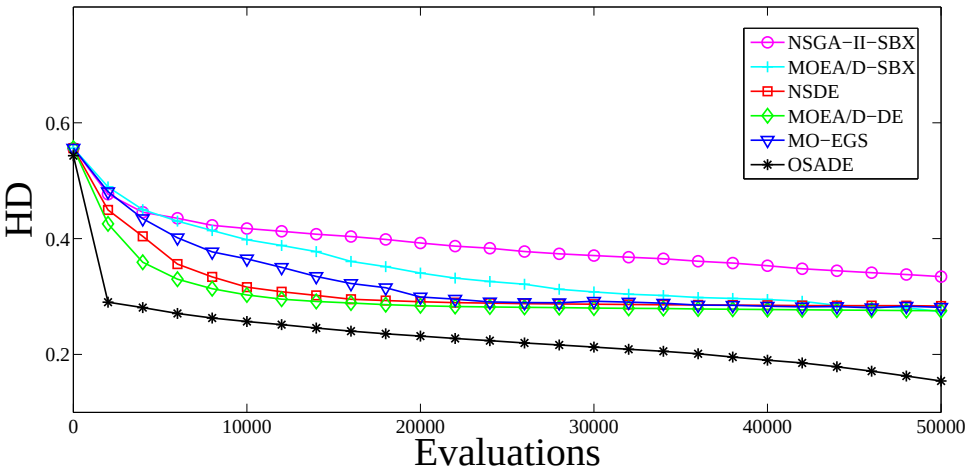

Figure 3.12: Evolutionary trajectories of mean HD by all the algorithms for WFG1

3.7.5 Summary of Comparative studies

Table 3.7: Frequencies of ranks on test problems using IGD

Algorithm	Rank 1	2	3	4	5	6
NSGA-II-SBX	4	6	14	6	5	3
MOEA/D-SBX	7	9	8	6	4	4
NSDE	2	2	4	18	16	4
MOEA/D-DE	3	5	14	11	7	4
MO-EGS	1	5	3	3	3	23
OSADE	25	7	1	3	2	0

The summary of the optimization results achieved by the different algorithms in this study is presented in terms of their rank frequencies shown in Tables 3.7-3.8. From these two tables,

Table 3.8: Frequencies of ranks on test problems using HD

	Rank					
Algorithm	1	2	3	4	5	6
NSGA-II-SBX	1	13	9	6	5	3
MOEA/D-SBX	11	8	9	4	2	4
NSDE	2	2	4	5	15	10
MOEA/D-DE	4	5	7	11	8	2
MO-EGS	1	4	4	5	8	16
OSADE	20	5	5	4	1	3

Table 3.9: Number of test problems where OSADE significantly outperforms a competing algorithm

	Number of Test Problems	
Algorithm	IGD	HD
NSGA-II-SBX	23	20
MOEA/D-SBX	22	18
NSDE	25	20
MOEA/D-DE	20	20
MO-EGS	25	20

it can be observed that the proposed algorithm OSADE has the overall best ranking in most of the test problems. The Wilcoxon ranksum test was also performed on the simulation results, and for every algorithm compared against OSADE, the number of test instances whereby the latter significantly outperforms it according to both the performance metrics (IGD and HD) is being stated in Table 3.9.

The overall best performance achieved by OSADE demonstrates that the novel opposition-based self-adaptive mechanism for the mutation scale factor in the DE operator is able to improve the optimization performance of basic differential evolution. This self-adaptive mechanism not only eliminates the need to fix the DE parameters (mutation scale factor F and crossover rate CR) at the start of the optimization process, but also increases the probability of finding optimal settings for the parameters. With optimal settings for the parameters, better solutions can be generated throughout the evolution process. This in turn allows OSADE to be able to achieve better and accelerated convergence for the test problems.

The promising results achieved by OSADE can also be attributed to hybridization of the earlier mentioned DE variant with the MO-EGS to act as a form of local search. For every generation, the DE variant, which is the earlier phase of OSADE, is capable of strong global search to find promising solutions in the search space. All the solutions obtained in the DE phase

will be stored in an archive before the algorithm gets enhanced by the embedded local search (MO-EGS) during the later phase of OSADE. By archiving the solutions from the DE phase, this allows the retention of good solutions found during global search. A certain number of solutions from the DE phase are chosen via binary tournament selection for local search to generate child solutions which are then updated to the archive as long they are not being dominated by any existing members in this archive. MO-EGS is effective in the exploitation of solutions in the search space. In this way, a larger selection pressure can be induced which helps towards overall convergence of the algorithm. Through the hybridization, exploration and exploitation of the search space can then be balanced for OSADE which aids in convergence while maintaining diversity.

3.7.6 Scalability Studies

In this section, the performance of the algorithms under comparison in terms of their scalability in decision variables and objective functions is examined for some selected test problems. Scalability in decision variables increases the difficulty of the problems due an enlargement of the search space, and this will lead to an increase in the number of possible moves towards optimality. For the case of scalability in objective functions, such problems are termed as many-objective problems [139–141] whereby there is generally lesser selection pressure devoted to every non-dominated individual in the problem. This is attributed to the fact that most individuals in the population are non-dominated in the earlier evolution stages, and this makes it difficult for the evolutionary algorithm in determining the fitter solutions. Moreover, as the dimensionality of the objective space increases, it is also harder for the algorithm to decide on which solutions are non-dominated against other solutions.

DTLZ2 and DTLZ3 were selected for the study of the scalability in the number of objective functions. In order to examine the performance of the algorithms in many-objective problems, these problems were scaled to a larger number of objective functions which poses greater

challenge to the algorithms in their search for a high dimensional Pareto front. The number of decision variables D was also fixed according to $D = M + K - 1$ where M is the number of objectives and constant $K = 10$. The population size was varied according to the number of objectives as indicated in Table 3.1. The performance metric of IGD for the two test problems with (a) 3 objectives, (b) 5 objectives, and (c) 7 objectives is as displayed in the box-plots in Figures 3.13-3.15. The indices of the algorithms as used in the box-plots are explained in Table 3.10.

Table 3.10: Indices of the algorithms

Index	Algorithm
1	NSGA-II-SBX
2	MOEA/D-SBX
3	NSDE
4	MOEA/D-DE
5	MO-EGS
6	OSADE

For DTLZ2 with 3 objectives, it is observed that OSADE achieves better performance than all the other algorithms except MO-EGS. However, when the number of objectives was being scaled to 5 or 7, OSADE is seen to achieve better, if not comparable, performance than the MO-EGS while outperforming all the other algorithms as well. For the case of DTLZ3 with 3 and objectives, OSADE achieves comparable performance with the MOEA/D algorithms by reaching convergence while outperforming the rest. When the number of objectives was scaled to 7 for the DTLZ3 problem, only OSADE and MOEA/D-SBX were able to reach full convergence. The good performance achieved by OSADE in these test instances demonstrates that OSADE is able to scale well with the number of objective functions when compared to other algorithms in this study.

As for the study of the scalability of decision variables, the UF1 and the DTLZ1 with 3 objectives were selected. In order to observe the behavior of the algorithms for problems with a very large number of decision variables, the performance metric of the HD values obtained by all the algorithms for these problems are plotted in Figures 3.16-3.17. In the plots, the number of decision variables spans from 100 to 1000 and 50 to 500 for UF1 and DTLZ1 respectively.

From the figures, it can be observed that OSADE performs very well for UF1 and outperforms all the other algorithms. For the case of DTLZ1, it can be observed that the performance of all the algorithms deteriorates with the increase in the number of decision variables due to the increasing ease of the algorithms getting trapped in the local optima in this highly multi-modal problem. This observation demonstrates the weakness of the OSADE as well as the other MOEAs. In summary, OSADE performs better than the other algorithms in terms of the scalability in the number of decision variables except for the MOEA/D algorithms as seen from the results obtained. This may also suggest that MOEA/D algorithms are less susceptible to be trapped at local optima as compared to the other algorithms for high dimensional problems.

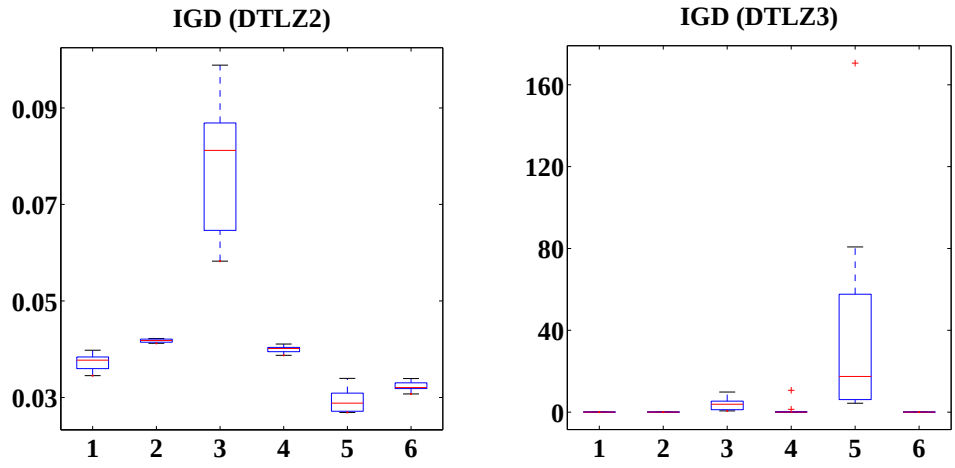

Figure 3.13: Performance metric of IGD for DTLZ2/DTLZ3 (3 objectives and 12 decision variables)

3.7.7 Sensitivity Analysis of Bounds for DE Control Parameters

In OSADE, the self-adaptive mechanism involves the setting of the mutation factor F and the crossover rate CR which are the two important parameters for the differential evolution (DE) operator. For all the experiments in the previous sections, they were conducted based on the setting of lower and upper bounds for both DE parameters as 0.1 and 0.9 respectively. This setting will be referred to as the original settings in this sub-section. This section investigates the effect when the bounds are varied, and tries to provide a recommended setting for the algorithm.

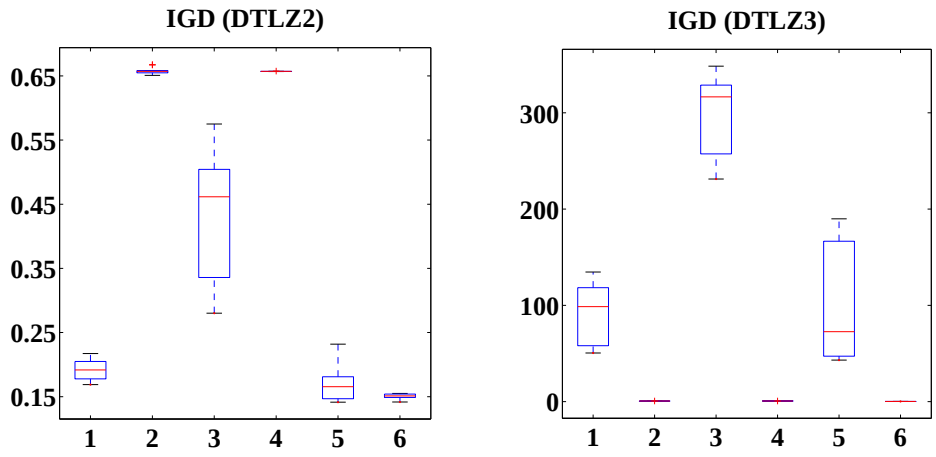

Figure 3.14: Performance metric of IGD for DTLZ2/DTLZ3 (5 objectives and 14 decision variables)

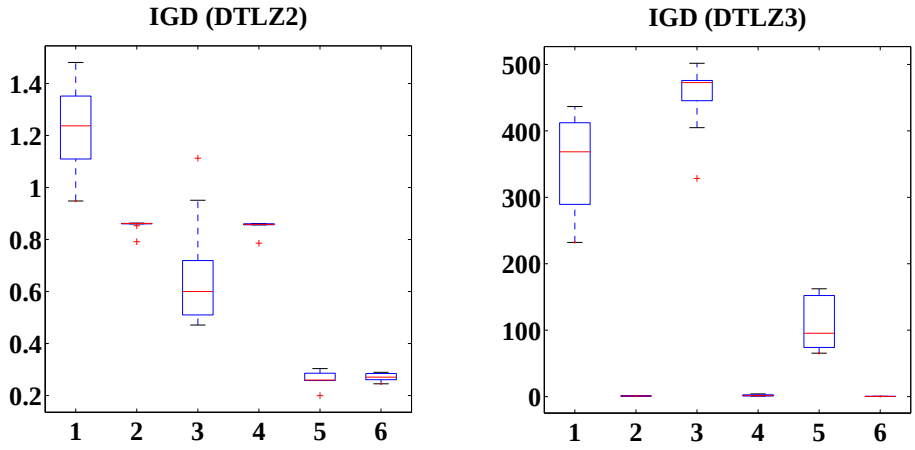

Figure 3.15: Performance metric of IGD for DTLZ2/DTLZ3 (7 objectives and 16 decision variables)

To study the sensitivity of OSADE to the lower and upper bounds, experiments for the UF and WFG test suites were repeated by using different combinations of F and CR. In Table 3.11, the values of the lower and upper bounds for all the different combinations of the bounds are stated. For all the experiments, the bounds were first varied for the mutation factor F but with the original settings being applied to the bounds for the crossover rate CR. Next the bounds for CR were varied while keeping the values of the bounds for F at the original settings. The other parameters in OSADE related to the self-adaptive mechanism were kept unchanged.

In Tables 3.12-3.15, the mean IGD values obtained by OSADE for all the UF problems using different combinations of the lower and upper bounds of both F and CR are being dis-

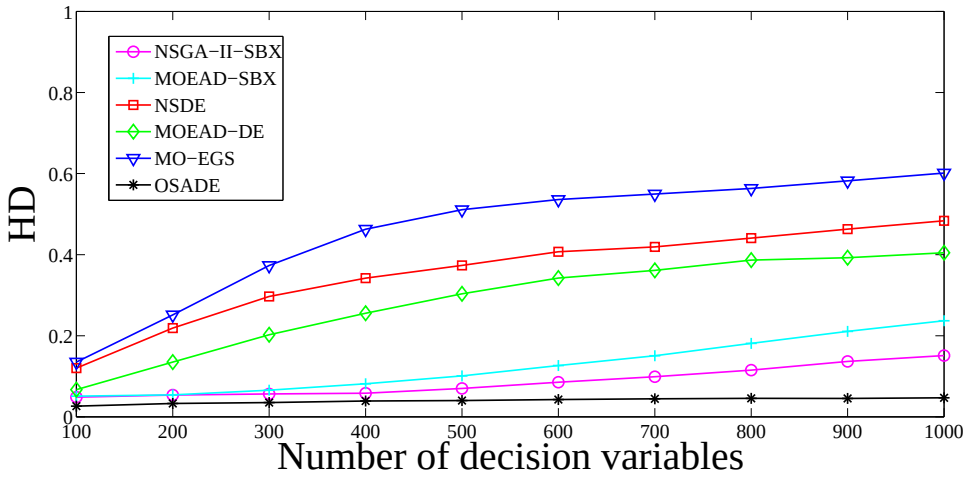

Figure 3.16: Performance metric of HD for UF1 versus number of decision variables

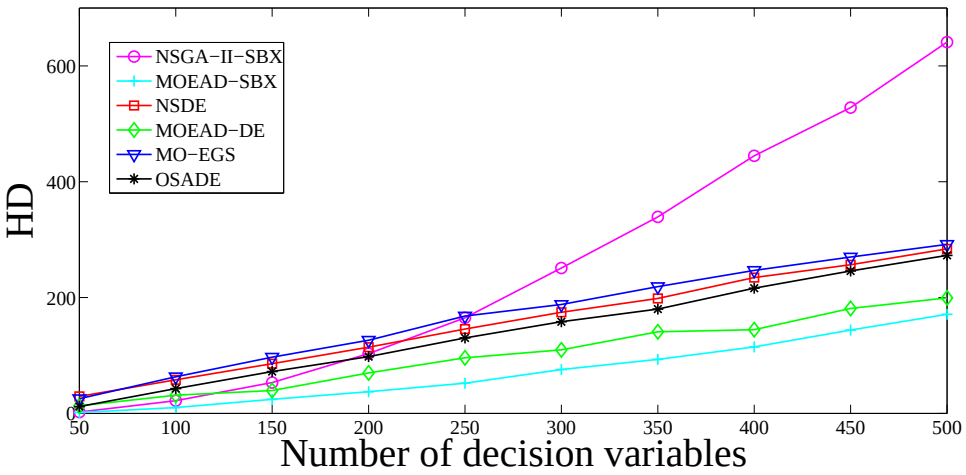

Figure 3.17: Performance metric of HD for DTLZ1 versus number of decision variables

Table 3.11: Lower and upper bounds for different combinations (cases) of DE parameters F and CR

	Case									
	1	2	3	4	5	6	7	8	9	10
Lower Bound	0.1	0.3	0.5	0.7	0.1	0.3	0.5	0.1	0.3	0.1
Upper Bound	0.9	0.9	0.9	0.9	0.7	0.7	0.7	0.5	0.5	0.3

played. The values in parentheses beside every IGD value represent the ranks for a particular combination of the lower and upper bounds as compared to the other combinations for the test problem. For every test problem, the top four entries for every case in terms of the lowest IGD values will be marked in boldface in all the tables.

Table 3.12: IGD measurement for UF1-UF5 with varying bounds for F

	Test Problem				
Case	UF1	UF2	UF3	UF4	UF5
1	0.0404(5)	0.0204(6)	0.2087(7)	0.0408(5)	0.1494(7)
2	**0.0370(2)**	**0.0184(3)**	**0.1987(2)**	**0.0405(3)**	0.1419(5)
3	**0.0326(1)**	**0.0177(2)**	**0.2030(4)**	**0.0404(2)**	**0.1354(2)**
4	**0.0377(3)**	**0.0174(1)**	**0.2006(3)**	0.0424(9)	0.1649(9)
5	0.0485(6)	0.0258(7)	0.2054(6)	0.0414(7)	0.1487(6)
6	0.0517(8)	0.0195(5)	**0.1978(1)**	0.0411(6)	**0.1221(1)**
7	**0.0395(4)**	**0.0194(4)**	0.2129(9)	**0.0401(1)**	0.1756(10)
8	0.0550(9)	0.0284(10)	0.2108(8)	0.0418(8)	**0.1401(4)**
9	0.0486(7)	0.0273(8)	0.2157(10)	**0.0405(3)**	0.1567(8)
10	0.0659(10)	0.0281(9)	0.2053(5)	0.0426(10)	**0.1400(3)**

Table 3.13: IGD measurement for UF6-UF10 with varying bounds for F

	Test Problem				
Case	UF6	UF7	UF8	UF9	UF10
1	0.0240(6)	**0.0178(4)**	0.1098(7)	0.0805(6)	0.3197(5)
2	**0.0193(3)**	0.0551(7)	**0.0982(3)**	0.0778(5)	0.3451(7)
3	**0.0105(1)**	**0.0102(1)**	**0.0132(4)**	0.1216(8)	0.4320(9)
4	**0.0117(2)**	**0.0107(2)**	0.1126(10)	0.1737(10)	0.6647(10)
5	0.0333(7)	0.0665(8)	0.1087(6)	**0.0484(3)**	**0.2872(4)**
6	0.0201(5)	0.0297(6)	**0.0955(1)**	0.0877(7)	0.3202(6)
7	**0.0199(4)**	0.0269(5)	0.1123(9)	0.1355(9)	0.3706(8)
8	0.0371(8)	0.1000(10)	**0.0973(2)**	**0.0302(2)**	**0.2601(2)**
9	0.0374(9)	**0.0107(3)**	0.1054(5)	**0.0513(4)**	**0.2748(3)**
10	0.0614(10)	0.0845(9)	0.1121(8)	**0.0202(1)**	**0.2521(1)**

Table 3.14: IGD measurement for UF1-UF5 with varying bounds for CR

	Test Problem				
Case	UF1	UF2	UF3	UF4	UF5
1	0.0404(7)	0.0204(5)	0.2087(7)	0.0408(5)	0.1494(8)
2	**0.0374(4)**	**0.0194(3)**	0.1951(5)	0.0426(7)	0.1468(7)
3	0.0375(5)	**0.0191(2)**	**0.1932(3)**	0.0446(9)	0.1465(6)
4	**0.0283(1)**	**0.0190(1)**	**0.1794(1)**	0.0457(10)	**0.1266(3)**
5	0.0427(9)	0.0209(6)	0.2041(6)	**0.0393(3)**	0.1454(5)
6	0.0415(8)	**0.0201(4)**	**0.1886(2)**	0.0415(6)	**0.1275(4)**
7	**0.0348(2)**	0.0209(7)	**0.1949(4)**	0.0436(8)	**0.1144(1)**
8	0.0446(10)	0.0216(10)	0.2202(9)	**0.0379(2)**	0.1721(9)
9	0.0403(6)	0.0215(9)	0.2175(8)	**0.0403(4)**	**0.1147(2)**
10	**0.0350(3)**	00214(8)	0.2225(10)	**0.0356(1)**	0.1734(10)

Table 3.15: IGD measurement for UF6-UF10 with varying bounds for CR

	Test Problem				
Case	UF6	UF7	UF8	UF9	UF10
1	0.0240(8)	**0.0178(4)**	0.1098(7)	0.0805(7)	0.3197(8)
2	**0.0153(3)**	0.0288(6)	**0.1010(3)**	**0.0560(4)**	0.2982(6)
3	**0.0121(2)**	**0.0134(3)**	0.1243(10)	**0.0395(1)**	0.3032(7)
4	**0.0102(1)**	**0.0103(1)**	**0.1009(2)**	0.0614(5)	**0.2945(4)**
5	0.0201(5)	0.0471(9)	0.1045(5)	0.0975(9)	0.3304(10)
6	**0.0174(4)**	0.0300(7)	0.1230(9)	**0.0556(3)**	0.3281(9)
7	0.0219(6)	0.0327(8)	0.1195(8)	**0.0423(2)**	**0.2772(1)**
8	0.0296(9)	0.0281(5)	**0.1044(4)**	0.1030(10)	0.2967(5)
9	0.0385(10)	**0.0133(2)**	0.1055(6)	0.0861(8)	**0.2843(2)**
10	0.0232(7)	0.0514(10)	**0.0947(1)**	0.0668(6)	**0.2857(3)**

For UF problems, it is observed that Case 3 (lower bound = 0.5, upper bound = 0.9) and Case 4 (lower bound = 0.7, upper bound = 0.9) have the highest occurrences of being in the top four cases in the experiments of varying the mutation factor F and crossover rate CR respectively. It was mentioned in Section 3.7.3 that UF problems require the algorithm to be able to have strong exploratory capability so as to generate a diverse set of solutions. If the mutation factor F is ranged between 0.5 and 0.9, it could potentially be adapted to larger values during evolutionary process to provide stronger exploration in the search space, and could also be adapted to a smaller value of 0.5 to increase the convergence rate. As such, this may allow a balance between exploration and exploitation when OSADE handles the UF problems. As for the crossover rate, a range between 0.7 to 0.9 may allow sufficiently fast convergence and yet allowing adequate chance for mutation of the individuals in the population.

The results for the mean IGD values obtained by OSADE for all the WFG problems using different combinations of the lower and upper bounds of both F and CR are displayed in Tables 3.16-3.19. The values in parentheses beside every IGD value represent the ranks for a particular combination of the lower and upper bounds as compared to the other combinations for the test problem. Similarly, the top four entries for every case in terms of the lowest IGD values for every test problem will also be marked in boldface in all the tables.

For WFG problems, Case 9 is observed to be the case of having the most occurrences of being in the top four cases in both experiments of varying the DE parameters separately. Due to the nature of the WFG problems whereby the solutions are located at certain part of the search

Table 3.16: IGD measurement for WFG1-WFG5 with varying bounds for F

	Test Problem				
Case	WFG1	WFG2	WFG3	WFG4	WFG5
1	**0.6882(4)**	0.0339(6)	0.0338(6)	0.0186(6)	**0.0608(3)**
2	0.8082(7)	0.0338(5)	**0.0325(4)**	**0.0185(4)**	0.0615(5)
3	0.8805(8)	**0.0328(2)**	**0.0311(3)**	0.0187(8)	0.0626(7)
4	0.9184(10)	**0.0321(1)**	**0.0296(1)**	0.0187(8)	0.0627(9)
5	**0.6196(3)**	0.0342(9)	0.0348(8)	0.0186(6)	0.0626(7)
6	0.7830(6)	**0.0337(4)**	0.0333(5)	**0.0180(1)**	**0.0607(2)**
7	0.8812(9)	**0.0330(3)**	**0.0309(2)**	0.0188(10)	**0.0605(1)**
8	**0.5263(2)**	0.0341(8)	0.0359(9)	**0.0183(3)**	**0.0610(4)**
9	0.7547(5)	0.0339(6)	0.0343(7)	**0.0185(4)**	0.0624(6)
10	**0.4150(1)**	0.0594(10)	0.0369(10)	**0.0182(2)**	0.0628(10)

Table 3.17: IGD measurement for WFG6-WFG9 with varying bounds for F

	Test Problem			
Case	WFG6	WFG7	WFG8	WFG9
1	0.0354(7)	0.0159(8)	0.0705(5)	**0.0194(2)**
2	0.0342(6)	0.0155(6)	0.0717(7)	0.0197(8)
3	0.0376(9)	0.0157(7)	0.0739(8)	**0.0193(1)**
4	0.0390(10)	0.0169(10)	0.0764(10)	0.0198(9)
5	0.0337(5)	**0.0151(4)**	**0.0699(3)**	0.0199(10)
6	**0.0307(1)**	0.0154(5)	0.0714(6)	0.0195(6)
7	0.0360(8)	0.0161(9)	0.0741(9)	0.0196(7)
8	**0.0330(3)**	**0.0149(2)**	**0.0682(2)**	**0.0194(2)**
9	**0.0332(4)**	**0.0149(2)**	**0.0702(4)**	**0.0194(2)**
10	**0.0323(2)**	**0.0144(1)**	**0.0679(1)**	**0.0194(2)**

Table 3.18: IGD measurement for WFG1-WFG5 with varying bounds for CR

	Test Problem				
Case	WFG1	WFG2	WFG3	WFG4	WFG5
1	0.6882(9)	0.3339(6)	0.0338(6)	0.0186(6)	**0.0608(3)**
2	**0.6175(3)**	**0.0327(1)**	**0.0305(1)**	0.0188(9)	0.0623(9)
3	0.6543(6)	**0.0337(3)**	**0.0316(3)**	0.0186(6)	0.0617(7)
4	0.8206(10)	**0.0327(1)**	**0.0305(1)**	0.0188(9)	0.0623(9)
5	0.6618(7)	0.0340(8)	0.0345(8)	**0.0183(3)**	0.0615(6)
6	**0.5787(1)**	0.0341(10)	0.0337(5)	0.0185(5)	**0.0608(3)**
7	**0.6428(4)**	**0.0337(3)**	**0.0317(4)**	0.0187(8)	0.0617(7)
8	0.6446(5)	0.0339(6)	0.0352(9)	**0.0182(2)**	**0.0605(2)**
9	**0.5795(2)**	0.0340(8)	0.0344(7)	**0.0184(4)**	**0.0599(1)**
10	0.6665(8)	0.0338(5)	0.0355(10)	**0.0177(1)**	0.0613(5)

Table 3.19: IGD measurement for WFG6-WFG9 with varying bounds for CR

	Test Problem			
Case	WFG6	WFG7	WFG8	WFG9
1	0.0454(5)	0.0159(7)	0.0705(5)	**0.0194(4)**
2	0.0573(10)	0.0165(9)	0.0727(9)	0.0201(9)
3	0.0394(9)	0.0160(8)	0.0717(7)	0.0200(7)
4	0.0393(8)	0.0165(10)	0.0727(9)	0.0201(9)
5	**0.0337(3)**	**0.0152(4)**	**0.0695(1)**	**0.0192(2)**
6	0.0369(6)	0.0153(5)	0.0706(6)	0.0198(6)
7	0.0382(7)	0.0157(6)	0.0720(8)	0.0200(7)
8	**0.0334(2)**	**0.0150(2)**	**0.0699(4)**	0.0196(5)
9	**0.0352(4)**	**0.0151(3)**	**0.0698(2)**	**0.0192(2)**
10	**0.0311(1)**	**0.0148(1)**	**0.0698(2)**	**0.0191(1)**

space, the need for diversity may not be as great as compared to UF problems. Hence, a small mutation factor between 0.3 and 0.5 will allow smaller jumps to other points in the search space. As for the adoption for the similar range for crossover rate, this will allow good control of which and the number of components to be mutated in every individual in the current population.

In conclusion, it is recommended that the bounds for F are to be set between 0.5 to 0.9 and CR to be between 0.7 to 0.9 for the case of UF problems. On the other hand, it may be suitable to adopt the range of bounds for both DE parameters to be between 0.3 and 0.5 for the case of WFG test suite. These analyses suggest that the setting of the bounds for the self-adaptive mechanism in OSADE may be problem-dependent in order to achieve the best results for the problems to be handled.

3.8 Summary

In view of the increasing complexity and dimensionality seen in several optimization problems today, there is a strong need to find ways to improve the efficiency and effectiveness of evolutionary algorithms. With memetic algorithms as a possible solution to allow fast convergence and strong global search capability with robustness, this gave motivation for the proposal of a novel memetic algorithm based on differential evolution in the context of multi-objective optimization to be presented in this chapter. Unlike most other differential evolution variants, OSADE eliminates the need of fixing the DE control parameters at the start of evolution which usually requires a tedious trial-and-error process. With opposition-based learning being incorporated into the self-adaptive mechanism, this may give a higher probability of finding near-optimal settings for the DE control parameters in the proposed algorithm throughout the evolutionary process. As the control parameters of OSADE gets adapted to near-optimal settings in the evolutionary process, better solutions can be generated throughout the evolutionary process as well. In OSADE, its DE operator is also hybridized with the multi-objective evolutionary gradient search (MO-EGS) to act as a form of local search which helps to improve the exploitation abilities for the overall

algorithm. As such, OSADE will be able to complement the strengths of a novel opposition-based self-adaptive differential evolution (DE) algorithm, which balances the exploration and exploitation abilities as found in the original DE, together with MO-EGS which utilizes gradient information which brings forth higher selection pressure. Through the validation of OSADE using a suite of carefully selected test benchmark problems for continuous multi-objective optimization, it is observed that OSADE achieves overall better performance in terms of convergence and diversity over the other algorithms compared in this chapter. In addition, OSADE can be recommended for solving multi-modal problems as well as problems with complicated Pareto sets as seen from its superior optimization performance in such types of problems when compared to the other algorithms in this study.

Chapter 4

A Grid-based Differential Evolution for Many-objective Optimization

In this chapter, the OSADE algorithm presented in the previous chapter is being extended to handle many-objective optimization problems. The opposition-based self-adaptive mutation scheme found in OSADE is first incorporated with a newly formulated opposition-based local mutation scheme, whereby one of the schemes will be selected using a linear decreasing probability rule. The resultant mutation scheme is then integrated into a grid-based framework to form a novel grid-based differential evolution variant called GrDE. Empirical studies show that GrDE displays promising results in terms of finding a well-approximated and well-distributed set of solutions for the many-objective problems tested on.

4.1 Chapter Objectives

The main objective of this chapter is to explore the possibility of extending OSADE into a grid-based differential evolution algorithm for handling many-objective problems. The chapter also seeks to explore the formulation of another novel mutation strategy and integrating it into a grid-based framework that is catered towards many-objective optimization.

4.2 Introduction

Many real-world optimization problems involve several objectives to be met simultaneously, and these problems are termed as multi-objective optimization problems (MOOPs) [4, 108–110]. As the objectives are conflicting in nature, there is usually no single optimal solution to MOOPs but rather a set of Pareto-optimal solutions. Through the decades, population-based evolutionary algorithms (EAs) [3] have been successfully applied to a wide range of multi-objective optimization problems as they are able to obtain an approximate set of Pareto-optimal solutions in a single run.

In recent years, many-objective optimization has been gaining increasing attention and popularity among researchers in the evolutionary multi-objective optimization (EMO) community [142], [143]. Many-objective optimization problems are multi-objective optimization problems with four or more objectives, and these problems are seen in various engineering applications [140, 144–148]. The motivation for the increasing efforts to work on this area is attributed to the realization that existing state-of-the-art EMO algorithms do not scale well with the number of objective functions [149, 150]. This is due to the fact that most solutions in the current population will become non-dominated to each other as the number of objectives increases. As such, the Pareto dominance-based fitness assignment method commonly used in most EMO algorithms will have problems inducing a strong selection pressure towards the Pareto front.

Another difficulty encountered in many-objective problems is in the aspect of the visualization of solutions. Usually, the decision maker for a problem will choose the final solution set based on a set of non-dominated solutions according to his or her preference. When there is an increase in the number of objectives, the visualization of non-dominated solutions obtained will be much tougher. As such, the selection of the final solutions will be very difficult in many-objective optimization.

As seen from literature, researchers have proposed the modification of Pareto dominance relation in order to impose stronger selection pressure towards the Pareto front. Some exam-

ples of the modifications reflected in these studies include ϵ-dominance [151], preference-order ranking [152], k-optimality [144], and fuzzy Pareto dominance [153] just to name a few. Experimental results for these methods demonstrate that they are able to steer a better search towards the Pareto front as compared to the conventional Pareto dominance relation. It is also observed that non-Pareto based fitness assignment techniques like ranking dominance [154], average rank [155], as well as some distance-based ranking methods [141, 156–158] are able to be considered for use in evolutionary many-objective optimization due to promising results that they displayed in terms of convergence towards the optimum. However, the drawback in these non-Pareto based techniques is that they may direct the final Pareto set to a certain sub-regions of the Pareto front [159, 160]. Another approach of non Pareto-based fitness assignment techniques is to incorporate quality indicator functions such as hypervolume or epsilon indicators for scalability improvement [161, 162]. Another idea proposed is to assign fitness to solutions in the population based on a number of different scalarizing functions [163, 164].

Another perspective of handling many-objective problems is to enhance the diversity maintenance mechanism in EMO algorithms. As the size of the objective space is being increased in these problems, there will be stronger contention between convergence and diversity requirements [165, 166]. In most of the diversity maintenance techniques like niche, crowding distance, and clustering, they are not able to provide strong selection pressure towards the Pareto front in high objective space, and these techniques may even cause hindrance to the evolutionary search process due to their preference for dominance resistant solutions [166]. A possible way of handling this problem is to lower the diversity requirement during the selection process. For this, a diversity management operator which is able to balance both the convergence and diversity requirements was proposed by Adra and Fleming [165]. In [167], it is demonstrated that the performance of NSGA-II could be improved should zero distance instead of infinity distance be assigned to boundary solutions.

Dimensionality reduction is another approach which involves the objective selection for the

problem. This approach is based on the idea that not every objective in the problem is required for the definition of the Pareto front. The decrease in the number of objectives provides a remedy for the difficulty of the visualization of solutions in high dimensional space whereby the objective vectors are mapped into a lower-dimensional space for visualization [168–170] which was done via the use of both linear and nonlinear correlation-based dimensionality reduction techniques based on principal component analysis and maximum variance unfolding. On the other hand, we are also able to see dimensionality reduction approach based on Pareto dominance proposed by Xue *et al.* [171].

From above-mentioned studies, we are able to find several different promising approaches to address the difficulties found in many-objective optimization problems. However, there is still a strong need to explore further improvements on EMO algorithms so that they can be more effectively applied on many-objective optimization problems [172]. From literature, we are able to see past attempts for the use of grid in EMO algorithms. This is motivated by the fact that the use of grid is able to allow the distribution of solutions to be reflected clearly during the evolutionary process as every solution can be represented by its own grid location. This in turn allows the information on both convergence and diversity to be available concurrently. As the location of a solution is able to be determined, its performance in terms of convergence and diversity can be estimated through the comparison of grid locations with other solutions. In addition, the use of grid-based criterion is able to provide the quantitative difference between the objective values among solutions instead of just comparing them qualitatively as seen in the case of Pareto dominance criterion. As such, the use of grid-based criterion might be more promising than Pareto dominance criterion in many-objective problems in consideration of rising selection pressure from the quantitative comparison of objective values among solutions [140, 159]. However, the potential of the use of grid has not been fully exploited in most of the existing grid-based EMO algorithms as they are only able to achieve good performance on problems with two or three objectives and not for the case of many-objective optimization problems.

As seen from the promising results from the Grid-based Evolutionary Algorithm (GrEA) proposed by Yang *et al.* [173], this gives forth motivation to propose a grid-based Differential Evolution (DE) algorithm called the GrDE for the solving of many-objective optimization problems. The aim of this study is to explore how the integration of a novel differential evolution variant into the framework of the grid-based EA (GrEA) would be able to enhance differential evolution as an EMO algorithm to be considered for many-objective optimization. The performance of the GrDE algorithm will be investigated by comparing it with five other state-of-the-art EMO algorithms using a set of test benchmark many-objective optimization problems.

4.3 Related Works

As seen in the past decade, we are able to find several grid-based EMO algorithms [174–177] being proposed and studied by many researchers in the EMO community. A very early initiative in the use of grid can be found in the Pareto-based evolution strategy (PAES) that was introduced by Knowles *et al.* [178] to ensure that the diversity of the archive population is being maintained. To achieve this, the estimation of the crowding distance of a solution has to done with the number of solutions having the same grid location as itself. If the size of an archive is reached, a non-dominated solution can enter it by replacing one of the solutions possessing the highest crowding degree only if it has a lower crowding distance.

In the dynamic multi-objective evolutionary algorithm (DMOEA) [179], the use of grid is different from the approach in PAES as it is being used to contain the convergence and diversity information of the solutions. In the approach here, there is assignment of a grid-based rank and a density value to every cell located in the grid according to the Pareto dominance relation and the grid position of solutions.

The idea of ϵ-dominance as pioneered by Laumanns *et al.* [151] can be perceived as a grid-based technique that balances the convergence and diversity requirements of an EMO algorithm. Based on this concept, the ϵ-MOEA was developed by Deb *et al.* [174] which segregates the

objective space into hyperboxes according to the size of epsilon ϵ whereby there is only one solution present in each hyperbox. However, ϵ-dominance may lead to the loss of boundary solutions during the evolutionary process of ϵ-MOEA. As such, a variant of the algorithm known as Pareto adaptive ϵ-dominance was being proposed by Hernandez Diaz *et al.* [180].

Other efforts involving the use of grid-based techniques include a dynamic grid resizing strategy introduced by Rachmawati *et al.* [176] which is able to discover the settings for the grid sizes according to certain metrics, and perform the varying of the size of hyperboxes when required. In another recent work, a steady-state algorithm known as the territory-based EMO algorithm was being developed by Karahan *et al.* [181] which basically marks out a territory around individuals in order to achieve diversity maintenance.

Studies for the above-mentioned implementations show that they are able to perform well on problems with two or three objectives. However, there are not much works on these grid-based algorithms being applied on many-objective problems. A noticeable attempt to utilize the properties of grid for many-objective optimization can be found in the grid-based fitness strategy (GrFS) [182]. In this initiative, the authors formulated three fitness assignment criteria based on the grid coordinates of the individuals which will be used to increase the selection pressure towards the Pareto optimal front. The promising experimental results obtained from this initiative led to the motivation towards the development of the grid-based evolutionary algorithm (GrEA) which encompasses an extended and more comprehensive study in the handling of many-objective optimization. In GrEA, the concept of grid dominance was proposed for the comparison of individuals during the mating and environmental selection processes. Another distinctive feature in GrEA lies in a carefully designed density estimator for the population individuals. The density estimation for an individual factors in the consideration of the number of its neighbouring individuals and also the difference in the distance between the neighbours and itself. In addition, the fitness adjustment technique in GrEA is an improved version from the one in GrFS which aims to prevent partial overcrowding of the solutions and to steer the evolutionary

search towards different directions in the archive population.

4.4 Background

4.4.1 Grid Environment

In the proposed grid-based differential evolution algorithm, the framework as found in GrEA [173] is adopted whereby the positions of individuals in the objective space are based on grid. As the population gets refreshed during the evolutionary process, the location and size of the grid are required to be adjusted accordingly so that the new population will be enclosed within it. The construction of the adaptive grid environment in GrEA is based on the concepts drawn from the adaptive genetic algorithm (AGA) proposed by Knowles and Corne [175].

In the grid environment, every individual has a grid location which needs to be calculated using the lower and upper boundaries of the grid in every objective. The below formulae from [173] demonstrates how the boundaries are being calculated for a population P:

$$\mathbf{lb_M} = min_M(P) - (max_M(P) - min_M(P))/(2 \times div) \tag{4.1}$$

$$\mathbf{ub_M} = max_M(P) + (max_M(P) - min_M(P))/(2 \times div) \tag{4.2}$$

In the formulae, lb_M and ub_M represent the lower and upper grid boundaries of the Mth objective respectively, and $min_M(P)$ and $max_M(P)$ denote the the minimum and maximum values for the Mth objective among the individuals of a population P. For div, it represents the number of divisions of the objective space in every dimension. From here, the objective space of M dimension will be segregated into div^M hyperboxes, and the width w_M of each hyperbox is calculated as follows:

$$\mathbf{w_M} = (ub_M - lb_M)/div \tag{4.3}$$

where lb_M and ub_M represent the lower and upper grid boundaries of the Mth objective respectively. With equation 4.3, the grid coordinate (location) of an individual in the Mth objective $G_M(x)$ is formulated as follows:

$$\mathbf{G_M}(\mathbf{x}) = \lfloor F_M(x) - lb_M / w_M \rfloor \tag{4.4}$$

where $\lfloor \bullet \rfloor$ represents the floor function, and $F_M(x)$ is the objective function value of the individual x.

With the grid coordinate parameter being described for an individual x, this leads to the explanation of the following two concepts from [173] that are required to be used for the comparison between different individuals.

1. Definition of Grid Dominance:

Let $x, y \in P$ where P is the population which is used for the construction of the grid environment. The individual x is grid-dominated by another individual y if and only if $G_i(x) \leq G_i(y)$ for all $i \in (1, 2, ..., M)$ and $G_j(x) < G_j(y)$ for some $j \in (1, 2, ..., M)$ whereby M is the number of objectives. There is similarity between Pareto dominance and Grid dominance if the actual objective values of the individuals are being replaced by their grid coordinates. The relationship between Pareto dominance and Grid dominance is as described. If a solution X dominates solution Y in terms of Pareto dominance, Y will not dominate X in terms of Grid dominance, and vice versa. However, solution X is said to grid-dominate solution Y if X is slightly inferior to Y in some objectives but much better than Y in all the other objectives.

Hence grid dominance is a relaxed form of Pareto dominance relation. The degree of relaxation is dependent on the number of divisions div which is a fixed parameter set by the user beforehand, and the setting will lead to convergence and diversity requirements to be adjusted adaptively with the evolution of the population. If the population is widely distributed in the objective space, which is often the case at the initial evolution stage, the relaxation degree will

be larger due to a larger size of a cell in the grid environment and this leads to higher selection pressure to be provided. However, as the population evolves towards a more concentrated Pareto front region, the relaxation degree will become smaller, and this will lead to the emphasis on diversity instead.

2. Definition of Grid Difference:

Let $x, y \in P$, and the grid difference between them can be represented as follows:

$$\mathbf{GD}(\mathbf{x}, \mathbf{y}) = \sum_{\mathbf{M=1}}^{\mathbf{N}} |\mathbf{G_M}(\mathbf{x}) - \mathbf{G_M}(\mathbf{y})| \tag{4.5}$$

The number of divisions div has an impact on the grid difference. When div is larger, the size of a cell in the grid becomes smaller which leads to a higher grid difference value between individuals.

4.4.2 Grid-based Criteria for Fitness Assignment

The proposed algorithm in this chapter, GrDE, adopts the grid-based criteria from the algorithm GrEA [173] for the purpose of fitness assignment for the individuals. In order to evolve the population towards the optimum and encourage diversity of solutions along the obtained trade-off surface, the fitness of individuals should contain information in terms of both convergence and diversity. To achieve this, three grid-based criteria from [173], which are namely grid ranking $\mathbf{GR}$, grid crowding distance $\mathbf{GCD}$, and grid coordinate point distance $\mathbf{GCPD}$, will be used for the fitness assignment of individuals. GR and $GCPD$ are meant for the evaluation of the convergence of individuals whereas GCD will be used for gauging the diversity of individuals in the population.

The definition of GR is as described:

$$\mathbf{GR}(\mathbf{x}) = \sum_{M=1}^{N} G_M(x) \tag{4.6}$$

where $G_M(x)$ represents the grid coordinate of individual x in the Mth objective, and N is the total number of objectives.

GR can be used to provide rankings of individuals based on their grid locations. If a solution performs better than other solutions in most of the objectives, it would have a higher possibility of attaining a lower GR value. However, the GR value of an individual can also be influenced by the difference in a single objective. This is illustrated in Figure 4.1 whereby individual C has a worse GR value compared to individual A (4 versus 3) because the advantage in $f1$ is lesser than the disadvantage in $f2$.

For GCD, it is regarded as a density estimator, and it is defined as follows:

$$\mathbf{GCD}(\mathbf{x}) = \sum_{y \in N(x)} M - GD(x, y) \tag{4.7}$$

where M is the number of objectives and $N(x)$ represents the set of neighbours of x. Density estimation of solutions is also of high importance in the fitness assignment process as a good distribution of solutions can direct the search towards the entire Pareto front. In most of the existing grid-based density estimation methods, the number of solutions is being tracked in a single hyperbox, and this leads to a limitation as the distribution of solutions is not clear when the number of hyperboxes increases exponentially especially for the case of many-objective problems. To overcome this limitation, the region for the solutions under consideration is being expanded, and the distribution of the neighbours of a solution is also being considered in the density estimation here. Hence in the context here, a solution y is regarded as a neighbour of solution x if the grid difference between them is less than the number of objectives M. Figure 4.1 gives an illustration of how the GCD is being calculated. In the figure, the neighbours of an individual C are B and D, and the GCD of C is calculated to be (2-1) + (2-1) = 2.

In certain situations, the GR and GCD may not be able to provide discrimination between individuals despite them being able to give a good measure of convergence and diversity between

individuals. This is because GR and GCD are of integer values as the computation of GR and GCD are based on the grid coordinates of individuals. This in turn means that some individuals may possess similar GR and GCD values as seen in an example shown in Figure 4.1 whereby individuals B and D have the same GR and GCD values. Therefore in such cases, it would not possible to differentiate them based on GR and GCD values. To overcome this, another metric termed as grid coordinate point distance **GCPD** is formulated as an additional method of differentiating individuals in the grid environment. The $GCPD$ metric is defined as follows:

$$\mathbf{GCPD}(\mathbf{x}) = \sqrt{\sum_{M=1}^{N} (\frac{F_M(x) - (lb_M + G_M(x) \times w_M)}{w_M})^2} \tag{4.8}$$

where $F_M(x)$ and $G_M(x)$ represent the objective function value and grid coordinate (location) of the individual x respectively in the Mth objective, and lb_M and w_M denote the lower bound of the grid and the hyperbox width respectively in the Mth objective, and N is the total number of objectives. For the case of $GCPD$ metric, a lower value is preferred. The illustration of the $GCPD$ criterion is as shown in Figure 4.1 with the use of individuals E and F.

These three grid-based criteria GR, GCD and $GCPD$ as described in this section will be used for the comparison of individuals during the selection process in the GrDE algorithm as they are able to provide indication of how well the population individuals have evolved during the optimization process.

4.5 Development of Proposed Algorithm GrDE

In this section, the development of the proposed grid-based differential evolution GrDE is discussed. GrDE adopts the basic procedures found in NSGA-II [50] but the differences lie in the fitness assignment and the mating and environmental selection processes, as well as the reproduction stage whereby a novel differential evolution operator is used.

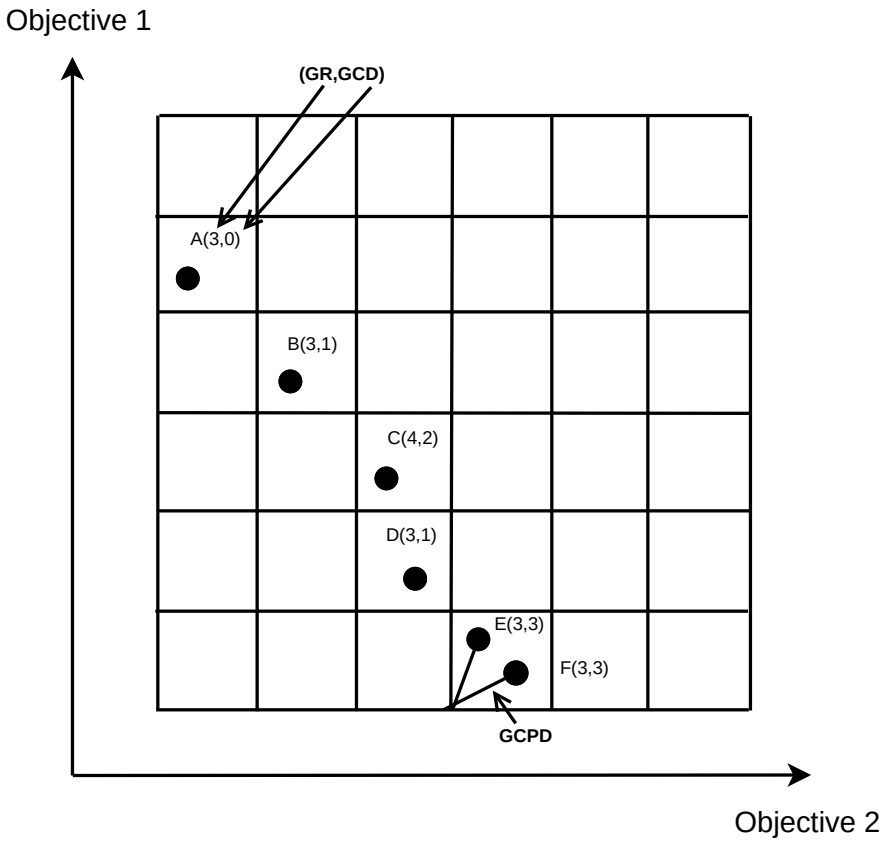

Figure 4.1: Grid fitness assignment using GR, GCD and GCPD

4.5.1 Mating Selection

The mating selection in GrDE is executed using a binary tournament selection strategy. For this process, two different individuals are first selected from the current population on a random basis. These two individuals are compared in terms of both Pareto and grid dominance. The individual that Pareto-dominates or grid-dominates the other one will be chosen to enter the mating pool for reproduction. However if both individuals are non-dominated to each other in terms of both Pareto and grid dominance, the GCD metric will be employed to compare the two individuals, and the individual with a lower GCD value will be chosen. If there is still a tie between the two individuals when their GCD values are compared, then one of them will be randomly selected for the mating pool.

4.5.2 Novel Mutation Strategy in GrDE

In this study, a novel mutation strategy is formulated for the reproduction stage in GrDE. For this, the opposition-based self-adaptive differential evolution operator from OSADE is extended by incorporating it with a modified local mutation scheme that originated from RDEL [183] so as to enhance the local search tendency and to accelerate the convergence of the differential evolution technique. In the local mutation scheme of RDEL, the best and worst individuals of the population of every generation are required to be identified. However, RDEL is catered towards single-objective optimization and hence the way to identify the best and worst individuals will not be applicable in the context of multi-objective optimization. As every individual in GrDE is being assigned GR, GCD and $GCPD$ values, the values of these metric will be used for the comparison of all the individuals so as to identify the best and worst individuals. The comparison processes to identify the best and worst individuals in a generation is as explained in Figure 4.2 and Figure 4.3.

Require: Define best solution as b in Population P, and every j-*th* individual as p_j
Begin
1: **Initialization**: Let $b = p_1$
2: **for** j = 2 to |P| **do**
3: **if** GR(p_j) < GR(b) **then**
4: $b = p_j$
5: **else if** GR(p_j) = GR(b) **then**
6: **if** GCD(p_j) < GCD(b) **then**
7: $b = p_j$
8: **else if** GCD(p_j) = GCD(b) **then**
9: **if** GCPD(p_j) < GCPD(b) **then**
10: $b = p_j$
11: **end if**
12: **end if**
13: **end if**
14: **end for**
15: **return** b
End

Figure 4.2: Pseudo-code of function in GrDE for finding best individual

The novel mutation strategy is made up of an opposition-based self-adaptive mutation scheme based on DE/rand/1 [71] and a local mutation scheme with opposition-based learning [30] incorporated. One of them will be selected for the mutation of an individual as decided by a linear decreasing probability rule.

The opposition-based self-adaptive mutation scheme in GrDE as based on OSADE is as

```
Require: Define worst solution as w in Population P, and every j-th individual as p_j
Begin
1:  Initialization: Let w = p_1
2:  for j = 2 to |P| do
3:     if GR(p_j) > GR(w) then
4:        w = p_j
5:     else if GR(p_j) = GR(w) then
6:        if GCD(p_j) > GCD(w) then
7:           w = p_j
8:        else if GCD(p_j) = GCD(w) then
9:           if GCPD(p_j) < GCPD(w) then
10:             w = p_j
11:          end if
12:       end if
13:    end if
14: end for
15: return w
End
```

Figure 4.3: Pseudo-code of function in GrDE for finding worst individual

described. At the start of the parent generation G, three vectors x_{r1}^G, x_{r2}^G and x_{r3}^G are randomly selected for every target vector x_i^G whereby $r1 \neq r2 \neq r3 \neq i$, and $r1, r2, r3 \in \{1, 2, ..., NP\}$ and NP is the size of the population. In GrDE, every individual will be encoded with values of the DE control parameters F and CR that exist as extended variables. These encoded values, which are initialized as zero at the start of every run, are needed in the opposition-based self-adaptive mutation scheme for the self-adaption process of the control parameters.

In the self-adaptation mechanism, the four vectors x_i^G, x_{r1}^G, x_{r2}^G and x_{r3}^G will first be compared with each other in terms of their GR, GCD and $GCPD$ values. This approach is slightly different from the original scheme in OSADE whereby the comparison is based on Pareto ranks and niche counts. Through the comparison, the best individual ranked among the four in terms of the lowest GR and/or GCD or $GCPD$ will be given the highest weight factor ω of 0.4 for its contribution towards the calculation of the current average values of F and CR. The weight factors assigned to the subsequent ranked individuals will be decremented by 0.1. The encoded values of F and CR in all the four vectors will be used for the calculation of the current average values of F and CR for the parent generation G which are denoted by $\langle F_G \rangle_i$ and $\langle CR_G \rangle_i$ and the computation is as follows:

$$\langle F_G \rangle_i = \frac{\omega_1 \times F_{i,G} + \omega_2 \times F_{r1,G} + \omega_3 \times F_{r2,G} + \omega_4 \times F_{r3,G}}{\omega_1 + \omega_2 + \omega_3 + \omega_4} \tag{4.9}$$

$$\langle CR_G \rangle_i = \frac{\omega_1 \times CR_{i,G} + \omega_2 \times CR_{r1,G} + \omega_3 \times CR_{r2,G} + \omega_4 \times CR_{r3,G}}{\omega_1 + \omega_2 + \omega_3 + \omega_4} \tag{4.10}$$

where ω_1, ω_2, ω_3 and ω_4 are the weight factors assigned to the four different individuals x_i^G, x_{r1}^G, x_{r2}^G, and x_{r3}^G . The values of the F and CR for the child generation $G+1$ which are denoted by $\bar{F}_{i,G+1}$ and $\bar{CR}_{i,G+1}$ are calculated by the following formulae as follows:

$$\bar{F}_{i,G+1} = \langle F_G \rangle_i \times e^{\tau N(0,1)} \tag{4.11}$$

$$\bar{CR}_{i,G+1} = \langle CR_G \rangle_i \times e^{\tau N(0,1)} \tag{4.12}$$

where $\tau = \frac{1}{8 \times \sqrt{2D}}$, D is the number of decision variables in the problem, and $N(0,1)$ represents a randomly generated number under Gaussian distribution. Both F and CR are bounded between a predefined set of lower and upper bounds of 0.1 and 0.9 respectively. The Opposite Number [35] of the mutation factor $\bar{F}_{i,G+1}$ is then generated as follows:

$$\bar{F\_opp}_{i,G+1} = F_{upper} + F_{lower} - \bar{F}_{i,G+1} \tag{4.13}$$

where F_{upper} and F_{lower} are the upper and lower bounds for the parameter F.

The local mutation scheme used in GrDE as based on RDEL allows every vector to learn from the position of the best and worst individuals in the population of a particular generation. Through this scheme, the generation of the new position of every mutant vector will be guided towards the same direction of the best individual while avoiding the direction of the worst. In the original local mutation scheme from RDEL, it involves two mutation factors which are randomly generated for the mutant vector generation of the DE algorithm. However, in the local mutation

scheme in GrDE, opposition-based learning is incorporated for the mutation factors. A random value between [0, 1] will be generated for the first mutation factor for the local mutation scheme, and the second mutation factor will be the Opposite Number of the first mutation factor. This is performed as the use of the Opposite Number as based on opposition-based learning may increase the probability of finding an optimal solution, and in the case here, an optimal setting for the mutation factors. The resultant local mutation scheme is as follows:

$$V_{i,G+1} = X_{r1,G} + F \cdot (X_{best,G} - X_{r1,G}) + F_{opp} \cdot (X_{worst,G} - X_{r1,G}) \tag{4.14}$$

The opposition-based self-adaptive mutation scheme is then joined with the local mutation scheme through a linearly decreasing function as presented in Figure 4.4.

If $u(0,1) \geq \left(1 - \frac{G}{GENS}\right)$ *Then*

// Opposition-based Local Mutation Scheme

$V1_{i,G+1} = x_{r1,G} + F1 * \left(x_{best,G} - x_{r1,G}\right) + F1\_opp * \left(x_{r1,G} - x_{worst,G}\right)$

$V2_{i,G+1} = x_{r1,G} + F1\_opp * \left(x_{best,G} - x_{r1,G}\right) + F1 * \left(x_{r1,G} - x_{worst,G}\right)$

Else

//Opposition-based Self-Adaptive Mutation Scheme

$V1_{i,G+1} = x_{r1,G} + F2 * \left(x_{r2,G} - x_{r3,G}\right)$

$V2_{i,G+1} = x_{r1,G} + F2\_opp * \left(x_{r2,G} - x_{r3,G}\right)$

End

Figure 4.4: Pseudo-code of Novel Mutation Strategy in GrDE

In the pseudo code of the novel mutation strategy, the current generation number and the maximum number of generations are represented by G and $GENS$ respectively. $F1$ is a randomly generated number within [0, 1] and it will be used together with its Opposite Number $F1\_opp$ in the local mutation scheme. As for $F2$, it is the mutation factor generated using equation 4.11, and it will also be used together with its Opposition Number $F2\_opp$ in the opposition-based self-adaptive mutation scheme. One of the schemes will be selected using a linear decreasing probability rule, and the selected scheme will generate two different mutant vectors which will be compared against each other. The dominant one in terms of grid and/or Pareto dominance will be chosen to enter the offspring population that will be combined with the parent population for environmental selection.

Lastly, the encoded values of both F and CR in the individual $x_{i,G+1}$ will be updated if the opposition-based self-adaptive mutation scheme is being used. The updating of F will depend on the outcome of the selection of the mutant vector. If the selected mutant vector is the one created using $\bar{F}_{i,G+1}$ as generated by equation 4.11, then the encoded value of F in $x_{i,G+1}$ will be updated with this F value; otherwise the encoded value will be updated with the Opposite Number of $\bar{F}_{i,G+1}$. However the encoded value of CR in $x_{i,G+1}$ will take on the value of $\bar{CR}_{i,G+1}$ as generated by equation 4.12 regardless of which mutant vector is being selected as a trial vector for the next generation.

4.5.3 Environmental Selection

In GrDE, the environmental selection process is based on the one found in GrEA [173]. The process starts by partitioning the combined population, which is made up of the previous parent population and a newly created child population, into different fronts $(F_1, F_2, ..., F_i)$ by using the fast non-dominated sorting technique similar to the one found in NSGA-II. Likewise to NSGA-II, GrDE also identifies the critical front which is often the first front in many-objective problems due to the fact that many solutions in these problems are non-dominated to each other. The fitter solutions are then selected to enter the parent population of the next generation based on their fitness values. However, this may lead to a loss in diversity due to the fact that adjacent solutions usually have similar fitness values. Hence there is a high chance of adjacent solutions being eliminated or preserved together. Due to this shortcoming, a GR adjustment mechanism is included in the environmental selection process of GrEA, and this will also be followed in GrDE. The aim of the GR adjustment is to achieve a good balance of convergence and diversity in the archive. The mechanism will be elaborated in a later section of this chapter.

Upon the completion of the segregation of the candidate solutions in the population different fronts, the individuals in the first non-dominated fronts starting F_1 will be updated into the archive. Once the updating is completed, the grid-related information of the individuals will be

initialized, whereby the fitness of the individuals in terms of GR, GCD and $GCPD$ will be calculated. Calculation of GR and $GCPD$ will be done using Equations 4.6-4.7 and they will be assigned to the individuals. As for the GCD value for an individual, the calculation has to be performed by taking into account the crowding relation among the individuals in the archive set. As such, the GCD value is first initialized to zero before performing the re-calculation. Next the best individual of the considered front will be identified by comparing the three criteria GR, GCD and $GCPD$ among the individuals in the front. A lower value in all the criteria is preferred. Lastly, GR adjustment will be performed on some of the adjacent individuals of those individuals that are being picked to enter the archive.

The GR adjustment mentioned earlier will be done by penalizing three groups of individuals that are related to the picked individuals using different degrees of punishment. For the group of individuals that possess the same grid coordinate as that of the picked individuals, their GR values will be increased by $M + 2$ where M is the number of objectives of the optimization problem. For a second group of individuals which are being grid-dominated by the picked individual, their GR will be also increased by M. The final group contains individuals which are not grid-dominated by the picked individual and also having a different grid coordinate from the picked individual. The GR of these individuals will be increased by a value between 0 to $M - 1$. For this, a neighbour individual y of a picked individual x is imposed a punishment of having its GR increased by $(M - GD(x, y)$ where $GD(x, y)$ is the grid difference between individual x and individual y. The individuals that are grid-dominated by the neighbour individual y will also be imposed the same punishment as that of y. This is to prevent these individuals from being archived earlier than competing individuals that are fitter than them in the aspect of grid dominance.

4.5.4 Algorithmic Framework

The GrDE algorithm starts with the creation of an initial population through random generation of N solutions. A grid environment as described in Section 4.4.1 is then set up for the current population $Pop(G)$, and the individuals in $Pop(G)$ will undergo fitness assignment based on the three grid-based criteria as mentioned in Section 4.4.2. Next, mating selection is executed to identify promising individuals to undergo reproduction using the novel DE mutation strategy in GrDE to generate offspring. This is followed by the environmental selection process to pick N best solutions through elitism for survival to the next generation. The process is iterated until the stopping criterion is achieved. The pseudo code of GrDE is as explained in Figure 4.5.

Begin
1. Initialization: At generation g = 0, randomly generate N solutions as the initial population, *Pop(g)*
2. Evaluation: Evaluate all solutions in *Pop(g)*.
Do while ("Stopping criterion is not met")
3. Grid Setting: Construct the grid environment for *Pop(g)*.
4. Fitness Assignment: Assign fitness to every solution in *Pop(g)* in terms of GR, GCD and GCPD values.
5. Mating Selection: Select *N* parent solutions using binary tournament selection.
6. Reproduction: Identify the best and worst individuals of the current population. Use the novel mutation strategy to create new offspring.
7. Evaluation: Evaluate all offspring and store them in an archive *A*.
8. Archiving: Combine the parent solutions and all the offspring *Pop(g)* ∪ *A*.
9. Environmental Selection: Partition the combined population *Pop(g)* ∪ *A* into different fronts using the fast non-dominated sorting approach and identify the critical front to be archived. Select *N* best solutions from the critical front according to their fitness in terms of GR, GCD and GCPD values to form a new population *Pop(g+1)*. Impose penalty in terms of GR adjustment to the "related" individuals of selected individuals. g = g +1.
End Do
10. Ouput: Output the final set of solutions *Pop(g)*
End

Figure 4.5: Pseudo-code of GrDE

4.6 Implementation

4.6.1 Test Problems

A total of 62 benchmark test instances from the DTLZ and WFG test suites as mentioned in Section 2.8 were chosen to test the optimization performance of the proposed algorithms in terms of converging to the true Pareto front as well as its ability in maintaining a set of diverse solutions in a higher objective space. The selected test instances are also scalable to any number of objectives

and decision variables.

In the DTLZ test suite, DTLZ2, DTLZ4, DTLZ5 and DTLZ7 are able to challenge the ability of an evolutionary algorithm in handling problems with different shapes and locations. As for DTLZ1, DTLZ3 and DTLZ6, they are able to pose more difficulties for algorithms in terms of converging to the Pareto front due to their characteristics like multi-modality. As for the WFG test suite, the problems involve different transformation types like bias or shift transformations which are not found in other test suites, and they can also create challenges for evolutionary algorithms.

4.6.2 Performance Metrics

In this study, Inverted Generational Distance (IGD) [136] and Hypervolume (HV) [3] were chosen as the performance metrics for the assessment of the optimization performance achieved by the algorithms. IGD is a unary indicator whereby the distance of every solution in a reference optimal Pareto front to the obtained Pareto front is being calculated, and it will be used for the evaluation of the performance of the algorithms on the DTLZ problems as their optimal fronts are known. A lower IGD value will indicate better performance. HV will be used to assess the performance of the algorithms on the WFG problems as their Pareto front is unknown. For HV calculation, the scaling of the search space and the choice of the reference point are two important issues to be considered. As the objectives in the WFG problems have different ranges of values, the objective values of the obtained solutions for the WFG problems were normalized according to the range of the obtained Pareto front of the problem. As for the reference point r, it was set to be at 1.1 times of the upper bound of the Pareto front as per the recommendation in [184] so as to emphasize the balance between convergence and diversity of the obtained solution set. For the case of HV metric, a higher value attained will mean that better performance is being achieved by the algorithm.

4.6.3 Algorithms Under Comparison

For this study, five state-of-the-art EMO algorithms were chosen for performance comparison with the proposed algorithm GrDE. The ϵ-dominance based Multiobjective Evolutionary Algorithm (ϵ-MOEA) [174] is a steady-state algorithm that exploits the ϵ-dominance relation to enhance selection pressure towards the Pareto front. The algorithm basically segregates the objective space into several hyperboxes of size ϵ. Each hyperbox is allocated only a single solution based on ϵ-dominance as well as the distance from different solutions to the utopia point in the hyperbox. From some works, it is seen that ϵ-MOEA is able to display good performance on many-objective problems [167, 185] despite not being specifically designed for them.

Hypervolume Estimation Algorithm (HypE) [186] is an indicator-based algorithm for many-objective optimization. Monte Carlo simulation is being used in this algorithm for the approximation of the hypervolume value (HV), and this leads to significant reduction in the amount of time required for HV calculation. This in turn enables hypervolume-based search to be readily applied on many-objective problems.

Multiple Single Objective Pareto Sampling (MSOPS) [187] is an aggregation-based approach that adopts the idea of single-objective aggregation optimization to perform parallel search for points that exist on the Pareto front. As MSOPS is able to strike a good balance between convergence and diversity, this makes it a popular algorithm to be considered for many-objective optimization problems [167].

Multiobjective Evolutionary Algorithm based on Decomposition (MOEA/D) [52] is a popular aggregation-based EMO algorithm that basically converts a multiobjective problem into several single-objective problems that are to be handled simultaneously. MOEA/D is found to display good performance in both multi-objective [53] and many-objective [188] optimization problems.

Grid-based Evolutionary Algorithm (GrEA) [173] is a recently proposed algorithm catered towards many-objective optimization. The basic procedures in GrEA are similar to the ones

found in NSGA-II, but it exploits the potential of grid-based approach to strengthen the selection pressure towards the optimal direction while maintaining a uniform distribution among solutions. As such, the concepts of grid dominance and grid difference are introduced for the determination of the mutual relationship of individuals in grid environment. Three grid-based relations, namely grid ranking, grid crowding distance and grid coordinate point distance, are incorporated into the fitness of individuals so as to distinguish them during the selection processes. GrEA is seen to be competitive with other state-of-the-art algorithms in terms of finding a well-approximated and well-distributed solution set when tested on some many-objective optimization problems.

4.6.4 Experimental Settings

Comparative studies of the proposed algorithm GrDE with the other algorithms mentioned in Section 4.6.3 were carried out to examine their performance in the test problems. The codes for GrDE, GrEA and MOEA/D were implemented in C++, while the codes for ϵ-MOEA, HypE and MSOPS were implemented in C. The simulations were performed on an Intel(R) Core(TM) i3-2100T CPU, 2.5GHz PC. Table 4.1 provides the summary of the general experimental settings for all the algorithms used in this study. The number of fitness evaluations for the DTLZ test suite followed the similar settings as found in [173] so as to ensure fair comparison for the algorithms. As for the WFG test suite, the number of fitness evaluations followed the recommendations from [189]. The additional settings in terms of the number of grid divisions div in GrDE and GrEA, as well as the ϵ values for the ϵ-MOEA are stated in Table 4.2 and Table 4.3 for the DTLZ and WFG problems respectively according to the number of objectives M in the problem.

Table 4.1: Parameter settings

Parameter	Settings
Population size	100 for all algorithms except MOEA/D and ϵ-MOEA
Population size (MOEA/D)	120 for 4 objectives, 126 for 5 objectives, 126 for 6 objectives, 84 for 7 objectives, 120 for 8 objectives, and 55 for 10 objectives
Population size (ϵ-MOEA)	Determined by the ϵ value, and the value of ϵ is set accordingly so that the population size is close to 100
Stopping criterion	30000 evaluations DTLZ2, DTLZ4, DTLZ5, and DTLZ7 100000 evaluations for DTLZ1, DTLZ3, and DTLZ6 50000 evaluations for all WFG problems
Number of independent runs	30
Number of objectives for DTLZ problems	4, 5, 6, 8, 10
Number of objectives for WFG problems	4, 5, 7
Number of decision variables for DTLZ problems	$n = m + K - 1$, where m is the number of objectives. K is 5 for DTLZ1, 10 for DTLZ2 to DTLZ6, and 20 for DTLZ7
Number of decision variables for WFG problems	$(2m - 1) + 20$ where m is the number of objectives.
Crossover probability	1.0
Mutation probability	1/n (where n denotes the number of decision variables
Distribution index in SBX	20
Distribution index in polynomial mutation	20
Number of weight vectors in MSOPS	100
Scalarizing function in MOEA/D	Tchebycheff function
Neighbourhood size for MOEA/D	10% of population size
Initial encoded F and CR values for vectors in GrDE	0
Lower Bound of F and CR in GrDE	0.1
Upper Bound of F and CR in GrDE	0.9

Table 4.2: Additional settings for GrDE, GrEA and ϵ-MOEA for DTLZ suite with different objectives M

	M=4		M=5		M=6		M=8		M=10	
Problem	div	ϵ	div	ϵ	div	ϵ	div	ϵ	div	ϵ
DTLZ1	10	0.0520	10	0.0590	10	0.0554	10	0.0549	11	0.0565
DTLZ2	10	0.1312	9	0.1927	8	0.2340	7	0.2900	8	0.3080
DTLZ3	11	0.1385	11	0.2000	11	0.2270	10	0.1567	11	0.8500
DTLZ4	10	0.1312	9	0.1927	8	0.2340	7	0.2900	8	0.3080
DTLZ5	35	0.0420	29	0.0785	14	0.1100	11	0.1272	11	0.1288
DTLZ6	36	0.1200	24	0.3552	50	0.7500	50	1.1500	50	1.4500
DTLZ7	9	0.1050	8	0.1580	6	0.1500	5	0.2250	4	0.5600

Table 4.3: Additional settings for GrDE, GrEA and ϵ-MOEA for WFG suite with different objectives M

	M=4		M=5		M=7	
Problem	div	ϵ	div	ϵ	div	ϵ
WFG1	9	0.0920	9	0.0950	9	0.0980
WFG2	9	0.5250	9	0.5550	9	0.5950
WFG3	9	0.9275	9	0.9560	9	0.9850
WFG4	9	1.5250	9	1.7250	9	1.8750
WFG5	9	1.3450	9	1.5655	9	1.7550
WFG6	9	1.5465	9	1.6540	9	1.7550
WFG7	9	1.2535	9	1.4355	9	1.6545
WFG8	9	1.6550	9	1.7650	9	1.8545
WFG9	9	1.8335	9	1.9550	9	2.0545

4.7 Simulation Results and Discussions

4.7.1 Performance Comparison for DTLZ problems

Comparative studies were conducted for the performance evaluation of the six algorithms under a comprehensive suite of DTLZ test functions. The DTLZ problems created by Deb *et al.* [138] are

scalable to any number of objectives and decision variables. For this study, the DTLZ problems used contain four, five, six, eight or ten objective functions. The number of decision variables in these problems was set accordingly to the number of objectives as shown in Table 4.1. Simulation results in terms of the measurement of the mean values of the Inverted Generational Distance (IGD) are presented in Table 4.4 according to test problems. The best entries in terms of mean values are also marked in boldface. In order to judge whether the results of the best performing algorithm differ from the results of competitors in a statistically significant way, the Wilcoxon ranksum test [190] was conducted at the 5% significance level. The entries which are significantly different from the best entries are indicated by the symbol †.

DTLZ2 is a problem that comes with a spherical Pareto front, and it might be considered as a relatively easy test problem. However, this test problem can be used to investigate the ability of a multi-objective evolutionary algorithm (MOEA) in scaling up its performance in a large number of objectives. As seen from the IGD results, the proposed algorithm GrDE achieves the best performance among all the other algorithms in this study for this problem with respect to all the considered number of objectives. DTLZ4 is a modified DTLZ2 with a different meta-variable mapping, and is used to investigate the ability of an MOEA in maintaining a good distribution of solutions. Once again, GrDE outperforms all the other algorithms for DTLZ4 for all considered number of objectives.

For DTLZ1 and DTLZ3, these two problems are highly multi-modal, and they bring forth a stiff challenge for the algorithms to find the global optimal front. From the IGD results, it is clear that GrDE generally outperforms all the other algorithms for most of the DTLZ1 problems for all considered number of objectives except 10. As for DTLZ3, this problem is characterized by the presence of a vast number of local optima, and algorithms face the challenge of preventing themselves from getting trapped in the local optima. Again, GrDE is able to achieve the best IGD values in DTLZ3 test instances with four, five and six objectives. For the case of DTLZ3 with eight objectives, GrEA achieves the best performance while MOEA/D outperforms the rest

for both DTLZ1 and DTLZ3 with 10 objectives.

In both DTLZ5 and DTLZ6, these problems possess degenerate Pareto optimal fronts. DTLZ5 tests the ability of an MOEA in finding a lower-dimensional Pareto front while working with a higher-dimensional objective space. For this problem, GrEA achieves the best performance on the problem with four objectives, but MSOPS takes the overall lead by attaining the best performance for this problem with respect to all the other considered number of objectives. However, GrDE is unable to perform as well as the other algorithms for the case of the DTLZ5 problem, and this suggests that the DE operator may not be that effective in handling problems with degenerate Pareto optimal front and therefore this could be a potential weakness of DE. On the other hand for DTLZ6, GrDE achieves good performance for this problem with four and five objectives. However, its performance deteriorates as the number of objectives increased. Instead, MOEA/D emerges as the top performing algorithm for this problem with six, eight and 10 objectives.

Lastly for DTLZ7, this problem is characterized by having a large number of disconnected Pareto-optimal regions in the search space, and challenges the ability of an MOEA in maintaining sub-populations in different Pareto-optimal regions. For this problem, GrDE generally achieves better, if not comparable performance over HypE, MSOPS and MOEA/D. It also performs better than ϵ-MOEA for this problem with five and 10 objectives. However, GrEA still emerges as the overall best performing algorithm for DTLZ7 with respect to all the considered number of objectives.

In order to explore further on the convergence performance of all the algorithms used in this study, the evolutionary trajectories of the average IGD for the six algorithms on the DTLZ2 and DTLZ4 problems with 10 objectives are plotted as shown in Figures 4.6-4.7.

As seen from both the evolutionary trajectories, it is clear that GrDE achieves better performance than the other five algorithms. In the plot for DTLZ2, it is observed that both the ϵ-MOEA and GrEA achieve fast convergence during the initial evolutionary stage, but GrDE outperforms

Table 4.4: Results in terms of IGD measurement for DTLZ problems

Algorithm	Test Instance (m, n)				
	DTLZ1(4,8)	DTLZ1(5,9)	DTLZ1(6,10)	DTLZ1(8,12)	DTLZ1(10,14)
ϵ-MOEA	4.85E2±2.4E-3†	6.95E2±8.3E-3†	9.68E2±8.5E-2†	3.17E1±3.2E-1†	4.08E2±3.7E-1†
HypE	1.46E1±7.4E-3†	3.25E+1±4.2E-1†	5.47E1±6.1E-1†	1.15E+0±1.6E+0†	1.58E+0±1.6E+0†
MSOPS	5.82E2±3.0E-3†	8.57E2±3.6E-3†	1.85E1±1.2E-1†	7.82E1±7.0E-1†	1.65E+0±1.0E+0†
MOEA/D	9.63E2±1.1E-4†	1.21E1±1.5E-4†	1.39E1±6.2E-3†	1.85E1±5.6E-3†	**2.23E1±6.4E-3**
GrEA	4.65E2±5.2E-3†	6.36E2±7.6E-3†	8.57E2±1.2E-2†	1.09E1±5.0E-3	2.92E1±1.0E-1†
GrDE	**3.84E2±1.1E-3**	**5.32E2±4.1E-3**	**6.87E2±3.3E-3**	**1.07E1±1.1E-2**	3.26E1±1.7E-1†
Algorithm	DTLZ2(4,13)	DTLZ2(5,14)	DTLZ2(6,15)	DTLZ2(8,17)	DTLZ2(10,19)
ϵ-MOEA	1.36E1±2.6E-3†	1.98E1±1.4E-2†	3.12E1±6.6E-3†	4.51E1±1.5E-2†	5.42E1±2.3E-2†
HypE	2.47E1±3.8E-2†	4.26E1±7.3E-2†	4.81E1±5.4E-2†	6.12E1±5.0E-2†	6.89E1±5.8E-2†
MSOPS	1.77E1±1.4E-2†	3.64E1±2.6E-2†	3.71E1±2.0E-2†	5.78E1±3.4E-2†	7.67E1±4.2E-2†
MOEA/D	2.15E1±2.0E-3†	2.69E1±1.0E-3†	4.17E1±1.8E-2†	6.34E1±7.2E-2†	7.32E1±6.6E-2†
GrEA	1.29E1±2.3E-3†	1.76E1±2.9E-3†	2.96E1±5.4E-3†	3.93E1±4.8E-3†	4.88E1±2.8E-3†
GrDE	**1.08E1±1.4E-3**	**1.56E1±4.4E-3**	**1.96E1±9.9E-3**	**3.22E1±2.2E-2**	**2.60E1±2.1E-2**
Algorithm	DTLZ3(4,13)	DTLZ3(5,14)	DTLZ3(6,15)	DTLZ3(8,17)	DTLZ3(10,19)
ϵ-MOEA	1.42E1±8.8E-3†	2.27E1±3.5E-2†	4.68E1±1.2E-1†	1.21E1±1.4E+3†	2.05E1±2.4E-3†
HypE	1.04E+0±7.2E-1†	4.75E+0±5.4E+0†	2.66E+0±1.8E+0†	7.45E+0±9.6E+0†	6.24E+0±6.0E+0†
MSOPS	1.07E+1±6.7E+0†	2.91E+1±1.4E+1†	4.67E+1±1.6E+1†	6.12E+1±1.8E+1†	6.38E+1±2.0E+1†
MOEA/D	2.15E1±1.8E-3†	2.63E1±8.0E-4†	4.11E1±3.4E-2†	6.14E1±7.6E-2	**6.36E1±5.4E-2**
GrEA	1.54E1±4.6E-2†	2.84E1±8.1E-2†	4.42E1±1.6E-1†	**5.56E1±2.2E-1**	7.72E1±2.6E-1†
GrDE	**1.09E1±2.4E-3**	**1.51E1±2.4E-3**	**4.02E1±2.4E-3**	6.05E1±2.4E-3	7.65E1±2.4E-3†
Algorithm	DTLZ4(4,13)	DTLZ4(5,14)	DTLZ4(6,15)	DTLZ4(8,17)	DTLZ4(10,19)
ϵ-MOEA	4.17E1±2.6E-1†	6.35E1±3.4E-1†	6.05E1±1.8E-1†	6.52E1±1.2E-1†	6.29E1±9.4E-2†
HypE	4.96E1±3.4E-1†	7.03E1±2.7E-1†	6.72E1±1.2E-1†	9.31E1±6.3E-2†	1.12E+0±6.2E-2†
MSOPS	1.45E1±4.5E-3†	3.24E1±3.4E-2†	3.79E1±1.2E-2†	5.52E1±2.6E-2†	8.23E1±4.3E-2†
MOEA/D	5.27E1±2.6E-1†	5.75E1±3.2E-1†	6.48E1±1.4E-1†	7.58E1±8.5E-2†	8.34E1±8.3E-2†
GrEA	1.93E1±1.2E-1†	2.18E1±9.6E-2†	3.01E1±4.8E-3†	4.05E1±3.0E-3†	4.94E1±2.8E-3†
GrDE	**1.09E1±1.6E-3**	**1.51E1±4.8E-3**	**1.97E1±9.3E-3**	**2.71E1±1.2E-2**	**2.42E1±1.9E-2**
Algorithm	DTLZ5(4,13)	DTLZ5(5,14)	DTLZ5(6,15)	DTLZ5(8,17)	DTLZ5(10,19)
ϵ-MOEA	4.83E2±5.4E-3†	8.97E2±7.8E-3†	1.30E1±1.3E-2†	1.61E1±2.2E-2†	1.72E1±2.3E-2†
HypE	1.23E1±4.2E-2†	1.53E1±5.5E-2†	1.76E1±6.0E-2†	1.78E1±6.8E-1†	1.57E1±4.9E-2†
MSOPS	3.02E2±3.2E-3†	**3.00E2±4.4E-3**	**1.88E2±1.8E-3**	**2.56E2±2.6E-3**	**4.07E2±4.1E-3**
MOEA/D	2.66E2±1.2E-4†	4.65E2±1.4E-3†	6.98E2±4.5E-3†	1.09E1±7.6E-3†	1.97E1±1.2E-2†
GrEA	**1.83E2±3.4E-3**	4.36E2±2.1E-2†	9.62E2±1.5E-2†	2.35E1±3.6E-2†	3.48E1±6.0E-2†
GrDE	3.73E2±2.0E-3†	1.18E1±6.8E-2†	1.81E1±2.5E-2†	3.41E1±7.2E-2†	7.58E1±5.9E-2†
Algorithm	DTLZ6(4,13)	DTLZ6(5,14)	DTLZ6(6,15)	DTLZ6(8,17)	DTLZ6(10,19)
ϵ-MOEA	4.65E1±2.6E-2†	1.69E+0±1.7E-1†	2.72E+0±2.9E-1†	1.96E+0±1.4E+0†	3.75E+0±2.2E+0†
HypE	2.24E+0±3.0E-1†	1.82E+0±4.3E-1†	2.26E+0±6.2E-1†	5.82E+0±4.0E+0†	8.96E+0±1.6E-1†
MSOPS	4.21E+0±6.3E-1†	6.61E+0±5.3E-1†	6.83E+0±5.4E-1†	6.79E+0±4.3E-1†	6.71E+0±4.8E-1†
MOEA/D	8.11E2±2.8E-2†	1.22E1±3.6E-2†	**1.58E1±3.5E-2**	**1.87E1±2.6E-2**	**2.73E1±3.0E-2**
GrEA	7.05E2±3.2E-2†	1.46E1±4.2E-2†	4.55E1±9.0E-2†	5.89E1±3.8E-1†	9.38E1±7.6E-1†
GrDE	**3.70E2±4.3E-3**	**1.11E1±1.1E-2**	7.25E1±3.3E-2†	7.45E1±1.4E-2†	1.21E+0±2.7E-3†
Algorithm	DTLZ7(4,23)	DTLZ7(5,24)	DTLZ7(6,25)	DTLZ7(8,27)	DTLZ7(10,29)
ϵ-MOEA	3.48E1±1.7E-1†	6.32E1±2.2E-1†	5.88E1±2.0E-1†	8.94E1±5.2E-1†	1.20E+0±3.8E-1†
HypE	4.87E1±1.8E-1†	8.92E1±2.0E-1†	9.91E1±1.7E-1†	1.06E+0±4.3E-2†	1.23E+0±8.2E-2†
MSOPS	1.56E+0±4.2E-1†	7.60E+0±1.4E+0†	1.07E+1±2.6E+0†	1.97E+1±2.0E+0†	2.69E+1±3.3E+0†
MOEA/D	5.15E1±7.3E-2†	6.46E1±8.6E-2†	7.57E1±6.2E-2†	1.05E+0±1.4E-1†	1.58E+0±2.1E-1†
GrEA	**1.87E1±6.4E-3**	**3.25E1±1.3E-2**	**4.86E1±1.5E-2**	**7.66E1±3.6E-2**	**1.08E+0±3.7E-2**
GrDE	3.58E1±1.2E-2†	4.68E1±5.8E-2†	6.08E1±6.7E-2†	1.10E+0±2.9E-2†	1.12E+0±2.2E-2

them with an even faster initial convergence, and having a clear advantage of having lower IGD values than these two algorithms till the end of 30,000 evaluations. As for the case of DTLZ4, the ϵ-MOEA comes close to GrDE in terms of the convergence rate at the initial stage, but GrDE outperforms it as well as the other competing algorithms with lower IGD values being attained throughout the remaining stages of the evolutionary process.

In summary, GrDE achieves the lowest mean IGD values for 19 out of the 35 DTLZ test

instances, and significantly outperforms all the other algorithms in 17 of these instances according to statistical test. The good performance achieved by GrDE in DTLZ problems may be attributed to the ability of the mutation strategies in the proposed algorithms in striking a good balance between exploration and exploitation. Through the linearly decreasing probability rule in the mutation strategy, the opposition-based self-adaptive mutation scheme, which is based on the DE/rand/1 mutation strategy that favours exploration, will have a higher chance of being engaged during the earlier evolutionary stages. During the later stages of the evolutionary process, the local mutation scheme that favours exploitation will have a greater chance of being employed and this helps in inducing a higher selection pressure. The use of grid criteria also played an important role by helping to select better solutions in higher objective space.

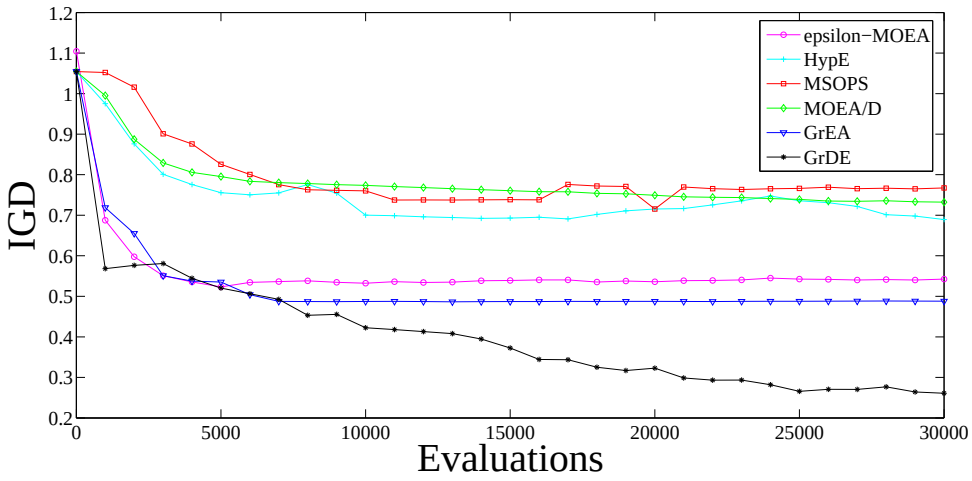

Figure 4.6: Evolutionary trajectories of average IGD for the algorithms on 10-objective DTLZ2

4.7.2 Performance Comparison for WFG problems

Comparative studies were also conducted for the performance evaluation of the six algorithms using the WFG test suite whereby its problems involve various transformation types. WFG test suite creates a big challenge for algorithms in terms of obtaining a well-converged and well-distributed solution set due to different complexities being introduced in the WFG problems which include multi-modality, non-separability and flat bias. WFG problems are also scalable in

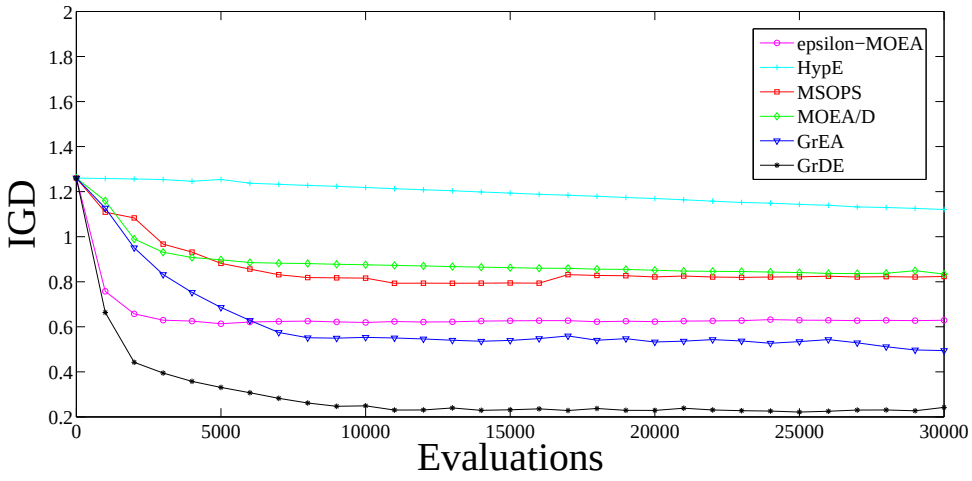

Figure 4.7: Evolutionary trajectories of average IGD for the algorithms on 10-objective DTLZ4

terms of the number of objectives and the number of decision variables. In this study, the WFG problems used involve four, five and seven objectives, and the number of decision variables for these problems was also set accordingly to the number of objectives as found in Table 4.1. Simulation results in terms of the measurement of the average Hypervolume (HV) values over 30 simulation runs are presented in Table 4.5 according to the WFG test problems. The best entries in terms of mean values are also marked in boldface. The Wilcoxon ranksum test was also conducted at the 5% significance level to determine whether the results of best performing algorithm differ from the results of the competitors in a statistically significant way. The entries which are significantly different from the best entries will be indicated by the symbol †.

WFG1 is characterized by the presence of flat bias and it is observed that MSOPS emerges as best performing algorithm for all the considered number of objectives, and the proposed algorithm GrDE achieves competitive results when compared to MSOPS. For the case of WFG2 and WFG3, both these two problems contain non-separable reduction and is characterized by a linear shift from WFG1. In addition, WFG2 is characterized by a disconnected front while the WFG3 has a degenerate one. For these two problems, GrDE achieves the overall best results for all the considered number of objectives.

WFG4 is characterized by multi-modality and results show that GrDE outperforms all the

other algorithms for this problem for all considered number of objectives. As for WFG5, it is a deceptive problem with weighted sum reduction from WFG4, and GrDE also takes the overall lead for this problem in all the considered number of objectives as well. WFG6 is another problem that has nonseparable reduction which is more difficult than the one found in WFG2 and WFG3. As for WFG7, it is separable and uni-modal like WFG1, but the position-related parameters of WFG7 are dependent on its distance-related parameters and other position-related parameters. For both WFG6 and WFG7 with four and five objectives, GrEA is seen to be the best performing algorithm, whereas GrDE outperforms all the other algorithms for WFG6 and WFG7 with 7 objectives.

Lastly for WFG8 and WFG9, they are also problems that involve parameter dependency, whereby the distance-related parameters in WFG8 have dependency on its position-related parameters and other distance-related parameters which in turn makes it a nonseparable problem as well. As for WFG9, it has the similar type of parameter dependency as WFG7, and its distance-related parameters also depend on other position-related parameters. On top of this, WFG9 is also deceptive on its position-related parameters. From the simulation results, it is seen that GrDE achieves the best outcome for WFG8 and WFG9 in all the number of objectives considered.

In summary, the proposed algorithm GrDE outperforms all the other algorithms in 20 out of the 27 WFG test instances in terms of mean HV values, Out of the 20 test instances, GrDE significantly outperforms all the other algorithms in 18 of them according to statistical test. In order to demonstrate the evolutionary process of the algorithms under comparison, the trajectories of the average HV achieved by the algorithms for the WFG2 problem with seven objectives are plotted in Figure 4.8.

As seen from the plot, the trajectory of the proposed algorithm GrDE increases more rapidly than the other algorithms, and maintains an advantage over the rest till the end of the evolutionary process. The overall good results seen for GrDE on the WFG problems may be attributed to the overall design of the proposed algorithm that allows good balance of exploration and exploitation

Table 4.5: Results in terms of HV measurement for WFG problems

Algorithm	Test Instance (m, n)		
	WFG1(4,26)	WFG1(5,28)	WFG1(7,32)
ϵ-MOEA	6.96E1±6.3E-2†	6.72E1±6.9E-2†	6.83E1±6.2E-2†
HypE	6.08E1±8.6E-2†	5.75E1±6.4E-2†	6.11E1±6.5E-2†
MSOPS	**9.35E1±1.1E-2**	**9.50E1±8.1E-3**	**9.43E1±6.6E-3**
MOEA/D	6.63E1±9.2E-2†	6.81E1±9.9E-2†	6.83E1±8.6E-2†
GrEA	7.51E1±4.5E-2†	7.20E1±5.3E-2†	6.90E0±4.1E-2†
GrDE	9.11E1±7.6E-3†	9.17E1±1.1E-2†	9.31E1±1.7E-2†
Algorithm	WFG2(4,26)	WFG2(5,28)	WFG2(7,32)
ϵ-MOEA	8.21E1±5.7E-2†	7.98E1±5.1E-2†	7.79E1±5.5E-2†
HypE	7.98E1±5.1E-2†	6.53E1±8.4E-2†	5.99E1±1.1E-1†
MSOPS	9.21E1±1.0E-2†	9.36E1±8.7E-3†	9.48E1±6.6E-3†
MOEA/D	8.73E1±1.0E-1†	8.64E1±2.9E-1†	8.18E1±2.7E-1†
GrEA	9.00E1±1.3E-2†	9.23E1±9.2E-3†	9.29E1±7.5E-3†
GrDE	**9.31E1±1.1E-2**	**9.43E1±9.5E-3**	**9.55E1±1.1E-2**
Algorithm	WFG3(4,26)	WFG3(5,28)	WFG3(7,32)
ϵ-MOEA	5.57E1±4.1E-2†	4.86E1±6.1E-2†	4.93E1±5.3E-2†
HypE	4.81E1±5.7E-2†	4.15E1±3.4E-2†	3.62E1±3.4E-2†
MSOPS	5.93E1±8.2E-3†	6.01E1±1.7E-2†	5.66E1±2.0E-2†
MOEA/D	4.87E1±1.7E-2†	4.11E1±1.6E-2†	3.13E1±3.5E-2†
GrEA	8.19E1±8.0E-3†	8.99E1±8.5E-3†	9.11E1±7.8E-3†
GrDE	**8.68E1±4.7E-3**	**9.21E1±9.3E-3**	**9.23E1±7.9E-3**
Algorithm	WFG4(4,26)	WFG4(5,28)	WFG4(7,32)
ϵ-MOEA	4.64E1±2.7E-2†	3.97E1±2.3E-2†	3.59E1±1.3E-2†
HypE	2.33E1±4.8E-2†	1.94E1±4.4E-2†	1.87E1±1.8E-2†
MSOPS	4.45E1±1.5E-2†	5.28E1±1.6E-2†	4.66E1±1.9E-2†
MOEA/D	3.00E1±4.3E-2†	3.41E1±7.3E-2†	2.84E1±5.2E-2†
GrEA	5.49E1±2.8E-3†	6.26E1±5.2E-3†	6.94E1±4.9E-3†
GrDE	**5.69E1±2.9E-3**	**6.35E1±3.4E-3**	**7.12E1±5.6E-3**
Algorithm	WFG5(4,26)	WFG5(5,28)	WFG5(7,32)
ϵ-MOEA	4.74E1±2.4E-2†	4.41E1±3.5E-2†	4.04E1±2.7E-2†
HypE	2.40E1±3.8E-2†	1.74E1±4.1E-2†	1.88E1±2.7E-2†
MSOPS	4.24E1±4.4E-3†	4.58E1±1.4E-2†	3.97E1±2.9E-2†
MOEA/D	2.74E1±5.6E-2†	3.11E1±5.4E-2†	3.95E1±6.2E-2†
GrEA	5.44E1±2.4E-3†	5.85E1±4.5E-3†	6.27E1±3.1E-3†
GrDE	**5.58E1±4.0E-3**	**5.99E1±1.1E-2**	**6.42E1±1.3E-2**
Algorithm	WFG6(4,26)	WFG6(5,28)	WFG6(7,32)
ϵ-MOEA	4.97E1±2.3E-2†	4.98E1±3.0E-2†	4.51E1±2.4E-2†
HypE	2.65E1±5.2E-2†	2.41E1±7.2E-2†	2.24E1±8.3E-2†
MSOPS	4.65E1±1.5E-2†	5.32E1±1.8E-2†	4.47E1±3.2E-2†
MOEA/D	2.87E1±2.2E-2†	3.76E1±1.1E-2†	3.46E1±1.0E-2†
GrEA	**5.58E1±2.4E-3**	**6.43E1±3.6E-3**	7.01E1±6.4E-3
GrDE	5.48E1±5.2E-3†	6.29E1±1.1E-2†	**7.03E1±2.1E-2**
Algorithm	WFG7(4,26)	WFG7(5,28)	WFG7(7,32)
ϵ-MOEA	4.64E1±2.7E-2†	4.41E1±4.3E-2†	4.05E1±2.6E-2†
HypE	2.67E1±5.4E-2†	1.87E1±4.7E-2†	1.68E1±3.0E-2†
MSOPS	4.56E1±8.5E-3†	5.44E1±9.1E-3†	4.53E1±2.6E-2†
MOEA/D	2.82E1±8.5E-3†	3.45E1±1.4E-†2	2.58E1±3.9E-2†
GrEA	**5.52E1±1.6E-3**	**6.41E1±2.4E-3**	6.98E1±1.9E-3
GrDE	5.50E1±1.6E-3	6.39E1±3.6E-3	**7.01E1±8.7E-3**
Algorithm	WFG8(4,26)	WFG8(5,28)	WFG8(7,32)
ϵ-MOEA	4.63E1±3.4E-2†	4.39E1±2.7E-2†	4.21E1±3.2E-2†
HypE	2.85E1±3.9E-2†	2.73E1±6.3E-2†	2.62E1±4.1E-2†
MSOPS	4.54E1±9.1E-3†	5.22E1±1.7E-2†	3.55E1±1.8E-2†
MOEA/D	4.81E1±2.0E-2†	5.55E1±4.3E-2†	7.30E2±4.8E-2†
GrEA	5.36E1±8.3E-3†	6.08E1±5.3E-3†	6.28E1±9.1E-3†
GrDE	**5.47E1±6.2E-3**	**6.23E1±8.4E-3**	**6.47E1±5.2E-2**
Algorithm	WFG9(4,26)	WFG9(5,28)	WFG9(7,32)
ϵ-MOEA	4.46E1±2.9E-2†	4.19E1±4.3E-2†	3.81E1±3.5E-2†
HypE	2.47E1±4.3E-2†	1.87E1±5.3E-2†	1.82E1±3.6E-2†
MSOPS	4.26E1±2.4E-2†	4.81E1±2.0E-2†	4.03E1±4.0E-2†
MOEA/D	1.83E1±7.9E-3†	2.07E1±3.3E-2†	1.91E1±2.8E-2†
GrEA	5.39E1±3.3E-3†	6.19E1±8.2E-3†	6.77E1±6.2E-3†
GrDE	**5.48E1±2.1E-3**	**6.37E1±5.3E-3**	**6.95E1±7.4E-3**

in its reproduction operators, while also having the ability to select fitter solutions in higher objective space due to the use of grid-based criteria.

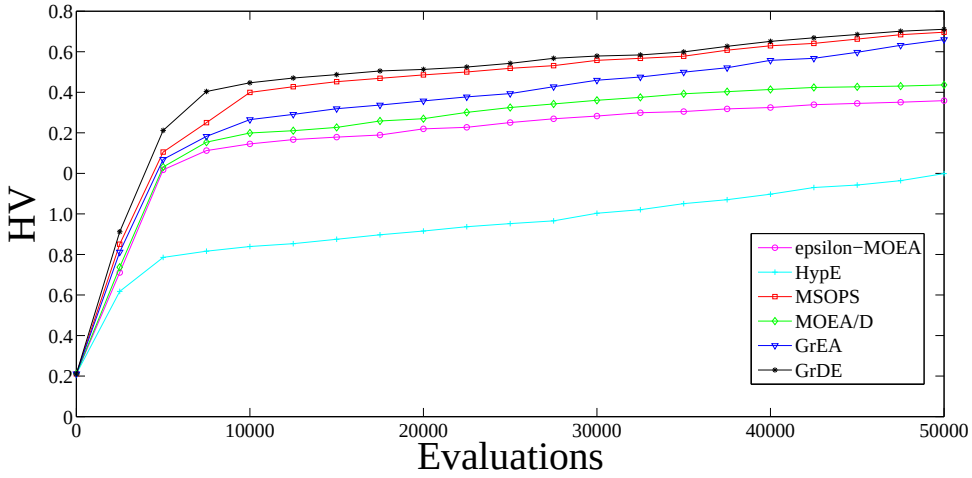

Figure 4.8: Evolutionary trajectories of average HV for the algorithms on 7-objective WFG2

4.7.3 Computational Time Analysis

The primary focus of this section is to compare the computational cost between the proposed algorithm GrDE and the other competing algorithms in this chapter. Table 4.6 presents the average computational times required by all the algorithms used in this study for solving the DTLZ2 problem with different number of objectives. From the results shown in Table 4.6, it is observed that lesser computational time are incurred by ϵ-MOEA, MSOPS, MOEA/D and GrEA when compared to the proposed algorithm GrDE in all the number of objectives considered. However, HypE generally incurs more computational time than all the other algorithms including GrDE. It is noted that the computational cost of GrDE is slightly more than that of GrEA, and this is attributed to the use of the novel mutation strategy that is based on the synthesization of an opposition-based self-adaptive differential evolution mutation scheme with a local mutation scheme. Despite a slower implementation of the proposed algorithm compared to the GrEA, the computational cost incurred is within 5.5 seconds even if the number of objectives hits 10. This is considered to be acceptable in view of the advantages witnessed in the proposed algorithm as

seen from its superiority over the other algorithms under comparison for certain test problems as demonstrated in the previous sections.

Table 4.6: Computational time (in seconds) used by the various algorithms for solving DTLZ2 with 4, 5, 6, 8 and 10 objectives

Algorithm	4 objectives	5 objectives	6 objectives	8 objectives	10 objectives
ϵ-MOEA	1.62	1.87	2.13	2.62	3.03
HypE	8.36	11.45	14.03	17.22	20.54
MSOPS	1.83	2.04	2.39	2.72	3.12
MOEA/D	1.97	2.18	2.47	2.88	3.25
GrEA	2.53	2.86	3.35	4.27	5.21
GrDE	2.76	3.05	3.62	4.52	5.45

4.7.4 Effects of Population Sizing on Optimization Performance

Test problems of DTLZ1 with 5 objectives and 9 decision variables (DTLZ1-5-9) and DTLZ2 5 objectives and 14 decision variables were used in the experimental studies for this section. Simulations were conducted on the proposed algorithm GrDE and GrEA with a population size of 100, 200, 300, 400, and 500, and the results are tabulated in Table 4.7. The simulations stopped at 100,000 fitness evaluations for the case of DTLZ1 and 30,000 fitness evaluations for DTLZ2. In Table 4.7, the best entries for values of the mean, median and standard deviation among the different cases of population sizes used in both the algorithms are marked in boldface.

From the results shown in Table 4.7, it can be deduced that GrEA prefers a larger number of generations rather than a larger population size for better performance. On the other hand, it is observed from the results that the proposed algorithm GrDE generally prefers a larger population size rather than a larger number of generations for better performance. This could be due to the difference in the type of genetic operators used in the algorithms. In GrDE, it basically uses the DE genetic operators which involves the random selection of three distinct individuals from the population for the generation of the mutant vector for every target vector. With a larger population size, there is a larger pool of distinct population individuals for the selection which not only enhances diversity but may also lead to a higher probability of finding the global minimum as well. However, as seen from the results achieved by GrDE for the DTLZ2 problem, it is

observed that there is a slight deterioration of optimization performance when the population size was increased beyond 300 for the DTLZ2. Hence this suggests that the population sizing in GrDE may be problem-dependent. In summary, a larger population sizing for GrDE is generally preferred but this needs to carefully managed so that optimal performance can be achieved.

Table 4.7: IGD metric for DTLZ1 and DTLZ2 with different population size

	GrDE					
	DTLZ1-5-9			DTLZ2-5-14		
Population Size	Mean	Std Dev	Median	Mean	Std Dev	Median
100	0.0532	0.0016	0.0525	0.1590	0.0044	0.1586
200	0.0413	0.0019	0.0411	0.1341	0.0059	0.1327
300	0.0405	**0.0015**	0.0401	0.1290	0.0066	0.1260
400	0.0381	0.0017	0.0375	0.1307	**0.0039**	0.1312
500	**0.0338**	0.0018	**0.00334**	**0.1390**	0.0050	**0.1380**
	GrEA					
	DTLZ1-5-9			DTLZ2-5-14		
Population Size	Mean	Std Dev	Median	Mean	Std Dev	Median
100	**0.0636**	**0.0597**	**0.0076**	**0.1760**	**0.0029**	**0.1756**
200	0.3907	0.1420	0.3986	0.1849	0.0032	0.1861
300	0.3915	0.1350	0.4238	0.1971	0.0036	0.1965
400	0.3921	0.1008	0.4344	0.2134	0.0084	0.2112
500	0.3421	0.1362	0.3727	0.2516	0.0056	0.2502

4.7.5 Sensitivity Analysis of Bounds for DE Control Parameters

In the proposed algorithm GrDE, the opposition-based self-adaptive mechanism requires the definition of the lower and upper bounds for the DE parameters which are namely the mutation factor F and crossover rate CR. In all the experiments mentioned in the previous sections, the bounds were set to 0.1 and 0.9 for the lower and upper bound respectively for both DE parameters. This setting will be referred to as the original settings in this section. This section investigates the effect when the bounds are varied, and tries to provide a recommended setting of the bounds for the proposed algorithm. The results on the DTLZ problems with 5 objectives are demonstrated for brevity. In order to study the sensitivity of GrDE to the lower and upper bounds, experiments for the DTLZ problems were repeated by using different combinations of F and CR. In Table 4.8, the values of the lower and upper bounds for all the different combinations of the bounds are stated. For all the experiments, the bounds were first varied for the mutation factor F but with the original settings being applied to the bounds for the crossover rate CR. Next the bounds for CR

were then varied and the bounds for F were fixed at the original settings. The other parameters in GrDE related to the self-adaptive mechanism remained unchanged.

Table 4.8: Values of bounds for the different combinations (cases) of DE parameters F and CR

	Case									
	1	2	3	4	5	6	7	8	9	10
Lower Bound	0.1	0.3	0.5	0.7	0.1	0.3	0.5	0.1	0.3	0.1
Upper Bound	0.9	0.9	0.9	0.9	0.7	0.7	0.7	0.5	0.5	0.3

In Tables 4.9-4.10, the average IGD values obtained by the proposed algorithm GrDE for all the experiments using different combinations of the lower and upper bounds of both F and CR are displayed. The values in parentheses beside every IGD value represent the ranks for a particular combination of the lower and upper bounds as compared to the other combinations for the test problem. For every test problem, the top four entries for every case in terms of the lowest IGD values are marked in boldface in all the tables.

Table 4.9: Average IGD values for GrDE with varying bounds (all 10 cases) for F for DTLZ problems

	Test Problem						
Case	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
1	**0.0532(1)**	0.1593(7)	**0.1511(1)**	**0.1571(3)**	0.1183(8)	0.4675(5)	0.1106(9)
2	0.2685(7)	0.1642(9)	0.5846(9)	0.1600(8)	0.1429(9)	**0.4449(3)**	0.0718(6)
3	**0.1413(4)**	**0.1572(3)**	**0.1873(3)**	**0.1571(3)**	0.1130(7)	**0.4402(1)**	0.0906(7)
4	**0.1008(3)**	0.1717(10)	**0.2908(4)**	0.1746(10)	0.1458(10)	**0.4440(2)**	0.2110(10)
5	0.2716(8)	**0.1571(2)**	**0.1694(2)**	**0.1559(1)**	**0.0805(3)**	0.4972(9)	**0.0443(1)**
6	**0.0542(2)**	0.1627(8)	0.3105(5)	**0.1568(2)**	0.0994(5)	0.4949(8)	0.0967(8)
7	0.2370(6)	0.1584(6)	0.4118(7)	0.1599(7)	0.1001(6)	**0.4481(4)**	0.0705(5)
8	0.2236(5)	**0.1574(4)**	0.3964(6)	0.1598(6)	**0.0543(1)**	0.4823(6)	**0.0524(3)**
9	0.3582(10)	0.1578(5)	0.7260(10)	0.1607(9)	**0.0759(2)**	0.4927(7)	**0.0652(4)**
10	0.2720(9)	**0.1560(1)**	0.5157(8)	0.1579(5)	**0.0808(4)**	0.5205(10)	**0.0501(2)**

Table 4.10: Average IGD values for GrDE with varying bounds (all 10 cases) for CR for DTLZ problems

	Test Problem						
Case	DTLZ1	DTLZ2	DTLZ3	DTLZ4	DTLZ5	DTLZ6	DTLZ7
1	**0.0532(2)**	0.1593(8)	**0.1511(1)**	0.1571(5)	0.1183(7)	0.1106(8)	0.4675(5)
2	**0.0534(4)**	0.1585(6)	0.4989(8)	**0.1555(1)**	0.1052(6)	0.1353(9)	**0.4580(4)**
3	0.0548(9)	0.1622(10)	0.5754(9)	0.1627(10)	0.1419(10)	0.0870(7)	**0.4258(1)**
4	**0.0521(1)**	0.1620(9)	0.6604(10)	0.1580(6)	0.1251(9)	0.1455(10)	**0.4355(3)**
5	0.0560(8)	0.1572(5)	**0.1523(2)**	**0.1569(4)**	0.1192(8)	**0.0640(4)**	0.5257(10)
6	0.0544(6)	**0.1556(3)**	**0.2730(4)**	0.1613(8)	**0.0642(1)**	0.0649(5)	0.4719(6)
7	0.0544(6)	0.1587(7)	0.4265(7)	0.1586(7)	0.1005(5)	**0.0529(2)**	**0.4353(2)**
8	**0.0532(2)**	**0.1554(2)**	0.3307(5)	**0.1558(2)**	**0.0889(4)**	**0.0623(3)**	0.4826(7)
9	0.0541(5)	**0.1545(1)**	**0.1535(3)**	0.1621(9)	**0.0686(2)**	**0.0455(1)**	0.5020(8)
10	0.0746(10)	**0.1556(3)**	0.3537(6)	**0.1562(3)**	**0.0799(3)**	0.0665(6)	0.5156(9)

For the mutation factor F, it is observed that through the use of the bounds under Case 5 (lower bound = 0.1, upper bound = 0.7), there were 2 DTLZ problems that achieve first ranking in

terms of the lowest IGD among all the different combinations for the bounds, and also achieving second placing for another 2 DTLZ problems and a third placing for a last DTLZ problem. This performance is seen to be better than all the other cases of the combinations for the mutation factor parameter. As such, Case 5 is considered as the best combination for the bounds for the mutation factor in the opposition-based self-adaptive mechanism in the proposed algorithms.

For the case of the crossover rate CR, the use of the bounds under Case 9 (lower bound = 0.3, upper bound = 0.5) generated results of 2 DTLZ problems achieving first placing and another 2 more DTLZ problems with a second and a third ranking respectively. With this performance, this combination for the bounds is regarded as the best one for the crossover rate in the opposition-based self-adaptive mechanism used in the proposed algorithm.

With this, it may be recommended that the bounds for the mutation factor F to be set between 0.1 and 0.7, and the crossover rate CR to be between 0.3 and 0.5. Through the recommended bounds, the mutation factor could potentially be adapted to an adequately large value for exploration in the search space, and be adapted to lower values to increase the convergence rate. This would help to strike a balance between exploration and exploitation when the proposed algorithm handles many-objective problems. As for the crossover rate, the range of 0.3 to 0.5 will allow the proposed algorithm to achieve good control on which and the number of components to be mutated in every individual of the current population. As sensitivity analysis for the parameters was only performed for the DTLZ problems, future work could be performed to investigate the effects of varying the bounds for WFG problems as the setting of the bounds for the self-adaptive mechanism in the proposed algorithm may be problem-dependent in order to achieve the best outcome for the problems handled.

4.8 Summary

In view of the growing need for better and efficient methodologies for evolutionary many-objective optimization, a grid-based differential evolution algorithm termed as GrDE has been

proposed in this study to handle many-objective optimization problems. For the development of the algorithm, a novel mutation strategy has been formulated and then incorporated into a grid-based framework to form the proposed grid-based differential evolution algorithm termed as GrDE. In the novel mutation strategy for GrDE, the opposition-based self-adaptive mutation scheme from OSADE is integrated with a newly formed opposition-based local mutation scheme via a linear decreasing probability rule.

Comprehensive simulation studies have been conducted to investigate how the differential evolution variant proposed in this study would perform when being incorporated into a grid-based framework to handle a suite of widely used test problems that are able to pose different challenges for evolutionary algorithms. Through an extensive comparison of the proposed algorithms with five other state-of-the-art evolutionary algorithms, it is seen that GrDE is very competitive with the other algorithms in terms of achieving a well-approximated and well-distributed solution set for the many-objective test problems. In total, the proposed algorithm significantly outperforms the other algorithms in 35 out of all 62 test instances. The good performance achieved by the proposed algorithm may be attributed to the complementary effect of using the opposition-based self-adaptive mutation scheme which has strong exploratory abilities together with the local mutation scheme that comes with strong exploitation abilities. With this coupling, a good balance between convergence and diversity can be achieved which is an important factor in evolutionary many-objective optimization. Furthermore, the use of the opposition-based self-adaptive mutation not only eliminates the need to pre-define the DE control parameters, but also increases the probability of finding optimal settings for the control parameters throughout the evolutionary process and this helps to generate better solutions as well. Moreover, the use of the grid-based criteria in the proposed algorithms allows a good refection of the information of convergence and diversity simultaneously, and this helps to increase the selection pressure in the proposed algorithm to derive better solutions in higher objective space. From the results obtained in this study, it is recommended that GrDE can be a suitable optimization algorithm for handling many-

objective problems with multi-modality. However, it is observed that the GrDE did not fare as well as the other algorithms for problems with degenerate Pareto optimal front like the DTLZ5. This could be a limitation of the DE operator, and requires further investigation.

Investigation of the effects of different population sizing for GrDE has also been conducted. It is observed that a larger population size is generally preferred over a larger number of generations for the proposed algorithm. However, the sizing might be problem-dependent, and therefore the sizing of the population needs to be carefully managed to ensure that optimal performance can be achieved. Besides this, the effects of varying the lower and upper bounds for the differential evolution control parameters (mutation factor and crossover rate) in the opposition-based self-adaptive mutation mechanism is also being studied as well. Through the study, it is observed that an appropriate setting for the bounds for both control parameters may bring upon better general performance by the proposed algorithm. Lastly, computational time analysis was also carried out and it is observed that the GrDE incurs slightly more computational cost than the GrEA. However, the increase in the computational cost is acceptable due to the better performance achieved by it over other algorithms when compared over certain test problems.

Chapter 5

An Opposition-based Self-adaptive Differential Evolution with Decomposition for Solving the Multi-objective Multiple Traveling Salesman Problem

The multiple Traveling Salesman Problem (mTSP) is a complex combinatorial optimization problem, which is a generalization of the well-known Traveling Salesman Problem (TSP), where one or more salesmen can be used in the solution. In this chapter, a novel differential evolution algorithm termed as D-OSADE is developed to solve a Multi-objective Multiple Salesman Problem (MmTSP). In D-OSADE, the opposition-based self-adaptive differential evolution operator from OSADE is incorporated into the decomposition-based framework, and is then hybridized with the multi-point evolutionary gradient search (EGS) which acts as a form of local search to enhance the search behaviour. Simulation studies will be carried out to examine the optimiza-

tion performances of the proposed algorithm on the MmTSP with different number of objective functions, salesmen, and problem sizes in terms of the number of cities to be visited. The effectiveness and efficiency of the algorithm will also be tested and benchmarked against several state-of-the-art multi-objective evolutionary paradigms.

5.1 Chapter Objectives

The main objectives of this chapter is to look into the adaptation of OSADE, which is originally intended for continuous optimization, for the handling of a combinatorial optimization problem the multi-objective multiple travelling salesman problem (MmTSP). The chapter also seeks to extend OSADE into a decomposition-based approach for multi-objective optimization, and investigates the optimization performance of the resultant algorithm by comparing it with several state-of-the-art evolutionary algorithms using the MmTSP.

5.2 Introduction

The multiple traveling salesman problem (mTSP) is a complex combinatorial optimization problem, which is considered as a generalization of the famous and widely known Traveling Salesman Problem (TSP), whereby more than one salesman is allowed to be used in the solution. In the mTSP, there is a set of n cities to be visited by m salesmen (where $m < n$), and all the salesmen will be required to start and end at a single depot after routing through the ordered cities. The objective of the mTSP is to determine a route for every salesman such that the total cost of the route is minimized and every city is only to be visited once by only one salesman. For the cost, it can in terms of time, distance, expense etc. In the event if m = 1, the mTSP problem will become the classical TSP. mTSP naturally becomes more sophisticated than the TSP as there is requirement to assign a set of cites to every salesman in an optimal order, while ensuring that the total cost for all the salesmen is being minimized. However, the mTSP has not received as much attention in

terms of research efforts when compared to the TSP. Due to the fact that the TSP belongs to the class of NP-complete problems, it is clear that the mTSP is an NP-hard problem due to the high complexity involved in it. Hence the mTSP will require heuristic approaches as they are able to obtain approximate optimal solutions within specific time or computational limitations.

Moreover, most scheduling problems in real-world situations involve the simultaneous optimization of several conflicting objectives. In multi-objective optimization framework, there is no single optimal solution but rather a set of non-dominated solutions which is representative of the tradeoff between the multiple objectives of a problem. As such, fitness assignment is a critical component of the multi-objective evolutionary framework as it contributes to the identification of better solutions for the next generation. Over the decades, a substantial amount of research efforts has been placed in formulating different fitness assignment techniques for multi-objective optimization, and the Pareto domination-based approach is considered to be one of the most popular [50]. However, the weakness in this approach is seen when the number of objectives in the problem scales up. This is due to the fact that most of the solutions are non-dominated to each other when the number of objectives increases and this result in difficulties during the selection of promising solutions. In recent years, a decomposition-based approach is being introduced whereby it makes use of traditional aggregation methods to convert the task of approximating the Pareto front into the optimization of a number of single-objective optimization subproblems. As such, this approach does not require the differentiation of the domination behaviour among the solutions in a population, and hence the decomposition approach can be a good alternative to the frequently used domination-based approach in multi-objective optimization.

Differential Evolution (DE) is a relatively new evolutionary algorithm developed by Storn and Price [25], and is considered to be one of the most powerful and effective stochastic real-parameter optimization algorithms in current use. The advantages in using DE lies in its simplicity and ease of implementation, and yet reliable and fast. In addition, it requires only a few control parameters to be set by a user. As such, there is a rapidly growing interest in the research

in DE as seen by numerous reported works on DE. However, there are very few studies on the use of differential evolution for solving the mTSP.

This study investigates the integration of an opposition-based self-adaptive DE operator from OSADE with the decomposition-based frameworks, and then hybridizing the resultant DE variant with multi-point evolutionary gradient search (EGS) which acts as a form of local search before using the overall algorithm to handle the multi-objective multiple traveling salesmen problem (MmTSP). To the best of our knowledge, there are no recorded works on the use of any decomposition-based self-adaptive differential evolution algorithms that are being proposed to solve the MmTSP, and this gives motivation for us to propose D-OSADE to tackle such a problem. The DE operator from OSADE is used in the proposed algorithm because it is simple and efficient, and the need to tune and fix the DE control parameters is eliminated due to the opposition-based self-adaptive mechanism in it. As the DE operator is originally intended for continuous optimization with floating point values, a heuristic rule needs to be applied in order to enable it to be used for combinatorial optimization. For this, the Smallest Position Value (SPV) rule [191] is applied in the proposed algorithm here. The MmTSP found in Shim *et al.* [192] is employed as the problem in this study whereby the objective function is being formulated to be the weighted sum of the total traveling cost of all salesmen and the highest traveling cost by any single salesman. The proposed algorithm is then compared with several state-of-the-art multi-objective evolutionary algorithms through simulation studies conducted on different test instances of the MmTSP with different number of objectives, salesmen and cities.

The rest of this chapter is organized as follows. The following section presents a literature review on some of the previous works on the mTSP, as well as a brief explanation of the multi-point EGS and the Smallest Position Value (SPV) that are used in D-OSADE. The description of the problem used in this study is presented in Section 5.4, and the details of the proposed algorithm will be highlighted in 5.5. The test problems and parameter settings for the experiments are outlined in Section 5.6. This will be followed by Section 5.7 which presents the simulation

results and discussion. The summary of this study will be given in Section **??**.

5.3 Background

In this section, a literature review is presented on some of the previous works on the mTSP where focus is placed on some of the evolutionary approaches for the mTSP. For a more rigorous review on the works that engaged the use of non-evolutionary techniques for the mTSP, it can be found in [193]. An overview of the multi-point EGS and the SPV that are used in the proposed algorithm will also be presented.

5.3.1 Related Works

Over the years, the traveling salesman problem received a large amount of attention with several approaches being introduced for the solving of the problem and these approaches include neural network [194], tabu search [195] and branch-and-bound [196]. Some of these methods are exact algorithms, and the others are near-optimal or approximate algorithms. For the exact algorithms, they employ integer linear programming approaches with additional constraints.

As for the mTSP, it received less attention compared to the TSP. A detailed review of the known approaches for the mTSP can be found in [197]. Some of the approaches are exact algorithms of the mTSP whereby some of the constraints of the problem are being relaxed, and an example of this is found in [198]. There is also another piece of work found in [199] whereby the approach is based on the Branch-and-Bound algorithm. Due to the combinatorial complexity of the mTSP, heuristic is being applied in the solution. Russell [200] proposed one of the first heuristic approach for the mTSP, and another such approach can be found in Potvin *et al.* [201]. Another solution based on neural network can be in found in Hsu *et al.* [202].

More recently, the use of genetic algorithms (GAs) for the solving of TSP emerged [203], and a survey of the solving of the general TSP using GA can be found in [204] . The pioneering effort of using GA to solve the mTSP was initiated by Zhang *et al.* [205] whereby simple GA

with natural representation is being used for the scheduling of multiple teams of photographers to visit a large number of schools. In the problem, the goal is to minimize both the total distance traveled and the time taken while satisfying the time constraints as well as the requirement for two schools to be visited by each team daily. However in this work, there is no elaboration on how the manipulation of the multiple teams is done in their chromosome representation.

In Tang *et al.* [206], a hot rolling scheduling problem in Shanghai Baoshan Iron and Steel Complex was being investigated whereby the problem was being modeled as an mTSP first. Real-life production constraints were then put into consideration in the problem whereby the goal is to schedule multiple turns within the same shift. With the use of a proposed one-chromosome representation, the mTSP was then transformed into a classical TSP. In this implementation, the best solutions obtained during every stage are always being selected to become one of the parent chromosomes for the crossover operation.In another study [207], a vehicle scheduling problem was treated like an mTSP due to the involvement of multiple vehicles in the routing. For this study, a two-chromosome representation was proposed whereby the first chromosome specifies the cities, and the second one is used to indicate which vehicle is to be tasked to visit the city specified in the first chromosome. In Carter *et al.* [208], a two-part chromosome representation was proposed for the mTSP. In this implementation, the chromosome for a gene consists of two different parts. The permutation of the cities is being assigned in the first part of the chromosome whereas the second part of the chromosome is catered for the number of cities to be visited by each salesman. As such, there is an additional number of m genes in the chromosome for m salesmen. From the results obtained, it is seen that the proposed representation derived better results when compared to the one-chromosome representation for most of the test instances.

An improved version of the genetic algorithm that is based on one-chromosome representation was proposed by Zhao *et al.* [209] for the tackling of the mTSP. In this algorithm, a pheromone-based crossover operator that utilizes heuristic information including edge lengths and adjacency relations as well as pheromone levels is used for the creation of offspring. In

addition, local search is also being used to act as the mutation operator.

In a recent study by Kiraly *et al.* [210], a multi-chromosome representation of mTSP solutions was proposed whereby every chromosome contains the route allocated to every salesman. As such, each solution has m associated chromosomes. It is to be mentioned that the crossover and mutation operators were also specially designed to handle the representation for this work. In another recent study by Shim *et al.* [192], the estimation of distribution algorithm (EDA) is being integrated into the decomposition framework, and is further enhanced by hybridizing the decomposition EDA with different local search metaheuristics. In that study, the problem is being reformulated to tailor the focus on the mTSP with single depot, and to consider the minimization of the total traveling cost while balancing the workload among all the salesmen. The proposed algorithms in that study were tested and compared against several state-of-the-art algorithms to test their effectiveness and efficiency. It was observed that the proposed algorithms achieved the best performance in most of the problem settings studied.

5.3.2 Evolutionary Gradient Search (EGS)

Local search has been proven to be able to enhance the performance of global search in evolutionary algorithms [211, 212] where it is being used to exploit the local optimal in a specific region. The evolutionary gradient search (EGS) is used as a form of local search in this chapter, and an overview of the EGS will be presented here.

Gradient search is a classical method for continuous parameter optimization whereby gradient direction is being retrieved to guide in the search. In every run, a single solution will be produced. Evolutionary gradient search (EGS) is a recent approach proposed by Arnold and Salomon [92] which is basically a hybrid of both gradient search and evolutionary strategies which encompasses the merits of their features. Due to it potential, the use of EGS has also been extended in [94,213]. In EGS, multi-directional searches are conducted, and the gradient will be the direction derived from the evolutionary movement instead of the single movement of a solution.

The pseudo-code of the multi-point EGS can be referred to Figure 5.1.

The EGS starts by defining the initial step size σ_0. A step size σ is required in EGS to control the mutation strength to be applied for the generation of the local neighbors and the offspring. Upon the selection of an initial solution to undergo local search, N local neighbours will be randomly generated by perturbing the initial solution using normal mutation with zero mean and σ^2 variance. Next the global gradient direction is to be estimated from the local neighbours according to the formula in step 6 of Figure 5.1. A gradient offspring will then be created in step 7, and this is followed by the updating the mutation step size in step 8 with the control of a factor ε which is recommended to be 1.8 according to [214]. The solution will then be updated in step 9 and the EGS process is iterated until the stopping criterion is achieved.

Begin

1. Input: Define the initial step size σ_0. $t = 1$

Do while ("Stopping criterion is not achieved")

For $i = 1$ ***to*** X (Number of parent solutions)

2. Initial Solution: Select a solution x_i from the selection pool

3. Reproduction: Randomly generate N local neighbours s_j by perturbing x_i using normal mutation $N(0, \sigma^2)$

4. Evaluation: Compute the objective function values F(x_i) of x_i

5. Archiving: Update the non-dominated solutions in an archive

6. Direction: Estimate the global gradient vector as follows:

$$\vec{g} = \frac{\sum_{i=1}^{N}[F(s_j) - F(x_i)](s_j - x_i)}{\left\|\sum_{i=1}^{N}[F(s_j) - F(x_i)](s_j - x_i)\right\|}$$

7. Offspring generation: Produce an offspring q as follows:

$$q = x_i - \sigma_t \vec{g}$$

8. Update parameter: Update the mutation step size σ_{t+1} as follows:

$$\sigma_{t+1} = \begin{cases} \sigma_t \varepsilon & if\ F(q) < F(x_i) \\ \sigma_t / \varepsilon & otherwise \end{cases}$$

where $\varepsilon = 1.8$

9. Update solution: Update the parent solution x_i as follows:

$$x_i = q\ if\ F(q) < F(x_i)$$

10. Output: Output x_i

End for i

End Do

End

Figure 5.1: Pseudo-code of the evolutionary gradient search algorithm

5.3.3 Small Position Value (SPV) Rule

A heuristic rule known as the Smallest Position Value (SPV) was proposed by Tasgetiren *et al.* [215] to allow the continuous particle swarm optimization to be applied to all classes of sequencing problems, which are NP-hard in literature. The Smallest Position Value approach was also applied in [191] which involves the use of a standard differential evolution algorithm along with the SPV and a unique solution representation to solve the generalized traveling salesman problem.

The SPV rule can be explained using Table 5.1 which shows the solution representation of a vector in differential evolution (DE). In every vector, there is a continuous set of floating values, and these values do not represent any sequence. Hence the SPV rule can be used to determine the sequence of the cities visited by salesmen in the traveling salesman problem as implied by the variable values in the vector. According to the SPV rule, the smallest position value is at x_{i4}^k = 0.153, and hence the dimension j = 4 is assigned to be the first city s_{i1}^k = 4 in the sequence (S_i) of the cities to be visited. The second smallest position value is at x_{i2}^k = 0.204, and this would mean that the dimension j = 2 is assigned to be the second city to be visited, and so on and so forth for the remaining dimensions. Therefore this means that dimensions are sorted according to the SPV rule, which in turn means that x_{ij}^k values are used to construct the sequence S_i. In addition, every value in the sequence will be unique and there will not be any repetition of the integer values in the sequence. This representation is also unique in finding new solutions as the positions in every vector will be updated at each iteration k in the differential evolution algorithm. Thus there will be different sequences at each iteration k.

Table 5.1: Solution representation of vector in DE (for Smallest Position Value rule)

j	1	2	3	4	5
x_{ij}^k	0.678	0.204	0.893	0.153	0.432
s_{ij}^k	4	2	5	1	3

5.4 Problem Description

As seen from literature, the goal in the mTSP is to either minimize the total traveling cost of all salesmen or to minimize the highest traveling cost incurred by any single salesman [216]. However in the study here, it shall focus on the problem formulated in [192]. This mTSP will be based on a single depot, and taking into consideration of both the objective of minimizing the total traveling cost as well as the objective of balancing the workload among all the salesmen. To achieve this, the objective functions in the mTSP are formulated as a weighted sum of the total traveling cost of all salesman and the highest traveling cost by any single salesman. As the mTSP here is to be considered in a multi-objective setting, the problem will be formulated as follows:

Minimize:

$$F_T(\mathbf{x}) = \omega_1 \times TTC_T\mathbf{x} + \omega_2 \times HTC_T\mathbf{x}, T = 1, 2, ..., K$$

where

$$TTC_T(\mathbf{x}) = \sum_{j=1}^{m} ITC_T^j(\mathbf{x})$$

$$HTC_T(\mathbf{x}) = \max_{1 \leq j \leq m} (ITC_T^j(\mathbf{x}))$$

$$ITC_T^j(\mathbf{x}) = \left(\sum_{i=1}^{n_j - 1} C_T^j(v_{i,j}, v_{i+1,j}) \right) + C_T^j(v_{n_j,j}, v_{1,j})$$

In the above formulated problem, $x \in \pi$, where π represents the decision space, $v_{i,j}$ is the ith city to be visited by salesman j, and T is the number of objective functions in the problem. In the objective functions, TTC is the total traveling cost of all salesmen, HTC is the highest traveling cost incurred by any single salesman, and ITC is the individual traveling cost of a salesman. The number of salesmen is denoted by m, and n_j represents the number of cities traveled by the salesman j. $C_T^j(v_{i,j}, v_{i+1,j})$ is the traveling cost (for the Tth objective) between cities at locations i and $i + 1$ for salesman j. Lastly, ω_1 and ω_2 denote the weights that are used to

balance between the total traveling cost and the highest traveling cost such that $\omega_1 + \omega_2 = 1.0$. In this study, the weights of ω_1 and ω_2 are set to 0.5 each as per the recommendation from the study conducted in [192]. In addition, all the cities must be visited only once and each salesman must be tasked to visit at least one city in his traveling route for the problem here.

5.5 Proposed Algorithm

The framework of the proposed algorithm is presented in this section. The algorithm is basically made up of four main mechanisms which are namely the chromosome representation, decomposition, reproduction based on an opposition-based self-adaptive differential evolution operator and local search. For the chromosome representation, one-chromosome representation from [206] is employed to represent the order of the cities to be traveled by the m salesmen. This representation approach creates $m - 1$ pseudo-cities (denoted by integer values < 0) for the chromosome. The pseudo-cities will represent the same initial starting city for all salesmen. Hence, every chromosome will consist of $n + m - 1$ genes.

An illustration of the chromosome representation can be found in Figure 5.2 whereby there are three salesmen and nine cities. The sequence in the representation shows that the first salesman starting from the initial city 0 and then visiting cities 4, 8 and 3 in that order of, and the second salesman will start from the initial city (city indicated by -1) and then visits cities 5 and 1 in that order. The third salesman starts from the initial city (city indicated by -2) and visits cities 2, 7 and 6 in that order.

0	4	8	3	-1	5	1	-2	2	7	6

Figure 5.2: One-chromosome representation

With the chromosome representation in place, the proposed algorithm will then involve the incorporation of the opposition-based self-adaptive differential evolution operator into the decomposition framework from MOEA/D. For the implementation here, the assignment of the

fitness to every solution is based on the Tchebycheff approach. In this classical aggregation approach, the reference point $\mathbf{z}^*$ and a set of uniformly distributed weight vectors $\boldsymbol{\lambda}^1, ..., \boldsymbol{\lambda}^S$, where S is the number of subproblems, are being generated. The population will be decomposed into S scalar optimization subproblems based on the Tchebycheff formulation and the fitness value of the ith subproblem is formulated as follows:

$$g^{te}(\mathbf{x}|\boldsymbol{\lambda}^i, \mathbf{z}^*) = \max_{1 \le k \le T} \left\{ \lambda_k^i | F_k(\mathbf{x}) - z_k^* | \right\} \tag{5.1}$$

Inputs:
1) Stopping criterion
2) Population size, X or number of subproblems, S
3) A set of evenly distributed weight vectors for decomposition, $\lambda_1, \dots, \lambda_S$
4) Number of neighbours for each weight vector, P
5) Number of traveling salesmen in the problem, m
6) Number of cities for the problem, n
7) Number of local neighbours for the local search, N

Output:
1) A set of solutions in both decision and objective space as found by the optimizing algorithm

Step 1: Initialization: (at Generation G = 0)
a) Randomly generate initial population, Pop1_DE (G) that contains S vectors with floating values.
b) Compute the Euclidean distance between all the weight vectors and group the P closest neighbours to every sub-problem i as denoted by $B(i) = \{i_1, \dots, i_P\}$ where $i \in [1, S]$
c) Convert the vectors in Pop1_DE (G) with floating values into chromosome-based solutions with integer values using the Smallest Position Value (SPV) rule, and place the chromosome-based solutions denoted by x_i where $i \in [1, S]$ in another population Pop1 (G)
d) Evaluate the fitness of every chromosome individual in Pop1 (G)
e) Initialize the reference point $z^* = (z_1^*, \dots, z_T^*)$ by setting it to the minimum objective value of the initial population

Step 2: Reproduction:
a) Use the opposition-based self-adaptive differential evolution operator, and store the resultant trial vectors from the the DE operation in an offspring population PopQ_DE (G+1)
b) Convert the vectors in PopQ_DE (G+1) into their equivalent chromosome offspring solution denoted by y_i using SPV rule and place them in PopQ (G+1).
c) Evaluate the fitness of every chromosome-based solution y_i in PopQ (G+1).
d) Penalize the chromosome solution if any salesman is not assigned any city by multiplying the objective value with a constant K

Step 3: Update solution:
For $i = 1, \dots, S$ do
a) Update of z^*: For $j = 1, \dots, T$, if $z_j^* > f_j(y^i)$, then set if $z_j^* = f_j(y^i)$
b) Update of neighbouring solutions: For $j \in B(i)$, if $gt(y^i | \lambda^j, z^*) \le gt(x^i | \lambda^j, z^*)$, then set $x^j = y^i$ and $FV^j = f(y^i)$
End do

Step 4: Local search:
a) Perform local search if activated. Next apply Step 3 to update the newly created offspring solutions from local search

Step 5: Stopping criterion:
a) If stopping criterion is achieved, then terminate. Else, go back to Step 2

Figure 5.3: Pseudo-code of D-OSADE

The pseudo code of the proposed algorithm is as described in Figure 5.3. In the initialization stage, the algorithm starts by determining the P nearest neighbours to each weight vector

according to Euclidean distance calculation. This means that every subproblem i would have P neighbours which are denoted by $B(i) = \{i_1, ..., i_P\}$. As a differential evolution variant is being used as the reproduction operator in the proposed algorithm, the initial population is then randomly generated with population candidates which are vectors with floating values instead of chromosomes with integer values as per the structure as shown in Figure 5.2. This is to facilitate the reproduction process for generation of new individuals using the opposition-based self-adaptive differential evolution operator. The objective values are also required to be calculated based on the problem described in Section 5.4 during the initialization stage so that the reference point $\mathbf{z}^*$ can be initialized. In order to evaluate the initial population, the initial population of vectors with floating point values has to be converted into chromosome-based solutions with integer values and stored in a separate population meant for the chromosome-based solutions. To achieve this, the Smallest Position Value (SPV) rule mentioned in Section 5.3.3 will be applied. The converted solutions will be evaluated according to the problem, and the F-values (FV) will be updated with the objective values from evaluation of the chromosome-based solution y. The minimum objective value of the initial population will also be used to initialize the reference point $\mathbf{z}^*$.

For the next stage, reproduction based on the opposition-based self-adaptive DE mutation scheme from OSADE will be performed, and the detailed pseudo code of this scheme is provided in Figure 5.4. At the start of a parent generation G, three randomly selected vectors x_{r1}^G, x_{r2}^G and x_{r3}^G, whereby $i \neq r_1 \neg r_2 \neq r_3$, and $r_1, r_2, r_3 \in 1, 2, ..., S$ where S is the size of the population, will be used for the DE mutation operation together with the target vector x_i^G. In every individual, the values of the control parameters (mutation factor F and crossover rate CR) will exist as extended variables whereby they will be initialized to zero at the start of every run. These encoded values are required in the opposition-based self-adaptive mutation scheme for the self-adaption process of the parameters whereby the current F and CR values for a particular generation will be calculated based on the averaging of the encoded values from the four indi-

For i = 1 to S **do**

a) Randomly select 3 vectors x_{r1}^G, x_{r2}^G and x_{r3}^G, whereby $i \neq r_1 \neq r_2 \neq r_3$, and $r_1, r_2, r_3 \epsilon \{1,2 \ldots, S\}$

b) Retrieve the encoded F and CR values from x_i^G, x_{r1}^G, x_{r2}^G and x_{r3}^G

c) Calculate the average values of F and CR which are denoted by $\langle F_G \rangle_i$ and $\langle CR_G \rangle_i$ for the parent population using the encoded values from x_i^G, x_{r1}^G, x_{r2}^G and x_{r3}^G:

$$\langle F_G \rangle_i = \frac{F_{i,G} + F_{r1,G} + F_{r2,G} + F_{r3,G}}{4}, \qquad \langle CR_G \rangle_i = \frac{CR_{i,G} + CR_{r1,G} + CR + CR_{r3,G}}{4}$$

d) Calculate the updated F and CR denoted by $\bar{F}_{i,G+1}$ and $\overline{CR}_{i,G+1}$ for the offspring population as follows:

$$\bar{F}_{i,G+1} = \langle F_G \rangle_i \; x \; e^{\tau N(0,1)}, \qquad \overline{CR}_{i,G+1} = \langle CR_G \rangle_i \; x \; e^{\tau N(0,1)}$$

where $\tau = 1/(8\sqrt{2D})$, D is the number of decision variable, and $N(0,1)$ represents a randomly generated number. F and CR are bounded between 0.1 and 0.9

e) Generate the Opposite Number of $\bar{F}_{i,G+1}$ as follows:

$$\overline{F\_opp}_{i,G+1} = F_{upper} + F_{lower} - \bar{F}_{i,G+1}$$ where F_{upper} and F_{lower} are the bounds for F

f) Create 2 trial vectors $V1$ and $V2$

g) Randomly select an integer from (1,D) to be j_{rand}, and perform differential mutation:

For j = 1 to D **do**

If rand (0, 1) $\leq CR_{i,G+1}$ or j = j_{rand} **then**

$$V1_{i,j} = X_{r1,j} + F_{i,G+1} * (X_{r2,j} - X_{r3,j})$$
$$V2_{i,j} = X_{r1,j} + F\_opp_{i,G+1} * (X_{r2,j} - X_{r3,j})$$

Else

$$V1_{i,j} = X_{i,j}, \quad V2_{i,j} = X_{i,j}$$

End If

End For

h) Let the offspring be represented by $U_{i,G+1}$

i) Convert the trial vectors $V1_{i,j}$ and $V2_{i,j}$ from vectors with floating point values to become two trial chromosome solutions ($C1$ and $C2$) with integer values using SPV rule.

j) Evaluate both $C1$ and $C2$ according to the problem, and compare them as follows:

If $C1$ dominates $C2$ **then**

$$U_{i,G+1} = V1; \quad F_{i,G+1} = \bar{F}_{i,G+1}; \; CR_{i,G+1} = \overline{CR}_{i,G+1}$$

Else if $C2$ dominates $C1$ **then**

$$U_{i,G+1} = V2; \quad F_{i,G+1} = \overline{F\_opp}_{i,G+1}; \; CR_{i,G+1} = \overline{CR}_{i,G+1}$$

Else if $C1$ and $C2$ are non-dominated to each other **then**

If rand (0, 1) < 0.5 **then**

$$U_{i,G+1} = V1; \quad F_{i,G+1} = \bar{F}_{i,G+1}; \; CR_{i,G+1} = \overline{CR}_{i,G+1}$$

Else

$$U_{i,G+1} = V2; \quad F_{i,G+1} = \overline{F\_opp}_{i,G+1}; \; CR_{i,G+1} = \overline{CR}_{i,G+1}$$

End if

End if

k) $U_{i,G+1}$ enters the offspring population PopQ_DE (G+1)

End do

Figure 5.4: Pseudo-code of Opposition-based self-adaptive DE mutation scheme in D-OSADE

viduals (x_i^G, x_{r1}^G, x_{r2}^G and x_{r3}^G). For this study, simple averaging of the encoded values will be done instead of weighted averaging from the original OSADE algorithm. This is because the decomposition-based approach is used in D-OSADE instead of the domination-based approach, and hence Pareto ranks and niche counts will not be available for the use of comparison of individuals as used in the weighted averaging of F and CR as performed in the original OSADE

algorithm. The average values of the parameters F and CR in the parent generation as denoted by $\langle F_G \rangle_i$ and $\langle CR_G \rangle_i$ are calculated using the encoded values of F and CR from all the four vectors ($x_i^G, x_r1^G, x_r2^G and x_r3^G$) as per step c of Figure 5.4. With $\langle F_G \rangle_i$ and $\langle CR_G \rangle_i$, the values of F and CR for the child generation $G+1$ which are denoted by $\bar{F_{i,G+1}}$ and $\bar{CR_{i,G+1}}$ are then calculated by the formulae shown in step d of Figure 5.4. The Opposite Number of the mutation factor $\bar{F_{i,G+1}}$ will then be generated in step e of Figure 5.4, and is denoted by $\bar{F\_opp_{i,G+1}}$.

Next, two mutant vectors ($V1$ and $V2$) will be created for the actual DE mutation operation as in step g of Figure 5.4. $\bar{F_{i,G+1}}$ will be used in the mutation operation for the generation of the mutant vector $V1$, whereas $\bar{F\_opp_{i,G+1}}$ will be used for the case of the mutant vector $V2$. The two mutant vectors will then be converted using SPV rule into two equivalent chromosome solutions, namely $C1$ and $C2$, and they will be evaluated according to the problem before being compared to find the dominant one. If both are non-dominated to each other, one of them will be randomly selected. The outcome of the comparison will determine which chromosome will enter the chromosome-based offspring population. If the outcome is favourable towards solution $C1$, the vector $V1$ will enter the DE offspring population, otherwise the vector $V2$ will be the one entering the DE offspring population. At the same time, the encoded value of the mutation factor F in the individual X_i will be updated with $\bar{F_{i,G+1}}$ if the comparison outcome mentioned above favours chromosome $C1$, otherwise the encoded value of the mutation factor will be updated with $\bar{F\_opp_{i,G+1}}$. For the case of crossover rate, the encoded value in individual X_i will be updated with $\bar{CR_{i,G+1}}$ regardless of the comparison outcome.

As the SPV rule is used to handle the conversion of vectors in the DE offspring with floating point values into their equivalent chromosome-based solutions with integer values representing the sequence of the cities visited by the salesmen, every gene in the chromosome will have a unique integer value. As such, there will not be any cases of chromosome-based solutions having repeated or unallocated cities. Hence there will be no requirement for any repair of chromosomes [192] as per what is being performed in all the other algorithms compared in this

study. However, in the event if some salesmen are not being allocated any city in their traveling route, then there will be a penalty given to the objective values of these solutions which do not satisfy all the constraints. This will be done by multiplying the original objective values of these solutions by a constant value K which is set to 10 in the implementation here, and the penalty will also be applied to all the other competing algorithms as well. Once the reproduction stage is completed, the reference point $\mathbf{z}^*$ mentioned above will be updated with the minimum value of the objective functions. Next, every of the chromosome-based solutions y, as converted from the solutions from the opposition-based self-adaptive differential mutation operation, will be compared with all the neighbouring solutions where the inferior solutions will be replaced by the fitter ones.

Local search in the form of EGS will be performed when it is being activated after a certain number of generations depending on the number of cities in the test instance. Once the local search is activated, it will be applied in every generation after that. The entire procedure will be repeated until the stopping criterion is achieved. As mentioned in [192], EGS is a suitable local search meta-heuristic to be employed in the decomposition-based framework due to the fact that aggregation function is readily available in the framework and hence it can be used in the determination of the gradient in EGS. However, as EGS is originally meant for continuous optimization problems, hence a modification was done in [192] so as to adapt the EGS for the MmTSP which is a permutation-based problem. In this modified version of the EGS, N local neighbours are first generated through the swapping of genes in the chromosomes. Aggregation is then done on the chromosome-based solutions with the weighted sum approach by using the set of uniformly distributed weight vectors $\boldsymbol{\lambda}^1, ..., \boldsymbol{\lambda}^S$ derived earlier in the algorithm.

Next, the estimation of the global gradient direction will be performed according to step 5 in Figure 5.1. An offspring will be produced next as per step 6 of Figure 5.1 which requires x_i to be floating point value. However, the chromosome-based solution x_i cannot be directly applied for step 6 as it contains integer values representing the cities. To handle this, an average

cost is calculated between the chromosome solution and its local neighbours, and the cost will be assigned to the x_i in step 6. The offspring q is then updated according to step 6, but q is of floating point value and cannot be used to represent a city. As such, a candidate city that has the nearest q cost value to the original city has to be determined. An example can be used to illustrate this whereby the value of q at a particular instance is 50 with city 1 being the original city, and a salesman is tasked to visit cities 4 and 6. The traveling cost between the original city and the two other cities will first be calculated. If the traveling cost between city 1 and 4 is 150 and 100 between city 1 and 6, then city 6 will be chosen as the candidate city as the traveling cost of 100 is closer to the current cost value of q which is 50 as mentioned above. In step 7 of Figure 5.1, the common mutation step size σ_t that is used in the generation of the offspring q will be adjusted according to the quality of the offspring derived from the estimated gradient. The gradient offspring will also be updated to the population according to step 3 of Figure 5.3. The process goes back to step 2 of Figure 5.3 until the stopping criterion is met.

It is to be noted that diversity of solutions is maintained through the pre-selected weight vectors in this decomposition-based DE algorithm. The concept is similar to what is being utilized in classical aggregation algorithms whereby different weight settings are used to generate an estimated optimal solution for every weight setting. However, the difference is that a set of solutions is maintained in every simulation run for the algorithm here rather than performing several runs as seen in classical aggregation approaches. Even though there is no specific steps for applying elitism, elitism is still implicitly ensured in step 3b of Figure 5.3, whereby the P closest neighbours (parent solutions) are being updated through the comparison of their fitness values with those of the offspring solutions.

5.6 Experimental Design

Comparative studies of the proposed algorithm D-OSADE with some state-of-the-art algorithms were carried out to examine their performance in all the test instances of the MmTSP formulated

Table 5.2: Parameter settings for experiments

Parameter	Setting
Population size, X	Number of cities, n
Number of subproblems, S	X
Number of cities, n	100, 300, 500
Number of salesmen, m	2, 5, 10 for 100 cities 2, 5, 10, 30 for 300 cities 2, 5, 10, 20, 50 for 500 cities
Number of objective functions, T	2 and 5
Number of local neighbours, N	10
Stopping criterion (Number of fitness evaluations)	200,000 for 100 cities; 600,000 for 300 cities; 1,000,000 for 500 cities
Crossover and mutation rate in NSGA-II and MOEA/D	0.8 and 0.005
Initial step size (σ_0) and ε in EGS	300 and 1.8
Local search activation	After 100,000 fitness evaluations for 100 cities After 300,000 fitness evaluations for 300 cities After 500,000 fitness evaluaions for 500 cities

in this study. A total of 24 multi-objective multiple salesmen problem (MmTSP) test instances with different number of objectives T, salesmen m and cities n were studied in this chapter. These problems are represented in the following convention $T5m5n100$, which indicates that the MmTSP test instance is a five-objective problem that involves five salesmen and 100 cities. For these problems, an n x n cost matrix is randomly formed for each objective in the range between 0 and 1000. The proposed algorithm was pitted against five other state-of-the-art algorithms to observe how well they are able to handle the MmTSP whereby the results are compared using the Inverted Generational Distance (IGD) performance metric [217] and Pareto front. IGD is a unary indicator whereby the Euclidean distance of every solution in a reference optimal Pareto front to the obtained Pareto front is being calculated. A smaller IGD implies better proximity and spread. As the optimal solution set to the MmTSP is unknown, estimated optimal Pareto fronts will be derived from the non-dominated solutions found from all the algorithms under comparison in all the simulation runs.

Five state-of-the-art EMO algorithms were chosen for performance comparison with the proposed algorithm. These five algorithms are namely the decomposition-based Estimation of Distribution Algorithm (EDA) with Evolutionary Gradient Search (UMEGS) [192], decomposition-based Estimation of Distribution Algorithm (UMDAD) [192], Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D) [52], Estimation of Distribution Algorithm with

non-dominated sorting genetic algorithm-II (UMGA) [218] and the non-dominated sorting genetic algorithm-II (NSGA-II) [50]. It is to be mentioned that the EDA used in the algorithms mentioned here are based on univariate modeling (UM) [219, 220]. The UMEGS is an evolutionary multi-objective optimization (EMO) algorithm that is formed by hybridizing a decomposition-based EDA and the EGS. The UMDAD is a pure decomposition-based EDA that uses UM. As for the MOEA/D, it is a popular aggregation-based EMO algorithm that basically converts a multi-objective problem into several single-objective problems that are to be handled simultaneously. UMGA is another EDA that is integrated with the NSGA-II. Lastly for the NSGA-II, it is a very popular and widely used MOEA that is based on the concept of domination [50]. Table 5.2 provides the summary of the parameter settings required for the experiments for all the algorithms in this study.

5.7 Results and Discussions

Simulations were carried out to study the performance of the six algorithms applied on the MmTSP with different number of objective functions, salesmen, and cities. In this study, the proposed algorithm D-OSADE was applied on the MmTSP with two and five objective functions which are also varied with different problem settings in terms of the number of salesmen and cities.

5.7.1 Results for Two Objective Test Instances

In this section, the simulation results on the MmTSP instances with two objectives are presented. For ease of visualization, the optimization results (mean $\pm$ standard deviation) for the different test instances are presented in table form. Table 5.3 present the IGD metric for the total traveling cost of all salesmen of the solutions obtained by the different algorithms, and the corresponding IGD metric for the highest traveling cost of any single salesman in the solutions obtained by the algorithms will be presented in Table 5.4. In each table, the best result for every test instance

with different problem settings, in terms of lowest mean IGD value, will highlighted in bold.

From the results shown in Table 5.3, it is seen that D-OSADE generates the best solutions in most of the problem settings. This could be attributed to the complementary effect of the use of the opposition-based self-adaptive DE operator which is strong in global exploration, and the usage of the gradient information through the EGS which enhances local exploitation leading to the enhancement of the algorithm in the search for more diverse solutions. Another observation derived from Table 5.3 is that the total traveling cost increases with the increase in the number of salesmen. This is due to the rise in difficulty of the optimization task when there are more salesmen involved as the optimizers will need to determine the route for every salesman while keeping the total traveling cost to a minimum at the same time [192]. Moreover, all the salesmen will also need to return to the home city, and additional traveling cost may be incurred if the last city assigned to the salesmen is far from the depot. In both D-OSADE and UMEGS, gradient information is used as EGS is embedded in both the algorithms. With the increase in the number of salesmen, the gradient information gets weakened. Therefore the use of local search to exploit the information for the search may become less effective. However, D-OSADE is able to perform better than the UMEGS and this may be attributed to the DE operator which is of strong exploration capability and can result in the promotion of diversity of solutions.

In Table 5.4, IGD metric for the highest traveling cost of any salesman of the solutions generated by the algorithms is presented. It can be observed that both D-OSADE and UMEGS achieve good performance in this aspect in terms of achieving the best IGD results in most of the test instances. This observation suggests that D-OSADE and UMEGS are able to allow a good distribution of the workload among the salesmen in these test instances. Another observation is that MOEA/D is able to produce solutions with the lowest traveling cost for any single salesman for test instances of problem settings of 10 salesmen in both 100 and 300 cities as well as 30 salesmen in 300 cities even though it did not achieve any best results in terms of the total traveling cost of all the salesmen in all the two objective test instances. As for UMDAD, it tends to

produce better outcome for test instances with a more cities (500) but its results are inferior for test instances with lesser cities (100).

In Figure 5.5, the evolved Pareto front of the total traveling cost generated by all the algorithms applied on the MmTSP with two objective functions, two salesmen, and 100 cities is as displayed. From the figure, it is observed that D-OSADE is able to produce a set of diverse solutions with good proximity when compared to the other algorithms. In terms of IGD measurement, D-OSADE has the lowest value for this test instance among all the algorithms, and this indicates better convergence performance and better ability in terms of producing a set of solutions with better diversity when compared with the other algorithms. From the figure, it can also observed that UMEGS produces a diverse set of solutions but is inferior to the other algorithms in terms of proximity for the solutions. On the other hand, UMDAD is seen to have the worst performance as it has poor solution diversity despite having good proximity. It is also observed that the domination-based algorithms (NSGA-II and UMGA) achieve better performance than the decomposition-based algorithms except D-OSADE and UMEGS.

For the similar test instance with two objective functions, two salesmen, and 100 cities, the convergence plots by the different algorithms are as shown in Figure 5.6. It is clear that the convergence speed of D-OSADE is better than all the other algorithms at the initial stage of evolution. After the initial stage, the other algorithms achieve about the same convergence rate as D-OSADE except UMGA. It is observed that UMDAD converges at a slower rate and was outperformed by most of the other algorithms. This could be due to the possibility of poorer diversity preservation ability in UMDAD as seen from its Pareto front in Figure 5.5. The poor diversity preservation in UMDAD is likely to result in the algorithm failing to further explore and exploit the other promising regions in the search space. For the case of UMEGS, it has the similar convergence speed as the UMDAD until the number of fitness evaluations reaches 100,000. This is because both UMDAD and UMEGS are almost similar algorithms except that the latter has EGS being embedded as a form of local search that is activated after a certain

number of fitness evaluations which is 100,000 for the case of this test instance. As such, the convergence speed for UMEGS improved tremendously after local search kicks in. However for D-OSADE, its opposition-based self-adaptive DE mutation operation allows a good exploration and exploitation of the search space and hence giving it an edge over UMEGS from the start of the evolution. With the local search being activated after 100,000 fitness evaluations for D-OSADE, it is observed that there is further improvement in its optimization performance in terms of IGD measurement as there is further enhancement of diversity through the further exploitation of more neighbouring solutions. This in turn helps D-OSADE to become the best performing algorithm during the later stages of evolution.

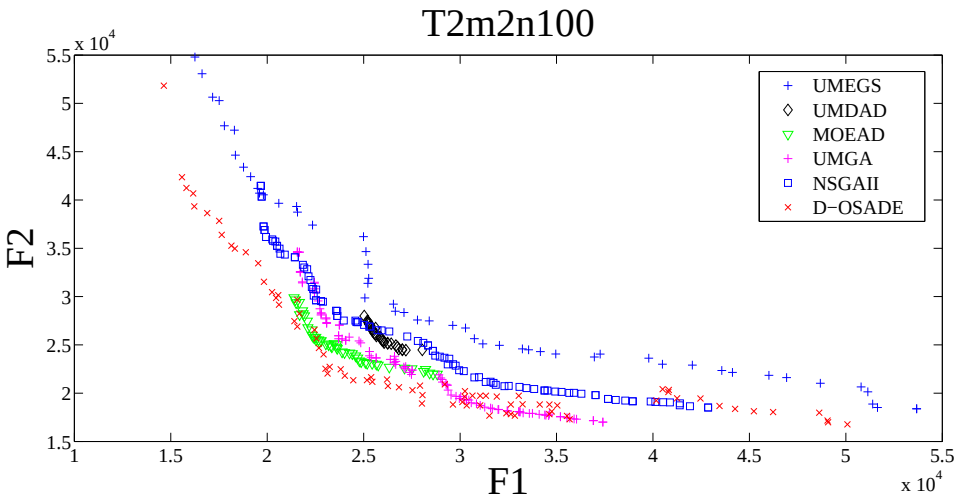

Figure 5.5: Evolved Pareto front of total travelling cost generated by the different algorithms applied to the MmTSP with two objective functions, two salesmen, and 100 cities

Table 5.3: IGD metric for total travelling cost for all salesmen of the solutions obtained by various algorithms for the MmTSP with two objective functions, m salesmen, and n cities

Test Instance	UMEGS	UMDAD	MOEA/D	UMGA	NSGA-II	D-OSADE
T2m2n100	3588±823	10985±473	7538±963	5875±753	6024±628	**2455±647**
T2m5n100	4876±1074	11638±792	8375±695	7784±794	6972±683	**3968±805**
T2m10n100	8875±653	12332±525	8926±728	9628±837	8477±692	**8267±583**
T2m2n300	5648±794	28126±695	28482±1032	26238±1526	25928±1503	**5382±706**
T2m5n300	8103±1775	28632±785	28916±1442	26962±1513	27564±2356	**7753±1548**
T2m10n300	17730±4303	29736±1688	29854±735	32395±1594	28182±2415	**16988±4020**
T2m30n300	**31205±1785**	35374±1073	325842±1684	43288±1337	38635±2508	31865±1802
T2m2n500	**8358±2735**	57362±662	59673±1325	54845±2331	56788±1846	8672±2845
T2m5n500	12114±2428	58384±843	58926±2021	55837±1471	57439±2125	**11428±2133**
T2m10n500	**33680±13853**	58827±786	59947±1855	62018±24532	58003±1730	34090±13039
T2m20n500	52195±2016	59546±684	61549±1642	66438±2348	62824±1056	**50014±1884**
T2m50n500	**58870±2924**	65681±1495	69838±1705	82725±2810	77581±1304	59258±2842

Next, the Pareto front for the total traveling cost generated by all the algorithms applied

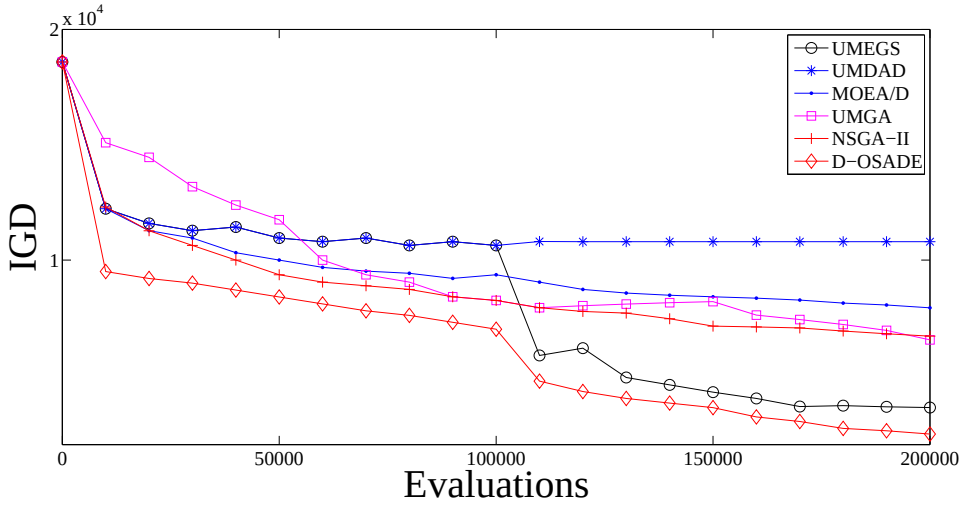

Figure 5.6: Convergence curve of total travelling cost generated by the various algorithms applied to the MmTSP with two objective functions, two salesmen, and 100 cities

on the MmTSP with two objective functions, 20 salesmen, and 500 cities is being displayed in Figure 5.7. From the figure, it is observed that the decomposition-based algorithms (D-OSADE, UMEGS, UMDAD, and MOEA/D) achieve better Pareto fronts than the domination-based algorithms (UMGA and NSGA-II). For the decomposition-based algorithms, D-OSADE and UMEGS are seen to generate more diverse solutions compared to UMDAD. However, the solutions generated by D-OSADE and UMDAD have better proximity than UMEGS. In terms of IGD measurement shown in Table 5.3, it is clear that D-OSADE has a better performance than both UMEGS and UMDAD. This is because D-OSADE has a set of diverse solutions which is much better than UMDAD, and has better proximity when compared to UMEGS. For the case of the domination-based algorithms (UMGA and NSGA-II), they are inferior in proximity as seen from their Pareto fronts in Figure 5.7, and this explains why the IGD values of the decomposition-based algorithm are generally better than the IGD values of the domination-based algorithms. In terms of convergence speed of the different algorithms as indicated in Figure 5.8, there are similar observations between this test instance and the previous one with two objective functions, 2 salesman and 100 cities. From the observations, it can be concluded that the decomposition-based algorithms have better scalability in the number of decision variables as compared to the

domination-based algorithms. For the case of D-OSADE, it has the best IGD value among all the algorithms for this test instance. This is due to its strong exploratory capability which helps to guide the search process, which is then complemented by the use of local information in the evolutionary process which helps the algorithm to further explore and exploit the search space. Hence this leads to D-OSADE being able to achieve both good proximity as well as a diverse set of solutions for this test instance.

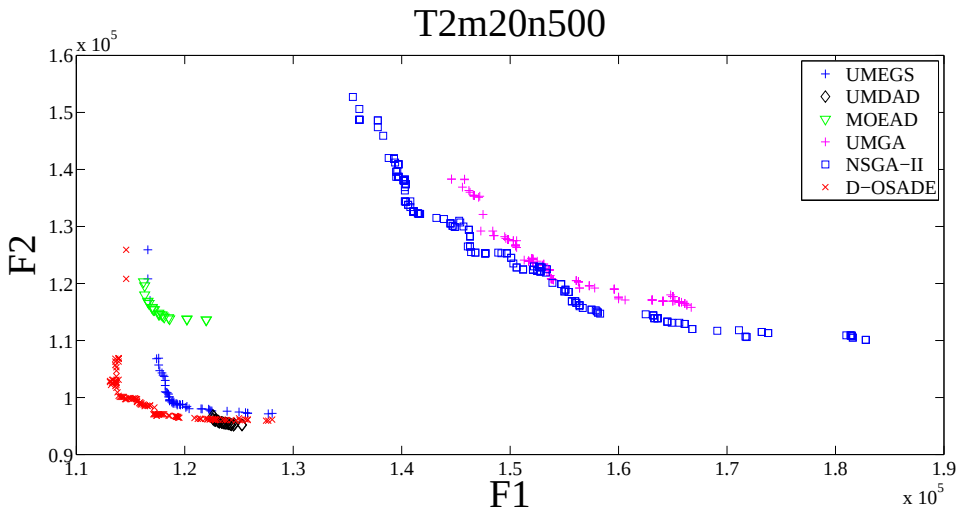

Figure 5.7: Evolved Pareto front of total travelling cost generated by the different algorithms applied to the MmTSP with two objective functions, 20 salesmen, and 500 cities

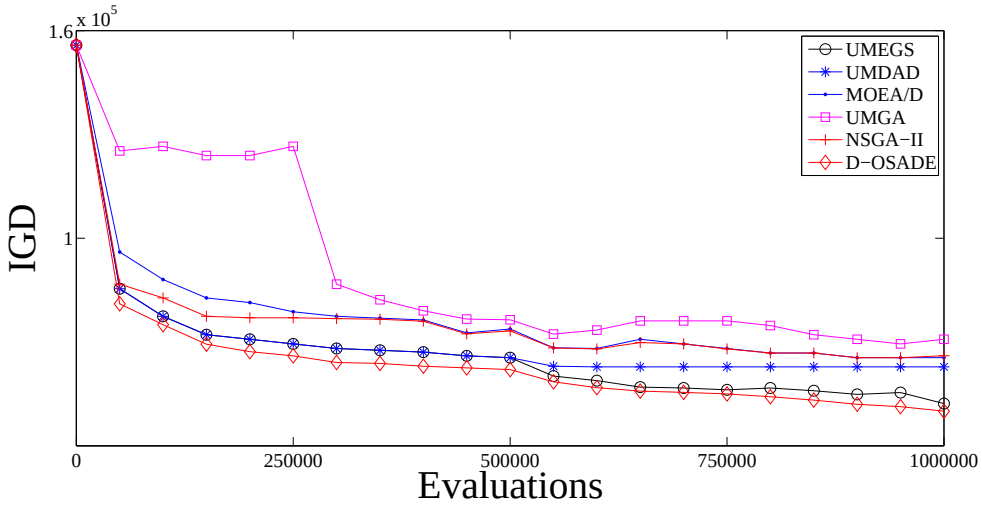

Figure 5.8: Convergence curve of total travelling cost generated by the various algorithms applied to the MmTSP with two objective functions, 20 salesmen, and 500 cities

Table 5.4: IGD metric for highest travelling cost of any salesman of the solutions obtained by various algorithms for the MmTSP with two objective functions, m salesmen, and n cities

Test Instance	UMEGS	UMDAD	MOEA/D	UMGA	NSGA-II	D-OSADE
T2m2n100	1853±594	7459±254	5253±682	4630±614	4430±558	**1537±483**
T2m5n100	1804±543	3045±563	2134±452	2353±463	2265±623	**1685±517**
T2m10n100	1275±502	1453±538	**535±391**	1237±405	1435±349	1053±509
T2m2n300	3502±1218	20534±704	19364±587	17883±1437	18135±1012	**3362±1035**
T2m5n300	3755±1756	12004±375	9913±489	9345±583	8743±980	**3549±1516**
T2m10n300	2014±883	2588±892	**1035±635**	3863±1164	3176±993	1842±728
T2m30n300	1023±358	984±426	**632±424**	2734±553	2395±514	904±464
T2m2n500	**5728±2163**	43459±1188	42845±803	38662±1993	38854±1834	5822±2036
T2m5n500	6354±2819	18043±9835	16034±8735	16534±7452	17135±1238	**6163±2762**
T2m10n500	3939±1663	4073±1402	2948±1432	6703±1303	6654±1504	**3766±1483**
T2m20n500	1788±937	1885±969	1833±1125	4652±670	3645±915	**1703±1053**
T2m50n500	**935±415**	1124±428	1310±573	3254±612	3569±846	1022±528

5.7.2 Results for Five Objective Test Instances

In this section, the number of objectives of the mMTP is increased to five, and the optimization results (mean ± standard deviation) achieved by the different algorithms will be presented. Table 5.5 shows the IGD metric for the total travelling cost of all salesmen of the solutions obtained by the algorithms for the MmTSP with five objective functions and different number of salesmen and cities. The corresponding IGD metric for the highest traveling cost of any single salesman of the solutions obtained by the algorithms will be presented in Table 5.6. In each table, the best result for every test instance with different problem settings, in terms of lowest mean IGD value, will be highlighted in bold.

From Table 5.5, it can be observed that the performance of the decomposition-based algorithms (D-OSADE, UMEGS and MOEA/D) are generally better compared to the domination-based algorithms (UMGA and NSGA-II) for most of the test instances. The superiority in the performance of the decomposition-based algorithms is attributed to the use of aggregation-based fitness assignment for the solutions. In the decomposition-based approach, the aggregated fitness values of the solutions are being compared during the selection of the solutions for the next generation. In this way, solutions of better fitness will always be selected for reproduction in the next generation. However, for the concept of domination as used in UMGA and NSGA-II, the fitness assignment to each solution is performed based on the ranks of domination. For the case here with five objective functions, it is considered as a many-objective problem where most of

the solutions are non-dominated against each other and hence they are given lower domination ranks. As such, it will be more difficult for the promising solutions to be selected for survival and reproduction in the domination-based algorithms. Among the decomposition-based algorithms, D-OSADE and UMEGS are able to produce better solutions than UMDAD and MOEA/D, and this is due to the use of local information derived from the EGS embedded in them which helps to further explore and exploit the search space. However, D-OSADE has the best IGD values in more of the five-objective test instances when compared with the UMEGS, and this demonstrates the good ability of the opposition-based self-adaptive DE operator in the exploration of the search space for promising solutions.

In Table 5.6, the IGD metric for the highest traveling cost of any salesman of the solutions generated by the algorithms is presented. It can be observed that both D-OSADE and UMEGS achieve good performance in this aspect. For MOEA/D, it is able to produce solutions with lowest traveling cost for any single salesman for test instances with problem settings of 100 cities with both five and 10 salesmen even though its performance in terms of total traveling costs is not considered superior. In summary, D-OSADE achieves the overall best performance in most of the five objective test instances under different problem settings as seen from the IGD results from Tables 5.5-5.6. This suggests that D-OSADE is able to achieve the objectives of minimizing the total traveling cost as well as to balance the workload among all the salesmen.

Table 5.5: IGD metric for total travelling cost for all salesmen of the solutions obtained by various algorithms for the MmTSP with five objective functions, m salesmen, and n cities

Test Instance	UMEGS	UMDAD	MOEA/D	UMGA	NSGA-II	D-OSADE
T5m2n100	8675±703	15035±893	8965±253	14983±568	14215±836	**8048±652**
T5m5n100	10537±1202	14208±677	11284±716	18043±805	17835±897	**9855±1024**
T5m10n100	13058±1004	15865±452	14853±952	23955±1388	23782±1225	**12631±959**
T5m2n300	**17452±2305**	44872±1283	46388±1643	56124±2405	55268±1624	18024±2490
T5m5n300	22843±3528	44620±7825	46894±2341	57305±3562	57253±3812	**21953±3227**
T5m10n300	30638±1525	43956±1493	52884±1978	62360±4216	51038±2750	**28945±1238**
T5m30n300	**47114±2013**	53285±1205	74195±4125	96234±2538	94239±3627	48033±2406
T5m2n500	25843±2214	63046±802	96238±463	96378±2937	94035±3210	**24793±2004**
T5m5n500	**24385±5317**	63848±1925	97035±4528	96817±3914	95294±3575	25154±5428
T5m10n500	38240±1263	63724±1136	103205±2952	103852±3820	98253±4783	**37228±1184**
T5m20n500	**42855±2854**	64083±1572	109450±1954	108836±4028	101205±6231	43016±2977
T5m50n500	**70140±4052**	79370±3015	151287±4782	151635±5063	152745±5274	71556±4365

Table 5.6: IGD metric for highest travelling cost of any salesman of the solutions obtained by various algorithms for the MmTSP with five objective functions, m salesmen, and n cities

Test Instance	UMEGS	UMDAD	MOEA/D	UMGA	NSGA-II	D-OSADE
T5m2n100	4983±1130	7012±421	5204±443	9612±624	9623±442	**4811±1032**
T5m5n100	3715±1458	3617±602	**2725±374**	7523±1327	5882±1235	3588±1325
T5m10n100	2023±445	2210±352	**1723±493**	4884±1103	4793±812	1925±484
T5m2n300	9342±1132	27303±998	26302±1304	32167±1445	31248±745	**9036±988**
T5m5n300	8548±2845	12184±558	9610±1388	19335±4193	17738±4828	**8395±2651**
T5m10n300	**3651±1679**	4305±1563	4362±1123	9523±1983	9693±2366	4052±1823
T5m30n300	1326±410	1832±516	3305±510	6546±695	6814±1142	**1223±365**
T5m2n500	**11035±1820**	48838±1668	49635±1834	53037±1772	50127±1635	12226±1633
T5m5n500	**9937±2655**	16932±1134	22346±5882	35238±5108	26304±4020	10233±2912
T5m10n500	5573±2417	5922±1824	12164±3784	17443±2237	16637±1884	**5433±2104**
T5m20n500	2738±1218	3380±1327	8136±1299	14374±1821	12205±2568	**2556±1050**
T5m50n500	**1517±663**	1894±683	4612±873	7102±1002	7203±773	1735±754

5.7.3 Sensitivity Analysis of Bounds for DE Control Parameters

In the proposed algorithm D-OSADE, the opposition-based self-adaptive mutation scheme requires the definition of the lower and upper bounds for the DE parameters (mutation factor F and crossover rate CR). In all the simulation runs performed in the previous sections, the bounds were set to 0.1 and 0.9 for the lower and upper bound respectively for both DE parameters. This setting will be referred to as the original settings in this section. This section investigates the effect when the bounds are varied, and tries to provide a recommended setting for the bounds for D-OSADE.

The results on the T5m2n100, T5m2n300 and T5m2n500 are demonstrated for brevity. For the study of the sensitivity of D-OSADE to the lower and upper bounds, experiments for these test instances were repeated by using different combinations of F and CR. In Table 5.7, the values of the lower and upper bounds for all the different combinations of the bounds are stated. For all the experiments, the bounds were first varied for F but with the original settings being applied to the bounds for CR. Next the bounds for CR were varied with the bounds for F being fixed at the original settings. The other parameters in D-OSADE related to the self-adaptive mechanism remained unchanged.

Table 5.7: Values of bounds for the different combinations (cases) of DE parameters F and CR

	Case									
	1	2	3	4	5	6	7	8	9	10
Lower Bound	0.1	0.3	0.5	0.7	0.1	0.3	0.5	0.1	0.3	0.1
Upper Bound	0.9	0.9	0.9	0.9	0.7	0.7	0.7	0.5	0.5	0.3

In Tables 5.8-5.9, the mean IGD values obtained by D-OSADE for all the experiments using different combinations of the lower and upper bounds of both F and CR are displayed. The values in parentheses beside every IGD value represent the ranks for a particular combination of the lower and upper bounds as compared to the other combinations for the test instance. For every test instance, the top four entries for every case in terms of the lowest IGD values will be marked in boldface in all the tables.

Table 5.8: Mean IGD values for D-OSADE with varying bounds for F for different test instances

	Test Instance		
Case	T5m2n100	T5m2n300	T5m2n500
1	**8048(4)**	**18024(3)**	**24793(2)**
2	8219(5)	18993(7)	25329(6)
3	**7983(2)**	**17645(1)**	**24603(1)**
4	**8012(3)**	**17836(2)**	**24994(4)**
5	8316(7)	**18327(4)**	**24885(3)**
6	8297(6)	18530(5)	25124(5)
7	**7810(1)**	18678(6)	25594(7)
8	8428(8)	19252(9)	25660(8)
9	8471(9)	19003(8)	26017(10)
10	8507(10)	19493(10)	25881(9)

Table 5.9: Mean IGD values for D-OSADE with varying bounds for CR for different test instances

	Test Instance		
Case	T5m2n100	T5m2n300	T5m2n500
1	**8048(3)**	**18024(3)**	**24793(4)**
2	8578(9)	19027(9)	25667(8)
3	8524(8)	19206(10)	25843(9)
4	8706(10)	18803(8)	26018(10)
5	8345(5)	18338(5)	25126(6)
6	8449(7)	18529(6)	24990(5)
7	8401(6)	18661(7)	25439(7)
8	**7783(1)**	**17895(2)**	**24504(3)**
9	**7921(2)**	**17783(1)**	**24018(1)**
10	**8153(4)**	**18265(4)**	**24378(2)**

For the mutation factor F, it is observed that through the use of the bounds under Case 3 (lower bound = 0.5, upper bound = 0.9), there are 2 out of the 3 test instances achieving first ranking in terms of the lowest IGD among all the different combinations for the bounds. As such, Case 3 is considered a better combination for the bounds for the mutation factor F in D-OSADE. As for the case of crossover rate CR, Case 9 (lower bound = 0.3, upper bound = 0.5) is considered to be a better combination for the bounds for the crossover rate as 2 of the 3 test instances achieved first ranking with this combination of the bounds. With this, it is recommended that the bounds for the mutation factor F to be set between 0.5 and 0.9, and

the crossover rate CR to be set between 0.3 and 0.5. Through these recommended bounds, the mutation factor could potentially be adapted to an adequately large value for exploration in the search space, and be adapted to lower values to increase the convergence rate. This in turn helps to strike a balance between exploration and exploitation. As for the crossover rate, the range of 0.3 to 0.5 will allow D-OSADE to achieve good control on which and the number of components to be mutated in every individual of the current population.

5.8 Summary

This chapter studied the potential of an opposition-based self-adaptive differential evolution variant hybridized with local search in dealing with the multi-objective multiple traveling salesman problem (MmTSP). For the MmTSP, it is formulated as a weighted sum of the total traveling cost of all salesman and the highest traveling cost by any single salesman, and this allows the consideration of both the goals of minimizing the total traveling cost as well as to balance the workload among all the salesmen. The proposed algorithm takes into account of both the global information from its differential evolution operation together with the local information in terms of the trajectory of movements for the overall search process. Furthermore, the utilization of the decomposition-based approach of multi-objective optimization also helps the proposed algorithm D-OSADE in generating a set of promising tradeoff solutions in most of the instances of the MmTSP. Comparative studies were conducted between the proposed algorithm and five state-of-the-art MOEAs, and the results demonstrate that D-OSADE is able to scale well in both the decision space and objective space as it is able to generate a set of diverse solutions with good proximity for most of the test instances of the MmTSP used in this chapter.

Chapter 6

Conclusions

6.1 Conclusions

The primary aim of this thesis is to propose novel differential evolution variants that are able to solve a variety of multi-objective optimization problems (MOOPs) effectively. In chapter 3, a memetic algorithm that hybridizes a novel opposition-based self-adaptive differential evolution with the multi-objective evolutionary gradient search has been designed. The novel self-adaptive mechanism based on opposition-based learning as found in OSADE not only eliminates the need of fixing the DE control parameters at the start of evolution which usually requires a tedious trial-and-error process, but also give a higher probability of finding near-optimal settings for the DE control parameters in the proposed algorithm throughout the evolutionary process. As the control parameters of OSADE gets adapted to near-optimal settings in the evolutionary process, better solutions can be generated throughout the evolutionary process as well. Simulation results showed that the proposed algorithm OSADE is able to achieve overall better performance in terms of convergence and diversity over several state-of-the-art algorithms when tested in a wide range of unconstrained continuous multi-objective optimization problems. The performance of OSADE in terms of its scalability in both the number of decision variables and the number of objective functions has also been investigated, and it is observed that OSADE scales well with an

increase in the number of decision variables as well as the number of objective functions when compared to the other state-of-the-art algorithms. These results suggest that OSADE is suitable for use in solving complex and difficult MOOPs. Despite the strengths seen in OSADE, it is seen to be less effective in handling problems with degenerate Pareto front, and this also indicates a potential weakness in differential evolution.

In chapter 4, the optimization performance of a proposed novel grid-based differential evolution algorithm (GrDE) extended from OSADE has been extensively examined by comparing the algorithm with several state-of-the-art algorithms in a suite of many-objective optimization problems. From the simulation results, GrDE is seen to be very competitive with the the other algorithms in terms of achieving a well-approximated and well-distributed solution set for the many-objective test problems. The results suggest that the coupling of the opposition-based self-adaptive mutation scheme with the local mutation scheme in GrDE can lead to a good balance of convergence and diversity which is crucial in solving many-objective problems well. In addition, GrDE is formulated in a grid-based environment which involves the use of certain grid-based criteria. The use of these criteria can help to reflect the information of both convergence and diversity concurrently in a better manner, and this can lead to an increase in the selection pressure within GrDE which in turn helps to derive better solutions in higher objective space. From further investigation, it is also found out that a larger population size is generally preferred over a larger number of generations for the proposed algorithm. In addition, it is also noticed that GrDE shares the same weakness as OSADE, whereby it is seen to be less effective in handling problems with degenerate Pareto front.

In chapter 5, a study has been conducted to examine the capability of OSADE in solving the multi-objective multiple travelling salesmen problem (MmTSP). For this, the mutation scheme from OSADE is incorporated into the decomposition-based approach of multi-objective optimization, which is then hybridized with a multi-point evolutionary gradient search to act as a form of local search. Through the simulation studies, it is found that the proposed algorithm

D-OSADE scales well in both the decision space and objective space as it is able to generate a set of diverse tradeoff solutions with good proximity as indicated by better IGD results achieved when compared to the other competing algorithms in the study. The capability of D-OSADE in terms of generating a sequence of cities in approximate optimal ordering also implies that D-OSADE can be used to deal with any scheduling or logistic problems which are permutation-based in nature. Through the study in this chapter, the use of a differential evolution variant for permutation-based problems is demonstrated and this is of considerable importance and contribution as the differential evolution algorithm is originally intended for continuous optimization.

6.2 Future Work

Based on the research work conducted in this thesis, some possible research directions that deserve future investigations are recommended here. Firstly, further work can be performed to look into the self-adaptation of the population size in the proposed algorithms in this thesis. This is because the population size is also an important control parameter, and it would be useful for the user if this control parameter is not required to be pre-defined at the start of simulation runs. Moreover, if the population can be adapted to optimal settings throughout the evolutionary process, it may lead to the generation of better individuals that will in turn bring upon better optimization performance.

The performance of differential evolution is also greatly dependent on the mutation strategies involved for the generation of trial vectors. Future work can be placed on the research and formulation of effective mutation strategies that possess diverse characteristics, which also means that the employed strategies in the differential evolution operator should be able to demonstrate distinct capabilities when handling different types of problems at different evolution stages during optimization. With this, a better candidate pool can also be created for the differential evolution algorithm. As such, this can be considered as an attractive area for future research for the extension of the proposed algorithms in this thesis.

In chapter 5, the D-OSADE has been proposed to handle the multi-objective multiple traveling salesman problem which is considered a real-life permutation-based problem. With such a potential witnessed in this opposition-based self-adaptive differential evolution algorithm, future work can be placed to look into the use of D-OSADE to handle other scheduling problems in the area of logistics for example. Even though the traveling salesman problem is closely related to other real-world scheduling and logistic problems, the application of D-OSADE in the traveling salesman problem and other real-world scheduling problems may consist of several differences. These differences may arise in the solution representation, constraints, and the level of conflict between objective functions, among others. As such, there is another possible avenue of future work to adapt D-OSADE in general for the handling of different real-work permutation-based problem like school timetable scheduling problems, network routine problems, and gene sequencing problem, just to name a few. When handling these real-world problems, *a priori* knowledge of the problems may need to be taken into account through local search or a specific enhancement operator that is to be used during the optimization process. With the incorporation of this knowledge, the search capability of the algorithm may be better enhanced to allow better solutions to be generated.

This thesis has implemented opposition-based self-adaptive differential evolution variants in general, to study global continuous MOOPs, scalable MOOPs, many-objective problems as well as permutation-based MOOPs. Many other characteristics and issues of MOOPs, which are out of the scope of this thesis, could be studied in future. These issues include multi-modality, presence of constraints, uncertainty, and framework. Multi-modality is always a challenging issue in optimization, and efforts can be channeled to look into this area. Besides this, MOOPs in real-life applications often come with constraints, and hence the area of constrained optimization can also be studied.

Many cost functions of real-world problems are uncertain in nature. The cost functions may be subject to noise in the objective space, which is commonly known as noisy MOOPs.

The objective functions may also be subject to noise in the decision space, and this is commonly referred to as robust MOOPs. In the cost functions of some MOOPs, the Pareto optimal solutions may vary over time, and this is commonly known as dynamic MOOPs. Studies of self-adaptive DE algorithms in these aspects are considerably lacking, and therefore they can be explored in future work. As for the framework issue, the proposed algorithms in this thesis are being developed in both domination-based and decomposition-based approaches. However, there has been no study of the development of self-adaptive DE algorithms using the preference-based approach. Hence, this is also a possible research direction for exploration in the future.

Last but not least, the concept of the novel self-adaptive scheme proposed in this thesis may also be employed in other population-based algorithms. The self-adaptive scheme in this thesis is general enough to used for the adaptation of the control parameters in other population-based algorithms like Genetic Algorithms (GAs) or the Particle Swarm Optimization (PSO). Thus, this is also a promising research direction that is worthy of exploration in the future.

Bibliography

[1] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*. Addison-Wesley, 1989.

[2] P. Husbands, "Genetic algorithms in optimization and adaptation," in *Advances in parallel algorithms* (L. Kronsjö and D. Shumsherudin, eds.), ch. 8, pp. 227–276, John Wiley and Sons, 1992.

[3] K. Deb, *Multi-objective optimization using evolutionary algorithms*. Chichester John Wiley and Sons, 2001.

[4] K. C. Tan, E. F. Khor, and T. H. Lee, *Multiobjective evolutionary algorithms and applications*. Springer, 2005.

[5] J. Carretero and F. Xhafa, "Using genetic algorithms for scheduling jobs in large scale grid applications," *Journal of Technological and Economic Development*, vol. 12, no. 1, pp. 11–17, 2006.

[6] D. Erni, D. Wiesmann, M. Spühler, S. Hunziker, E. Moreno, B. Oswald, J. Fröhlich, and C. Hafner, "Application of evolutionary optimization algorithms in computational optics," *The Applied Computational Electromagnetics Society Journal*, vol. 15, no. 2, 2000.

[7] E. Alba, "Evolutionary algorithms in telecommunications," in *IEEE Mediterranean Electrotechnical Conference*, pp. 795–798, 2006.

[8] Y. Wu, L. Xu, and J. Xue, "Improved multiobjective particle swarm optimization for environmental/economic dispatch problem in power system," in *Proceedings of the Second International Conference on Advances in Swarm Intelligence*, pp. 49–56, 2011.

[9] M. Pant, R. Thangaraj, C. Grosan, and A. Abraham, "Hybrid differential evolution - particle swarm optimization algorithm for solving global optimization problems," in *Third International Conference on Digital Information Management*, pp. 18–24, 2008.

[10] N. Dong and Y. Wang, "A memetic differential evolution algorithm based on dynamic preference for constrained optimization problems," *Journal of Applied Mathematics*, vol. 2014, 2014.

[11] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing*, vol. 9, no. 6, pp. 448–462, 2005.

[12] F. Xue, A. C. Sanderson, P. P. Bonissone, and R. J. Graves, "Fuzzy logic controlled multiobjective differential evolution," in *Proceedings of the 14th IEEE International Conference on Fuzzy Systems*, pp. 720–725, 2005.

[13] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 945–958, 2009.

[14] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Computing*, vol. 10, no. 8, pp. 673–686, 2006.

[15] E. Zitzler, *Evolutionary algorithms for multiobjective optimization: Methods and applications*. Ph.D. Dissertation, Swiss Federal Institute of Technology, Zurich, 1999.

[16] P. Dasgupta, P. P. Chakrabarti, and S. C. Desarkar, *Multiobjective heuristic search: An introduction to intelligent search methods for multicriteria optimization*. Springer, 1999.

[17] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: Analyzing the state-of-the-art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125–147, 2000.

[18] C. K. Goh and K. C. Tan, *Evolutionary multi-objective optimization in uncertain environments: Issues and algorithms*. Springer, 2009.

[19] C. Y. Cheong, *Evolutionary multi-objective optimization in scheduling problems*. Ph.D. Dissertation, National University of Singapore, 2009.

[20] D. A. Coley, *An introduction to genetic algorithms for scientists and engineers*. World Scientific Publishing, 1999.

[21] R. Sivaraj and T. Ravichandran, "A review of selection methods in genetic algorithm," *International Journal of Engineering Science and Technology*, vol. 3, no. 5, pp. 3792–3797, 2011.

[22] K. Deb and H.-G. Beyer, "Self-adaptive genetic algorithms with simulated binary crossover," *Evolutionary Computation*, vol. 9, no. 2, pp. 197–221, 2001.

[23] R. Dawkins, *The selfish gene*. Oxford University Press, New York, 1976.

[24] R. Storn and K. Price, "Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces," Technical Report TR-95-012, ICSI, 1995.

[25] R. Storn and K. Price, "Differential evolution - a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[26] K. Zielinski and R. Laur, "Variants of differential evolution for multi-objective optimization," in *IEEE Symposium on Computational Intelligence in Multicriteria Decision Making (MCDM'07)*, pp. 91–98, 2007.

[27] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. C. Coello, "A comparative study of differential evolution variants for global optimization," in *Proceedings of the 2006 conference on Genetic and evolutionary computation*, pp. 485–492, 2006.

[28] T. Robic and F. B., "DEMO: Differential evolution for multiobjective optimization," in *Proceedings of the Third International Conference on Evolutionary Multi-criterion Optimization*, pp. 520–533, 2005.

[29] B. V. Babu and M. M. L. Jehan, "Differential evolution for multi-objective optimization," in *IEEE Congress on Evolutionary Computation*, pp. 2696–2703, 2003.

[30] H. R. Tizhoosh, "Opposition-based learning: A new scheme for machine intelligence," in *Proceedings of the International Conference on Computational Intelligence for Modeling, Control and Automation*, pp. 695–701, 2005.

[31] M. A. Ahandani and H. Alavi-Rad, "Opposition-based learning in the shuffled differential evolution algorithm," *Soft Computing*, vol. 16, no. 8, pp. 1303–1337, 2012.

[32] H. Wang, Y. Liu, S. Y. Zeng, H. Li, and C. H. Li, "Opposition-based particle swarm optimization with Cauchy mutation," in *IEEE Congress on Evolutionary Computation*, pp. 4750–4756, 2007.

[33] M. El-Abd, "Generalized opposition-based artificial bee colony algorithm," in *IEEE Congress on Evolutionary Computation*, pp. 1–4, 2012.

[34] M. Kaucic, "A multi-start opposition-based particle swarm optimization algorithm with adaptive velocity for bound constrained global optimization," *Journal of Global Optimization*, vol. 55, no. 1, pp. 165–188, 2013.

[35] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "Opposition-based differential evolution," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 64–79, 2008.

[36] A. Tuson and P. Ross, "Adapting operator settings in genetic algorithms," *Evolutionary Computation*, vol. 6, no. 2, pp. 161–184, 1998.

[37] J. Gomez, D. Dasgupta, and F. Gonzalez, "Using adaptive operators in genetic search," in *Proceedings of the Genetic and Evolutionary Computation - GECCO2003*, pp. 1580–1581, 2003.

[38] B. R. Julstrom, "What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm," in *Proceedings of the 6th International Conference on Genetic Algorithms*, pp. 81–87, 1995.

[39] J. E. Smith and T. C. Fogarty, "Operator and parameter adaptation in genetic algorithms," *Soft Computing*, vol. 1, no. 2, pp. 81–87, 1997.

[40] H. G. Beyer, "Toward a theory of evolution strategies: Self-adaptation," *Evolutionary Computation*, vol. 3, no. 3, pp. 311–347, 1996.

[41] H. G. Beyer and K. Deb, "On self-adaptive features in real-parameter evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 3, pp. 250–270, 2001.

[42] K. Mietten, *Nonlinear multiobjective optimization*. Boston: Kluwer, 1999.

[43] L. Rachmawati and D. Srinivasan, "Multiobjective evolutionary algorithm with controllable focus on the knees of the Pareto front," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 4, pp. 810–824, 2009.

[44] L. Thiele, K. Miettinen, P. J. Korhonen, and J. M. Luque, "A preference-based evolutionary algorithm for multi-objective optimization," *Evolutionary Computation*, vol. 17, no. 3, pp. 411–436, 2009.

[45] A. Zhou, B. Qu, H. Li, S. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: A survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, pp. 32–49, 2011.

[46] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multi-objective optimization: formulation, discussion and generalization," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, pp. 416–423, 1993.

[47] N. Srinivas and K. Deb, "Multiobjective optimization using nondominated sorting in genetic algorithms," *Evolutionary Computation*, vol. 2, no. 3, pp. 221–248, 1994.

[48] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: A comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, p. 257271, 1999.

[49] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the strength Pareto evolutionary algorithm," Technical Report 103, Computer Engineering and Networks Laboratory, Swiss Federal Institute of Technology, Zurich, Switzerland, 2001.

[50] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[51] H. Ishibuchi, T. Yoshida, and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *IEEE Transaction on Systems, Man, and Cybernetics: Part C (Applications and Reviews)*, vol. 28, no. 3, pp. 204–223, 1998.

[52] Q. Zhang and H. Li, "MOEA/D: A multiobjective evolutionary algorithm based on decomposition," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 712–731, 2007.

[53] H. Li and Q. Zhang, "Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 284–302, 2009.

[54] A. J. Nebro and J. J. Durillo, "A study of the parallelization of the multi-objective metaheuristic MOEA/D," in *Fourth International Conference on Learning and Intelligent Optimization*, pp. 303–317, 2010.

[55] P. C. Chang, S. H. Che, Q. Zhang, and J. L. Lin, "MOEA/D for flowshop scheduling problems," in *IEEE Congress on Evolutionary Computation*, pp. 1433–1438, 2008.

[56] T. J. Yuen and R. Raml, "Comparison of computational efficiency of MOEA/D and NSGA-II for passive vehicle suspension optimization," in *Proceedings of 24th European Conference on Modelling and Simulation*, pp. 219–225, 2010.

[57] S. Pal, B. Qu, S. Das, and P. N. Suganthan, "Optimal synthesis of linear antenna arrays with multi-objective differential evolution," *Progress in Electromagnetics Research B*, vol. 21, pp. 87–111, 2010.

[58] A. Konstantinidis, C. Charalambous, A. Zhou, and Q. Zhang, "Multi-objective mobile agent-based sensor network routing using MOEA/D," in *IEEE Congress on Evolutionary Computation*, pp. 1–8, 2010.

[59] K. Price, "Genetic annealing," *Dr Dobb's Journal*, pp. 127–132, 1994.

[60] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A parameter study for differential evolution," *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, vol. 10, pp. 293–298, 2002.

[61] J. Rönkkönen, S. Kukkonen, and K. Price, "Real-parameter optimization with differential evolution," in *IEEE Congress on Evolutionary Computation*, pp. 506–513, 2005.

[62] S. Das, A. Konar, and U. K. Chakraborty, "Two improved differential evolution schemes for faster global search," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pp. 991–998, 2005.

[63] D. Zaharie, "Control of population diversity and adaptation in differential evolution algorithms," in *9th International Conference on Soft Computing*, pp. 41–46, 2003.

[64] D. Zaharie and D. Petcu, "Adaptive Pareto differential evolution and its parallelization," in *5th International Conference on Parallel Processing and Applied Mathematic*, pp. 261–268, 2003.

[65] A. H. A., "The self-adaptive Pareto differential evolution algorithm," in *IEEE Congress on Evolutionary Computation*, pp. 831–836, 2002.

[66] M. G. H. Omran, A. Salmsan, and A. P. Engelbrecht, "Self-adaptive differential evolution," in *Proceedings of International Conference in Computational Intelligence and Security*, pp. 192–199, 2005.

[67] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 646–657, 2006.

[68] K. Price, "Differential evolution vs the functions of the 2nd iceo," in *IEEE Congress on Evolutionary Computation*, pp. 153–157, 1997.

[69] H.-Y. Fan and J. Lampinen, "A trigonometric mutation operation to differential evolution," *Journal of Global Optimization*, vol. 27, no. 1, pp. 105–129, 2003.

[70] K. Price, *An introduction to differential evolution*. McGraw-Hill, 1999.

[71] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer, 2005.

[72] V. Feoktistov and S. Janaqi, "Generalization of the strategies in differential evolution," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, p. 165, 2004.

[73] X. Yao, Y. Liu, and G. Lin, "Evolutionary programming made faster," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 82–102, 1999.

[74] W.-J. Zhang and X.-F. Xie, "DEPSO: Hybrid particle swarm with differential evolution operator," in *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, pp. 3816–3821, 2003.

[75] S. Das, A. Konar, and U. K. Chakraborty, "Improving particle swarm optimization with differentially perturbed velocity," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pp. 177–184, 2005.

[76] J. Sun, Q. Zhang, and E. Tsang, "DE/EDA: A new evolutionary algorithm for global optimization," *Information Sciences*, vol. 169, no. 3-4, pp. 249–262, 2005.

[77] D. Zaharie, "A multi-population differential evolution algorithm for multi-modal optimization," in *10th International Conference on Soft Computing*, pp. 17–22, 2004.

[78] Z. V. Hendershot, "A differential evolution algorithm for automatically discovering multiple global optima in multidimensional, discontinuous spaces.," in *Proceedings of the Fifteenth Midwest Artificial Intelligence and Cognitive Sciences Conference*, pp. 92–97, 2004.

[79] R. Thomsen, "Multimodal optimization using crowding-based differential evolution," in *IEEE Congress on Evolutionary Computation*, pp. 1382–1389, 2004.

[80] R. Storn, "Differential evolution design of an iir-filter," in *IEEE Congress on Evolutionary Computation*, pp. 268–273, 1996.

[81] H. K. Kim, J. K. Chong, K. Y. Park, and D. A. Lowther, "Differential evolution strategy for constrained global optimization and application to practical engineering problems," *IEEE Transactions on Magnetics*, vol. 43, no. 4, pp. 1565–1568, 2007.

[82] V. P. Plagianakos and M. N. Vrahatis, "Training neural networks with 3-bit integer weights," in *Proceedings of the 1999 conference on Genetic and evolutionary computation*, pp. 910–915, 1999.

[83] J. G. Marin-Blazquez, Q. Shen, and A. Tuson, "Tuning fuzzy membership functions with neighbourhood search techniques: A comparative study," in *Proceedings of the 3rd IEEE International Conference on Intelligent Engineering Systems*, pp. 337–342, 1999.

[84] Y. C. Lin, K. S. Hwang, and F.-S. Wang, "Plant scheduling and planning using mixed-integer hybrid differential evolution with multiplier updating," in *IEEE Congress on Evolutionary Computation*, pp. 593–600, 1996.

[85] K. Rzadca and F. Seredynski, "Heterogeneous multiprocessor scheduling with differential evolution," in *IEEE Congress on Evolutionary Computation*, pp. 2840–2847, 2005.

[86] P. Thomas and D. Vernon, "Image registration by differential evolution," in *Proceedings of the First Irish Machine Vision and Image Processing Conference*, pp. 221–225, 1997.

[87] M. G. H. Omran, A. P. Engelbrecht, and S. A., "Differential evolution methods for unsupervised image classification," in *IEEE Congress on Evolutionary Computation*, pp. 966–973, 2005.

[88] D. K. Tasoulis, V. P. Plagianakos, and M. N. Vrahatis, "Differential evolution algorithms for finding predictive gene subsets in microarray data," in *3rd International Federation for Information Processing (IFIP) Conference on Artificial Intelligence Applications and Innovations*, pp. 484–491, 2006.

[89] D. W. Corne, J. D. Knowles, and M. J. Oates, "The Pareto envelope-based selection algorithm for multiobjective optimization," in *Proceedings of the sixth international conference on Parallel Problem Solving from Nature*, pp. 839–848, 2000.

[90] M. Lahanas, N. Milickovic, D. Baltas, and N. Zamboglou, "Application of multiobjective evolutionary algorithms for dose optimization problems in brachytherapy," in *Proceedings of the first international conference on Evolutionary Multi-Criterion Optimization*, pp. 574–587, 2001.

[91] B. W. Silverman, *Density estimation for statistics and data analysis*. Chapman and Hall, 1986.

[92] D. V. Arnold and R. Salomom, "Evolutionary gradient search revisited," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 4, pp. 480–495, 2007.

[93] D. S. Liu, K. C. Tan, C. K. Goh, and W. K. Ho, "A multiobjective memetic algorithm based on particle swarm optimization," *IEEE Transactions on Systems, Man, and Cybernetics: Part B (Cybernetics)*, vol. 37, no. 1, pp. 42–50, 2007.

[94] C. K. Goh, Y. S. Ong, K. C. Tan, and E. J. Teoh, "An investigation on evolutionary gradient search for multi-objective optimization," in *IEEE Congress on Evolutionary Computation*, pp. 3741–3746, 2008.

[95] J. D. Knowles, "PaeEGO: A hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 1, pp. 50–66, 2006.

[96] E. Zitzler, T. L., M. Laumanns, C. M. Fonseca, and V. G. Fonseca, "Performance assessment of multiobjective optimizers: An analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.

[97] A. Jaszkiewicz, "On the performance of multiple-objective genetic local search on the 0/1 knapsack problem - A comparative experiment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 4, pp. 402–412, 2002.

[98] D. A. Van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithm test suites," *ACM Symposium on Applied Computing*, pp. 351–357, 1999.

[99] D. A. Van Veldhuizen and G. B. Lamont, "On measuring multiobjective evolutionary algortihm performance," in *IEEE Congress on Evolutionary Computation*, pp. 204–211, 2000.

[100] K. Deb, "Multi-objective genetic algorithm: Problem difficulties and construction of test problems," *Evolutionary Computation*, vol. 7, no. 3, pp. 205–230, 1999.

[101] D. A. Van Veldhuizen, *Multiobjective evolutinoary algorithm: Classification, analyzes, and new innovations*. Ph.D. Dissertation, Air Force Institute of Technology , Wright-Patterson AFB, 1999.

[102] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff, "On test functions for evolutionary multi-objective optimization," in *Proceedings of the Eighth International Conference on Parallel Problem Solving from Nature*, pp. 792–802, 2004.

[103] V. L. Huang, A. K. Qin, K. Deb, E. Zitzler, P. N. Suganthan, J. J. Liang, M. Preuss, and S. Huband, "Problem definitions for performance assessment of multi-objective optimizatio algorithms," Technical Report, Nanyang Technological University, Singapore, 2007.

[104] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: Empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.

[105] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multi-objective optimization," KanGAL Report 2001001, Kanpur Genetic Algorithms Lab, Indian Institute of Technology, 2001.

[106] Q. Zhang, A. Zhou, P. N. Zhao, S. Suganthan, W. Liu, and S. Tiwari, "Multiobjective optimization test instances for the CEC 2009 special session and competition," Technical Report CES-487, The School of Computer Science and Electronic Engineering, 2009.

[107] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.

[108] C. A. Coello Coello, "Evolutionary multi-objective optimization: A historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, 2006.

[109] E. Zitzler, T. L., and J. Bader, "On set-based multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 1, pp. 58–79, 2010.

[110] J. H. Kim and L. C. H., "Multi-objective evolutionary generation process for specific personalties of artificial creature," *IEEE Computational Intelligence Magazine*, vol. 3, no. 1, pp. 45–53, 2008.

[111] H. A. Abbass and R. Sarker, "The Pareto differential evolution algorithm," *International Journal of Artificial Intelligence Tools*, vol. 11, no. 4, pp. 531–552, 2002.

[112] S. Kukkonen and J. Lampinen, "GDE3: The third evolution step of generalized differential evolution," in *IEEE Congress on Evolutionary Computation*, pp. 443–450, 2005.

[113] F. Xue, A. C. Sanderson, and R. J. Graves, "Pareto-based multiobjective differential evolution," in *IEEE Congress on Evolutionary Computation*, pp. 228–235, 2003.

[114] W. B. Langdon and R. Poli, "Evolving problems to learn about particle swarm optimizers and other search algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 5, pp. 561–578, 2007.

[115] R. Denysiuk, L. Costa, and I. E. Santo, "Many-objective optimizatio using differential evolution with variable-wise mutation restriction," in *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pp. 591–598, 2013.

[116] N. Noman and H. Iba, "Accelerating differential evolution using an adaptive local search," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 107–125, 2008.

[117] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighbourhood based mutation operator," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 526–553, 2009.

[118] J. Digalakis and K. Margaritis, "Performance comparison of memetic algorithms," *Journal of Applied Mathematics and Computation*, vol. 158, no. 1, pp. 237–252, 2004.

[119] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," *Evolutionary Computation*, vol. 12, no. 3, pp. 273–302, 2004.

[120] J. M. Renders and H. Bersini, "Hybridizing genetic algorithms with hill-climbing methods for global optimization: Two possible ways," in *IEEE Congress On Evolutionary Computation*, pp. 312–317, 1994.

[121] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 151–165, 2011.

[122] R. L. Galski, F. L. Sousa, F. M. Ramos, and I. Muraoka, "Application of a new hybrid evolutionary strategy to spacecraft thermal design," in *Proceedings of Genetic and Evolutionary Computation Conference*, 2004.

[123] P. Attaviriyanupap, H. Kita, E. Tanaka, and J. Hasegawa, "A hybrid ep and sqp for dynamic economic dispatch with nonsmooth fuel cost function," *IEEE Transactions on Power Systems*, vol. 17, no. 2, pp. 411–416, 2002.

[124] Z. S. Kapekan, D. A. Savic, and G. A. Walters, "A hybrid inverse transient model for leakage detection and roughness calibration in pipe networks," *Journal of Hydraulic Research*, vol. 41, no. 5, pp. 481–492, 2003.

[125] R. Thangaraj, M. Pant, A. Abraham, and Y. Badr, "Hybrid evolutionary algorithm for solving global optimization problems," in *Hybrid Artificial Intelligence Systems*, vol. 5572, pp. 310–318, Springer, 2009.

[126] C. Y. Chung, C. H. Liang, K. P. Wong, and X. Z. Duan, "Hybrid algorithm of differential evolution and evolutionary programming for optimal reactive power flow," *IET Generation, Transmission and Distribution*, vol. 4, no. 1, pp. 84–93, 2010.

[127] S. Das, A. Konar, and C. U. K., "Annealed differential evolution," in *IEEE Congress on Evolutionary Computation*, pp. 1926–1933, 2007.

[128] S. Tsutsui, M. Yamamura, and T. Higuchi, "Multi-parent recombination with simplex crossover in real coded genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation*, pp. 657–664, 1999.

[129] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," in *IEEE Congress on Evolutionary Computation*, pp. 1785–1791, 2005.

[130] V. L. Huang, A. K. Qin, P. N. Suganthan, and M. F. Tasgetiren, "Multi-objective optimization based on self-adaptive differential evolution algorithm," in *IEEE Congress on Evolutionary Computation*, pp. 3601–3608, 2007.

[131] V. L. Huang, S. Z. Zhao, R. Mallipeddi, and P. N. Suganthan, "Multi-objective optimization using self-adaptive differential evolution algorithm," in *IEEE Congress on Evolutionary Computation*, pp. 190–194, 2009.

[132] A. Zamuda, J. Brest, B. Boskovic, and V. Zumer, "Differential evolution for multiobjective optimization with self-adaptation," in *IEEE Congress on Evolutionary Computation*, pp. 3617–3624, 2007.

[133] C. K. Goh, K. C. Tan, D. S. Liu, and S. C. Chiam, "A competitive and cooperative coevolutionary approach to multi-objective particle swarm optimization algorithm design," *European Journal of Operational Research*, vol. 202, no. 1, pp. 42–54, 2010.

[134] A. W. Iorio and X. Li, "Solving rotated multiobjective optimization problems using differential evolution," in *Procdeeding of Artificial Intelligence: Advances in Artificial Intelligence*, pp. 861–872, 2004.

[135] R. Angira and B. V. Babu, "Non-dominated sorting differential evolution (nsde): An extension of differential evolution for multi-objective optimization," in *Proceedings of the 2nd Indian International Conference on Artificial Intelligence*, pp. 1428–1443, 2005.

[136] C. A. Coello Coello and C. N. C., "Solving multiobjective optimization problems using an artificial immune system," *Genetic programming and Evolvable Machines*, vol. 6, no. 2, pp. 163–190, 2005.

[137] O. Schutze, X. Esquivel, A. Lara, and C. A. Coello Coello, "Using the average hausdorff distance as a performance measure in evolutionary multiobjective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 4, pp. 504–522, 2012.

[138] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable multi-objective optimization test problems," in *IEEE Congress on Evolutionary Computation*, pp. 825–830, 2002.

[139] M. Garza-Fabre, G. Toscano-Pulido, and C. A. C. Coello, "Ranking methods for many-objective optimization," in *Proceedings of the 8th Mexican international conference on Artificial Intelligence*, pp. 633–645, 2009.

[140] N. Ishibuchi, H. Tsukamoto and Y. Nojima, "Evolutionary many-objective optimization: A short review," in *IEEE Congress on Evolutionary Computation*, pp. 2419–2426, 2008.

[141] X. Zou, Y. Chen, M. Liu, and L. Kang, "A new evolutionary algorithm for solving many-objective optimization problems," *IEEE Transactions on Systems, Man and Cybernetics: Part B Cybernetics*, vol. 38, no. 5, pp. 1402–1412, 2008.

[142] D. Saxena, J. Duro, A. Tiwari, K. Deb, and Q. Zhang, "Objective reduction in many-objective optimization: Linear and nonlinear algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 1, pp. 77–99, 2013.

[143] H. K. Singh, A. Isaacs, and T. Ray, "A Pareto corner search evolutionary algorithm and dimensionality reduction in many-objective optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 4, pp. 539–556, 2011.

[144] M. Farina and P. Amato, "A fuzzy definition of optimality for many-criteria optimization problems," *IEEE Transactions on Systems, Man and Cybernetics: Part A (Systems and Humans)*, vol. 34, no. 3, pp. 315–326, 2004.

[145] P. Fleming, R. Purshouse, and R. Lygoe, "Many-objective optimization: An engineering design perspective," in *Proceedings of the Third International Conference on Evolutionary Multi-criterion Optimization*, pp. 14–32, 2005.

[146] J. G. Herrero, A. Berlanga, and J. M. M. Lopez, "Effective evolutionary algorithms for many-specifications attainment," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 151–168, 2009.

[147] Y. Jin and B. Sendhoff, "A systems approach to evolutionary multiobjective structural optimization and beyond," *IEEE Computational Intelligence Magazine*, vol. 4, no. 3, pp. 62–76, 2009.

[148] A. Sulflow, N. Drechsler, and R. Drechsler, "Robust multi-objective optimization in high dimensional spaces," in *Proceedings of the Fourth International Conference on Evolutionary Multi-criterion Optimization*, pp. 715–726, 2007.

[149] V. Khara, X. Yao, and K. Deb, "Performance scaling of multi-objective evolutionary algorithms," in *Proceedings of the Second International Conference on Evolutionary Multi-criterion Optimization*, pp. 367–390, 2003.

[150] R. C. Purshouse and P. J. Fleming, "Evolutionary many-objective optimization: an exploratory analysis," in *IEEE Congress on Evolutionary Computation*, pp. 2066–2073, 2003.

[151] M. Laumanns, L. Thiele, K. Deb, and E. Zitzler, "Combining convergence and diversity in evolutionary multiobjective optimization," *Evolutionary Computation*, vol. 10, no. 3, pp. 263–282, 2002.

[152] D. F. Pierro, S. T. Khu, and D. A. Savic, "An investigation on preference order ranking scheme for multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 17–45, 2007.

[153] M. Köppen and K. Yoshida, "Substitute distance assignments in NSGA-II for handling many-objective optimization problems," in *Proceedings of the Fourth International Conference on Evolutionary Multi-criterion Optimization*, pp. 727–741, 2007.

[154] S. Kukkonen and J. Lampinen, "Ranking dominance and many-objective optimization," in *IEEE Congress on Evolutionary Computation*, pp. 3983–3990, 2007.

[155] M. Li, J. Zheng, K. Li, Q. Yuan, and R. Shen, "Enhancing diversity for average ranking method in evolutionary many-objective optimization," in *Proceedings of the 11th international conference on Parallel Problem Solving from Nature*, pp. 647–656, 2010.

[156] M. Garza-Fabre, G. Toscano-Pulido, and C. A. C. Coello, "Two novel approaches for many-objective optimization," in *IEEE Congress on Evolutionary Computation*, pp. 1–8, 2010.

[157] S. Mostaghim and H. Schmeck, "Distance based ranking in many-objective particle swarm optimization," in *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*, pp. 753–762, 2008.

[158] U. K. Wickramasinghe and X. Li, "Using a distance metric to guide pso algorithms for many-objective optimization," in *Proceedings of the 11th annual conference on Genetic and evolutionary computation*, pp. 667–674, 2009.

[159] D. W. Corne and J. 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*, pp. 773–780, 2007.

[160] A. L. Jaimes and C. A. C. Coello, "Study of preference relations in many-objective optimization," in *Proceedings of the 11th annual conference on Genetic and evolutionary computation*, pp. 611–618, 2009.

[161] N. Beume, B. Naujoks, and M. Emmerich, "SMS-EMOA: Multiobjective selection based on dominated hypervolume," *European Journal of Operational Research*, vol. 181, no. 3, pp. 1653–1669, 2007.

[162] E. Zitzler and S. Kunzli, "Indicator-based selection in multiobjective search," in *Proceedings of the 8th international conference on Parallel Problem Solving from Nature*, pp. 832–842, 2004.

[163] E. J. Hughes, "MSOPS-II: A general purpose many-objective optimiser," in *IEEE Congress on Evolutionary Computation*, pp. 3944–3951, 2007.

[164] H. Ishibuchi, T. Doi, and Y. Nojima, "Incorporation of scalarizing fitness function into evolutionary multiobjective optimization algorithms," in *Proceedings of the 9th international conference on Parallel Problem Solving from Nature*, pp. 493–502, 2006.

[165] S. F. Adra and P. J. Fleming, "Diversity management in evolutionary many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 2, pp. 183–195, 2011.

[166] S. F. Adra and P. J. Fleming, "On the evolutionary optimization of many conflicting objectives," *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 6, pp. 770–784, 2007.

[167] T. Wagner, N. Beume, and B. Naujoks, "Pareto-, Aggregation-, and Indicator-based methods in many-objective optimization," in *Proceedings of the Fourth International Conference on Evolutionary Multi-criterion Optimization*, pp. 742–756, 2007.

[168] K. Deb and D. K. Saxena, "Searching for pareto-optimal solutions through dimensionality reduction for certain large-dimensional multi-objective optimization problems," in *IEEE Congress on Evolutionary Computation*, pp. 3353–3360, 2006.

[169] D. Brockhoff and E. Zitzler, "Are all objectives necessary? On dimensionality reduction in evolutionary multiobjective optimization," in *Proceedings of the 9th international conference on Parallel Problem Solving from Nature*, pp. 533–542, 2006.

[170] D. Brockhoff and E. Zitzler, "Improving hypervolume-based multiobjective evolutionary algorithms by using objective reduction methods," in *IEEE Congress on Evolutionary Computation*, pp. 2086–2093, 2007.

[171] B. Xue, M. Zhang, and W. N. Browne, "Particle swarm optimization for feature selection in classification: A multi-objective approach," *IEEE Transactions on Cybernetics*, vol. 43, no. 6, pp. 1656–1671, 2013.

[172] R. C. Purshouse, C. Jalba, and P. J. Fleming, "Preference-driven coevolutionary algorithms show promise for many-objective optimisation," in *Proceedings of the Sixth International Conference on Evolutionary Multi-criterion Optimization*, pp. 136–150, 2011.

[173] S. Yang, M. Li, X. Liu, and J. Zheng, "A grid-based evolutionary algorithm for many-objective optimization," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 721–736, 2011.

[174] K. Deb, M. Mohan, and S. Mishra, "Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of pareto-optimal solutions," *Evolutionary Computation*, vol. 13, no. 4, pp. 501–525, 2005.

[175] J. Knowles and D. Corne, "Properties of an adaptive archiving algorithm for storing nondominated vectors," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 100–116, 2003.

[176] L. Rachmawati and D. Srinivasain, "Dynamic resizing for grid-based archiving in evolutionary multiobjective optimization," in *IEEE Congress on Evolutionary Computation*, pp. 3975–3982, 2007.

[177] Y. Zhou and J. He, "Convergence analysis of a self-adaptive multiobjective evolutionary algorithm based on grids," *Information Processing Letters*, vol. 104, no. 4, pp. 117–122, 2007.

[178] J. D. Knowles and D. W. Corne, "Approximating the nondominated front using Pareto archived evolutionary strategy," *Evolutionary Computation*, vol. 8, no. 2, pp. 149–172, 2000.

[179] G. G. Yen and H. Lu, "Dynamic multiobjective evolutionary algorithm: adaptive cell-based rank and density estimation," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 3, pp. 253–274, 2003.

[180] A. G. Hernandez, L. V. Santana-Quintero, C. A. C. Coello, and J. Molina, "Pareto-adaptive ϵ-domination," *Evolutionary Computation*, vol. 15, no. 4, pp. 493–517, 2007.

[181] I. Karahan and M. Koksalan, "A territory defining multiobjective evolutionary algorithm and preference incorporation," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 4, pp. 636–664, 2010.

[182] M. Li, J. Zheng, R. Shen, K. Li, and Q. Yuan, "A grid-based fitness strategy for evolutionary many-objective optimization," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, pp. 463–470, 2010.

[183] A. W. Mohamed, "RDEL: Restart differential evolution algorithm with local search mutation for global numerical optimization," *Egyptian Informatics Journal*, vol. 15, no. 3, pp. 175–188, 2014.

[184] H. Ishibuchi, N. Hitotsuyanagi, Y. Tsukamoto, and Y. Nojima, "Many-objective test problems to visually examine the behaviour of multiobjective evolution in a decision space," in *Proceedings of the 11th international conference on Parallel Problem Solving from Nature*, pp. 647–656, 2010.

[185] D. Hadka and P. Reed, "Diagnostic assessment of search controls and failure modes in many-objective evolutionary optimization," *Evolutionary Computation*, vol. 20, no. 3, pp. 423–452, 2007.

[186] J. Bader and E. Zitzler, "An algorithm for fast hypervolume-based many-objective optimization," *Evolutionary Computation*, vol. 19, no. 1, pp. 45–76, 2011.

[187] E. J. Hughes, "Multiple single objective pareto sampling," in *IEEE Congress on Evolutionary Computation*, pp. 2678–2684, 2003.

[188] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, "Evolutionary many-objective optimization by nsga-ii and moea/d with large populations," in *IEEE Congress on Evolutionary Computation*, pp. 3742–3747, 2009.

[189] S. B. Gee, K. C. Tan, V. A. Shim, and N. R. Pal, "Online diversity assessment in evolutionary multiobjective optimization: A geometrical perspective," *IEEE Transactions on Evolutionary Computation*, vol. 99, 2014.

[190] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 2011.

[191] F. Tasgetiren, A. Chen, G. Gencyilmaz, and S. Gattoufi, "Smallest position value approach," in *Differential Evolution: A Handbook for Global Permutation-Based Combinatorial Optimization*, vol. 175, pp. 121–138, Springer, 2009.

[192] V. A. Shim, K. C. Tan, and C. Y. Cheong, "A hybrid estimation of distribution algorithm with decomposition for solving the multiobjective multiple traveling salesman problem," *IEEE Transactions on Systems, Man and Cybernetics: Part C Applications and Reviews*, vol. 42, no. 5, pp. 682–691, 2012.

[193] T. Bektas, "The multiple traveling salesman problem: An overview of formulations and solution procedures," *The International Journal of Management Science*, vol. 34, no. 3, pp. 209–219, 2005.

[194] S. Bhide, N. John, and M. R. Kabuka, "A boolean neural network approach for the traveling salesman problem," *IEEE Transactions on Computers*, vol. 42, no. 10, p. 1271, 1993.

[195] F. Glover, "Artificial intelligence, heuristic frameworks and tabu search," in *Managerial and Decision Economics 11*, pp. 365–378, 1990.

[196] G. Laporte and Y. Nobert, "A cutting planes algorithm for the m-salesmen problem," *The Journal of the Operational Research Society*, vol. 31, no. 11, pp. 1017–1023, 1980.

[197] T. Bektas, "The multiple traveling salesman problem: an overview of formulations and solution procedures," *Omega*, vol. 34, no. 3, pp. 209–219, 2006.

[198] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

[199] A. I. Ali and J. L. Kennington, "The asymmetric M-travelling salesmen problem: A duality based branch-and-bound algorithm," *Discrete Applied Mathematics*, vol. 13, no. 2-3, pp. 259–276, 1986.

[200] R. A. Russell, "An effective heuristic for the m-tour traveling salesman problem with some side conditions," *Operations Research*, vol. 25, no. 3, pp. 517–524, 1977.

[201] J. Potvin, G. Lapalme, and J. Rousseau, "A generalized k-opt exchange procedure for the mtsp," *INFOR*, vol. 27, pp. 474–481, 1989.

[202] C. Y. Hsu, M. H. Tsai, and W. M. Chen, "A study of feature-mapped approach to the multiple travelling salesmen problem," in *IEEE International Sympoisum on Circuits and Systems*, pp. 1589–1592, 1991.

[203] M. Gen and R. Cheng, *Genetic algorithms and engineering design*. Wiley, 1997.

[204] J. Y. Potvin, "Genetic algorithms for the traveling salesman problem," *Annals of Operations Research*, vol. 63, no. 3, pp. 337–370, 1996.

[205] T. Zhang, W. A. Gruver, and M. H. Smith, "Team scheduling by genetic search," in *Proceedings of the Second International Conference on Intelligent Processing and Manufacturing of Materials*, pp. 829–844, 1999.

[206] L. Tang, J. Liu, A. Rong, and Z. Yang, "A multiple traveling salesman problem model for hot rolling scheduling in shanghai baoshan iron and steel complex," *European Journal of Operational Research*, vol. 124, no. 2, pp. 267–282, 2000.

[207] Y. B. Park, "A hybrid genetic algorithm for the vehicle scheduling problem with due times and times deadlines," *International Journal of Production Economics*, vol. 73, no. 2, pp. 175–188, 2001.

[208] A. E. Carter and C. T. Ragsdale, "A new approach to solving the multiple traveling salesperson problem using genetic algorithms," *European Journal of Operational Research*, vol. 175, no. 1, pp. 246–257, 2006.

[209] F. Zhao, J. Dong, S. Li, and X. Yang, "An improved genetic algorithm for multiple traveling salesman problem," in *Proceedings of the Second International Asia Conference on Informatics in Control, Automation and Robotics*, pp. 493–495, 2008.

[210] A. Király and J. Abonyi, "Optimization of multiple traveling salesman problem by a novel representation based genetic algorithm," in *Intelligent Computational Optimization in Engineering*, vol. 366, pp. 241–269, Springer, 2010.

[211] Y. S. Ong, M. Lim, and X. S. Chen, "Memetic computation - past, present & future," *IEEE Computational Intelligence Magazine*, vol. 5, no. 2, pp. 24–31, 2010.

[212] J. Y. Chia, C. K. Goh, K. C. Tan, and V. A. Shim, "Memetic informed evolutionary optimization via data dining," *Memetic Computing*, vol. 3, no. 2, pp. 73–88, 2011.

[213] W. T. Koo, C. K. Goh, and K. C. Tan, "A predictive gradient strategy for multiobjective evolutionary algorithms in a fast changing environment," *Memetic Computing*, vol. 2, no. 2, pp. 87–110, 2010.

[214] R. Solomon, "Evolutionary algorithms and gradient search: Similarities and differences," *IEEE Transaction on Evolutionary Computation*, vol. 2, no. 2, pp. 45–55, 1998.

[215] M. F. Tasgetiren, M. Sevkli, Y. C. Liang, and G. Gencyilmaz, "Particle swarm optimization algorithm for single machine total weighted tardiness problem," in *IEEE Congress on Evolutionary Computation*, pp. 1412–1419, 2004.

[216] C. Okonjo, "An effective method of balancing the workload amongst salesma," *Omega*, vol. 16, no. 2, pp. 159–163, 1998.

[217] M. A. Villalobos-arias, G. T. Pulido, and C. A. Coello Coello, "A proposal to use stripes to maintain diversity in a multi-objective particle swarm optimizer," in *Proceedings 2005 IEEE Swarm Intelligence Symposium*, pp. 22–29, 2005.

[218] V. A. Shim, K. C. Tan, and J. Y. Chia, "Probabilistic based evolutionary optimizers in bi-objective travelling salesman problem," in *Proceedings of the 8th International Conference on Simulated Evolution and Learning*, pp. 588–592, 2010.

[219] P. Larrañaga and J. A. Lozano, *Estimation of distribution algorithms. A new tool for evolutionary computation*. Kluwer Academic Publishers, 2001.

[220] H. Mühlenbein and G. Paass, "From recombination of genes to the estimation of distributions i. binary parameters," in *Proceedings of the Fourth International Conference on Parallel Problem Solving from Nature*, pp. 178–187, 1996.

[221] D. A. Van Veldhuizen and G. B. Lamont, "Evolutionary computation and convergence to a Pareto front," in *Late Breaking Papers at the Genetic Programming 1998 Conference*, pp. 221–228, 1998.

[222] Q. Zhang, A. Zhou, and Y. Jin, "RM-MEDA: A regularity model-based multiobjective estimation of distribution algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 12, no. 1, pp. 41–63, 2008.

Appendix A

Performance Metrics

Four performance metrics that are applied in this thesis are illustrated in this section. For description purposes, PF^* is a set of evolved solutions and PF is the set of Pareto optimal solutions. The definitions of the indicators are presented below.

1. **Generational Distance (GD):** Generational distance (GD) [221] is a unary performance indicator which is defined as

$$GD = \sqrt{\frac{\sum_{i=1}^{N} d(p^*, p)_i^2}{N}}$$

where N is the number of solutions in PF^*, $p \in PF$, $p^* \in PF^*$, and $d(p^*, p)_i$ is the minimum Euclidean distance in the objective space between p^* and p for each member i. GD illustrates the convergence ability of the algorithm by measuring the closeness between the Pareto optimal front and the evolved Pareto front. Thus, a lower value of GD shows that the evolved Pareto front is closer to the Pareto optimal front. This indicator is a representative metric which provides a quantitative measurement for the proximity goal of multi-objective optimization.

2. **Inverted Generational Distance (IGD):** Inverted generational distance (IGD) is a unary indicator which performs a near-similar calculation as done by GD [217, 222]. The difference is that GD calculates the distance of each solution in PF^* to PF while IGD calculates the distance of each solution in PF to PF^*. In this indicator, both convergence and diversity are taken into consideration. A lower value of IGD implies that the algorithm has better optimization performance.

3. **Hausdorff Distance (HD):** Hausdorff Distance (HD) is a unary indicator which combines the properties of GD and IGD, and it basically takes the maximum value between the modified GD and IGD. In this way, both convergence and diversity are taken into consideration. A lower value of HD implies that the algorithm has better optimization performance.

4. **Hypervolume (HV):** Hypervolume (HV) is a metric that gives the volume (in objective space) that is dominated by a solution set S. It can be defined as

$$HV = volume(\cup_{i=1}^{|S|} v_i)$$

where S is the solution set and R is the reference set, and v_i is the hypercube constructed with a reference point and a solution $\vec{s_i} \in S$ as the diagonal corners of the hypercube. The reference point can be found by constructing a vector of the worst objective function values. This metric quantifies and encapsulates both the convergence and diversity information of the solution set S. Hence, if the solutions in S are closer to the true Pareto front, the value of HV will be larger. In addition, a higher HV could also indicate that the solutions in S are distributed more evenly in the objective space.

Appendix B

Multi-objective Test Problems

A multi-objective optimization problem (MOOP) can be characterized by two main aspects which are namely fitness landscapes and Pareto optimal front geometries. For the case of fitness landscapes, an MOOP may be scalable in terms of its number of objective functions. As the number of conflicting objective functions in an MOOP increases to more than three, the problem will be harder to solve as the selection pressure for selecting fitter individuals will be reduced. This is because in an objective space of higher dimensionality, more individuals will be non-dominated against each other during the evolutionary process. This may lead to a hindrance in the search towards optimality or may cause the population to get trapped in a local optima. Besides this, it is also challenging for an optimizer to search over all the promising regions in a large fitness landscape. An MOOP may also be scalable in terms of the number of decision variables in it. The complexity of an MOOP will increase with an increase in the number of decision variables in it. This is due to an enlargement of the search space and also an increase in the number of possible moves towards optimality.

Another characteristic of fitness landscape is modality. If an MOOP consists of only a single optimum, it will be known as a unimodal problem. However, if an MOOP has many local optima, then it is a multimodal problem. A multimodal MOOP may create challenges for optimizers as they may get stucked in local optima. If the multimodal MOOP consists of a deceptive optimum, it will be more difficult to solve as the true optimum is placed in an unlikely place for the case of a deceptive MOOP. Another characteristic seen in the fitness landscape is the presence of the mapping from the decision space to the objective space. For this, a problem is bias if a set of evenly distributed samples are being mapped to an unevenly distributed region in an objective space. This is challenging for an MOEA as a set of evenly distributed tradeoff solutions has to be generated by the MOEA. Besides this, an MOOP may be separable or nonseparable. For separable problems, every decision variable can be optimized independently. However, this may not be possible for the case of nonseparable problems as there will be some level of dependencies between the decision variables in such problems.

For the aspect of Pareto optimal front geometries in MOOP, there can be a few types of geometries which are namely convex, concave, linear, disconnected, degenerate, and mixed geometries of Pareto optimal front. A Pareto optimal front is considered as convex if the set of tradeoff solutions encompasses its convex hull. If a Pareto optimal front is concave, the set of tradeoff solutions will cover its concave hull. If the set of tradeoff solutions is both convex and concave, the Pareto optimal front is considered as linear. An MOOP has a degenerate Pareto front when the optimal front is of one dimension lower than its objective space. This may challenge an MOEA in terms of generating a set of diverse tradeoff solutions. For the case of a disconnected Pareto optimal front, it may consist of several discontinuous subsets of solutions. Finally, a mixed Pareto optimal front is one that contains several connected subsets with different geometries. A more throughout review, description, and analysis of different multi-objective test problems can be referred to [107]. In Table 1, a list of test problems used in this thesis is given with their characteristics stated.

The ZDT test problems are extracted from [104].
ZDT1

$$f_1(\mathbf{x}) = x_1$$

$$f_2(\mathbf{x}) = g(\mathbf{x}) \left[1 - \sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}} \right]$$

$$g(\mathbf{x}) = 1 + 9 \left(\frac{\sum_{i=2}^{n} x_i}{n-1} \right)$$

ZDT2

Same as ZDT1, except

$$f_2(\mathbf{x}) = g(\mathbf{x}) \left[1 - \left(\frac{f_1(\mathbf{x})}{g(\mathbf{x})} \right)^2 \right]$$

Table 1: Multi-objective test problems. S refers to scalable, m is the number of objective functions, K is a scalar parameter, n is the number of decision variables, SP refers to separable, NS refers to nonseparable, D refers to deceptive, U refers to unimodal, and M refers to multimodal.

Instance	m	n	Domain	Geometry	SP/NS	U/M	Bias
ZDT1	2	30(S)	$[0,1]^n$	Convex	SP	U	NO
ZDT2	2	30(S)	$[0,1]^n$	Concave	SP	U	NO
ZDT3	2	30(S)	$[0,1]^n$	Disconnected	SP	M	NO
ZDT4	2	30(S)	$[0,1]\times[-5,5]^{n-1}$	Convex	SP	M	NO
ZDT6	2	30(S)	$[0,1]^n$	Concave	SP	M	YES
DTLZ1	3(S)	$m+K-1$(S)	$[0,1]^n$	Linear	SP	M	NO
DTLZ2	3(S)	$m+K-1$(S)	$[0,1]^n$	Concave	SP	U	NO
DTLZ3	3(S)	$m+K-1$(S)	$[0,1]^n$	Concave	SP	M	NO
DTLZ4	3(S)	$m+K-1$(S)	$[0,1]^n$	Concave	SP	U	YES
DTLZ5	3(S)	$m+K-1$(S)	$[0,1]^n$	Degenerate	NS	U	NO
DTLZ6	3(S)	$m+K-1$(S)	$[0,1]^n$	Degenerate	NS	U	YES
DTLZ7	3(S)	$m+K-1$(S)	$[0,1]^n$	Disconnected	SP	M	NO
UF1	2	30(S)	$[0,1]\times[-1,1]^{n-1}$	Convex	SP	M	NO
UF2	2	30(S)	$[0,1]\times[-1,1]^{n-1}$	Convex	NS	M	NO
UF3	2	30(S)	$[0,1]^n$	Convex	NS	M	NO
UF4	2	30(S)	$[0,1]\times[-2,2]^{n-1}$	Concave	NS	M	NO
UF5	2	30(S)	$[0,1]\times[-1,1]^{n-1}$	Linear	NS	M	NO
UF6	2	30(S)	$[0,1]\times[-1,1]^{n-1}$	Linear, Disconnected	NS	M	NO
UF7	2	30(S)	$[0,1]\times[-1,1]^{n-1}$	Linear	NS	M	NO
UF8	3	30(S)	$[0,1]^2\times[-2,2]^{n-2}$	Concave	SP	M	NO
UF9	3	30(S)	$[0,1]^2\times[-2,2]^{n-2}$	Linear, Disconnected	SP	M	NO
UF10	3	30(S)	$[0,1]^2\times[-2,2]^{n-2}$	Concave	NS	M	NO
WFG1	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Convex, Mixed	SP	U	YES
WFG2	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Convex, Disconnected	NS	U	NO
WFG3	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Linear, Degenerate	NS	M	NO
WFG4	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Concave	SP	M	NO
WFG5	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Concave	SP	D	NO
WFG6	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Concave	NS	U	NO
WFG7	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Concave	SP	U	YES
WFG8	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Concave	NS	U	YES
WFG9	2(S)	30(S)	$[0,2i]\ i\in\{1,...,n\}$	Concave	NS	M,D	YES

ZDT3

Same as ZDT1, except

$$f_2(\mathbf{x}) = g(\mathbf{x})\left[1-\sqrt{\frac{f_1(\mathbf{x})}{g(\mathbf{x})}}-\frac{f_1(\mathbf{x})}{g(\mathbf{x})}\sin(10\pi x_1)\right]$$

ZDT4

Same as ZDT1, except

$$g(\mathbf{x}) = 1+10(n-1)+\sum_{i=2}^{n}\left[x_i^2-10\cos(4\pi x_i)\right]$$

ZDT6

$$f_1(\mathbf{x}) = 1-exp(-4x_1)\sin^6(6\pi x_1)$$

$$f_2(\mathbf{x}) = g(\mathbf{x})\left[1-\left(\frac{f_1(\mathbf{x})}{g(\mathbf{x})}\right)^2\right]$$

$$g(\mathbf{x}) = 1+9\left[\frac{\sum_{i=2}^{n}x_i}{n-1}\right]^{0.25}$$

The DTLZ test problems are extracted from [105].

DTLZ1

$$f_1(\mathbf{x}) = (1 + g(\mathbf{x}))\, 0.5 \left(\prod_{i=1}^{m-1} x_i \right)$$

$$f_2(\mathbf{x}) = (1 + g(\mathbf{x}))\, 0.5 \left(\prod_{i=1}^{m-1} x_i \right) (1 - x_{m-1})$$

$$\vdots$$

$$f_m(\mathbf{x}) = (1 + g(\mathbf{x}))\, 0.5(1 - x_1)$$

$$g(\mathbf{x}) = 100 \left[(n - m + 1) + \sum_{i=m}^{n} \left((x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)) \right) \right]$$

DTLZ2

$$f_1(\mathbf{x}) = (1 + g(\mathbf{x})) \left[\prod_{i=1}^{m-1} \cos(0.5\pi x_i) \right]$$

$$f_2(\mathbf{x}) = (1 + g(\mathbf{x})) \left[\prod_{i=1}^{m-2} \cos(0.5\pi x_i) \right] \sin(0.5\pi x_{m-1})$$

$$\vdots$$

$$f_m(\mathbf{x}) = (1 + g(\mathbf{x})) \sin(0.5\pi x_1)$$

$$g(\mathbf{x}) = \sum_{i=m}^{n} (x_i - 0.5)^2$$

DTLZ3

Same as DTLZ2, except

$$g(\mathbf{x}) = 100 \left[(n - m + 1) + \sum_{i=m}^{n} \left((x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5)) \right) \right]$$

DTLZ4

Same as DTLZ2, except

$$x_i = x_i^{\alpha}, \;\; i \in \{m, ..., n\}, \;\; \alpha = 10$$

DTLZ5

Same as DTLZ2, except

$$x_i = \frac{1 + 2g(\mathbf{x})x_i}{2(1 + g(\mathbf{x}))}, \;\; i \in \{2, ..., m - 1\}$$

DTLZ6

Same as DTLZ5, except

$$g(\mathbf{x}) = \sum_{i=m}^{n} x_i^{0.1}$$

DTLZ7

$$f_1(\mathbf{x}) = x_1$$

$$\vdots$$

$$f_{m-1}(\mathbf{x}) = x_{m-1}$$

$$f_m(\mathbf{x}) = (1 + g(\mathbf{x})) \left[m - \sum_{i=1}^{m-1} \left(\frac{f_i(\mathbf{x})}{1 + g(\mathbf{x})} (1 + \sin(3\pi f_i(\mathbf{x}))) \right) \right]$$

$$g(\mathbf{x}) = 1 + 9 \left(\frac{\sum_{i=m}^{n} x_i}{n - m + 1} \right)$$

The UF test problems are extracted from [106].
UF1

$$f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|} \sum_{i \in J_1} \left[x_i - \sin(6\pi x_1 + \frac{i\pi}{n}) \right]^2$$

$$f_2(\mathbf{x}) = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{i \in J_2} \left[x_i - \sin(6\pi x_1 + \frac{i\pi}{n}) \right]^2$$

$$J_1 = \{i | i \text{ is odd and } 2 \leq i \leq n\} \text{ and } J_2 = \{i | i \text{ is even and } 2 \leq i \leq n\}$$

UF2

$$f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|} \sum_{i \in J_1} y_i^2$$

$$f_2(\mathbf{x}) = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \sum_{i \in J_2} y_i^2$$

$$J_1 = \{i | i \text{ is odd and } 2 \leq i \leq n\} \text{ and } J_2 = \{i | i \text{ is even and } 2 \leq i \leq n\}$$

$$y_i = \begin{cases} x_i - \left[0.3x_1^2 \cos(24\pi x_1 + \frac{4i\pi}{n}) + 0.6x_1 \right] \cos(6\pi x_1 + \frac{i\pi}{n}) & i \in J_1 \\ x_i - \left[0.3x_1^2 \cos(24\pi x_1 + \frac{4i\pi}{n}) + 0.6x_1 \right] \sin(6\pi x_1 + \frac{i\pi}{n}) & i \in J_2 \end{cases}$$

UF3

$$f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|} \left[4 \sum_{i \in J_1} y_i^2 - 2 \prod_{i \in J_1} \cos(\frac{20 y_i \pi}{\sqrt{i}}) + 2 \right]$$

$$f_2(\mathbf{x}) = 1 - \sqrt{x_1} + \frac{2}{|J_2|} \left[4 \sum_{i \in J_2} y_i^2 - 2 \prod_{i \in J_2} \cos(\frac{20 y_i \pi}{\sqrt{i}}) + 2 \right]$$

$$J_1 = \{i | i \text{ is odd and } 2 \leq i \leq n\} \text{ and } J_2 = \{i | i \text{ is even and } 2 \leq i \leq n\}$$

$$y_i = x_i - x_1^{0.5\left(1.0 + \frac{3(i-2)}{n-2}\right)}, \ i \in \{2, ..., n\}$$

UF4

$$f_1(\mathbf{x}) = x_1 + \frac{2}{|J_1|} \sum_{i \in J_1} h(y_i)$$

$$f_2(\mathbf{x}) = 1 - x_1^2 + \frac{2}{|J_2|} \sum_{i \in J_2} h(y_i)$$

$$J_1 = \{i | i \text{ is odd and } 2 \leq i \leq n\} \text{ and } J_2 = \{i | i \text{ is even and } 2 \leq i \leq n\}$$

$$y_i = x_i - \sin\left(6\pi x_1 + \frac{i\pi}{n}\right), \ i \in \{2, ..., n\}$$

$$h(t) = \frac{|t|}{1 + e^{2|t|}}$$

UF5

$$f_1(\mathbf{x}) = x_1 + \left(\frac{1}{2N} + \varepsilon\right)|\sin(2N\pi x_1)| + \frac{2}{|J_1|}\sum_{i \in J_1} h(y_i)$$

$$f_2(\mathbf{x}) = 1 - x_1 + \left(\frac{1}{2N} + \varepsilon\right)|\sin(2N\pi x_1)| + \frac{2}{|J_2|}\sum_{i \in J_2} h(y_i)$$

$$J_1 = \{i | i \text{ is odd and } 2 \leq i \leq n\} \text{ and } J_2 = \{i | i \text{ is even and } 2 \leq i \leq n\}, N = 10 \text{ and } \varepsilon = 0.1$$

$$y_i = x_i - \sin(6\pi x_1 + \frac{i\pi}{n}), \ i \in \{2, ..., n\}$$

$$h(t) = 2t^2 - \cos(4\pi t) + 1$$

UF6

$$f_1(\mathbf{x}) = x_1 + \max\{0, \ 2(\frac{1}{2N} + \varepsilon)\sin(2N\pi x_1)\} + \frac{2}{|J_1|}\left[4\sum_{i \in J_1} y_i^2 - 2\prod_{i \in J_1}\cos(\frac{20y_i\pi}{\sqrt{i}}) + 2\right]$$

$$f_2(\mathbf{x}) = 1 - x_1 + \max\{0, \ 2(\frac{1}{2N} + \varepsilon)\sin(2N\pi x_1)\} + \frac{2}{|J_2|}\left[4\sum_{i \in J_2} y_i^2 - 2\prod_{i \in J_2}\cos(\frac{20y_i\pi}{\sqrt{i}}) + 2\right]$$

$$J_1 = \{i | i \text{ is odd and } 2 \leq i \leq n\} \text{ and } J_2 = \{i | i \text{ is even and } 2 \leq i \leq n\}, N = 2 \text{ and } \varepsilon = 0.1$$

$$y_i = x_i - \sin\left(6\pi x_1 + \frac{i\pi}{n}\right), \ i \in \{2, ..., n\}$$

UF7

$$f_1(\mathbf{x}) = \sqrt[5]{x_1} + \frac{2}{|J_1|}\sum_{i \in J_1} y_i^2$$

$$f_2(\mathbf{x}) = 1 - \sqrt[5]{x_1} + \frac{2}{|J_2|}\sum_{i \in J_2} y_i^2$$

$$J_1 = \{i | i \text{ is odd and } 2 \leq i \leq n\} \text{ and } J_2 = \{i | i \text{ is even and } 2 \leq i \leq n\}$$

$$y_i = x_i - \sin\left(6\pi x_1 + \frac{i\pi}{n}\right), \ i \in \{2, ..., n\}$$

UF8

$$f_1(\mathbf{x}) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|}\sum_{i \in J_1}\left[x_i - 2x_2\sin(2\pi x_1 + \frac{i\pi}{n})\right]^2$$

$$f_2(\mathbf{x}) = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{|J_2|}\sum_{i \in J_2}\left[x_i - 2x_2\sin(2\pi x_1 + \frac{i\pi}{n})\right]^2$$

$$f_3(\mathbf{x}) = \sin(0.5x_1\pi) + \frac{2}{|J_3|}\sum_{i \in J_3}\left[x_i - 2x_2\sin(2\pi x_1 + \frac{i\pi}{n})\right]^2$$

$$J_1 = \{i | 3 \leq i \leq n, \text{ and } i-1 \text{ is a multiplication of 3}\}$$
$$J_2 = \{i | 3 \leq i \leq n, \text{ and } i-2 \text{ is a multiplication of 3}\}$$
$$J_3 = \{i | 3 \leq i \leq n, \text{ and } i \text{ is a multiplication of 3}\}$$

UF9

$$f_1(\mathbf{x}) = 0.5\left[\max\{0,\ (1+\varepsilon)(1-4(2x_1-1)^2)\} + 2x_1\right]x_2 + \frac{2}{|J_1|}\sum_{i\in J_1}\left[x_i - 2x_2\sin(2\pi x_1 + \frac{i\pi}{n})\right]^2$$

$$f_2(\mathbf{x}) = 0.5\left[\max\{0,\ (1+\varepsilon)(1-4(2x_1-1)^2)\} - 2x_1 + 2\right]x_2 + \frac{2}{|J_2|}\sum_{i\in J_2}\left[x_i - 2x_2\sin(2\pi x_1 + \frac{i\pi}{n})\right]^2$$

$$f_3(\mathbf{x}) = 1 - x_2 + \frac{2}{|J_3|}\sum_{i\in J_3}\left[x_i - 2x_2\sin(2\pi x_1 + \frac{i\pi}{n})\right]^2$$

$J_1,\ J_2,$ and J_3 are the same as $J_1,\ J_2,$ and J_3 from UF8, and $\varepsilon = 0.1$

UF10

$$f_1(\mathbf{x}) = \cos(0.5x_1\pi)\cos(0.5x_2\pi) + \frac{2}{|J_1|}\sum_{i\in J_1}\left[4y_i^2 - \cos(8\pi y_i) + 1\right]$$

$$f_2(\mathbf{x}) = \cos(0.5x_1\pi)\sin(0.5x_2\pi) + \frac{2}{|J_2|}\sum_{i\in J_2}\left[4y_i^2 - \cos(8\pi y_i) + 1\right]$$

$$f_2(\mathbf{x}) = \sin(0.5x_1\pi) + \frac{2}{|J_3|}\sum_{i\in J_3}\left[4y_i^2 - \cos(8\pi y_i) + 1\right]$$

$J_1,\ J_2,$ and J_3 are the same as $J_1,\ J_2,$ and J_3 from UF8

$$y_i = x_i - 2_x 2\sin\left(2\pi x_1 + \frac{i\pi}{n}\right),\ \ i \in \{3, ..., n\}$$

The WFG test problems are extracted from [107].
Common format of WFG test problems:
Given:

$$\mathbf{z} = \{z_1, ..., z_k, z_k + 1, ..., z_n\}$$

Minimize:

$$f_{j=1:m}(\mathbf{x}) = Dx_m + S_j h_j(x_1, ..., x_{m-1})$$

where

$$\begin{aligned}
\mathbf{x} &= \{x_1, ..., x_m\} \\
&= \{\max(t_m^p, A_1)(t_1^p - 0.5) + 0.5, ..., \max(t_m^p, A_{m-1})(t_{m-1}^p - 0.5) + 0.5, t_m^p\} \\
\mathbf{t^p} &= \{t_1^p, ..., t_m^p\} \leftarrow | \mathbf{t}^{p-1} \leftarrow | ... \leftarrow | \mathbf{t^1} \leftarrow | \mathbf{z}_{[\mathbf{0},\mathbf{1}]} \\
\mathbf{z}_{[\mathbf{0},\mathbf{1}]} &= \{z_{1,[0,1]}, ..., z_{n,[0,1]}\} \\
&= \{z_1/z_{1,\max}, ..., z_n/z_{n,\max}\}
\end{aligned}$$

Constants:

$S_{j=1:m} = 2m,\ \ D = 1,\ \ A_{1:m-1} = 1, k$ is the number of position-related parameters, and

l is the number of distance-related parameters

Shape functions ($h_{j=1:m}$):

Linear: represented by linear$_{1:m}(x_1, ..., x_{m-1})$
Convex: represented by convex$_{1:m}(x_1, ..., x_{m-1})$
Concave: represented by concave$_{1:m}(x_1, ..., x_{m-1})$
Mixed Convex and Concave: represented by mixed$_{1:m}(x_1, ..., x_{m-1})$
Disconnected: represented by disc$_{1:m}(x_1, ..., x_{m-1})$

Transformation functions:

Polynomial bias transformation: represented by b_poly(y, α)
Flat region bias transformation: represented by b_flat(y, A, B, C)
Parameter dependent bias transformation: represented by b_param$(y, \mathbf{y}^{'} A, B, C)$
Linear shift transformation: represented by s_linear(y, A)
Deceptive shift transformation: represented by s_decept(y, A, B, C)
Multi-modal shift transformation: represented by s_multi(y, A, B, C)
Weighted sum reduction transformation: represented by r_sum$(\mathbf{y}, \mathbf{w})$
Non-seperable reduction transformation: represented by r_nonsep$(\mathbf{y}, A)$

WFG1

$$
\begin{aligned}
h_{j=1:m-1} &= \text{convex}_j \\
h_m &= \text{mixed}_m \text{ (with } \alpha = 1 \text{ and } A = 5) \\
t^1_{i=1:k} &= y_i \\
t^1_{i=k+1:n} &= \text{s\_linear}(y_i, 0.35) \\
t^2_{i=1:k} &= y_i \\
t^2_{i=k+1:n} &= \text{b\_flat}(y_i, 0.8, 0.75, 0.85) \\
t^3_{i=1:n} &= \text{b\_poly}(y_i, 0.02) \\
t^4_{i=1:m-1} &= \text{r\_sum}(\{y_{(i-1)k/(m-1)+1}, ..., y_{ik/(m-1)}\}, \\
&\quad \{2[(i-1)k/(m-1)+1, ..., 2ik/(m-1)]\}) \\
t^4_m &= \text{r\_sum}(\{y_{k+1}, ..., y_n\}, \{2(k+1), ..., 2n\})
\end{aligned}
$$

WFG2

$$
\begin{aligned}
h_{j=1:m-1} &= \text{convex}_j \\
h_m &= \text{disc}_m \text{ (with } \alpha = \beta = 1 \text{ and } A = 5) \\
\mathbf{t}^1 &\text{ is the same as } \mathbf{t}^1 \text{ from WFG1 (linear shift)} \\
t^2_{i=1:k} &= y_i \\
t^2_{i=k+1:k+l/2} &= \text{r\_nonsep}(\{y_{k+2(i-k)-1}, y_{k+2(i-k)}\}, 2) \\
t^3_{i=1:m-1} &= \text{r\_sum}(\{y_{i-1k/(m-1)+1}, ..., y_{ik/(m-1)}\}, \{1, ..., 1\}) \\
t^3_m &= \text{r\_sum}(\{y_{k+1}, ..., y_{k+l/2}\}, \{1, ..., 1\})
\end{aligned}
$$

WFG3

$$
\begin{aligned}
h_{j=1:m} &= \text{linear}_j \\
\mathbf{t}^{1:3} &\text{ are the same as } \mathbf{t}^{1:3} \text{ from WFG2}
\end{aligned}
$$

WFG4

$$
\begin{aligned}
h_{j=1:m} &= \text{concave}_j \\
t^1_{1:n} &= \text{s\_multi}(y_i, 30, 10, 0.35) \\
t^2_{1:m-1} &= \text{r\_sum}(\{y_{(i-1)k/(m-1)+1}, ..., y_{ik/(m-1)}\}, \{1, ..., 1\}) \\
t^2_m &= \text{r\_sum}(\{y_{k+1}, ..., y_n\}, \{1, ..., 1\})
\end{aligned}
$$

WFG5

$$
\begin{aligned}
h_{j=1:m} &= \text{concave}_j \\
t^1_{i=1:n} &= \text{s\_decept}(y_i, 0.35, 0.001, 0.05)
\end{aligned}
$$

$\mathbf{t}^2$ is the same as $\mathbf{t}^2$ from WFG4

WFG6

$$
\begin{aligned}
h_{j=1:m} &= \text{concave}_j \\
\mathbf{t}^1 &\text{ is the same as } \mathbf{t}^1 \text{ from WFG1} \\
t^2_{i=1:m-1} &= \text{r\_nonsep}(\{y_{(i-1)k/(m-1)+1}, ..., y_{ik/(m-1)}\}, k/(m-1)) \\
t^2_m &= \text{r\_nonsep}(\{y_{k+1}, ..., y_n\}, l)
\end{aligned}
$$

WFG7

$$
\begin{aligned}
h_{j=1:m} &= \text{concave}_j \\
t^1_{i=1:k} &= \text{b\_param}\left(y_i, \text{r\_sum}(\{y_{i+1}, ..., y_n\}, \{1, ..., 1\}), \frac{0.98}{49.98}, 0.02, 50\right) \\
t^1_{i=k+1:n} &= y_i
\end{aligned}
$$

$\mathbf{t}^2$ is the same as $\mathbf{t}^1$ from WFG1

$\mathbf{t}^3$ is the same as $\mathbf{t}^2$ from WFG4

WFG8

$$
\begin{aligned}
h_{j=1:m} &= \text{concave}_j \\
t^1_{i=1:k} &= y_i \\
t^1_{i=k+1:n} &= \text{b\_param}\left(y_i, \text{r\_sum}(\{y_i, ..., y_{i-1}\}, \{1, ..., 1\}), \frac{0.98}{49.98}, 0.02, 50\right)
\end{aligned}
$$

$\mathbf{t}^2$ is the same as $\mathbf{t}^1$ from WFG1

$\mathbf{t}^3$ is the same as $\mathbf{t}^2$ from WFG4

WFG9

$$
\begin{aligned}
h_{j=1:m} &= \text{concave}_j \\
t^1_{i=1:n-1} &= \text{b\_param}\left(y_i, \text{r\_sum}(\{y_{i+1}, ..., y_n\}, \{1, ..., 1\}), \frac{0.98}{49.98}, 0.02, 50\right) \\
t^1_n &= y_n \\
t^2_{i=1:k} &= \text{s\_decept}(y_i, 0.35, 0.001, 0.05) \\
t^2_{i=k+1:n} &= \text{s\_multi}(y_i, 30, 95, 0.35)
\end{aligned}
$$

$\mathbf{t}^3$ is the same as $\mathbf{t}^2$ from WFG6