# DECOMPOSITION EVOLUTIONARY ALGORITHMS FOR NOISY MULTIOBJECTIVE OPTIMIZATION.

By

Hamid R. Jalalian

A thesis submitted for the degree of Doctor of Philosophy

School of Computer Science and Electronic Engineering

University of Essex

January 2016

***This thesis is dedicated to my Parents***

*for their endless love, support and encouragement.*

# Acknowledgements

First and foremost, I wish to thank my parents for giving me the gift of education from the best institutions and for supporting me throughout my life.

It is with immense gratitude that I acknowledge the support and help of my supervisor, Professor Edward Tsang. It is he who introduced me to computational finance and to constraint satisfaction problems. Without his guidance and constant assistance this thesis would not have been possible.

It gives me great pleasure to acknowledge the support and help of my supervisor, Professor Qingfu Zhang, who introduced me to computational intelligence and metaheuristic optimization algorithms.

I would like to thank my committee chair, Dr. Francisco Sepulveda, who has offered guidance and support throughout the years of my degree.

I wish to thank Mrs Marisa Bostock, the postgraduate research administrator of our department, for always helping with my requests over the years I have studied at the University of Essex.

Finally, I cannot find words to express my gratitude to my dearest friend, Sisi Zhou. Her emotional support over all these years has made it possible for me to complete my journey.

## Abstract

Multi-objective problems are a category of optimization problem that contain more than one objective function and these objective functions must be optimized simultaneously. Should the objective functions be conflicting, then a set of solutions instead of a single solution is required. This set is known as Pareto optimal.

Multi-objective optimization problems arise in many real world applications where several competing objectives must be evaluated and optimal solutions found for them, in the presence of trade offs among conflicting objectives. Maximizing returns while minimizing the risk of stock market investments, or maximizing performance whilst minimizing fuel consumption and hazardous gas emission when buying a car are typical examples of real world multi-objective optimization problems. In this case a number of optimal solutions can be found, known as non-dominated or Pareto optimal solutions. Pareto optimal solutions are reached when it is impossible to improve one objective without making the others worse.

Classical ways to address this problem used direct or gradient based methods that rendered them insufficient or computationally expensive for large scale or combinatorial problems. Other difficulties attended the classical methods, such as problem knowledge, which may not be available, or sensitivity to some problem features. For example, finding solutions on the entire Pareto optimal set can only be guaranteed for convex problems. Classical methods for generating the Pareto front set aggregate the objectives into a single or parametrized function before search. Thus, several runs and parameter settings are performed to achieve a set of solutions that approximate the Pareto optimals.

Subsequently new methods have been developed, based on computer experiments with meta-heuristic algorithms. Most of these meta-heuristics implement some sort of stochastic search method, amongst which the 'Evolutionary Algorithm' is garnering much attention. It possesses several characteristics that make it a desirable method for confronting multi-objective problems. As a result, a number of studies in recent decades have developed or modified the Multi-objective Optimization Evolutionary Algorithm

(MOEA) for different purposes. This algorithm works with a population of solutions which are capable of searching for multiple Pareto optimal solutions in a single run. At the same time, only the fittest individuals in each generation are offered the chance for reproduction and representation in the next generation. The fitness assignment function is the guiding system of MOEA. Fitness value represents the strength of an individual.

Unfortunately, many real world applications bring with them a certain degree of noise due to natural disasters, inefficient models, signal distortion or uncertain information. This noise affects the performance of the algorithm's fitness function and disrupts the optimization process. This thesis explores and targets the effect of this disruptive noise on the performance of the MOEA.

In this thesis, we study the noisy Multi-objective Optimization Problem (MOP) and modify the Multi-objective Optimization Evolutionary Algorithm based on Decomposition (MOEA/D) to improve its performance in noisy environments. To achieve this, we will combine the basic MOEA/D with the 'Ordinal Optimization' technique to handle uncertainties. The major contributions of this thesis are as follows.

- First, MOEA/D is tested in a noisy environment with different levels of noise, to give us a deeper understanding of where the basic algorithm fails to handle the noise.

- Then, we extend the basic MOEA/D to improve its noise handling by employing the ordinal optimization technique. This creates MOEA/D+OO, which will outperform MOEA/D in terms of diversity and convergence in noisy environments. It is tested against benchmark problems of varying levels of noise.

- Finally, to test the real world application of MOEA/D+OO, we solve a noisy portfolio optimization with the proposed algorithm. The portfolio optimization problem is a classic one in finance that has investors wanting to maximize a portfolio's return while minimizing risk of investment. The latter is measured by standard deviation of the portfolio's rate of return. These two objectives clearly make it a multi-objective problem.

# Contents

# List of Figures

# List of Tables

# List of Algorithms

# List of Acronyms

**MOEA** Multi-objective Optimization Evolutionary Algorithm

**SPEA** Strength Pareto Evolutionary Algorithm

**NSGA-II** Non-dominated Sorting Genetic Algorithm-II

**ELDP** Experimental Learning Directed Perturbation

**GASS** Gene Adaptation Selection Strategy

**MOEA/D** Multi-objective Optimization Evolutionary Algorithm based on Decomposition

**MOP** Multi-objective Optimization Problem

**OO** Ordinal Optimization

**MOEA/D+OO** Combined MOEA/D algorithm with OO technique

**VOO** Vector Ordinal Optimization

**SVR** Support Vector Regression

**ANN** Artificial Neural Network

**PF** Pareto Front

**PS** Pareto-optimal Set

**NTSPEA** Noise Tolerant Version of SPEA

**MOEA-RF** Robust Feature Multi-objective Evolutionary Algorithm

**MNSGA-II** Modified Non-dominated Sorting Genetic Algorithm-II

**MOPSEA** Multi-objective Probabilistic Selection Evolutionary Algorithm

**SOP** Single-objective Optimization Problem

**DMOEA-DD** Dynamical Multi-Objective Evolutionary Algorithm with Domain Decomposition

**OGA** Order Based Genetic Algorithm

**GOO** Genetic Ordinal Optimization

**OCBA** Optimal Computing Budget Allocation

**CPSOO** Combined Particle Swarm with Ordinal Optimization

|  |  |
|---|---|
| **PSO** | Particle Swarm Optimization |
| **VaR** | Value-at-risk |
| **CVaR** | Conditional value at Risk |

# List of Publications

- Chapter 4: MOEA/D With Ordinal Optimization Technique for Handling Noisy Problems.

    1. Hamid R. Jalalian, Qingfu Zhang and Edward Tsang, "*Combining MOEA/D with Ordinal Optimization for Noise Handling Purpose*", to be submitted to journal of "Computational Intelligence".

- Chapter 5: Noisy Portfolio Optimization Problem.

    1. Hamid R. Jalalian and Edward Tsang, "*An Investigation on Noisy Portfolio Optimization Problem*", to be submitted to journal of "Intelligent Systems in Accounting, Finance and Management".

# 1 Introduction

Many real life applications involve multiple (potentially conflicting) objective functions that must be optimized simultaneously. In the case of conflicting objectives, no single solution can be optimal to all objectives. Thus, a strong and powerful optimization algorithm is required to be capable of finding a set of solutions that will represent the best tradeoffs amongst the all objectives. This set of solutions is known as the Pareto optimal solution.

Evolutionary algorithms fall within a class of stochastic search methods that are capable of estimating Pareto optimal solutions in a single run. They are able to do this because the algorithms update their population of solutions at each generation. As a result, this method is proving itself to be very effective at solving complicated multi-objective optimization problems.

Finding a good Pareto-optimal estimation is not the only challenge facing the optimization algorithm, however. Uncertainty is also a disruptive phenomenon that characterizes many real world optimization problems in various forms. In recent decades numerous studies [4–6] have been conducted into different types of uncertainty and these are listed in the next section.

## 1.1 Sources of Uncertainty

- Uncertainty of environment: for example, temperature, moisture, perturbation in speed or dynamic fitness function.
- Uncertainty of optimization parameters: for instance, parameters of a solution subject to change or perturbation after implementation, but still required to function for manufacturing tolerance. This type of uncertainty is known as a search for robust solution.
- Uncertainty introduced due to unavailability of original fitness function or where the analytical fitness function is computationally very expensive. In this instance, the solution must be approximated.

The work presented in this thesis addresses this third version of uncertainty, also known as the optimization of noisy problems.

## 1.2 Thesis Motivation

A very important and also very sensitive research area is the study of noise, and the ways to cope with it, in the evolutionary multi-objective optimization algorithm. There are a number of studies that suggest different strategies or noise handling techniques for tackling disruptive noise by very well known MOEA. For instance,

- Strength Pareto Evolutionary Algorithm (SPEA) introduced by Zitzler in 1999 [7]. A Noise Tolerant Version of SPEA (NTSPEA) by Buche [8]. Zitzler and Buche proposed three modifications for handling noise for this particular dominance based MOEA, namely *i) Domination dependent lifetime*, which defines a lifetime for the solution that is related to the solution's dominance. *ii) Re-evaluation of solution*: instead of deleting the expired solutions, they are added to the population, giving them a second chance to reach a good solution and survive. *iii) Extended update of the secondary population*, which reduces loss of information by updating all non-expired lifetime solutions rather than only the current population. [8]

- A Robust Feature Multi-objective Evolutionary Algorithm (MOEA-RF). Goh and Tan proposed three noise handling techniques and incorporated them into a simple MOEA, naming the new algorithm MOEA-RF. The three noise handling features are the Experimental Learning Directed Perturbation (ELDP), the Gene Adaptation Selection Strategy (GASS) and a possibilistic archiving methodology [9].
- A Modified Non-dominated Sorting Genetic Algorithm-II (MNSGA-II). Deb introduced the Non-dominated Sorting Genetic Algorithm-II (NSGA-II) and Babbar introduced a modification of its ranking scheme to handle noise. The new scheme allows the algorithm to expand its Rank 1 frontier by adding close neighbouring solutions to the rank. It also incorporates a procedure to keep only reliable solutions in the final non-dominated solution set. [10].

MOEA/D, which is a very well established decomposition-based MOEA, introduced for the first time by Zhang and Li in 2007 [11], will be utilised in this thesis to confront noise problems.

As with the other algorithms detailed in Section 1.2, this thesis will investigate MOEA/D in a noisy environment. In order to reach our stated goal certain steps must be taken, the first of which being an answer to the following questions.

- How effective is MOEA/D in the presence of noise?

  It is important to analyse the performance of MOEA/D in the presence of different levels of noise, from low to medium to high. Will its performance deteriorate with increased noise? if so, by how much? In order to measure these qualities, different performance metrics will be implemented (see section 3.4).

- What technique best assists MOEA/D to handle noise?

  Due to the fact that most of the studies on noisy environments explore MOEAs that are based on dominance their noise-handling methods will not be useful for MOEA/D, which is a decomposition-based algorithm. Furthermore, their results are not comparable because parameter settings have a bearing on different algorithms' performance. Thus, in this work, we are seeking a novel technique for handling noise in conjunction with the basic MOEA/D. Our technique will ideally cope with noisy problems and estimate more reliable solutions for them.

Finally, we will assess the new algorithm as to its suitability for real life application.

## 1.3 Thesis Contribution

As previously mentioned, this work will study MOEA/D in the presence of different levels of noise. The major contributions of this thesis are listed as follows:

1. This is the first piece of research that studies the effect of noise on the performance of MOEA/D.
2. We will prove that the performance of MOEA/D deteriorates as noise levels intensify.
3. In Chapter 4 a new algorithm, MOEA/D+OO, based on the MOEA/D framework will be introduced. This is a modified version of MOEA/D that is significantly better suited to handling noise.
4. We will prove that MOEA/D+OO significantly outperforms MOEA/D in the noisy multi-objective optimization problem.
5. We study noisy portfolio optimization for the first time by adding noise only to the return values of the objective function.
6. In this thesis, the noisy portfolio optimization problem is used as a real world application to test the algorithms' performance.

7. We will demonstrate that MOEA/D+OO is better than MOEA/D at handling noise in the portfolio optimization problem.

8. Finally we illustrate that the portfolio optimization problem is very sensitive to noise.

## 1.4 Thesis Outline

The organization of this thesis is as follows:

Chapter 2 provides a review of multi-objective evolutionary algorithms. In this chapter, the fundamentals of evolutionary algorithms and MOP will be summarized. Chapter 3 assesses the performance of MOEA/D in a noisy environment. This chapter explains the theory and methodologies that have been used to examine and assess the algorithm. Chapter 4 proposes a noise handling technique to handle noisy problems. A new algorithm is developed that combines Ordinal Optimization (OO) with MOEA/D. Chapter 5 details the introduction of the algorithm into a real life problem: a classical finance problem in a noisy environment. Finally Chapter 6 presents conclusions, which will wrap up this thesis and propose possible future works.

# 2 Background and Literature Review

This chapter will briefly discuss the principals of optimization theory and the different types of optimization problem. A literature review of previous research studies into multi-objective optimization problems is delivered, along with a discussion of the traditional methods and evolutionary algorithms used for solving multi-objective problems. Thereafter, the major issues in multi-objective optimization evolutionary algorithms (MOEAs) are discussed, alongside a classification of the different MOEAs.

The base algorithm used in this thesis is MOEA/D. A detailed review of it has been prepared in Section 2.4, followed by a look at the literature on noisy MOEAs.

In conclusion, this chapter details the principals of ordinal optimization theory (OO) that are going to be used for noise handling in this study to assist MOEA/D in solving noisy multi-objective problems.

## 2.1 Optimization Theory

In the face of limited resources such as funds, time, space and so on, optimization has become an important area of research within the computational sciences. Different disciplines clearly need to optimize different quantities or possibilities, subject to the specific constraints of their area.

In this section, a succinct general summary and classification of the optimization problem is provided, alongside a look at other issues in this area such as different optima types or different problems.

### 2.1.1 Elements of an Optimization Problem

There are three major elements which are common to any optimization problem as follows [12]:

- **An objective function.** A system model, representing the quantity to be optimized.
- **A set of variables.** These impact the value of the objective function.
- **A set of constraints.** These restrict the values that can be assigned to the variables.

The goal of any optimization method is to assign values, from a given domain, to the variables of the objective function to be optimized such that all constraints are satisfied. In this research, the search space is denoted by $\Omega$ . In the case of a constraint problem, a solution is found in the feasible space that is denoted by $\mathcal{F}$. Always, $\mathcal{F} \subseteq \Omega$ .

### 2.1.2 Classification of Optimization Methods

The different classifications are made according to the specific characteristics of the methods used. For instance, optimization methods can be divided into two major classes [12], dictated by the solutions found, as follows.

- **Local search algorithm:** information local to the current solution is used to produce a new solution.
- **Global search algorithm:** the entire domain is searched for optima.

Further classifications can be introduced as follows:

- **Stochastic:** this method uses random elements to transform a candidate solution into a new solution.
- **Deterministic:** in which no random elements are applied.

### 2.1.3 Classification of Optimization Problems

Optimization problems can have many characteristics and classifications of these can be proposed according to the following [12]:

- **Number of variables:** single variable to multi-variable.
- **Type of variable:** continuous or discrete.
- **Degree of non-linearity:** linear, quadratic, etc.
- **Type of constraint:** boundary, equality and/or inequality.
- **Number of optima:** optimization problems can have one (unimodal) or many (multimodal) solutions.
- **Number of optimization criteria:** if only one objective function requires optimization, it is a 'Single Objective Problem'. If more than one objective function must be optimised simultaneously, the problem becomes 'Multi-objective'.

### 2.1.4 Multi-objective Optimization Problems

Most real-world search and optimization problems naturally involve multiple objectives. The extremist principle mentioned above cannot only be applied to one objective when the rest of the objectives are just as important. Different solutions may produce tradeoffs (conflicting scenarios) among different objectives. A solution that is extreme (in a better sense) with respect to one objective requires a compromise on other objectives.

**Definition 1 (The Multi-Objective Optimization Problem)** *A general MOP comprises a set of $n$ decision variables, a set of $m$ objective functions and a set of $r$ constraints. Objective functions and constraints are functions of the decision variables. The optimization goal is to*

$$\begin{aligned}
\min \quad & y = F(x) = (f_1(x), f_2(x), ..., f_m(x)) \\
s.t \quad & C(x) = (c_1(x), c_2(x), ..., c_r(x)) \leq 0 \\
& x = (x_1, x_2, ..., x_n) \in \Omega \\
& x_i^{(L)} \leq x_i \leq x_i^{(U)} \quad for \quad i = 1, 2, ..., n \\
& y = (y_1, y_2, ..., y_m) \in \Lambda
\end{aligned} \tag{2.1}$$

*where $x$ is the decision vector, $y$ is the objective vector, $\Omega$ is denoted as the decision space, and $\Lambda$ is the objective space. Mapping between the solution space and the objective space is illustrated in Figure 2.1. The constraints $C(x) \leq 0$ determine the set of feasible solutions [2].*

The solutions $x \in \Omega$ of continuous MOPs are a vector of $n$ real variables. Nevertheless the solutions of discrete MOPs are vectors of $n$ integer variables.

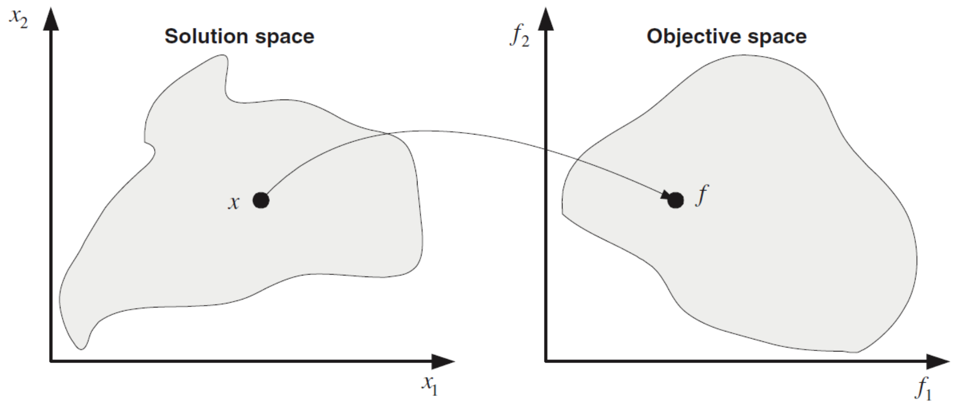


Figure 2.1: Mapping between the solution space and the objective space [1]

The definition of optimality is not straightforward, due to totally conflicting, non-conflicting or partially conflicting objective functions. It is therefore necessary to outline the specific definition of 'optimum' for the MOP: for an MOP the optimum means a balance point between all of the objectives. In other words, improving any one objective may bring about the degrading of other objectives. Thus, our task is to find solutions that balance these tradeoffs. A significant number of solutions may exist for our MOP, so in order to tackle this task, it is necessary to put forward a set of definitions.

Most multi-objective optimization algorithms use the concept of dominance in their search.

**Definition 2 (Dominance)** *A solution $x_1$ is said to dominate another solution $x_2$ , if both conditions 1 and 2 are true:*

1. *The solution $x_1$ is no worse than $x_2$ in all objectives, or $f_i(x_1) \leq f_i(x_2)$ for all $i = 1, 2, ..., m$.*
2. *The solution $x_1$ is strictly better than $x_2$ in at least one objective, or $f_j(x_1) < f_j(x_2)$ for at least one $j \in \{1, 2, \cdots m\}$.*

If either of the above conditions is violated, the solution $x_1$ does not dominate solution $x_2$. If $x_1$ does dominate solution $x_2$ (or mathematically $x_1 \preceq x_2$), it is customary to note any of the following [13]:

- $x_2$ is dominated by $x_1$
- $x_1$ is non-dominated by $x_2$
- $x_1$ is non-inferior to $x_2$.

**Definition 3 (Pareto Optimal Set)** *For a given MOP 2.1, the Pareto Optimal Set (see Figure 2.2), $P^*$, is defined as:*

$$P^* := \{x \in \Omega \ \mid \ \nexists \, x' \in \Omega \ \ F(x') \preceq F(x)\}. \tag{2.2}$$

The Pareto-optimal Set (PS) contains all balanced tradeoffs which represent the MOP solutions.

**Definition 4 (Pareto Front)** *For a given MOP 2.1, and a Pareto Optimal Set, $P^*$, the Pareto Front $PF^*$ is defined as:*

$$PF^* := \{u = F(x) \mid x \in P^*\}. \tag{2.3}$$

The Pareto front contains all the objective vectors corresponding to the decision vectors that are not dominated by any other decision vector (see Figure 2.2).

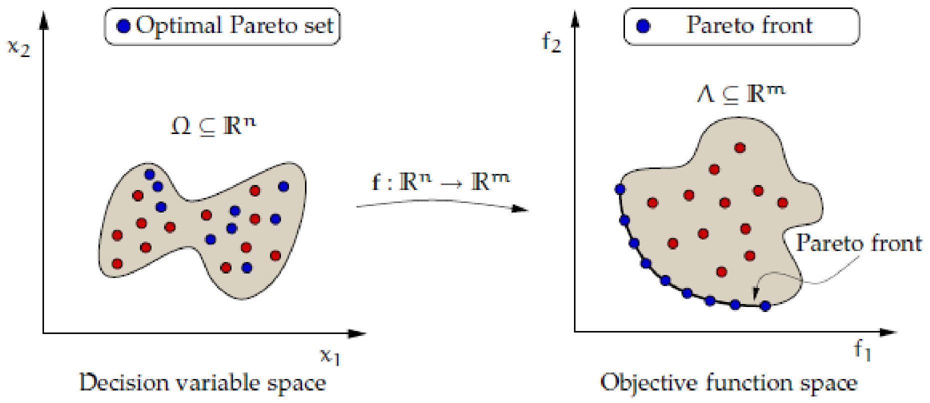


Figure 2.2: Illustration of Pareto front and Pareto set [2].

It is appropriate to note the characteristics of a Pareto front:

1. The Pareto front contains the Pareto-optimal solution and, in the case of a continuous front, divides the objective function space into two parts: non-optimal solutions and infeasible solutions.
2. A Pareto front is not necessarily continuous.
3. The Pareto front can be concave, convex, or a combination of either.
5. The Pareto front may continue towards infinity, even in the case of boundary constrained decision variables.
6. Due to mapping, neighbouring points in a Pareto front (objective function space) are not necessarily neighbours in the decision variable space.

#### 2.1.4.1 Ideal and Nadir Points (Objective Vectors)

We assume that the objective functions are bounded over a feasible region, with two special objective vectors ideal and nadir point to define the lower and upper bounds of PF. Figure 2.3 illustrates both points in the objective space of a hypothetical two objective minimization problem. Definitions of both points are given below.

**Definition 5 (Ideal point)** *A point $z^{idl} = \{z_1, \cdots, z_m\}$ in the objective space is called an ideal point if it has the best value for each objective: $z_i^{idl} = \min_{x \in \Omega} f_i(x) \quad \forall \ i = \{1, ..., m\}$ for problem 2.1.*

**Definition 6 (Nadir point)** *A point $z^{nad} = \{z_1, \cdots, z_m\}$ in the objective space is called a nadir point if it has the worst value for each objective: $z_i^{nad} = \max_{x \in \Omega} f_i(x) \quad \forall \ i = \{1, ..., m\}$ for problem 2.1.*

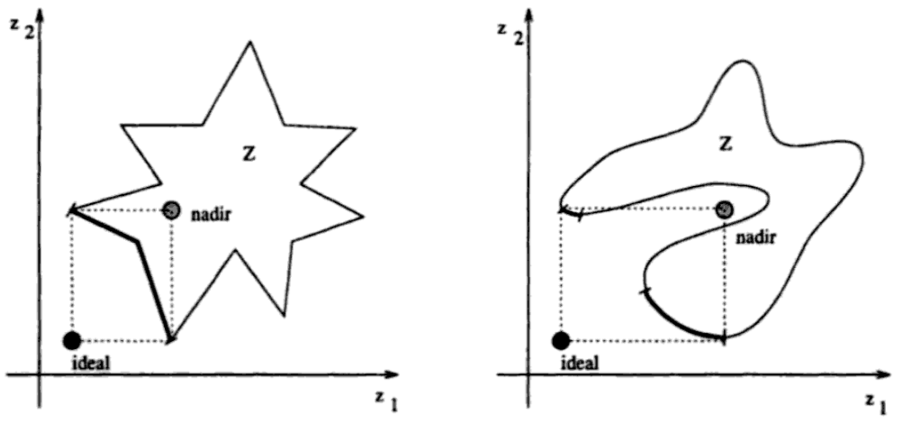


Figure 2.3: Illustration of Nadir and Ideal points [1].

### 2.1.5 Classification of an MOP

Multi-objective optimization problems have been around for at least the last four decades and many algorithms have been evolved to solve them. Researchers have attempted to classify these algorithms according to various considerations. Cohon [14] classified them into the following two types:

- Generating methods.

- Preference-based methods.

In the former, a few non-dominated solutions are generated for the decision-maker, who then chooses one solution from the obtained non-dominated solutions. No *a priori* knowledge of any objective is used. On the other hand, in the preference-based methods, some known preference for each objective is used in the optimization process. Hwang and Masud [15] and later Miettinen [1] fine-tuned the above classification and suggested the following four classes:

- No-preference methods.
- A posteriori methods.
- A priori methods.
- Interactive methods.

The no-preference methods assume no information about the importance of objectives, but a heuristic is used to find a single optimal solution. It is important to note that although no preference information is used, these methods do not make any attempt to find multiple Pareto-optimal solutions. Posteriori methods do use preference information on each objective and iteratively generate a set of Pareto-optimal solutions. The classical method of generating Pareto optimal solutions requires some knowledge of the algorithmic parameters that will guarantee the finding of a Pareto-optimal solution. On the other hand, A priori methods use more information about the preferences of objectives and usually find one preferred Pareto-optimal solution. Interactive methods use the preference information progressively during the optimization process as the decision-maker interacts with the optimization program during the optimization process. Typically the system provides an updated set of solutions and lets the decision-maker consider whether or not to change the weighting of individual objective functions.

The popularity of using a weighted sum of objective functions is obvious: it is trivial to implement and it effectively converts a multi-objective problem into a single objective one. A known drawback is that in the case of a high number of objective functions, the appropriate weighting is painful to choose a priori by the decision-maker. Furthermore, scaling of the individual objective function values is often required due to different function value ranges. With regard to the popularity of a posteriori techniques, especially Pareto-optimization techniques, there are two obvious candidate explanations:

1. The decision-makers are willing to perform unbiased searches.
2. The decision-makers are unwilling or unable to assign priorities without having further information about the other potential/effective solutions.

## 2.2 Traditional Methods of Solving MOPs

Classical ways to address this problem used direct or gradient based methods that rendered them insufficient or computationally expensive for large scale or combinatorial problems. Other difficulties attended the classical methods, such as problem knowledge, which may not be available, or sensitivity to some problem features. For example, finding solutions on the entire Pareto optimal set can only be guaranteed for convex problems. Classical methods for generating the Pareto front set aggregate the objectives into a single or parametrized function before search. Thus, several runs and parameter settings are performed to achieve a set of solutions that approximate the Pareto optimal.

### 2.2.1 The Weighted Sum Method

The idea behind this method is to associate each objective function with a weighting coefficient and minimize the weighted sum of the objective. In this way, multiple objective functions are transformed into a single objective function. More accurately, the multi-objective optimization problem is modified into the following problem, known as a weighted problem:

$$\begin{aligned} &minimize \qquad \sum_{i=1}^{m} w_i f_i(x) \\ &s.t \qquad x \in \Omega \end{aligned} \tag{2.4}$$

where $w_i \geq 0$ for all $i = \{1, ..., m\}$ and $\sum_{i=1}^{m} w_i = 1$.

**Theorem 1** *The solution of the weighted problem (2.4) is weakly Pareto optimal.*

**Theorem 2** *The solution of the weighted problem (2.4) is Pareto optimal if the weighting coefficient is positive, that is $w_i > 0$ for all $i = 1, ..., m$*

**Theorem 3** *Let the multi-objective optimization problem be convex if $x^*$ is Pareto optimal, then there exists a weighting vector $w$ ($w_i \geq 0$ , $i = \{1, ..., m\}$ , $\sum_{i=1}^{k} w_i = 1$.) such that $x^*$ is a solution of the weighted problem (2.4).*

For the proof of all theorems, refer to [1].

Theorem 1 to 3 state the solution of the weighting method is Pareto optimal if the weight coefficients are all positive [1]. The disadvantage of this method is that it is limited solely to convex problems, because a whole solution cannot be found for non-convex problems.

### 2.2.2 $\varepsilon$-Constraint Method

In the $\varepsilon$-constraint method one of the objective functions is selected to be optimized and all the other objective functions are converted into constraints by setting an upper bound to each of them. The problem to be solved is now of the following form

$$\begin{aligned} \textit{minimize} \quad & f_l(x) \\ s.t \quad & f_i(x) \leq \varepsilon_i, \ \ \forall \ \ i = 1, ..., m \ , \ i \neq l \\ s.t \quad & x \in \Omega \end{aligned} \tag{2.5}$$

where $l \in \{1, ..., m\}$. Problem (2.5) is called an $\varepsilon - constraint \ \ problem.$

**Theorem 4** *The solution of $\varepsilon - constraint \ \ problem$ (2.5) is weakly Pareto optimal.*

Proof in [1].

Theorem 4 states that the solutions of equation 2.5 are weakly Pareto optimal without any additional assumptions. After this the theorem 5 regarding the proper Pareto optimality of the solutions of the $\varepsilon - constraint$ problem can be introduced as follows,

**Theorem 5** *A decision vector $x^* \in \Omega$ is Pareto optimal if and only if it is a solution of $\varepsilon$-constraint problem (2.5) for every $l = 1, ..., m$, where $\varepsilon_i = f_i(x^*) \quad for \quad i = 1, ..., m, \quad i \neq l$.*

Proof in [1].

### 2.2.3 Value Function Method

In this method, the decision maker must be able to give an accurate and explicit mathematical form of the value function $U : \mathbb{R}^m \to \mathbb{R}$ that represents his or her preferences globally. This function provides a complete ordering in the objective space.

$$\begin{aligned} maximize \quad & U(f(x)) \\ s.t \quad & x \in \Omega \end{aligned} \tag{2.6}$$

The value function problem is then ready to be solved by any single objective optimization method.

**Theorem 6** *Let the value function $U : \mathbb{R}^m \to \mathbb{R}$ be strongly decreasing. Let $U$ attain its maximum at $f^*$. Then, $f^*$ is Pareto optimal.*

Proof in [1].

## 2.3 Multi-objective Evolutionary Algorithm

### 2.3.1 Evolutionary Algorithm

Evolution is an optimization process that improves the ability of a system to survive in competitive environments [12]. Inspired by Charles Darwin's theory of 'natural selection', evolutionary computation has adopted the following principles of Darwinian natural selection theory.

- Selection $\Longleftrightarrow$ Survival of the fittest.
- Two parents generate two offspring $\Longleftrightarrow$ Crossover or Recombination.
- Small changes in the location (decision variables) of the offspring $\Longleftrightarrow$ Mutation.

The evolutionary algorithm (EA) is a stochastic optimization method. The earliest study in this field dates back to the 1950s and, since the 1970s, several evolutionary methodologies have been proposed. All of these approaches operate on a set of candidate solutions. Using strong simplifications, this set is subsequently modified by two basic principles: selection and variation. While 'selection' mimics the natural world's competition for reproduction and resources among living beings, the other principle, variation, imitates the natural ability to create new beings by means of recombination and mutation.

Evolutionary algorithms such as evolution strategies and genetic algorithms are often used for solving optimization problems that are too complex to be solved using traditional mathematical programming methods [12]. EAs require little knowledge of the problem to be solved and are easy to implement, robust, and inherently parallel.

### 2.3.2 Multi-objective Optimization Problems using EAs

To solve an optimization problem by EA, one must be able to evaluate the objective (cost/loss) functions for a given set of input variables. Due to their ease of implementation, and fitness for parallel computing, EAs are eminently suited to complex problems. Most real-world problems involve simultaneous optimization of several often conflicting objectives. Multi-objective EAs are able to find a set of optimal trade-offs in a single run [2, 13].

EAs work with 'individuals' in a population. The number of individuals in the population is called 'popsize' and each individual has two properties:

- Location, known as 'decision variables'.

- Quality, known as 'fitness value'.

After obtaining the fitness values of all individuals, the selection process generates a 'mating pool'. Only individuals with higher fitness values are allowed into the mating pool. Selected individuals are called 'parents'.

Then, two parents might be selected randomly from the mating pool to generate two 'offspring'. After which, the newly generated individuals replace the old 'parents' and another generation starts.

### 2.3.3 Major Issues in MOEAs

MOEAs regulate the following processes in order to achieve a good approximation of a Pareto front.

#### 2.3.3.1 Reproduction Operators

Reproduction is the process of producing offspring from selected parents. Thus an operator needs to combine or change the value of the parents in the decision space to create new individuals.

The operator that combines the genome of the parents to produce a new individual is called the 'Crossover'. 'Mutation' changes the value of genes in a chromosome randomly. From the first evolutionary algorithm introduced to the current day, different reproduction operators have been proposed, including:

(i) Binary reproduction operators such as, one point, two point or uniform crossover and Gaussian or uniform mutation [2, 13].

(ii) Floating point operators such as, simulated binary crossover (SBX) [16], unimodal normal distribution operator (UNDX) [17], deferential evolution (DE) [18] and simplex crossover (SPX) [19] or polynomial mutation [13] and Gaussian mutation operator [20]. The floating point operator shows better performance when decision variables are floating point values (Real numbers).

In this thesis we will use the DE and Gaussian mutation operators.

**2.3.3.1.1 Deferential Evolution (DE)** In our study we employ DE to create new individuals. DE is a parallel direct search method which creates new candidate solutions by choosing three random individuals from the neighbourhood. DE generates new decision vectors by adding the weighted difference between two parental vectors to a third one. This step is called mutation [18]. The mutated vectors are then mixed with the decision variables from another predetermined vector to create a trial vector. Parameter mixing is often referred to as 'crossover'. There are two predetermined parameters, differential weight ($F \in [0, 2]$) and crossover probability ($CR \in [0, 1]$), that need to be set up either by practice or through a specific method, for instance rules of thumb for selecting parameters [18]. The basic DE algorithm is described in Algorithm 2.1.

**Algorithm 2.1** DE

```
Input: 1) Three randomly selected individuals $x^1, x^2, x^3 = (x_1, x_2, ...x_n)$.
       2) F differential weight.
       3) CR crossover probability
Output: New individual $x' = (x'_1, x'_2, ..., x'_n)$.
   Step 1) Create vector $U$ with uniformly distributed number $U = (u_1, u_2, ..., u_n)$
   Step 2) if $u_i < CR$ then $x'_i = x^1_i + F \times (x^3_i - x^2_i)$
   Step 3) otherwise set $x'_i = x^1_i$ for $i = 1, 2, ..., n$.
```

Sometimes, the newly created candidate falls out of the bounds of the decision variable space. We address this problem by simply replacing the candidate value that violated the boundary constraints with the closest boundary value [21].

**2.3.3.1.2 Gaussian Mutation** If the uniformly distributed number $u \sim U(0, 1)$ is greater than the mutation probability ($P_{mu}$) then this operator adds a Gaussian distributed random value to the decision variables of the chosen individual. If it falls out of the boundary of the decision variables then the violating values are replaced with the closest boundary value [22, 23]. The Gaussian density function is

$$f_{G(0,\sigma^2)}(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

where $\sigma^2$ is the variance [23].

#### 2.3.3.2 Fitness Assignment

As only the best performing individuals get the chance to reproduce, it is important to generate a function that will determine the fitness of each individual, known as 'Fitness Function'. A fitness function maps a fitness vector to a single value, which represents the quality or rank of the individual in the population. Moreover, the fitness function guides the MOEA to search into promising areas in the search space. Pareto dominance ranking, indicator-based and decomposition-based rankings are three major fitness assignment strategies used in MOEAs.

#### 2.3.3.3 Convergence

It is important for any optimization framework to find actual solutions to optimization problems or to make a good estimation of a solution. This process is called convergence. As with any optimization technique, converging to the true Pareto front is important for all MOEAs. Algorithms are comparative in their converging speed [24, 25].

#### 2.3.3.4 Diversity

Obtaining a good distribution of generated solutions along the Pareto front is called 'Diversity'. A diversity maintenance technique avoids convergence of a population to a single solution. Therefore, it is very important. It is a fact that an even spread of discovered solutions is more desirable and different techniques have been established to preserve the diversity of solutions along the Pareto front such as, niche sharing [26], clustering [27], crowding density estimation [28], and nearest neighbour method [29].

#### 2.3.3.5 Elitism

The process that guarantees survival of the best individual in the current population to the next generation is called 'Elitism'. To ensure this, a copy of the current population will be kept, without being mutated; in other words, elitism in MOEAs makes sure that the best (or elite) solutions are kept in a safe place between generations.

### 2.3.4 Classification of MOEAs

There are a diverse range of MOEA classifications in the literature, classified according to the mode of determining fitness function or specific techniques, such as a *Priori*, *Progressive(Interactive)* or a *Posteriori* [2].

In this study we classify the MOEA according to their fitness assignment methods and divide these into three categories including:

#### 2.3.4.1 MOEAs based on Pareto Dominance

One of the most popular approaches to fitness assignment appears to be the Pareto-based ranking. Since its inception, Pareto-based MOEAs such as MOGA [30], PAES [31], NSGA-II [32], SPEA-II [33] have emerged as the most widely used. However, both Fonseca and Fleming [34], [30] have highlighted the inadequacy of an MOEA based on Pareto assignment in high dimensional objectives. In this situation, the Pareto-based MOEA may not be able to produce sufficient selection pressure and also its performance does not scale well with respect to the number of objectives [35].

#### 2.3.4.2 MOEAs Based on Decomposition

This approach aggregates the objectives into a single scalar to approximate the Pareto front. It was in fact the failure of Pareto-based MOEAs in the high dimensional objective space that turned attention to decomposition-based methods. MOGLS [36] and MOEA/D [11] are the two most successful algorithms in this category.

#### 2.3.4.3 MOEAs Based on Indication

Here, the fitness function seeks to rank population members according to their performance in relation to the optimization goal. MOEAs then introduce a utility function to be maximized. For example, one possibility would be to sum up the indicator values for each population member with respect to the rest of the population [37], [38]. IBEA, which was introduced by Zitzler and Künzli, is an example of an indicator-based evolutionary algorithm. For more information see [37].

The Non-dominated Sorting Genetic Algorithm, NSGA-II, is undoubtedly the most well-known and referenced algorithm in the multi-objective literature. It is a GA with random mating of individuals within a population. It is based on obtaining a new population from the original one by applying the typical genetic operators (selection, crossover and mutation); then, the individuals in the two populations are sorted according to their rank, and the best solutions are chosen to create a new population. In the case of having to select some individuals with the same rank, a density estimation based on measuring the crowding distance to the surrounding individuals belonging to the same rank is used to get the most promising solutions [32]. In 2014 a new version of this algorithm was introduced based on adaptive updating and including new reference points on the fly. The resulting adaptive NSGA-III is shown to provide a denser representation of the Pareto-optimal front [39, 40].

The Strength Pareto Evolutionary Algorithm, SPEA2, works on the same random mating of individuals within a population as NSGA-II. In this algorithm, each individual has a fitness value assigned, which is the sum of its strength raw fitness and a density estimation. The algorithm applies the selection, crossover, and mutation operators to fill an archive of individuals; then, the non-dominated individuals of both the original population and the archive are copied into a new population. If the number of non-dominated individuals is greater than the population size, a truncation operator, based on calculating the distances to the $(k - th)$ nearest neighbour, is used [29].

## 2.4 MOEA/D as a Framework

In this thesis, MOEA/D has been studied for handling noisy MOP and this framework will be reviewed as follows:

In order to find a set of $N$ Pareto optimal solutions, MOEA/D decomposes an MOP to $N$ Single-objective Optimization Problem (SOP) (see Figure 2.4). It then solves each subproblem independently. (See Figure 2.5).

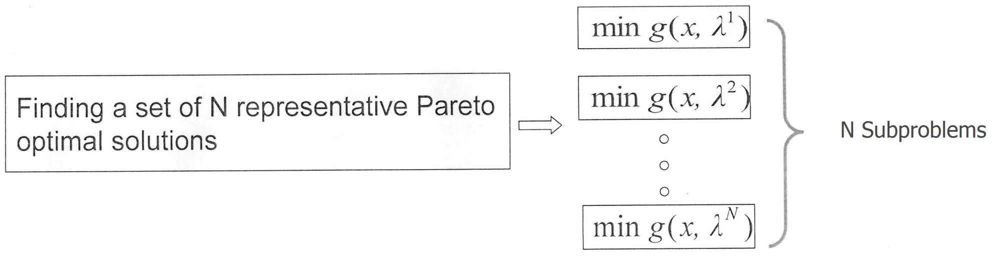


Figure 2.4: Decomposing MOP to N Subproblems.

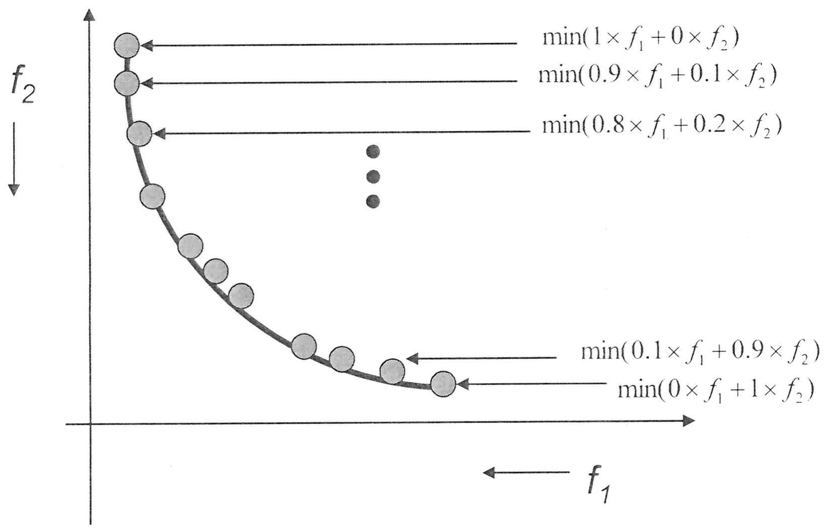

$f_2$
$f_1$
$\min(1\times f_1+0\times f_2)$
$\min(0.9\times f_1+0.1\times f_2)$
$\min(0.8\times f_1+0.2\times f_2)$
$\min(0.1\times f_1+0.9\times f_2)$
$\min(0\times f_1+1\times f_2)$

Figure 2.5: MOEA/D solve N subproblems simultaneously.

### 2.4.1 Decomposition Methods

Decomposition is a general approach to solving a problem by breaking it up into smaller ones and solving each of the smaller ones separately, either in parallel or sequentially. [41].

Decomposition in optimization is an old idea and appears in early work on large-scale $LPs$ [42]. The original primary motivation behind decomposition methods was to solve very large problems that were beyond the reach of standard techniques.

Decomposition of an MOP can be done at different levels. i) Decision variables: in [43] the authors introduced a Dynamical Multi-Objective Evolutionary Algorithm with Domain Decomposition (DMOEA-DD) by using a domain decomposition technique. The decomposition of decision variables is implemented by splitting the original set of decision variables into subgroups and optimizing each group as a subproblem. ii) Objective functions: in [11, 44] the authors introduced algorithms that decompose an MOP into multiple scalar optimization subproblems.

In the following the Tchebycheff decomposition method is introduced. This will be used later in this thesis.

#### 2.4.1.1 Tchebycheff Decomposition Method

The Tchebycheff approach was introduced in [45]. The aggregation function of this method is mathematically defined as follows,

$$\begin{aligned} &\text{minimize} \quad g^{te}(x|\lambda, z^*) = \max_{i\in 1,\cdots,m} \lambda_i |f_i(x) - z_i^*| \\ &\text{subject to} \quad x \in \Omega \subset \mathbb{R}^n. \end{aligned} \tag{2.7}$$

where $z^* = (z_1^*, \cdots, z_m^*)$ is the reference point. $z_i^* = \min\{f_i(x) \ \mid \ x \in \Omega\}$ for each $i = 1, \cdots, m$. The reference point guides the search procedure to converge. (see Figure 2.6).

According to the following theorem for any Pareto optimal solution $x^*$ there is a weight vector $(\lambda_1, \lambda_2)$ such that $x^*$ is the optimal solution to (2.7).

**Theorem 7** *If the Tchebycheff problem 2.7 has a unique solution, then it is Pareto-optimal.*

Proof of this theorem is available in [1].

### 2.4.2 Subproblems

Generating a diverse set of weight vectors is intransitive for the decomposition of the multi-objective problem into multiple single objective problems in order to achieve a good representation of Pareto Front (PF). Table 2.1 shows the process of creating subproblems based on an aggregation function. Every weight vector defines a subproblem and a diverse set of weight vectors leads to a diverse range of subproblems. This results in a diversity of Pareto optimal solutions because, as is mentioned in Section 2.4.1.1,

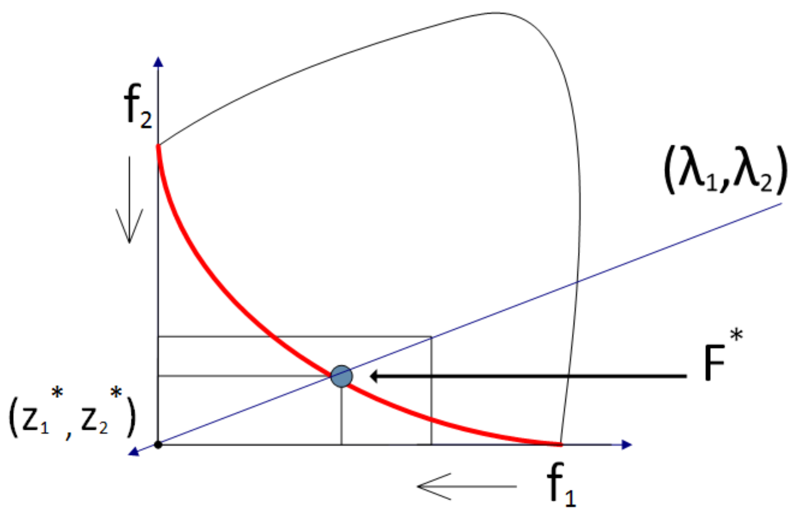


Figure 2.6: Tchebycheff Decomposition Method.

Table 2.1: Create subproblems with evenly distributed weight vectors

| Weight Vectors | Subproblems |
|---|---|
| $\lambda^1 = (1, 0)$ | $g(x, \lambda^1) = 1 \times f_1 + 0 \times f_2$ |
| $\lambda^2 = (0.9, 0.1)$ | $g(x, \lambda^2) = 0.9 \times f_1 + 0.1 \times f_2$ |
| $\lambda^3 = (0.8, 0.2)$ | $g(x, \lambda^3) = 0.8 \times f_1 + 0.2 \times f_2$ |
| . | . |
| . | . |
| . | . |
| $\lambda^N = (0, 1)$ | $g(x, \lambda^N) = 0 \times f_1 + 1 \times f_2$ |

the optimal solution of 2.7 is a Pareto optimal solution for 2.1. This fact is clearly illustrated in Figure 2.7. The authors in [11] introduced a method for generating uniform weight vectors.

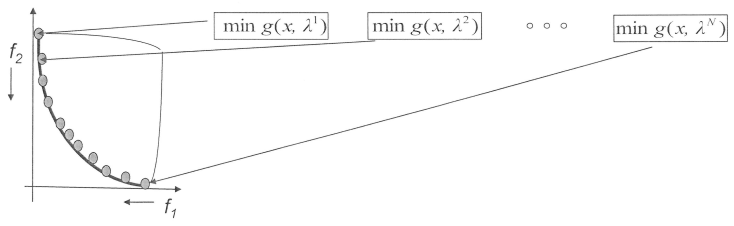


Figure 2.7: Pareto front constructed by optimal solutions of each subproblems.

### 2.4.3 Neighbourhood

Neighbourhood relation in MOEA/D is introduced by computing the Euclidean distances between any two weight vectors and then working out the $T$ closest weight vectors to each weight vector. $T$ is the size of neighbourhood which is set by the decision makers. For each $i = 1, \cdots, N$ set $B(i) = \{i_1, \cdots, i_T\}$ where $\lambda^{i_1}, \cdots, \lambda^{i_T}$ are the closest weight vectors to $\lambda^i$. Note that each weight vector is the closest vector to itself and the neighbourhoods of weight vectors remain unchanged during the whole search process. Figure (2.8) illustrates the neighbouring relations in MOEA/D. $T$ is a major

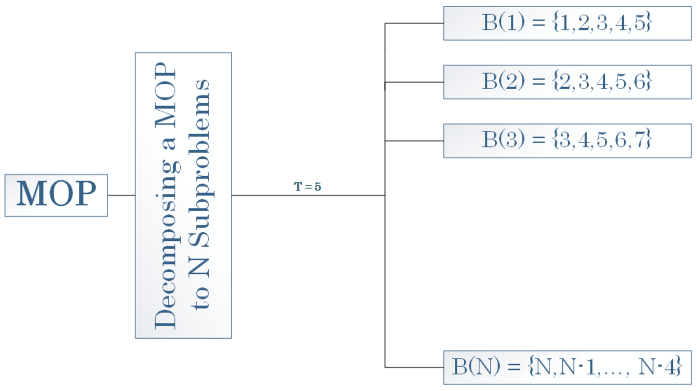


Figure 2.8: Illustration of neighbouring relation in MOEA/D.
(T is the size of neighbourhood)

control parameter in MOEA/D [11] because it is a mating restriction. Two solutions have a mating chance if they are in the same neighbourhood.

### 2.4.4 General Framework

In the framework of MOEA/D, a population of scalar optimization subproblems is maintained and each subproblem is formed by the following components:

- Solution x: is the current best solution of this subproblem.
- Weight $\lambda$: is the weight vector that characterizes this subproblem and determines its search direction.
- Neighbourhood B: the list for each subproblem that contains the indexes of neighbouring subproblems.

After initialization, MOEA/D starts the search process in its main loop. An offspring is generated for each subproblem $i$ by applying the selection, crossover and mutation operators. Then, two neighbouring subproblems of subproblem $i$ are selected randomly from B(i). The selected solutions produce a new solution $y$ by applying genetic operators (crossover and mutation); $y$ is then offered to all subproblems within the neighbourhood of subproblem $i$. If $y$ is fitter than any neighbours, then $y$ will replace that particular neighbour.

A stopping criteria is necessary to stop the algorithm from searching. In this thesis, the stopping criteria employed is a predetermined number of generations.

An external population that holds the best solutions is not practical in continuous MOPs, however. This is because the final generation of this population represents the best result found by MOEA/D an inherent elitism that plays an important role in discrete MOPs. However, in our experiments we focus purely on continuous MOP and for this reason the next chapters will not employ the external population.

Finally, the reference point is a vector which directs the algorithm towards the optimal solution. A reference point constructed as $z^* = (z_1^*, \cdots, z_m^*)$ where $z_i^* = \min \{f_i(x) \mid x \in \Omega\}$ for each $i = 1, \cdots, m$. This can be updated during the search or can be fixed as a predetermined parameter. Algorithm 2.2 describes MOEA/D in detail and more information is available in [11].

In the less than a decade since Zhang and Li introduced MOEA/D in 2007 [11] it has attracted much interest and numerous research studies have been published on the following aspects [46]:

1. Combining MOEA/D with other meta-heuristics, such as simulated annealing [47], colony optimization [48], particle swarm optimization [49, 50], tabu search [51], guided local search [52]and deferential evolution [53].
2. Changing the reproducing operators, such as guided mutation operator [54], nonlinear crossover and mutation operator [55], differential evolution schemes [53], and a new mating parent selection mechanism [46, 56].

3. Research on decomposition techniques. An NBI-style Tchebycheff decomposition approach is proposed to solve portfolio optimization problems by the authors in [57]. In [58, 59] different decomposition approaches are used simultaneously.

4. Improvement on weight vectors. Predetermined, uniformly distributed weight vectors are used to define scaler subproblems in MOEA/D. This reveals that the fixed weight vectors used in MOEA/D might not be able to cover the whole PF very well [47]. Therefore, in [60], the authors create weight vectors predictably based on the distribution of the current weight set. In [61], another weight adjustment method is developed by sampling the regression curve of the objective vectors of the solutions in an external population. The authors in [46] introduce (MOEA/D-AWA), which is an improved version of MOEA/D with an adaptive weight vector adjustment.

5. Applications of MOEA/D like the combinatorial optimization problem, known as the knapsack problem, [47, 58], the travelling salesman problem [47], the flow-shop scheduling problem [51,62] and the capacitated arc routing problem [63]. Or practical engineering problems like antenna array synthesis [64, 65], wireless sensor networks [66], robot path planning [67], missile control [68], a multi-objective optimization for rest-to-rest manoeuvres of flexible spacecraft [69], portfolio management [57] and rule mining in machine learning [70] have also been investigated.

**Algorithm 2.2** MOEA/D

```
Input:
    • A stopping criterion.
    • N: the number of subproblems considered in MOEA/D.
    • A uniform spread of the weight vectors: $\lambda^1, \cdots, \lambda^N$.
    • T: the number of weight vectors in the neighbourhood of each weight vector.
Output:
    • EP or $\{F(x^1), \cdots, F(x^N)\}$.
Step 1) Initialization:
    Step 1.1) Set EP = $\emptyset$.
    Step 1.2) Compute the Euclidean distances between any two weight vectors.
    For each subproblem $i = 1, ..., N$, set the neighbourhood $B(i) = \{i_1, ..., i_T\}$. where
    $\lambda^{i_1}, ..., \lambda^{i_T}$ are the $T$ closest weight vectors to $\lambda^i$.
    Step 1.3) Generate an initial population $x^1, ..., x^N$ randomly.
    Step 1.4) Evaluate the population.
    Step 1.5) Set the reference point $z = (z_1, ..., z_m)$ (see Section 2.4.4).
Step 2) Update:
    For $i = 1, \cdots, N$ do
    Step 2.1) Reproduction: Randomly select two solutions from $B(i)$ to generate
    a new solution $y$ by using genetic operators.
    Step 2.2) Improvement: Apply a problem-specific (repair/ improvement
    heuristic) on $y$ to produce $y'$.
    Step 2.3) Update of z: Update the reference point $z$.
    Step 2.4) Update of Neighbouring Solutions: For each index $j \in B(i)$, set
    $x^j = y'$ if $x^j$ is not fitter than $y'$ regarding to the subproblem $j$.
    Step 2.5) Update of EP: Add $F(y')$ to EP if no vector in EP dominates $F(y')$
    and remove all dominated vectors by $F(y')$ from EP.
Step 3) Stopping Criteria:
   If stopping criteria is satisfied stop and return EP or $\{F(x^1), \cdots, F(x^N)\}$ Otherwise,
   go to Step 2.
```

## 2.5 Ordinal Optimisation Technique.

Ordinal optimization is a ranking and selection approach to solve a simulated optimization problem [71].

### 2.5.1 Introduction

Ordinal optimization concentrates on ordinal comparison and achieves a much faster convergence rate [3]. The idea behind ordinal optimization is to effect a strategic change of goals.

#### 2.5.1.1 Problem Statement

Suppose a general simulation optimization problem was defined as follows:

$$\min_{x \in \Omega} J(x) \equiv E[f(x, \epsilon)] \tag{2.8}$$

Where $J(x)$ is the performance measure of the problem, $L(x, \epsilon)$ is the sample performance, $x$ is a system solution and $\Omega$ is the set containing all the feasible solutions. If $J(x)$ is a scalar function, the problem is a single objective optimization problem; whereas if it were to be a vector valued function, the problem would become a multi-objective optimization problem. The standard approach for estimating the expectation of performance $E[f(x, \epsilon)]$ is the mean performance measure as follows,

$$\bar{J} \equiv \frac{1}{n} \sum_{i=1}^{n} f(x, \epsilon_i) \tag{2.9}$$

Where, n shows the number of simulation samples for solution i.

Due to its huge search space, lack of structure and high uncertainty, solving problem 2.8 is very challenging, either computationally or analytically. The fact that many real world optimization problems remain unsolved is partly due to these very issues. A large number of human-made systems imply combinatorics, symbolic or categorical variables which make the calculus or real variable-based methods less applicable. Search-based methods are required to tackle the difficulty of those models. These allow for a narrowing of the search for the optimum to a 'good enough' subset, rather than the perfect best. After all, real world solutions to real world problems all involve compromise towards 'good enough' rather than perfect.

Further more, it is undoubtedly much easier to simply determine which solution is better than to struggle to find out how much better.

#### 2.5.1.2 Basic Ideas

The fundamental principles of the ordinal optimization method are as follows [3, 72]:

1. Goal softening.
2. Ordinal Comparison.
3. 'Order' converges exponentially fast.
4. 'Order' is much more robust against noise than 'value'.

The first principle, goal softening, holds that it is much easier to find a top-n solution than to find out the global best.

The second principle, namely ordinal comparison, holds that it is much easier to determine which solution is better than how much better. For example, were you to receive two parcels, it would be far easier to identify which one was heavier than to work out the exact weight difference between them.

The third principle, in which order converges faster than value, has been analysed in [73] (pp. 160-163). In addition, the interested reader could refer to [3].

#### 2.5.1.3 Notifications and Concepts

Assume that a subset of search space $\Omega$, defined as 'Good enough' and denoted by G, which could be the top-g solution or top-n% of the solutions of the sampled set of $M$ solutions. The size of the number G is denoted as g ($|G| = g$). Moreover, by selecting some other members of the population, either blindly or by some rule, another subset is defined called ‘Selected Subset’. It is denoted by ‘S’ with the same cardinality as G ($|G| = |S| = g$). Figure 2.9 illustrates the concept of ordinal optimization.

The question here is: what is the probability that among the set ‘S’ we have at least ‘$k$’ of the members of G, which is $P\{|G \cap S| \geq k\}$ and represents another concept of ordinal optimization known as ‘Alignment Probability’. It is a measure of the rightness of our selection rules. Alternatively there are some special cases of alignment probability which are denoted by $P(CS)$ and stand for probability of ‘Correct Selection’ [72]. This probability is calculated for discrete systems with blink picking in [3, 72].

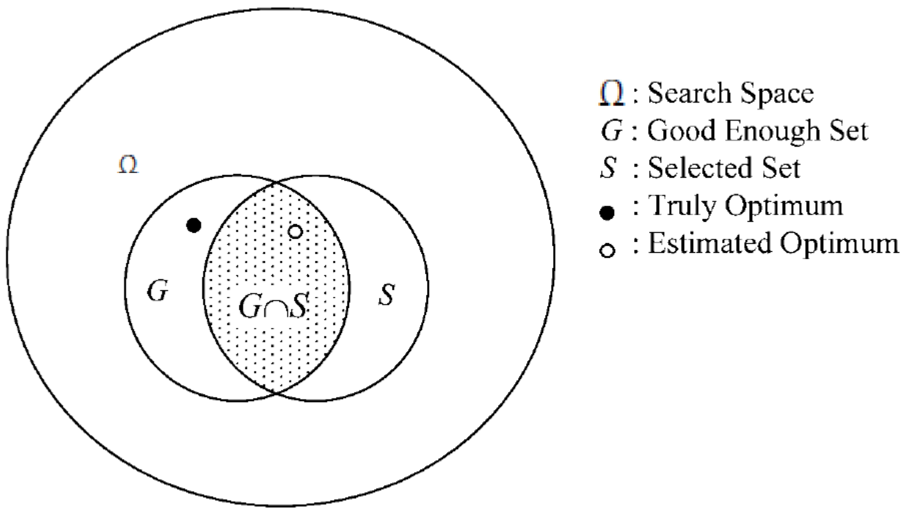


Figure 2.9: Generalized concept of Ordinal Optimization [3].

#### 2.5.1.4 Definitions, Terminologies and Concepts of OO

Ordinal optimization uses a crude system model to order the solutions in the search space. A crude model is one with a lower computational cost that allows the simulation to converge faster.

In addition, it utilizes a different method to select set $S$. A selection rule is a procedure that selects the set $S$ based on observed performance of the solutions, such as blind picking or horse racing etc.

The ordinal optimization (OO) procedure is summarized in Algorithm 2.3 [3]. As we study the multi-objective optimization problem, the concept of OO by itself is not

**Algorithm 2.3** Ordinal Optimization Procedure

```
Require: Search Space $\Omega$
    Step 1 : Pick M random solutions from $\Omega$
    Step 2 : Specify the size of the good enough set $G$ and alignment level $k$.
    Step 3 : Use crude model to estimate the performance of $N$ solutions.
    Step 4 : Estimate the noise level and the problem type.
    Step 5 : Calculate $s$, the size of selected set.
    Step 6 : Select the observed top-s solutions.
    Step 7 : Then employ OO theory to ensure there are at least $k$ truly good enough
    solutions in $S$ with a certain probability.
```

helpful. However in [74] the authors extended the concept of OO for vector optimization problems and called it Vector Ordinal Optimization (VOO). We will implement VOO later in this research.

### 2.5.2 Vector Ordinal Optimization

When ordinal optimization was first developed it was initially proposed to solve a stochastic simulation optimization with a single objective and no constraints [3, 74]. Very soon, however, the idea was extended to multi-objective problems, constrained optimization problems and so on [74].

#### 2.5.2.1 Definitions, terminologies and concepts of VOO

Practical problems in the finance or industry sectors involve multiple simulation-based objective functions and, in most cases, decision makers have no prior knowledge as to priority nor appropriate weighting amongst the objective functions.

Different studies have proposed various ways to introduce order amongst the solutions in vector ordinal optimization. The first and most common way is to follow the definition of Pareto front.

**Definition 7 (Dominance)** *Assume that we have two solutions, $x_1$ and $x_2$. $x_2$ dominates $x_1$, denoted by $x_2 \prec x_1$, if both the following conditions hold:*

$$\forall \;\; i \in \{1, 2, ..., m\}, \quad J_i(x_2) \leq J_i(x_1)$$

$$\exists \;\; j \in \{1, 2, ..., m\}, \quad J_j(x_2) < J_j(x_1)$$

where $m$ is the number of objective functions in the simulation-based optimization problem.

**Definition 8 (Pareto frontier)** *A set of solutions $L_1$ is called the Pareto frontier if it contains only the non-dominated solutions,*

$$L_1 \equiv \{x \mid x \in \Omega, \quad \nexists \;\; x' \in \Omega, \quad s.t. \;\; x' \prec x\}$$

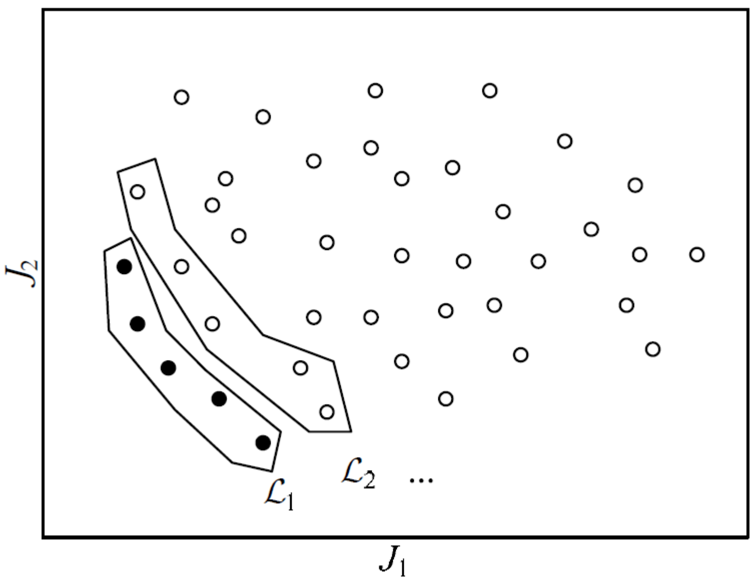


Figure 2.10: Illustration of Layers [3].

The concept of Pareto frontier introduces an operator $\omega$ that maps the solution space to the set of Pareto fronts with respect to the objective functions as $L_1 = \omega(\Omega)$ [74]. The concept of Pareto frontier can extend to a sequence of layers. This can be seen in Figure 2.10.

**Definition 9 (Layers)** *A series of solutions $L_{s+1} = \omega(\Omega \backslash \bigcup_{i=1,2,...,s} L_i)$ , $s = 1, 2, ....$ are called layers. $A \backslash B$ denotes the set containing all the solutions included in the set $A$ but not included in the set $B$.*

Without any additional problem information, there are no preferences as to objective function and no preferences as to solution in the same layer.

The procedure of VOO is summarized in Algorithm 2.4 that will be used later in this thesis.

```
Algorithm 2.4 Vector Ordinal Optimization
Require: Search Space $\Omega$
  Step 1 : Pick $M$ random solutions from $\Omega$
  Step 2 : Use crude model (computationally fast) to estimate the performance of $N$ solutions.
  Step 3 : Select the observed top-s layers. (selected set $S$).
  Step 4 : Evaluate the selected layers with exact model (more refined model) to estimate the optimal solutions.
```

The second method for introducing order among the solutions is to count the number of solutions that dominate a solution $x$, denoted as $n(x)$, then to sort all the solutions according to $n(x)$ in ascending order [75]. Solution $x_i$ is deemed better than $x_j$ if $n(x_i) < n(x_j)$. And solutions $x_i$ and $x_j$ are regarded as equally good solutions if $n(x_i) = n(x_j)$.

An Order Based Genetic Algorithm (OGA) was introduced in [76], based on the idea of ordinal optimization, to ensure the quality of the solution found with a reduction in computational effort.

The authors in [77] combine OO and Optimal Computing Budget Allocation (OCBA) within the search framework of GA to propose a novel Genetic Ordinal Optimization (GOO) algorithm to solve the stochastic travelling salesman problem.

In [78] the authors incorporate particle swarm along with OO for a stochastic simulation optimization problem. The new algorithm Combined Particle Swarm with Ordinal Optimization (CPSOO) is applied to solve the centralized broadband wireless network problem.

The authors in [78] combine evolution strategy with ordinal optimization to solve a wafer testing problem. They called this new algorithm (ES+OO). In another study [79] they solve the same problem with (GA+OO), which is a combination of a genetic algorithm with ordinal optimization.

An ordinal optimization-based algorithm is also used for the hotel booking limits problem in [80]. The authors construct a crude mode as a fitness evaluation function in Particle Swarm Optimization (PSO) Algorithm to select $M$ candidate solutions and then use OCBA to search for a good enough solution.

In this thesis we will use the ordinal optimization technique to handle uncertainty for the first time.

## 2.6 Noisy MOEAs

In real-world problems characterized by noise, precise determination of the fitness value for individual solutions is a major challenge. This is because the noise may be associated with different sources, including erroneous sensory measurements and randomized simulations. Such noise causes an uncertainty in the fitness evaluation of potential solutions and eventually adversely affects the search efficiency, convergence and self-adaptation of evolutionary algorithms (EAs) and other heuristic search algorithms.

Uncertainty in the context of evolutionary optimisation can be divided into four major categories [4], as follows:

1. Noise: The noisy fitness function ($F(\mathbf{X})$) may be described as:

   $$F(\mathbf{X}) = f(\mathbf{X}) + \zeta$$

   where $\mathbf{X}$ denotes the parameter-vector, $f(\mathbf{X})$ the fitness function without noise, and $\zeta$ the additive noise. In that, though $\zeta$ is often assumed to have a Gaussian distribution, it may have non-Gaussian distributions as well. Notably, given the randomness associated with the noise, different fitness values may be obtained for the same solution in different evaluations.

2. Robustness: Here, the parameter-vector is perturbed after the optimal solution has been obtained, and a solution is still required to work satisfactorily. In this case, the expected fitness function ($F(\mathbf{X})$), as below, may be used:

   $$F(\mathbf{X}) = f(\mathbf{X} + \delta)$$

where $\delta$ represents the perturbation.

3. Fitness approximation: In situations where either an analytical fitness function may not be available or its evaluation may be very expensive, the fitness function may need to be approximated based on experimental or simulation data. The approximated fitness function, often referred to as the meta-model ought to be used together with the original fitness function as follows:

$$F(\mathbf{X}) = \begin{cases} f(\mathbf{X}), & \text{if the original fitness function is used} \\ f(\mathbf{X}) + E(\mathbf{X}) & \text{if the meta-model is used} \end{cases}$$

where, $E(\mathbf{X})$ is the approximation error.

4. Time-varying fitness functions: Here, the fitness function is deterministic at any point in time but is dependent on time $t$, and may be described by:

$$F(\mathbf{X}) = f_t(\mathbf{X})$$

Among the above categories, the issue of handling noise in fitness evaluations is often an important one in several domains, including evolutionary robotics [81], evolutionary process optimization [82], and evolution of en-route cashing strategies [83]. In order to address this issue, three major approaches have been identified [4], as follows:

1. Explicit Averaging (Fitness Averaging): This calls for estimating the fitness by averaging over a number of samples taken over time. Notably, each sampling may be quite expensive, hence a balance between the sample size and performance becomes critical. The authors in [84, 85] suggested two adaptation schemes: *i*) increasing the sample size with generation number and using a higher sample size for individuals with higher estimated variance. The author in [86] concludes that for small population sizes, sampling is able to improve the learning performance. Moreover it is also mentioned that sampling does not help if the population size is generously large.

2. Implicit Averaging (Population Sizing): This calls for negating the effect of noise

by increasing the population size. For instance, the authors in [87] have demonstrated that when the population size is infinite, proportional selection is not affected by noise.

3. Modifying Selection: This calls for modifying the selection process in order to cope with noise. For instance, the authors in [88] proposed to de-randomize the selection process, and demonstrated that the effect of noise could be significantly reduced without a proportional increase in computational cost. Notably, this approach has also been studied in the context of multi-objective optimization, where Pareto-dominance is used for selection. In the latter, the authors in [89] and [90, 91] have proposed that an individual solutions Pareto-rank be replaced by its probability of being dominated.

A number of approaches have also been proposed to reduce the disruptive effect of noise such as population sizing [92, 93], fitness averaging and fitness estimation [94–96], specific selection method [97–99], and Kalman filtering [100].

A few noise handling techniques in MOEAs have been introduced which include periodic re-evaluation of achieved solutions [8], probabilistic Pareto ranking [90], the extended averaging scheme [101], experiential learning directed perturbation [102] and gene adaptation selection strategy [102].

There are some MOEAs which are facilitated by specific noise handling techniques to tackle the disruptive impact of noise, for instance NTSPEA [8], Multi-objective Probabilistic Selection Evolutionary Algorithm (MOPSEA) [103], a robust feature multi-objective evolutionary algorithm (MOEA-RF) [9] and MNSGA-II [10].

The authors in [104] examined the effect of noise on both local search and genetic search to understand the potential effects of noise on the search space.

Optimization in noisy and uncertain environments is regarded as one of the favourite application domains of evolutionary algorithms [6]. Research in the field of noisy MOEAs is still in its infancy. Compared to its practical relevance, the effect of noise and its influence on the performance of MOEAs has gained relatively little attention in EA research [8, 90, 105].

## 2.7 Conclusions

In this chapter we briefly reviewed the basic concepts of optimization theory by focusing on the multi-objective optimization problem. Having discussed the traditional approaches used to solve these problems, we outlined a modern heuristic method, the 'Evolutionary Algorithm', for solving multi-objective optimisation. Then followed a detailed discussion of the major issues confronting multi-objective evolutionary algorithms.

MOEA/D was reviewed in this chapter as a framework for optimizing multiobjective problem. We will use MOEA/D as a base algorithm for further research. A literature review on noisy MOEAs was provided.

Furthermore, an introduction to the ordinal optimization technique has been explored, covering both single and multi-objective optimization problems. We will combine this technique with the MOEA/D algorithm to handle noise.

# 3 MOEA/D in Noisy Environments

In the previous chapter we discussed MOEA and its major issues. Noise is one of these. It poses a significant challenge to MOEA because the noise, spread as it is from different sources, causes uncertainty in the fitness evaluation of potential solutions and eventually adversely affects search efficiency, elitism, convergence and the self-adaptation of Evolutionary Algorithms (EAs) and other heuristic search algorithms.

Does noise matter in the case of MOEA/D? Will its performance be affected by noise? If so, how seriously? Results obtained in this chapter do reveal a meaningful deterioration in the performance of MOEA/D when noise intensifies. Thus, this chapter will provide answers to the above questions, but in order to get to that point we will first define some common concepts in noisy multi-objective optimization.

## 3.1 Multi-objective Optimization Problems in Noisy Environments

A noisy MOP is an MOP whose objective function is disrupted by noisy terms. A noisy multi-objective problem can be described as follows:

$$\begin{aligned} &\min F(x) = (f_1(x) + \delta_1, \cdots, f_m(x) + \delta_m) \\ &s.t \qquad x \in \Omega \end{aligned} \tag{3.1}$$

where $x$ is a 'decision vector' and $\delta_i$ for $i = 1, 2, ..., m$ are disruptive noises with scalar values.

In this study, an unbiased (zero mean) Gaussian perturbation is added to the objective functions [102].

$$\begin{aligned} F(X) &= f(X) + \delta \\ \delta &\sim N(0, \sigma^2) \end{aligned} \tag{3.2}$$

where $\sigma^2$ denotes the level of noise, while $F(x)$ and $f(x)$ represent the objective functions with and without noise respectively.

In this thesis, it is assumed that noise has a disruptive influence on the value of each individual in the objective space and it is common practice as used in [9,90,91,95,96,106].

## 3.2 Evolutionary Multi-objective Optimization in Noisy Environments

In this section, we explain how the research reported in this chapter relates to other work in the literature.

To begin with, there are different ways to model noise. The majority of them, including this research, use the Gaussian model. In [107], Arnold and Beyer conducted a comparison of the influence of Gaussian, Cauchy and $\chi^2$ distributed noise on the performance of evolutionary strategy (ES) .

Secondly, most research into noisy optimization focuses on single objective problems [107]. In this thesis, we focus on MOP.

Thirdly, studies on EA for noisy MOPs have been conducted, [5,9], and a number of approaches have been proposed in recent decades by different studies aimed at decreasing the impact of noise on MOEA such as population sizing [93], fitness estimation [96] and modified selection schemes [9, 98].

The aim of this thesis is to improve MOEA/D in noisy MOPs. We are not comparing the performance of proposed methods to others in the literature at this stage. In any case, a beauty contest would not be straightforward or meaningful because different methods could perform better in different problems. Besides, performance could also be affected by the parameters and fitness measures used in different algorithms.

## 3.3 MOEA/D Algorithm

MOEA/D is a population-based algorithm that decomposes the MOP to $N$ scalar optimization problems and optimizes them simultaneously rather than seeking to solve the MOP as a whole. All traditional mathematical decomposition techniques are applicable such as *Weighted Sum*, *Tchebycheff Approach* and so on.

Diversity in the subproblems naturally brings diversity to the population. A properly chosen weight vector and decomposition method can result in an evenly distributed solution along the PF as described in Section 2.4 [108].

In MOEA/D a neighbourhood of subproblems is defined as $T$ closest subproblems. The closeness of subproblems is measured by the Euclidean distance of weight vectors between each subproblem. Subproblems share information such as optimal points with neighbouring subproblems.

In this research, we use MOEA/D with the Tchebycheff decomposition method that is described in Section 2.4.1.1. All the steps of this framework are listed in Algorithm 3.1 and further details are available in [11] and Section 2.4.

**Algorithm 3.1** MOEA/D for Solving Noisy MOP

```
Input:
  • MOP 3.1.
  • A stopping criterion;
  • N: the number of subproblems considered in MOEA/D.
  • A uniform spread of the weight vectors: $\lambda^1, \cdots, \lambda^N$.
  • T: the number of the weight vectors in the neighbourhood of each weight vector.
Output:
  • $\{F(x^1), \cdots, F(x^N)\}$.
Step 1) Initialization:
  Step 1.1) Compute the Euclidean distances between any two weight vectors.
  For each subproblem $i = 1, ..., N$, set the neighbourhood $B(i) = \{i_1, ..., i_T\}$. where
  $\lambda^{i_1}, ..., \lambda^{i_T}$ are the $T$ closest weight vectors to $\lambda^i$.
  Step 1.2) Generate an initial population $x^1, ..., x^N$ randomly.
  Step 1.3) Evaluate the population.
  Step 1.4) Set the reference point $z = (z_1, ..., z_m)$ (see Section 2.4.4).
Step 2) Update:
  For $i = 1, \cdots, N$, do
  Step 2.1) Reproduction: Randomly select two solutions from $B(i)$ to generate
  a new solution $y$ by using genetic operators.
  Step 2.2) Update of z: Update the reference point $z$.
  Step 2.3) Update of Neighbouring Solutions: For each index $j \in B(i)$, set
  $x^j = y$ if $x^j$ is not fitter than $y$ regarding to the subproblem $j$.
Step 3) Stopping Criteria:
  If stopping criteria is satisfied stop and return $\{F(x^1), \cdots, F(x^N)\}$. Otherwise, go
  to Step 2.
```

## 3.4 Performance Metrics

Performance metrics play an important role in returning a scalar value to represent the quality of a solution set with respect to a given measure. Due to the nature of MOP several performance metrics are needed to gauge the performance of an algorithm [13, 109].

1. Proximity Indicator: The generation gap between $PF_{true}$ and $PF_{approx}$ indicates the closeness of the approximated Pareto front and true Pareto front. The true Pareto front is the global Pareto optimal set [9, 110]. For ZDT problems, Zitzler

and others produced a very good approximation of the true Pareto front on their website[1]. Their approximation is also covered in Appendix A. Mathematically, generational distance(GD) is formalized as:

$$GD = \left(\frac{1}{n_{PF}} \sum_{i=1}^{n_{PF}} d_i^2\right)^{1/2} \tag{3.3}$$

where $n_{PF}$ is the number of elements in $PF_{true}$ and $d_i$ is the Euclidean distance (in objective space) between member $i$ of $PF_{true}$ and its nearest member of $PF_{approx}$. Notably, a lower value of GD implies a better approximation of the Pareto front.

2. Diversity Indicator (*Maximum Spread*): $MS$ measures how well the true Pareto front is covered by the approximated Pareto front [9,110]. To assess the diversity of solutions in $PF_{approx}$ vis-à-vis $PF_{true}$, the following metric will be used:

$$MS = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left[\frac{min(f_i^{max},F_i^{max}) - max(f_i^{min},F_i^{min})}{F_i^{max}-F_i^{min}}\right]^2} \tag{3.4}$$

where:

$m$ is the number of objective functions.

$f_i^{min} \& f_i^{max}$ are the minimum and maximum of $f_i$ in $PF_{approx}$.

$F_i^{min} \& F_i^{max}$ are the minimum and maximum of $f_i$ in $PF_{true}$.

Notably, by converging to 1, MS shows that the approximated Pareto front properly covers the true Pareto front.

3. Distribution Indicator (*Spacing*): To assess the uniformity of distribution between solutions along $PF_{approx}$ [9,110] the following metric will be used:

$$S = \left[\frac{1}{n_{PF}-1}\sum_{i=1}^{n_{PF}}(d_i - \bar{d})^2\right]^{\frac{1}{2}} \tag{3.5}$$

where

$$\bar{d} = \frac{1}{n_{PF}}\sum_{i=1}^{n_{PF}} d_i$$

[1]http://www.tik.ee.ethz.ch/sop/download/supplementary/testproblems/

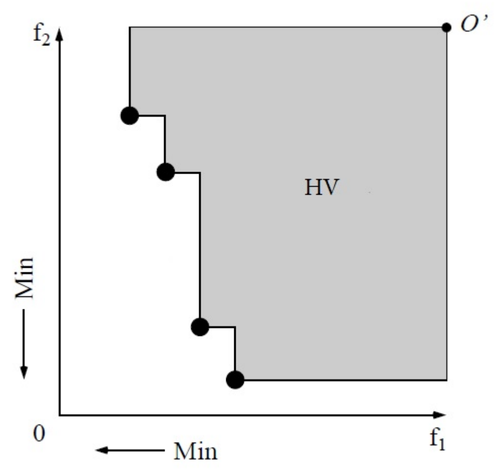


Figure 3.1: HV: Area that is dominated by solution set.

$d_i$ is the Euclidean distance between the i-th member and its nearest neighbour in PF and $n_{PF}$ is the number of elements in $PF_{approx}$. Notably, a smaller value of spacing implies a more uniform distribution of solutions in $PF_{approx}$.

4. General Quality Indicator: the hypervolume($HV$) metric indicates the general quality of a solution set by taking into account its performance in diversity and proximity [9, 110]. Hypervolume indicates the size of area that is dominated by a solution set as in Fig.3.1 below. A reference point $O' = (o_1, o_2, ..., o_m)$ is defined where $o_i$ represents the worst values for objective function $i$. Finally $HV$ metrics can be defined as follow:

$$HV = volume \bigcup_{i=1}^{n_{PF}} v_i \tag{3.6}$$

where $v_i$ is a hypercube between solution $i$ and the reference point which is constructed as the diagonal corner of the hypercube. Veldhuizen and Lamont expressed this metric as a ratio between the $PF_{approx}$ and $PF_{true}$

$$HVR = \frac{HV(PF_{approx})}{HV(PF_{true})} \tag{3.7}$$

Notably, $PF_{approx}$ is a good approximation of $PF_{true}$ if its hypervolume metric value is close enough to the hypervolume metric value of $PF_{true}$. Consequently it is desirable that the $HVR$ metric should merge to a value of 1 .

## 3.5 Experiment

The purpose of this experiment is to find the impact of noise on the performance of the MOEA/D. The behaviour of the MOEA/D is tested on different levels of noise, which helps us to detect the destructive effects of noise with some scale. Noises are added to the test functions that are summarized in Section 3.5.2, in the form that is mentioned in Section 3.1.

### 3.5.1 Design of Experiment

In order to study the impact of noise on MOEA/D, an experiment has been designed to challenge the algorithm in the presence of different levels of noise, from low $(1\%, 2\%)$, to medium $(5\%)$ to high $(10\%, 20\%)$.

In this thesis we implement the DE operation, along with a Gaussian mutation operator to generate new individuals. Both of these operators are reviewed in Sections 2.3.3.1.1 and 2.3.3.1.2. In our experiment, DE parameters are set as $(F = 0.5)$ for differential weight and $(CR = 0.5)$ for crossover probability. This setting has been implemented before (see MOEA/D homepage), meaning we are therefore using the same settings as those used by the original authors of MOEA/D. The new offspring just produced by the DE operator then undergoes the Gaussian mutation with the following mutation probability:

$$P_{mu} = \frac{1}{Number\ of\ Decesion\ Variables}$$

Finally, this mutated offspring will be referred as a new individual to the optimization framework.

The performance of MOEA/D is affected by its parameter settings. The most influential parameters are population size and neighbourhood size, as well as maximum iterations. We adopt the following set up in the experiment:

- Number of subproblems and population size: as MOEA/D decomposes the MOP into $N$ scalar subproblems and a population of $N$ solutions $x^1, ..., x^N$ is maintained, where $x^i$ is the best solution found so far for the $i-$th subproblem. Without loss of generality, a fixed population of one hundred individuals ($N = 100$) has been considered sufficient for this study. The amount is entirely arbitrary.
- Neighbourhood size: To ensure better exploration and exploitation, attention should be paid to the size of neighbourhood. As a result, twenty percent of the population is considered as the neighbourhood size, which offers a good chance for neighbouring solutions to mate [108].
- Number of iterations: The algorithms will stop when maximum generation is reached. In our study a total of 150 iterations will complete the search process. It has been empirically observed that there are no significant improvements after 150 iterations, hence our setting up the algorithm to stop at that point to reduce the computational cost.

Finally, fifty independent simulation runs are conducted for each of the noisy problems. We run our experiments in Matlab.

No other study covers computational cost in noisy research. For instance, see [9]. At this stage finding a proper and trustworthy method for handling noise is the primary concern in noisy optimization research. Computational cost and time remain minor considerations for the moment.

### 3.5.2 Benchmark Problems

To reveal capabilities, possible pitfalls and specific characteristics of the algorithms, researchers use benchmark problems. These have different features such as multi-modality, convexity, discontinuity and non uniformity of the Pareto front. These features may prevent the MOEAs from finding a diverse set of solutions.

Six benchmark problems, FON, KUR, ZDT1, ZDT3, ZDT4 and ZDT6 have been selected to be used in this research. Many researchers have applied these test problems

Table 3.1: Definition of the test functions.

| Problems (Characteristics) | Variables Number (n); bounds | Objective Functions |
|---|---|---|
| FON<br>Non-convex | 3; [-4 , 4] | $f_1(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i - \frac{1}{\sqrt{3}})^2)$<br>$f_2(x) = 1 - \exp(-\sum_{i=1}^{3}(x_i + \frac{1}{\sqrt{3}})^2)$ |
| KUR<br>Non-convex<br>Disconnected | 3; [-5 , 5] | $f_1(x) = \sum_{i=1}^{n-1}(-10\exp(-0.2\sqrt{x_i^2 + x_{i+1}^2}))$<br>$f_2(x) = \sum_{i=1}^{n}(\lvert x_i\rvert^{0.8} + 5\sin(x_i^3))$ |
| ZDT1<br>Convex | 30; [0 , 1] | $f_1(x) = x_1$<br>$f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$<br>$g(x) = 1 + 9(\sum_{i=1}^{n} x_i)/(n-1)$ |
| ZDT3<br>Non-convex<br>Disconnected | 30; [0 , 1] | $f_1(x) = x_1$<br>$f_2(x) = g(x)[1 - \sqrt{x_1/g(x)} - \frac{x_1}{g(x)}\sin(10\pi x_1)$<br>$g(x) = 1 + 9(\sum_{i=2}^{n} x_i)/(n-1)$ |
| ZDT4<br>Non-convex<br>Multimodal | 10; $x_1 \in [0,1]$<br>$x_i \in [-5,5]$<br>$i = 2, \cdots, n$ | $f_1(x) = x_1$<br>$f_2(x) = g(x)[1 - \sqrt{x_1/g(x)}]$<br>$g(x) = 1 + 10(n-1) + \sum_{i=2}^{n}[x_i^2 - 10\cos(4\pi x_i)]$ |
| ZDT6<br>Non-convex<br>Non-uniformly<br>distributed | 10; [0 , 1] | $f_1(x) = 1 - \exp(-4x_1)\sin^6(6\pi x_1)$<br>$f_2(x) = g(x)[1 - (f_1(x)/g(x))^2]$<br>$g(x) = 1 + 9[(\sum_{i=2}^{n} x_i)/(n-1)]^{0.25}$ |

to assess the effectiveness of their proposed algorithms [9, 11, 28, 29, 32, 111–114]. The definitions of the selected problems are outlined in the following and in Table 3.1.

- FON is a non-convex, non-linear problem. It is difficult for algorithms to maintain a stable evolving population for FON [9]. The challenge this problem poses to algorithms is finding and maintaining a uniform Pareto front [115, 116]. Thus, for this problem, the algorithms' performance can be easily assessed and compared via observations of the Pareto front.

- KUR uses two complicated objectives with a non-convex and disconnected Pareto front. There are a total of three distinct disconnected regions on the Pareto frontier. Furthermore, the decision variables are also disconnected in the decision space and difficult to discover. This challenges the algorithm's ability to cope with discontinuities and non-convexities. [117]

- ZDT1 is a problem with a large number of decision variables (30) to be optimized.

It has a convex Pareto front. This problem challenges the algorithm's ability to converge and maintain a diverse solution on a convex Pareto frontier

- ZDT3 is a problem with 30 variables to be optimized. Its Pareto front has several noncontiguous convex parts that represent the discreteness feature. These features challenge the algorithm to find the optimal. However, there is no discontinuity in the parameter space.
- ZDT4 is a problem with 10 variables where the first variable is in [0, 1] and the rest are in [-5 , 5]. This problem contains $21^9$ local Pareto optimals that challenge the ability of the algorithms to deal with multimodality [113, 114].
- ZDT6 is a problem with 10 variables and a non-convex formed Pareto front. This presents two difficulties. First, the Pareto optimal solutions are non-uniformly distributed along the global Pareto front and, second, the density of solutions is lowest near the Pareto front and highest away from the front. [113, 114]

The characteristics of these problems are highlighted in Table 3.1. These test functions are modified in the form of 3.1 in order to include the impact of noise.

### 3.5.3 Results

True Pareto front is the best recent estimation of each problem, therefore we plot our results on the true Pareto front to illustrate how the algorithms work.

For a convenient evaluation of the results, the desirable values of the performance metrics are restated here: for the GD and S metrics 0 is the goal, and for the MS and HVR metrics 1 is the desirable value. Closeness of the performance metric values to the desirable values indicates the quality of the estimated solutions.

- FON: Figure 3.2 shows the true Pareto front and estimated Pareto front of the noisy FON problem by MOEA/D. Table 3.2 includes the performance metric values of MOEA/D in the presence of different levels of noise. It is desirable that performance metrics GD and S converge to 0 and MS and HVR converge to 1 as indicated in Section 3.4. As can be seen from Table 3.2, the performance of the

MOEA/D algorithm deteriorates sharply when noise intensity increases. This is due to the fact that, when the noise level is 20%, the performance metric values are far from acceptable.

- KUR: Figure 3.3 shows the obtained result from the benchmark problem KUR. Table 3.3 represents the performance metric values of this benchmark problem in the presence of different noise levels. Similar to FON, it can be seen from the values in the table for this problem (KUR) that MOEA/D does not perform well when noise levels increase. For example, the GD values rocket up from 0.0326 to 0.05098 when noise levels vary from 1% to 20%. The gap between the estimated Pareto front and true Pareto front increases dramatically in Figure 3.3 and the performance metric values degenerate when noise levels increase. However, the obtained results would indicate that noise effects its most gentle impact on the performance of MOEA/D in this specific problem.

- ZDT1: Figure 3.4 shows the estimated Pareto front of ZDT1 found by MOEA/D in the presence of different levels of noise in comparison with the true Pareto front. An evolutionary algorithm, by its very nature, can handle low level noise, as is illustrated in Parts $(a)$ and $(b)$ of Figure 3.4. However, the algorithm fails to approximate a good solution in the presence of medium and high levels of noise. The solutions are not evenly distributed along the Pareto front and in some places we can see that dominated solutions are still present in the Pareto front. Performance metric values detailed in Table 3.4 also show the level of vulnerability of MOEA/D in a noisy environment.

- ZDT3: Figure 3.5 shows the true and estimated Pareto fronts for benchmark problem ZDT3. This problem has a discontinuous Pareto front. As can be clearly seen, even two percent of noise (low level noise) degrades the performance of MOEA/D. Table 3.5 shows the values of MOEA/D's performance metrics for this benchmark problem. They become far from desirable when noise levels increase. For example when 1% noise is present, GD returns 0.0369, a value relatively

Table 3.2: Performance metric values of estimated Pareto front by MOEA/D for noisy FON

| FON | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.0200 | 0.9938 | 0.0072 | 1.0745 |
| 2% Noise | 0.0520 | 0.9672 | 0.0115 | 1.1526 |
| 5% Noise | 0.1731 | 0.6445 | 0.0247 | 0.9985 |
| 10% Noise | 0.2429 | 0.5937 | 0.0359 | 1.0805 |
| 20% Noise | 0.2616 | 0.7515 | 0.0773 | 1.4873 |

close to 0. However, when noise levels increase to 20% the GD returns 0.6540, a significantly large value for this performance metric that shows how bad the situation is.

- ZDT4: Figure 3.6 shows the true and estimated Pareto fronts for benchmark problem ZDT4. This is a benchmark problem with a multimodal feature as discussed in Section 3.5.2. From Diagrams $(a)$ to $(e)$ it would appear that the performance of MOEA/D is quite satisfactory in the presence of only one percent of noise. However, the algorithm is challenged even by two percent noise, which is still considered low level noise. Table 3.6 includes the performance metric values of MOEA/D for noisy ZDT4. From the values of this table it can be seen that the performance of MOEA/D deteriorates when noise intensifies. For instance, HVR's value drops by 50% when noise levels increase to 20%.

- ZDT6: Figure 3.7 shows the behaviour of MOEA/D in the objective space for noisy benchmark problem ZDT6. The obtained result, with only one percent of noise, is near optimal, although the diversity and quality of solutions reduces as the noise level increases. Table 3.7 illustrates the performance metric values of MOEA/D for noisy ZDT6. It can be seen that MOEA/D's performance degenerates with respect to the performance metrics applied so far. For example the S metric value increases tenfold when noise levels reach 20% from 1%.

Table 3.3: Performance metric values of estimated Pareto front by MOEA/D for noisy KUR

| KUR | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.0326 | 0.9963 | 0.0828 | 1.0067 |
| 2% Noise | 0.0506 | 0.9957 | 0.0825 | 1.0149 |
| 5% Noise | 0.1146 | 0.9944 | 0.0791 | 1.0484 |
| 10% Noise | 0.2384 | 0.9898 | 0.1059 | 1.1072 |
| 20% Noise | 0.5098 | 0.9838 | 0.1630 | 1.2316 |

Table 3.4: Performance metric values of estimated Pareto front by MOEA/D for noisy ZDT1

| ZDT1 | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.0396 | 0.9963 | 0.0352 | 1.0317 |
| 2% Noise | 0.0596 | 0.9948 | 0.0424 | 1.0563 |
| 5% Noise | 0.2004 | 0.9052 | 0.0501 | 0.7948 |
| 10% Noise | 0.6336 | 0.7341 | 0.0655 | 0.2626 |
| 20% Noise | 1.1114 | 0.7081 | 0.0957 | 0.0799 |

Table 3.5: Performance metric values of estimated Pareto front by MOEA/D for noisy ZDT3

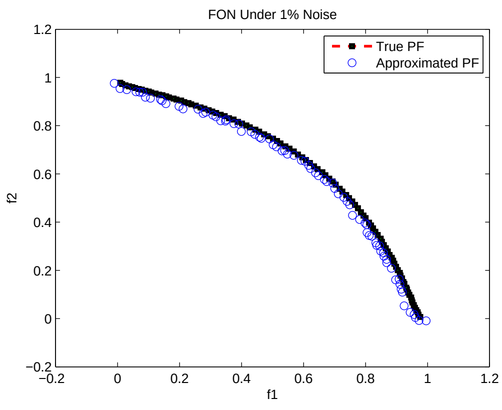
| ZDT3 | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.0396 | 0.9963 | 0.0352 | 1.0317 |
| 2% Noise | 0.0779 | 0.9711 | 0.0552 | 1.0547 |
| 5% Noise | 0.1942 | 0.9240 | 0.0711 | 0.9390 |
| 10% Noise | 0.5900 | 0.7745 | 0.0738 | 0.4446 |
| 20% Noise | 0.9540 | 0.7075 | 0.1114 | 0.2764 |

Table 3.6: Performance metric values of estimated Pareto front by MOEA/D for noisy ZDT4

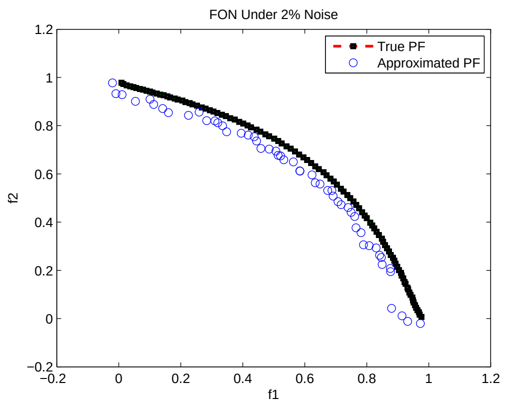
| ZDT4 | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.0396 | 0.9963 | 0.0352 | 1.0317 |
| 2% Noise | 1.6368 | 0.9828 | 1.5954 | 1.0053 |
| 5% Noise | 1.7076 | 0.8800 | 1.5809 | 0.9344 |
| 10% Noise | 2.9027 | 0.7153 | 2.5614 | 0.6479 |
| 20% Noise | 3.5134 | 0.6301 | 2.4304 | 0.5033 |

Table 3.7: Performance metric values of estimated Pareto front by MOEA/D for noisy ZDT6

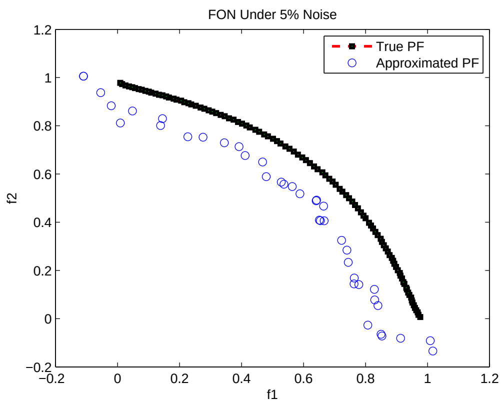
| ZDT6 | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.0396 | 0.9963 | 0.0352 | 1.0317 |
| 2% Noise | 0.1675 | 0.9998 | 0.1399 | 1.1227 |
| 5% Noise | 0.3286 | 0.9992 | 0.2014 | 1.3053 |
| 10% Noise | 0.5851 | 0.9988 | 0.2190 | 1.6211 |
| 20% Noise | 1.7030 | 1.2284 | 0.3245 | 1.9502 |



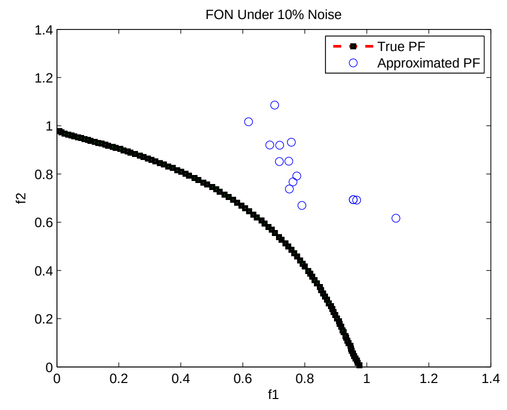
(a)



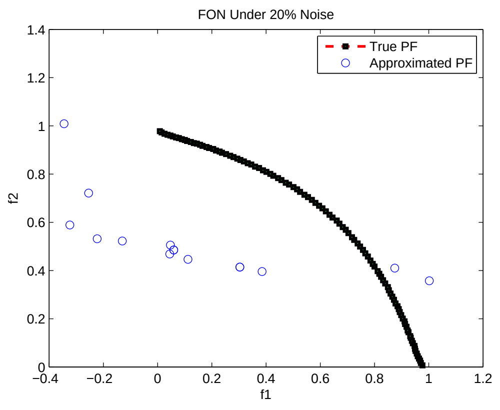
(b)



(c)

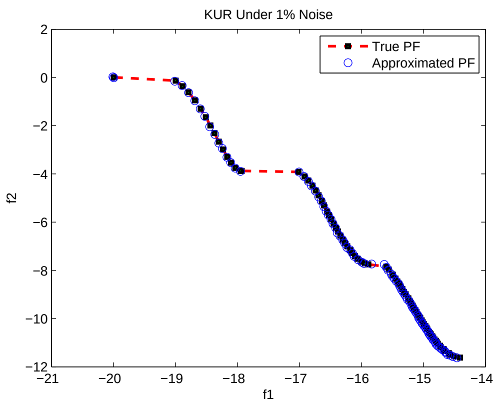


(d)

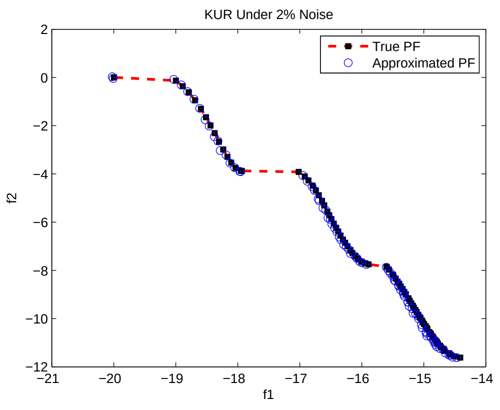


(e)

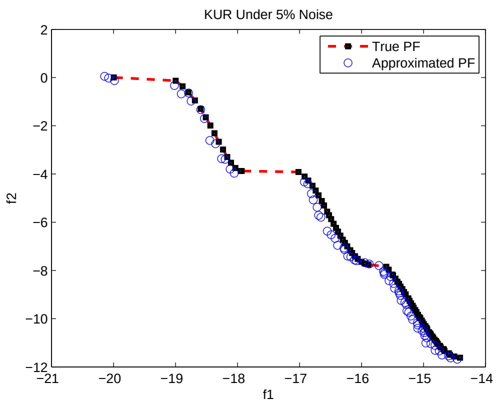
Figure 3.2: The evolved Pareto front of FON under the influence of noise levels (a)1%, (b)2%, (c)5%, (d)10% and (e)20% by MOEA/D.



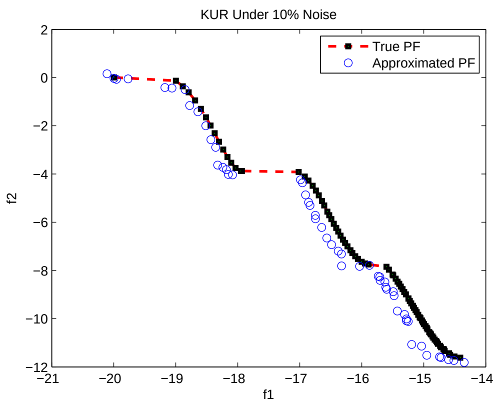
(a)



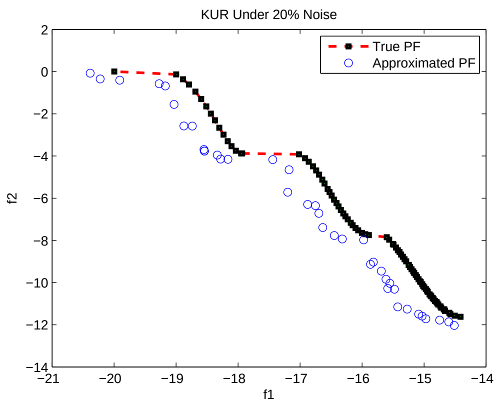
(b)



(c)

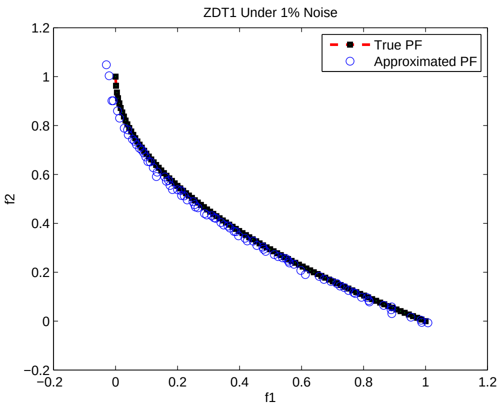


(d)

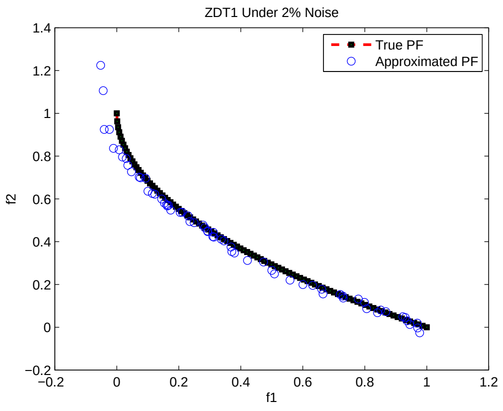


(e)

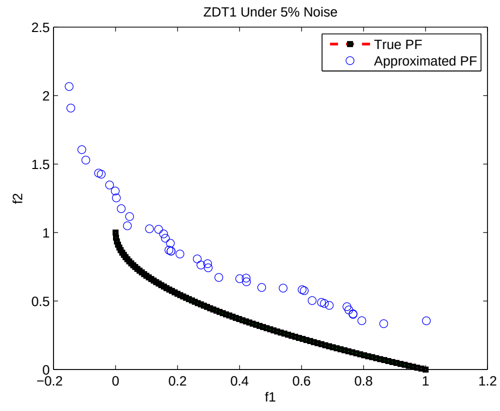
Figure 3.3: The evolved Pareto front of KUR under the influence of noise levels (a)1%, (b)2%, (c)5%, (d)10% and (e)20% by MOEA/D.



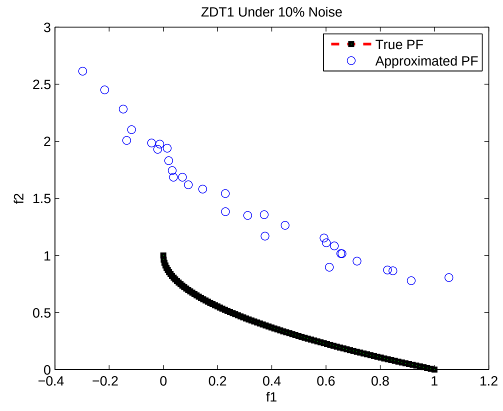
(a)



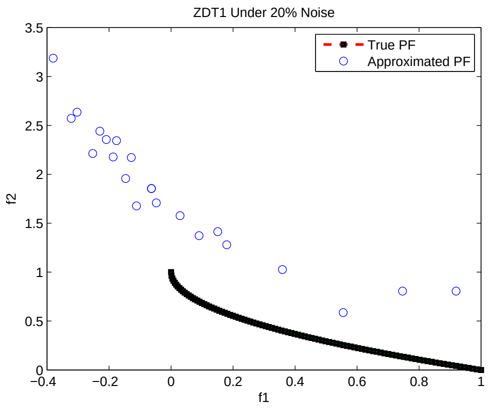
(b)



(c)

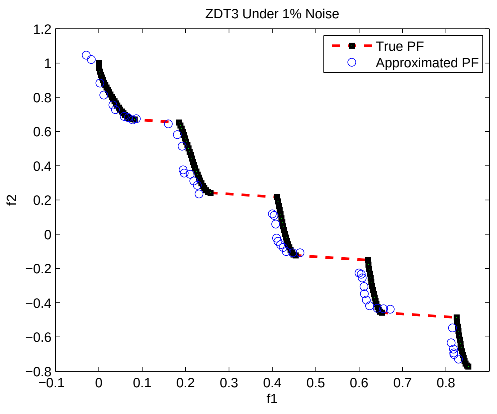


(d)

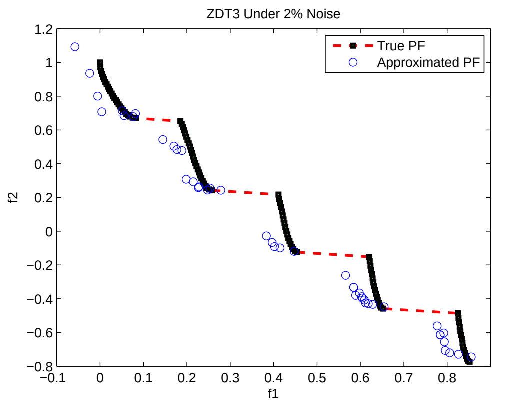


(e)

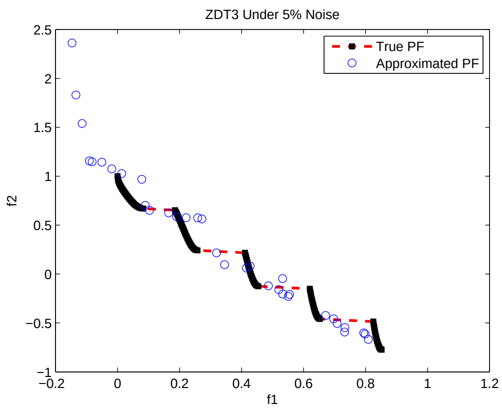
Figure 3.4: The evolved Pareto front of ZDT1 under the influence of noise levels (a)1%, (b)2%, (c)5%, (d)10% and (e)20% by MOEA/D.



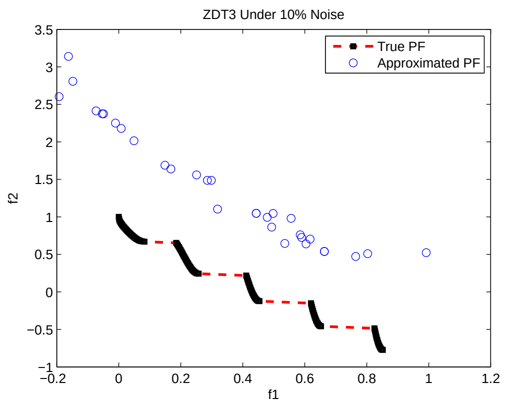
(a)



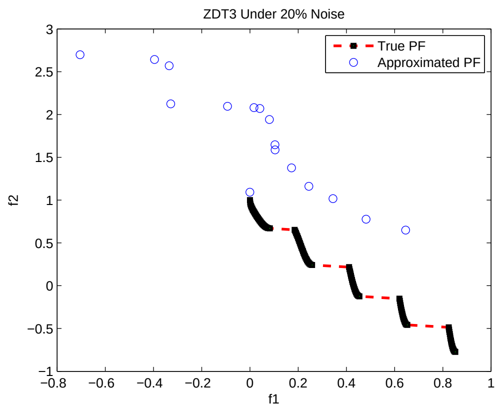
(b)



(c)

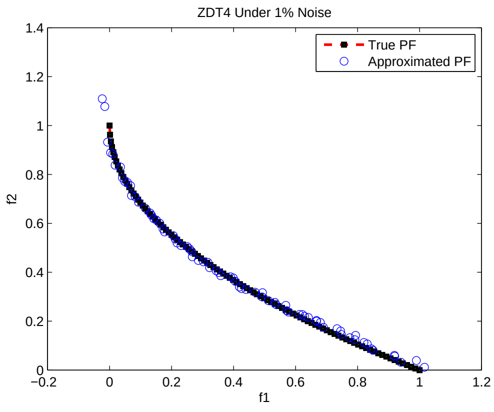


(d)

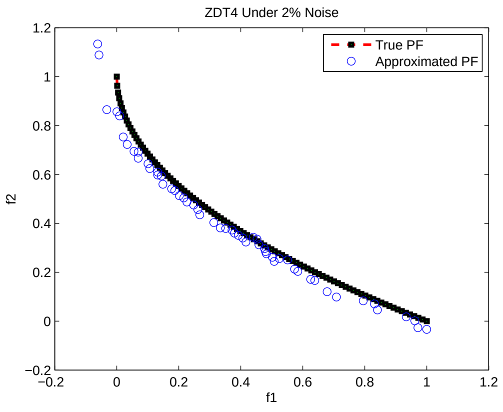


(e)

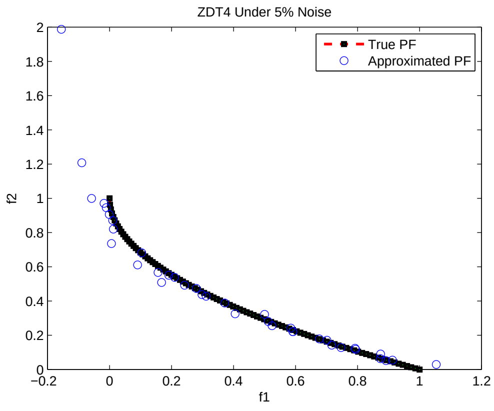
Figure 3.5: The evolved Pareto front of ZDT3 under the influence of noise levels (a)1%, (b)2%, (c)5%, (d)10% and (e)20% by MOEA/D.



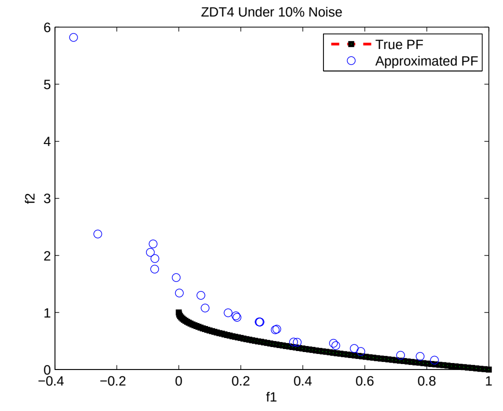
(a)



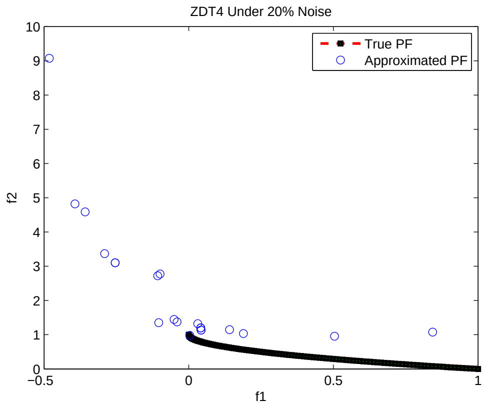
(b)



(c)

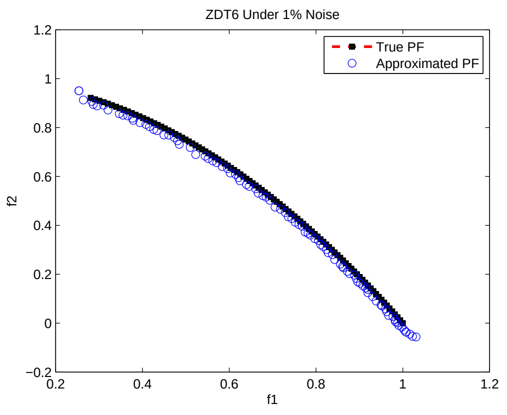


(d)

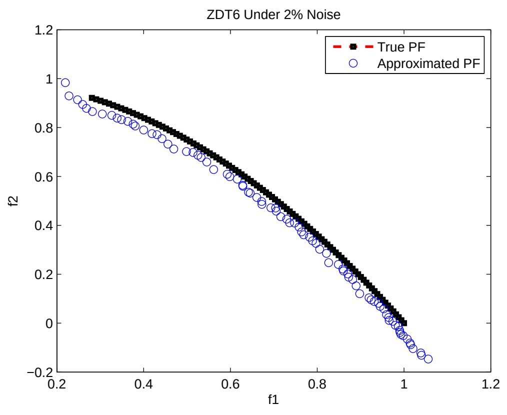


(e)

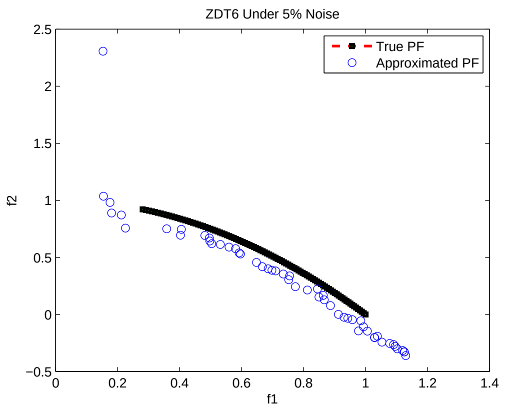
Figure 3.6: The evolved Pareto front of ZDT4 under the influence of noise levels (a)1%, (b)2%, (c)5%, (d)10% and (e)20% by MOEA/D.



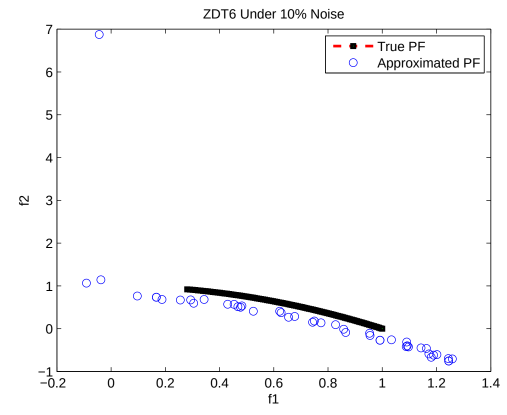
(a)



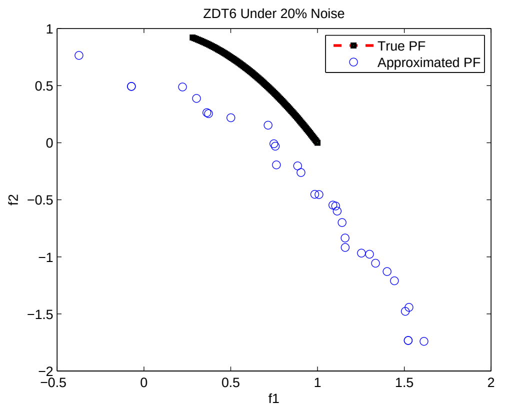
(b)



(c)

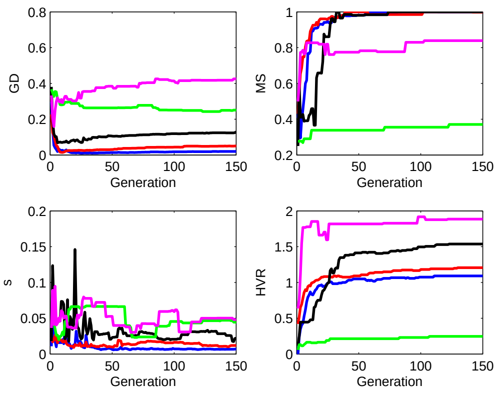


(d)

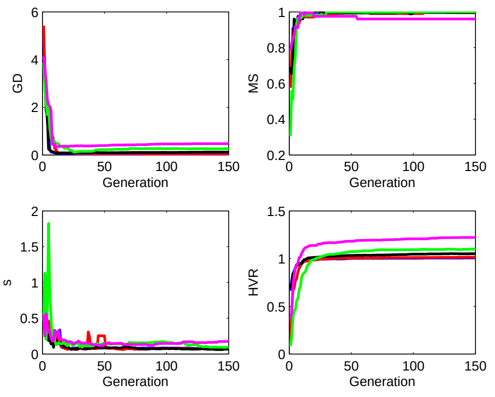


(e)

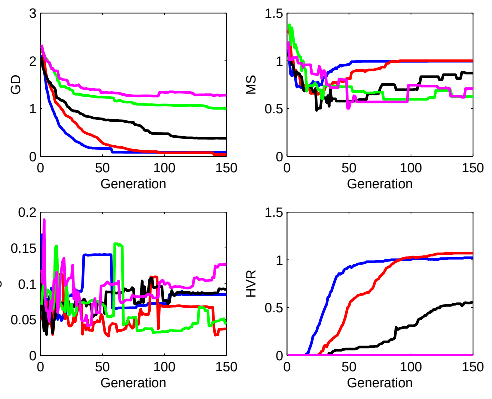
Figure 3.7: The evolved Pareto front of ZDT6 under the influence of noise levels (a)1%, (b)2%, (c)5%, (d)10% and (e)20% by MOEA/D.



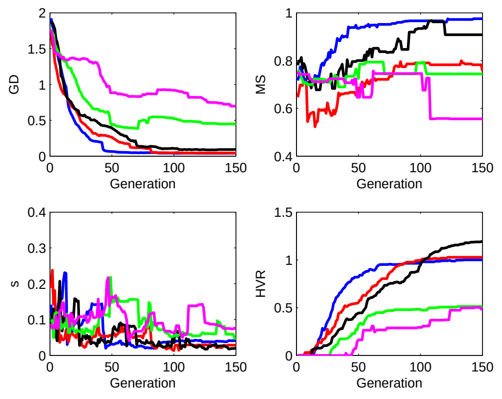
(a)



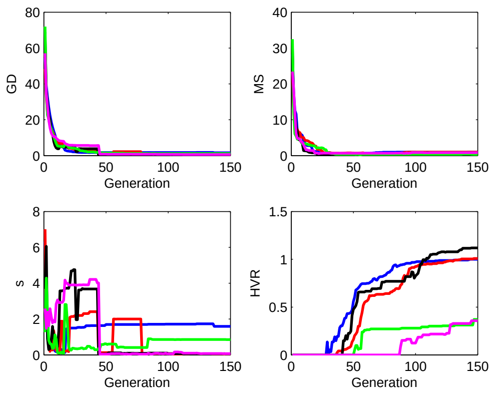
(b)



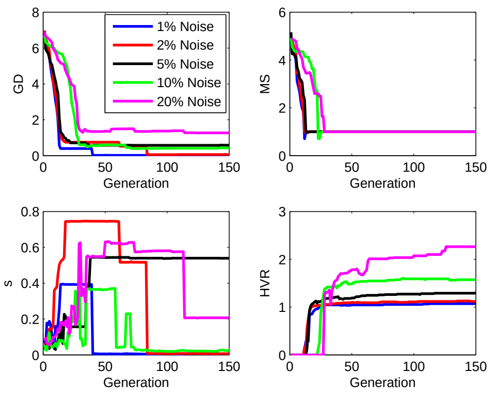
(c)



(d)



(e)



(f)

Figure 3.8: MOEA/D's trace of performance metrics for (a) FON, (b) KUR, (c) ZDT1, (d) ZDT3, (e) ZDT4, (f) ZDT6.

### 3.5.4 Discussion

From the results of Section 3.5.3, it is clear that the impact of noise on MOEA/D varies for each different benchmark problem, with their different features and difficulties.

As noise levels intensify, we can observe from Figures 3.2 to 3.7 that the range of solutions expands and the gap between true Pareto front and the estimated Pareto front becomes larger. In addition, the diversity of solutions drops badly as illustrated by the sharp reduction in the number of solutions found.

Similarly from Tables 3.2 to 3.7 it can be clearly seen that the different performance metrics obtain far from desirable values when noise intensities increase. As mentioned in Section 3.4, the goal was for the GD and S metrics to converge to 0 and the MS and HVR metrics to converge to 1. The results in these tables demonstrate that the performance of MOEA/D deteriorates for each of the tested benchmark problems when noise levels increase.

The impact of noise is observed to be severe on problems such as ZTD3, with its discontinued Pareto front, and ZDT4, with multimodality, although MOEA/D does evolve better solutions for some problems, such as ZTD1, in the presence of low level noise and the KUR benchmark problem.

Figure 3.8 plots the performance metrics for all the benchmark problems in the presence of different noise levels. Deterioration in the performance of MOEA/D is significant: for instance, the generational gap (GD) rockets up as noise intensifies. This trend is detectable in the other performance metrics as well. In other words, when noise intensifies the performance of MOEA/D significantly degenerates.

It is not clear why the S metric in some cases (such as Part (f) in Figure 3.8) behaves as random or why the HVR metric in Part (c) returns zero when noise levels reach 20%. Applying multiple performance metrics will guarantee that we do not lose any useful information, but we believe that added noise adversely influences the diversity of solutions in some problems more than in others. This can lead to the gap between solutions in approximated Pareto front. As the S metric calculates this gap, it therefore can be random if the algorithm is not able to keep the best solution during the search for the optimal.

## 3.6 Conclusions

This chapter opened with a description of noisy multi-objective optimization problems, outlined the challenges to evolutionary algorithms (EA) in handling such noisy problems and then analysed the decomposition based multi-objective optimization evolutionary algorithm (MOEA/D) in the presence of noise with different intensities.

Major contribution :

1- This is the first piece of research that studies the effect of noise on the performance of MOEA/D.

2- We have proved that the performance of MOEA/D deteriorates as noise levels intensify. [See Section 3.5.3 and section 3.5.4]

Minor contribution: The features of a problem must be taken into account. Problems with features such as multi-modality or discontinued Pareto fronts are faced with greater adversity. [See Section 3.5.4]

Significance: This is very important for the development of future algorithms: we now know that the standard MOEA/D must be modified to handle noise.

# 4 MOEA/D With Ordinal Optimization for Handling Noisy Problems

In the previous chapter we showed in detail the impact of noise on the performance of MOEA/D. The results of our experiments support the fact that MOEA/D deteriorates rapidly when noise intensities increase.

Ordinal optimization theory ensures that the order of solutions is likely to be preserved, even when using a crude model evaluation, in the presence of noise [3]. Thus, in order to ensure the selection of a set of good enough solutions, but with minimum computational cost, constructing a crude model is necessary.

In this chapter we will combine the MOEA/D framework with the ordinal optimization technique to handle the noisy multi-objective optimization problem.

## 4.1 Simulation Based Optimization

For computationally intensive objective functions the performance of system $F(x, w)$ can be measured via simulation [118], where $x$ is a vector of system parameters and $w$ represents either randomness or noise in the system. For a simulation-based optimization problem, an estimation of the expected system performance can be obtained by applying a Monto Carlo procedure as follows,

$$J(x) = E[F(x, w)] = \lim_{n\to\infty} (\frac{1}{n}) \sum_{i=1}^{n} F(x, w_i)$$

Limits can be approximated by

$$J(x) \equiv (\frac{1}{n}) \sum_{i=1}^{n} F(x, w_i) \tag{4.1}$$

When $n$ is large, this approximation is more accurate.

Thus, the algorithm optimizes as per Equation 4.1 instead of directly on $F(x, w)$, as the former is either noisy or computationally expensive. This problem, in our case a multi-objective optimization, can be modelled as follows,

$$\min_{x\in\Omega} J(x) \tag{4.2}$$

where $\Omega$ is the search space.

The simulation is conducted as a proxy for the actual system in an optimization process. Real world problems are too complex to be solved analytically, hence studying them via computer simulation [118]. Recent computer technology advances have moved simulation methods from a last resort to a primary technique for solving many real world problems.

Ordinal optimization is one technique among many that has evolved to cope with this sort of simulation-based evaluation problem [73]. Most of the early research focused on single objective optimization problems [3], but in the last decade the ordinal optimization method has been extended to multi-objective optimization problems and given the title 'vector ordinal optimization' [74], as described in Section 2.5.2.

**Algorithm 4.1** MOEA/D with OO

```
Input:
    • MOP (4.2);
    • A stopping criterion;
    • N: the number of subproblems considered in MOEA/D+OO.
    • A uniform spread of the weight vectors: $\lambda^1, \cdots, \lambda^N$;
    • T: the number of weight vectors in the neighbourhood of each weight vector.
Output:
    • EP : Good enough solutions found by algorithm;
Step 1) Initialization:
  Step 1.1) Set EP = $\emptyset$ .
  Step 1.2) Compute the Euclidean distances between any
   two weight vectors and then work out $T$ the closest weight
  vectors to each weight vector. For each $i = 1, ..., N$, set
  $B(i) = \{i_1, ..., i_T\}$, where $\lambda^{i_1}, ..., \lambda^{i_T}$ are the $T$ closest
  weight vectors to $\lambda^i$.
  Step 1.3) Generate initial population $x^1, ..., x^N$ at random.
  Step 1.4) Evaluate the population by crude model.
  Step 1.5) Initialize reference point $z = (z_1, ..., z_m)$ (see Section 2.4.4).
Step 2) Update:
  For $i = 1, \cdots, N$, do
  Step 2.1) Reproduction: Randomly select three solutions
  from $B(i)$ to generate a new solution $y$ by using DE and polynomial mutation.
  Step 2.2) Update of z: For each $j = 1, ..., m$, if
  $z_j < J_j(y)$, then set $z_j = J_j(y)$.
  Step 2.3) Update of Neighbouring Solutions: For each
  index $j \in B(i)$, if $g^{te}(y|\lambda^j, z) \leq g^{te}(x^j|\lambda^j, z)$, then set
  $x^j = y$ and $J(x^j) = J(y)$.
Step 3)Stopping Criteria:
  If stopping criteria is satisfied, then stop and go to 4.
  Otherwise, go to Step 2.
Step 4) Good Enough Set:
  Step 4.1) Select the first layer of solutions.
  Step 4.2) Evaluate the selected set with exact model and copy them in EP.
  Step 4.3) Return EP.
```

## 4.2 Combining MOEA/D with Ordinal Optimization

Ordinal optimization theory ensures that the order of solutions is likely preserved even through evaluation with a crude model in the presence of noise [3]. Thus for selecting a set of good enough solutions with minimum computational cost we need to construct a crude model to approximate Eq. 4.1. For this purpose, a rough model is constructed, based on a stochastic simulation with a small amount of test samples.

For a noisy multi-objective problem, similar to the problem in 4.2, we use the ordinal optimization technique along with the MOEA/D algorithm to handle noise. We call this new algorithm: Combined MOEA/D algorithm with OO technique (MOEA/D+OO).

In order to solve Equation (4.2), MOEA/D+OO involves three steps that are summarized as follows,

- First, construct a crude model to approximate the objective value for $E[f(x)]$ of a given $x$.
- Second, apply MOEA/D assisted by the crude model to solve (4.2) for a good enough subset of solutions $S$.
- Third, use the exact model to evaluate the objective value $E[f(x)]$ for each $x$ in $S$.

### 4.2.1 Crude model

Stochastic simulation is lengthy and computationally expensive. However, whilst not as accurate as a normal model, using a crude model based on stochastic simulation to approximate $E[f(x)]$ does reduce computational cost and complexity. We settled on 1000 as an appropriate number of test samples for this step.

### 4.2.2 MOEA/D with crude model

Using a crude model to evaluate the objective values, MOEA/D can efficiently search for $N$ good enough solutions for Problem 4.2. As the algorithm completes its last iteration, all final solutions are copied to $S$ as good enough solutions.

### 4.2.3 Exact model

Finally the selected solutions $S$ which are obtained by MOEA/D using the crude model are ready to be evaluated with a more refined model that we call Exact Model. An exact model can be constructed by a larger number of test samples. A sufficiently large sampling size is $10^6$ [119], but in this study we apply a model with a sampling size of $10^4$.

## 4.3 Experiment

The purpose of this experiment is to assess the performance of the new algorithm (MOEA/D+OO) in the presence of different levels of noise. We compare the performance of MOEA/D+OO and the original MOEA/D to see whether the modification has made the algorithm better at handling noise.

### 4.3.1 Design of Experiment

To begin with, for computational ease, a crude model is constructed based on stochastic simulation with a basic number of test samples, let us say 1000, whilst the number of simulations for the exact model is set at $10^4$.

The results depicted in Tables 4.1 to 4.6 were obtained after 50 runs. They show that the proposed algorithm does outperform the generic MOEA/D for each problem it was tested on and with different levels of noise, such as $\{1\%, 2\%, 5\%, 10\%, 20\%\}$. The tested problems are presented in Table 3.1 and Section 3.5.2.

For testing MOEA/D+OO we use the same parameter settings (neighbourhood and population size, maximum iteration and reproduction parameters) as mentioned in Section 3.5.1 for MOEA/D. We run our experiments in Matlab.

As indicated in Chapter 3, finding a proper and trustworthy method for handling noise is our primary concern for the noisy optimization problem rather than computational cost and time with these experiments. Neither was it a concern for the authors of [9].

Table 4.1: Performance metrics values of estimated Pareto front by MOEA/D+OO for noisy FON

| FON | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.0041 (0.0159) | 0.9979 (0.0041) | 0.0037 (0.0035) | 0.9848 (0.0897) |
| 2% Noise | 0.0045 (0.0475) | 0.9972 (0.0300) | 0.0043 (0.0072) | 0.9754 (0.1772) |
| 5% Noise | 0.0055 (0.1679) | 0.9915 (0.3470) | 0.0063 (0.0184) | **0.9661 (0.0324)** |
| 10% Noise | 0.0071 (0.2358) | 0.9881 (0.3944) | 0.0087 (0.2720) | 0.9531 (0.1274) |
| 20% Noise | 0.0102 (0.2514) | 0.9748 (0.2233) | 0.0119 (0.0654) | 0.9359 (0.5514) |

### 4.3.2 Results

Results are obtained from fifty independent runs. The average means of the collected data are represented in the following tables and graphs.

Tables 4.1 to 4.6 trace the values of MOEA/D+OO's different performance metrics on the benchmark problems outlined in Section 3.5.2. These tables contain four columns for the different performance metrics and five rows for the various noise levels. As mentioned in Section 3.4, a lower value of GD and S compels a better approximation of the Pareto front. For MS and HVR, a value closer to one is sensible.

The values in parentheses show the difference between the calculated values of the performance metrics for MOEA/D+OO and its basic version MOEA/D. As can be clearly seen, these values increase dramatically when noise levels intensify (shown top to bottom in each column). This indicates that MOEA/D+OO performs far better than MOEA/D in a noisy environment. Proof of this can be taken from how close to desirable are the performance metric values (shown in the tables) for MOEA/D+OO: GD and S are close to zero and MS and HVR are close to one.

The values shown in bold in Tables 4.1 to 4.6 indicate that MOEA/D performs better than MOEA/D+OO in those specific cases. There are a few instances, mostly in the presence of low noise levels, in which the performance of MOEA/D is better than that of MOEA/D+OO.

Figure 4.1 shows the returned values for performance metrics (a) GD, (b) MS, (c) S and (d) HVR for the benchmark problem FON by MOEA/D+OO under the influence of different noise levels. The desirable values for these four performance metrics are the same as those detailed in Section 3.4. As can be seen, the estimated values for GD and

Table 4.2: Performance metrics values of estimated Pareto front by MOEA/D+OO for noisy KUR

| KUR | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.0256 (0.0070) | 0.9967 (0.0004) | **0.0921 (0.0093)** | 1.0002 (0.0065) |
| 2% Noise | 0.0252 (0.0254) | 0.9967 (0.0010) | **0.0949 (0.8665)** | 0.9999 (0.0150) |
| 5% Noise | 0.0264 (0.6882) | 0.9963 (0.0019) | **0.0940 (0.0149)** | 0.9988 (0.0496) |
| 10% Noise | 0.0312 (0.2072) | 0.9955 (0.0057) | 0.0954 (0.0105) | 0.9975 (0.1097) |
| 20% Noise | 0.0404 (0.4694) | 0.9945 (0.0107) | 0.0959 (0.0671) | 0.9959 (0.2357) |

Table 4.3: Performance metrics values of estimated Pareto front by MOEA/D+OO for noisy ZDT1

| ZDT1 | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | 0.03112 (0.01202) | 0.99757 (0.00229) | 0.03056 (0.02099) | 0.99662 (0.03099) |
| 2% Noise | 0.03112 (0.02845) | 0.99757 (0.00274) | 0.03056 (0.01184) | 0.99662 (0.05968) |
| 5% Noise | 0.03416 (0.16623) | 0.99533 (0.09016) | 0.02870 (0.02138) | 0.98856 (0.19374) |
| 10% Noise | 0.07327 (0.56033) | 0.99129 (0.25719) | 0.04335 (0.02214) | 0.97448 (0.71185) |
| 20% Noise | 0.14747 (0.95346) | 0.98418 (0.27620) | 0.06011 (0.03595) | 0.93860 (0.85859) |

Table 4.4: Performance metrics values of estimated Pareto front by MOEA/D+OO for noisy ZDT3

| ZDT3 | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | **0.05602 (0.00227)** | **0.98389 (0.01614)** | **0.06642 (0.01157)** | 0.98661 (0.02569) |
| 2% Noise | 0.05602 (0.02185) | 0.98389 (0.01277) | **0.06642 (0.01127)** | 0.98661 (0.06805) |
| 5% Noise | 0.05366 (0.14058) | 0.98303 (0.05900) | 0.06113 (0.01000) | 0.97909 (0.04012) |
| 10% Noise | 0.09590 (0.49413) | 0.97811 (0.20363) | 0.07335 (0.00047) | 0.96164 (0.51703) |
| 20% Noise | 0.16674 (0.78728) | 0.96492 (0.25743) | 0.08411 (0.02729) | 0.92042 (0.64400) |

Table 4.5: Performance metrics values of estimated Pareto front by MOEA/D+OO for noisy ZDT4

| ZDT4 | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | **1.40825 (0.60464)** | **0.99570 (0.00384)** | **1.40726 (0.58915)** | 0.98913 (0.00535) |
| 2% Noise | 1.40825 (0.22853) | 0.99570 (0.01287) | 1.40726 (0.18809) | 0.98913 (0.01621) |
| 5% Noise | 0.97472 (0.73288) | 0.98906 (0.10908) | 0.95770 (0.62321) | 0.97111 (0.03671) |
| 10% Noise | 0.38387 (2.51878) | 0.98634 (0.27099) | 0.35458 (2.20685) | 0.95037 (0.30247) |
| 20% Noise | 0.52865 (2.98473) | 0.97086 (0.34076) | 0.46897 (1.96143) | 0.90115 (0.39786) |

Table 4.6: Performance metrics values of estimated Pareto front by MOEA/D+OO for noisy ZDT6

| ZDT6 | GD | MS | S | HVR |
|---|---|---|---|---|
| 1% Noise | **0.10887 (0.07971)** | 0.99955 (0.00031) | **0.10947 (0.08939)** | 0.99527 (0.05907) |
| 2% Noise | 0.10887 (0.05865) | **0.99955 (0.00021)** | 0.10947 (0.03044) | 0.99527 (0.12748) |
| 5% Noise | 0.07787 (0.25072) | **0.99882 (0.00042)** | 0.07951 (0.12190) | 0.99218 (0.31309) |
| 10% Noise | 0.02911 (0.55601) | **0.99534 (0.00350)** | 0.02828 (0.19073) | 0.99637 (0.62472) |
| 20% Noise | 0.10579 (1.59723) | 0.98465 (0.24375) | 0.05335 (0.27111) | 1.07163 (0.87855) |

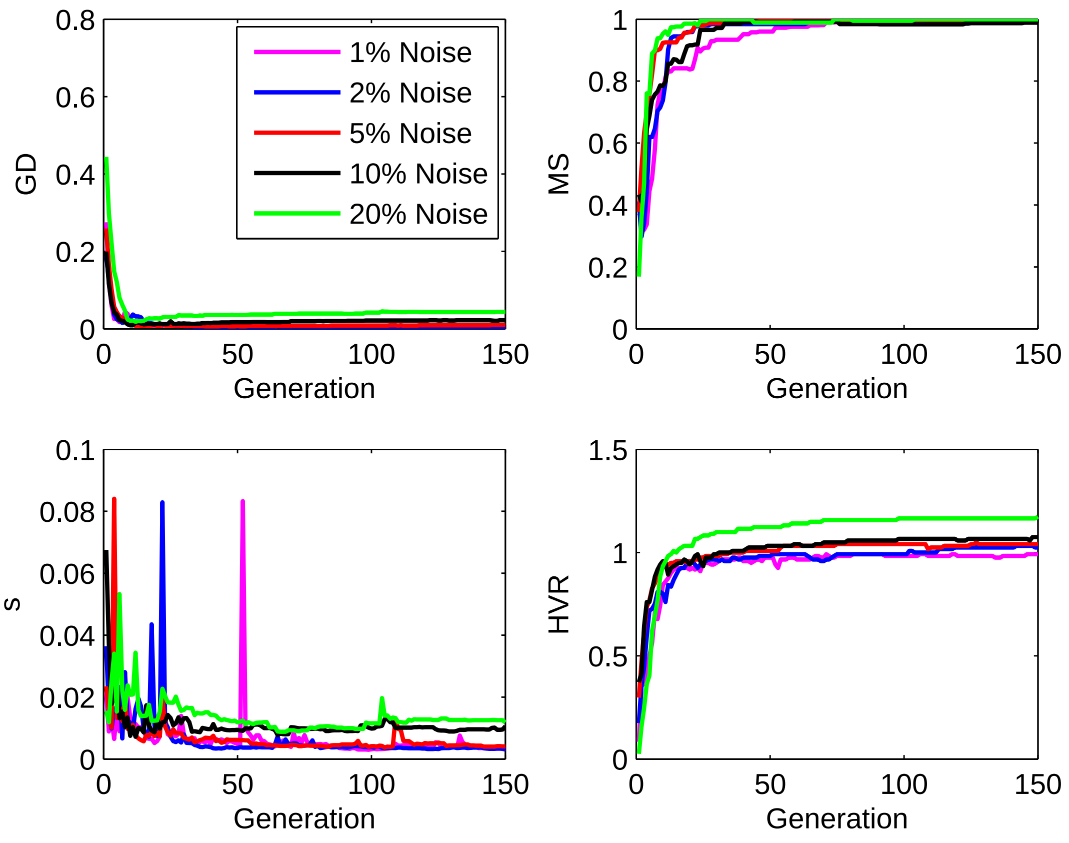


Figure 4.1: Performance Metrics of MOEA/D+OO in presence of different noise levels for noisy FON problem.

S are very close to 0 and the values for MS and HVR are converging to 1 in the presence of different noise levels on this FON benchmark problem.

Figure 4.2 illustrates the performance metrics (a) GD, (b) MS, (c) S and (d) HVR attained by MOEA/D+OO on the KUR benchmark problem under the influence of different levels of noise. As can been seen, the performance metrics return sensible values for all four metrics in the presence of both low and high levels of noise.

Figure 4.3 shows the performance metrics (a) GD, (b) MS, (c) S and (d) HVR attained by MOEA/D+OO for ZDT1 under the influence of different noise levels over 150 generations. According to this graph, MOEA/D+OO estimated the Pareto front extremely well for low, medium and high level noises - all except 20%. The spacing metric (S) and generational distance (GD) show a bit of violation under the influence

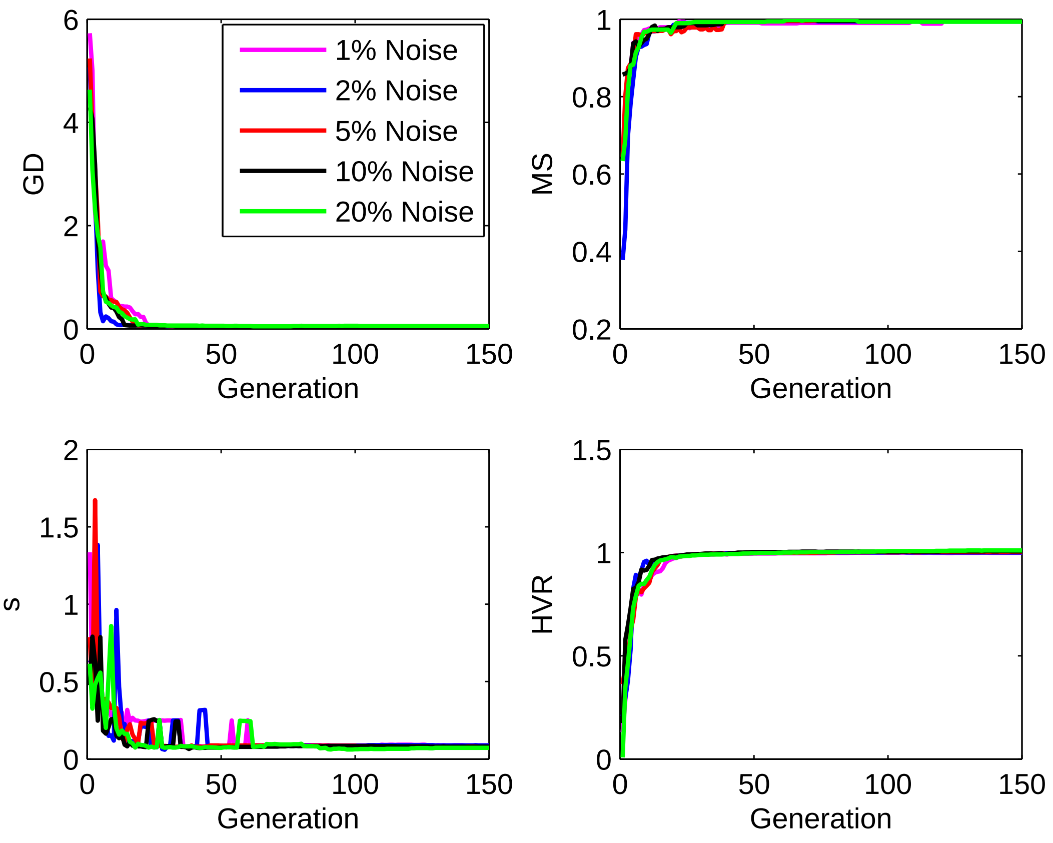


Figure 4.2: Performance Metrics of MOEA/D+OO in presence of different noise levels for noisy KUR problem.

of 20% noise, which can be caused by bad distribution of solutions and a meaningful gap between the estimated Pareto front and true Pareto front.

Figure 4.4 details the values for MOEA/D+OO's performance metrics on the ZDT3 problem, influenced by different levels of noise over 150 generations. According to this diagram, the algorithm had difficulty in maintaining a diverse and evenly spread solution set. The challenge to the algorithm on this problem was its disconnected Pareto front.

Figure 4.5 shows how MOEA/D+OO deals with the multimodal noisy ZDT4 problem. According to this graph, the spacing performance metric (S) gets disturbed when noise increases to twenty percent. This in turn impacts the diversity of the solution, but the other performance metrics do indicate that the modified algorithm (MOEA/D+OO) achieves a very close approximation of the true Pareto front for noisy ZDT4. The rea-

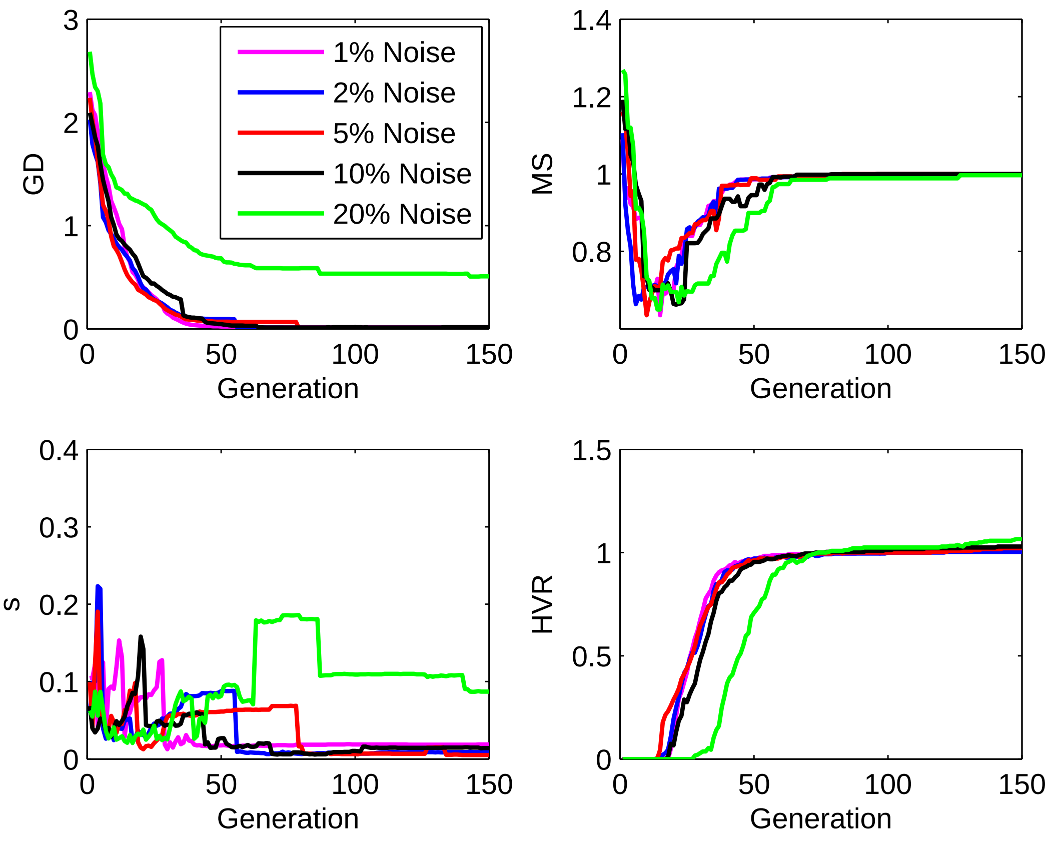


Figure 4.3: Performance Metrics of MOEA/D+OO in presence of different noise levels for noisy ZDT1 problem.

sons are not clear as to why the performance metric S returns unfavourable values. This may be caused by the multi-modality feature of this problem.

Figure 4.6 illustrates the performance metrics of MOEA/D+OO on the noisy ZDT6 problem. For this problem, as with ZDT4, the diversity of the solution is impacted by high levels of noise with regard to the spacing metrics. Other metrics, however, such as GD, serve to underline the rightness of the estimated Pareto front. Similar to the ZDT4 problem, the performance metric S remains unfavourable for unknown reasons.

It is not clear why the S metric seems random in only a few cases (ZDT4 and ZDT6), but we believe that added noise adversely influences the diversity of solutions in some problems more than others. This can lead to the gap between solutions in the approximated Pareto front. As the S metric calculates this gap, it can therefore

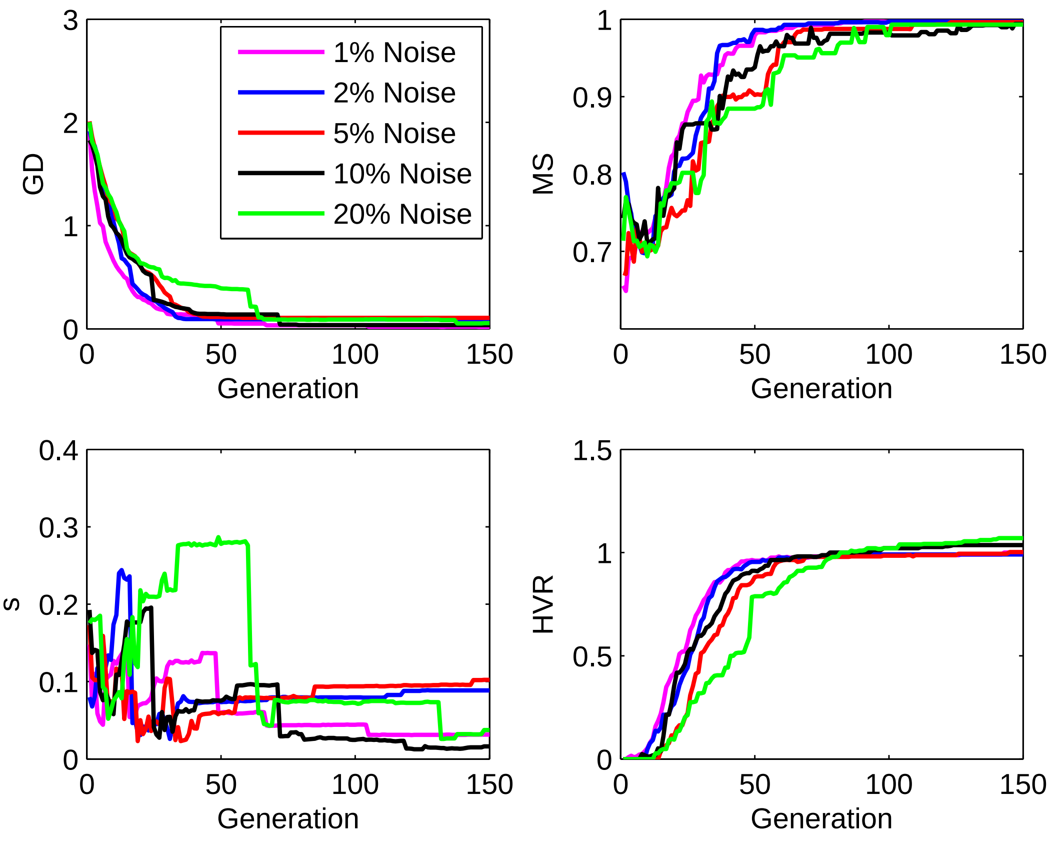


Figure 4.4: Performance Metrics of MOEA/D+OO in presence of different noise levels for noisy ZDT3 problem.

be random if the algorithm is not able to keep the best solution during the search for optimality.

True Pareto front is the best recent estimation of each problem, therefore we plot our results using the true Pareto front to represent how well the algorithms work. Figure 4.7 shows both the estimated and the true Pareto fronts for MOEA/D+OO on the noisy FON problem. The algorithm shows its ability to handle noise but, in the presence of twenty percent noise, there are some missing solutions that would indicate the impact of high noise on the diversity of solutions.

Figure 4.8 draws both the estimated and true Pareto fronts for noisy KUR by MOEA/D+OO. The algorithm shows its ability to handle noise. As can be seen from this graph, the algorithm maintains a good diversity and precision between the esti-

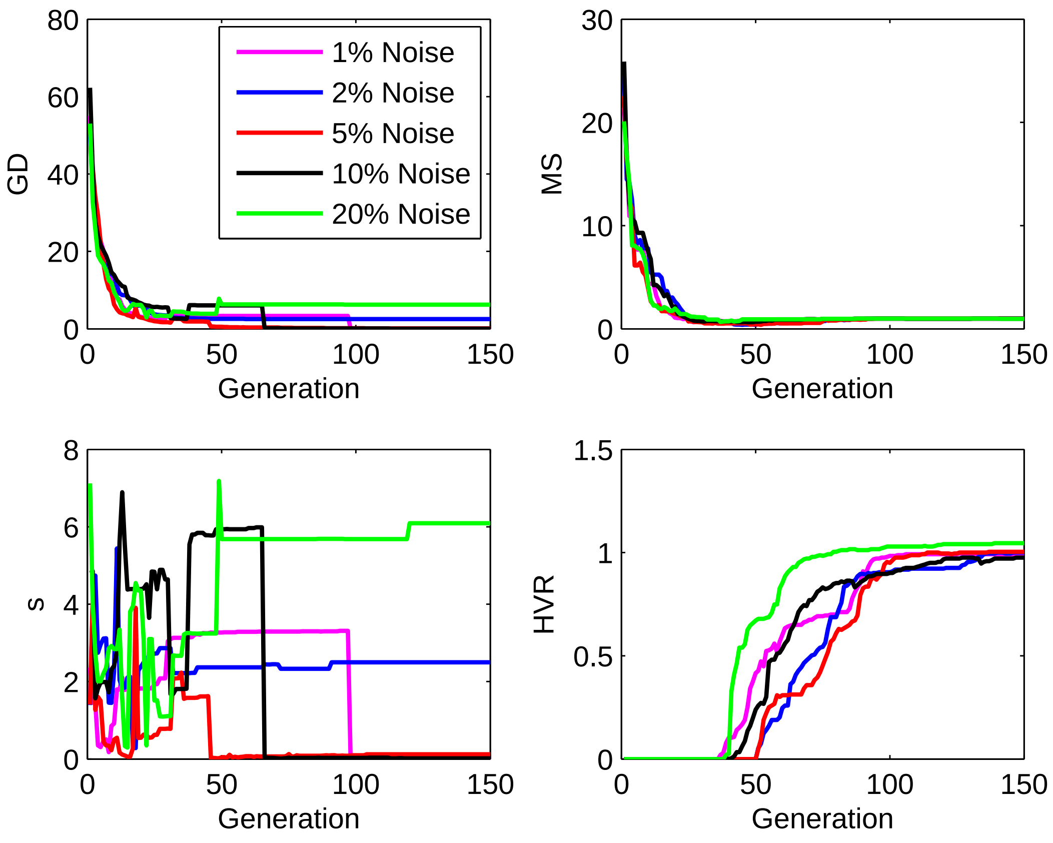


Figure 4.5: Performance Metrics of MOEA/D+OO in presence of different noise levels for noisy ZDT4 problem.

mated the true Pareto fronts in all scenarios.

Figure 4.9 draws both the estimated and true Pareto fronts for noisy ZDT1 by MOEA/D+OO. The algorithm shows its ability to handle noise but, in the presence of twenty percent noise, there are some missing solutions that would indicate the impact of high noise on the diversity of solutions.

Figure 4.10 depicts the estimated and true Pareto fronts of noisy problem ZDT3 by MOEA/D+OO. This problem has a disconnected Pareto front on which the new algorithm achieves a better performance than its basic version that failed to cover some parts of the Pareto front - see Section 3.5.3. As can be seen, even in the presence of 20% noise, the new algorithm finds solutions in all the disconnected parts of the Pareto front.

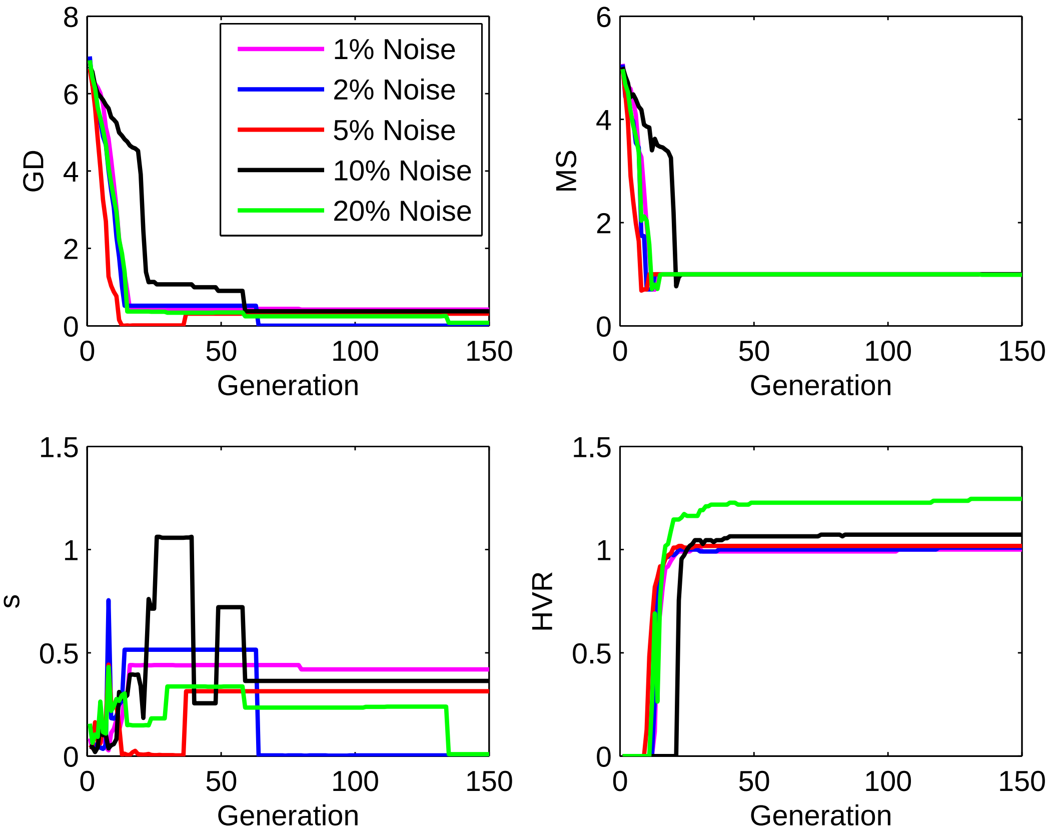
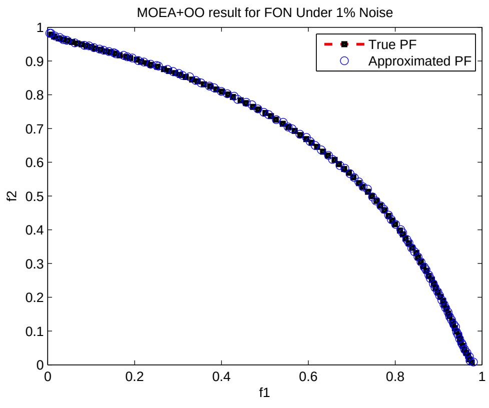


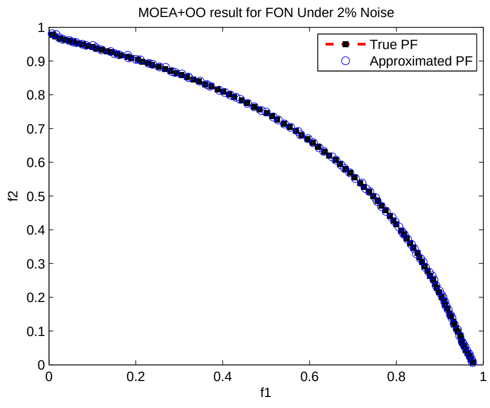
Figure 4.6: Performance Metrics of MOEA/D+OO in presence of different noise levels for noisy ZDT6 problem.

Figure 4.11 shows how well MOEA/D+OO approximates the Pareto front by comparison to the true Pareto front on the noisy ZDT4 problem. As mentioned in Section 3.5.2, the ZDT4 problem is a multimodal one. The algorithm estimates the Pareto front quite satisfactorily, although some solutions are still missing, which in turn impacts the diversity of solutions at high levels of noise. In the presence of noise, finding near optimal solutions is a big challenge for the MOEA algorithm. In contrast, our modified algorithm is quite adept at tackling this challenge.

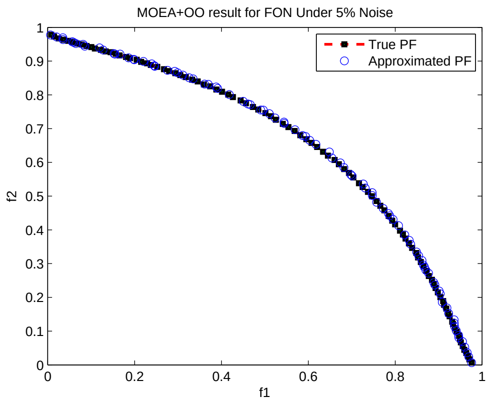
Figure 4.12 represents the approximated Pareto front of noisy ZDT6 by MOEA/D+OO. The performance of the algorithm is almost that of the true Pareto front, apart from when the noise gets to twenty percent. However its performance is still satisfactory, even at that level.



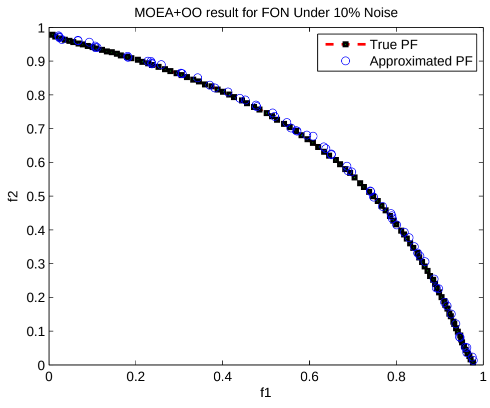
(a)



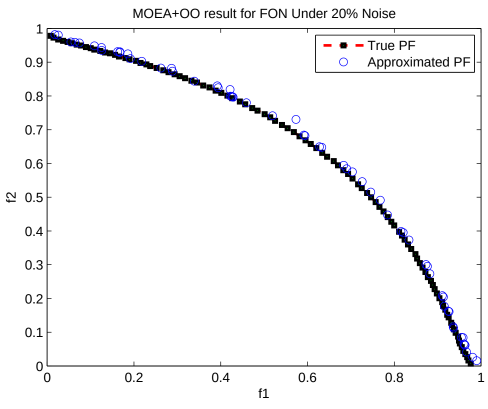
(b)



(c)

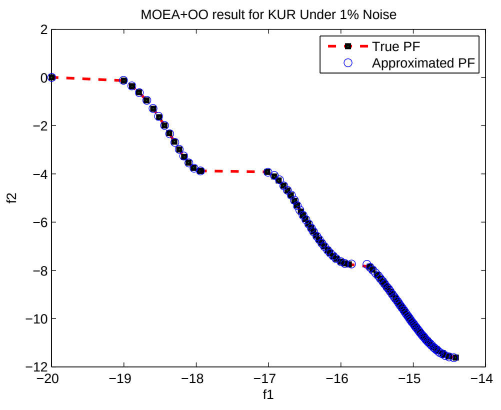


(d)

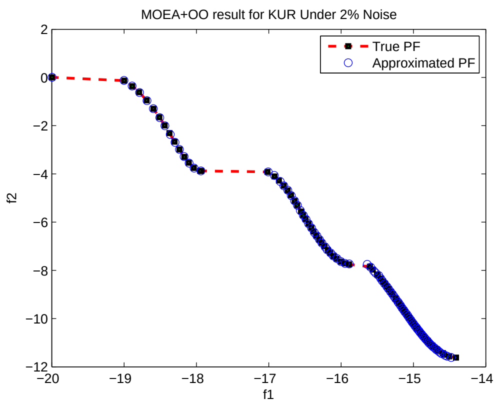


(e)

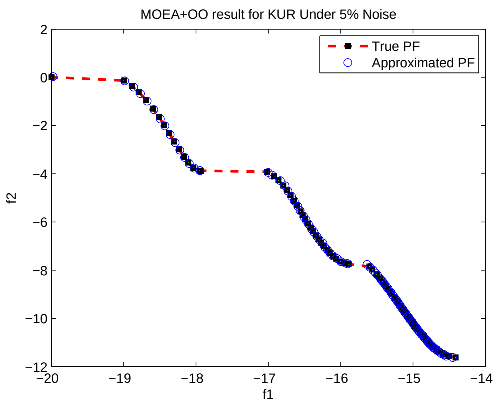
Figure 4.7: Pareto Front of noisy FON under the influence of noise level at (a)1%, (b)2%, (c)5%, (d)10%, (e)20% by MOEA/D+OO.



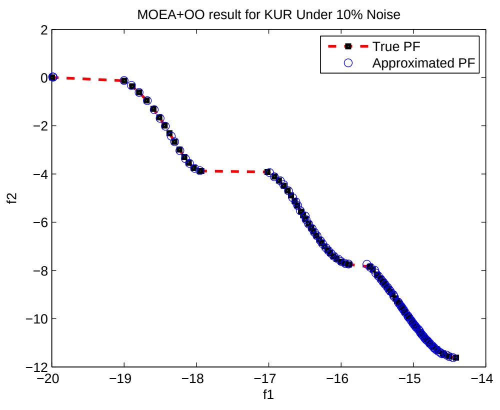
(a)



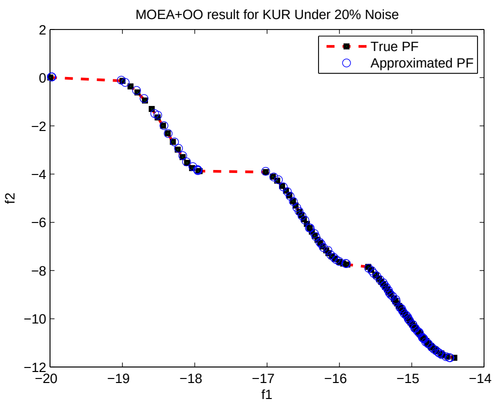
(b)



(c)

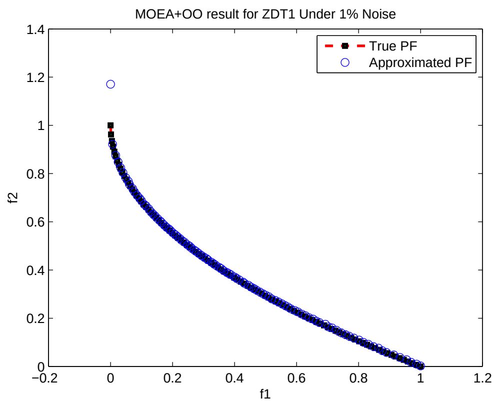


(d)

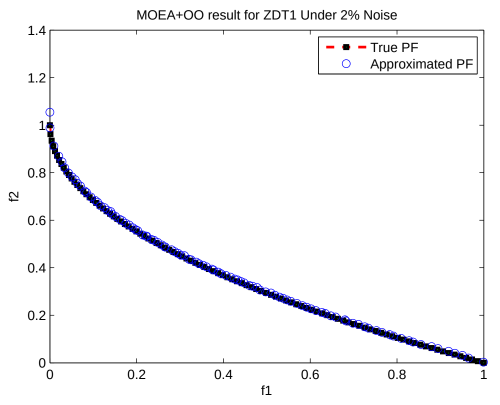


(e)

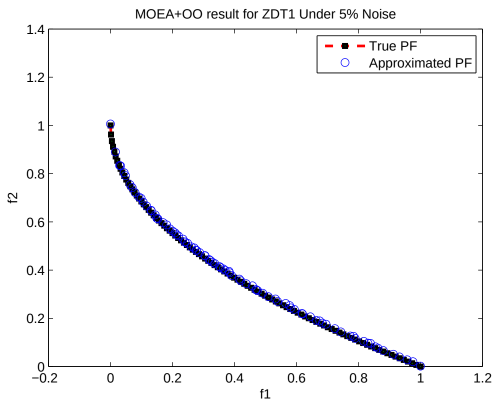
Figure 4.8: Pareto Front of noisy KUR under the influence of noise level at (a)1%, (b)2%, (c)5%, (d)10%, (e)20% by MOEA/D+OO.



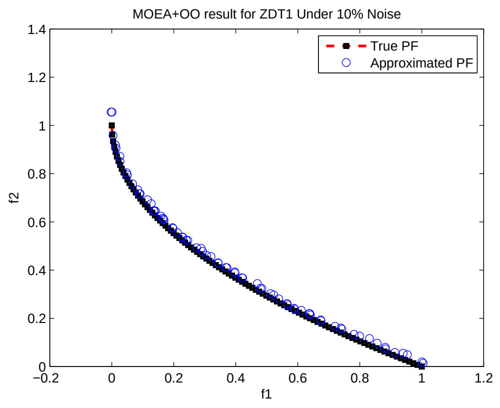
(a)



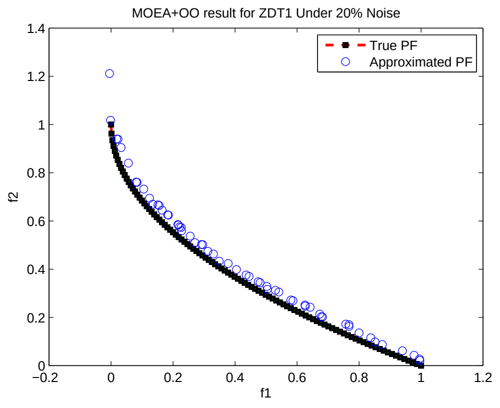
(b)



(c)

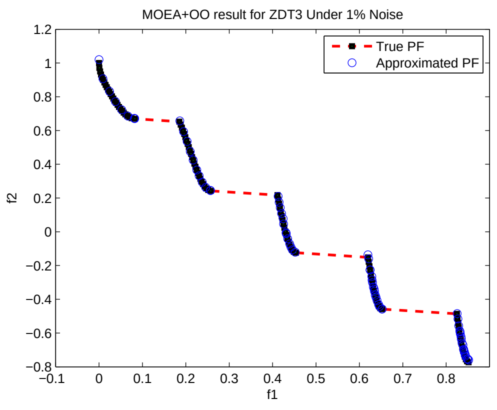


(d)

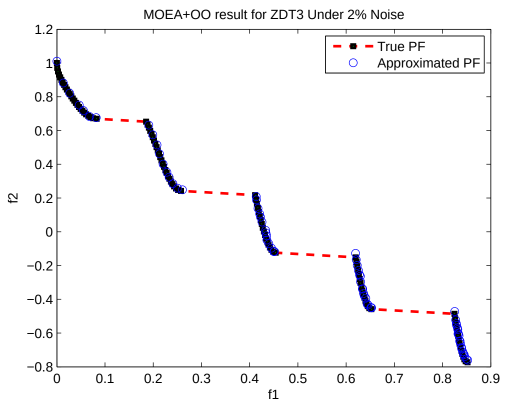


(e)

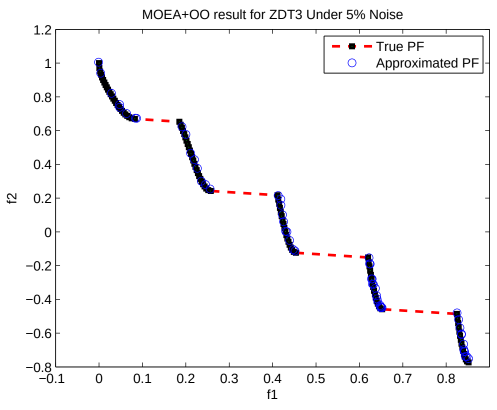
Figure 4.9: Pareto Front of noisy ZDT1 under the influence of noise level at (a)1%, (b)2%, (c)5%, (d)10%, (e)20% by MOEA/D+OO.



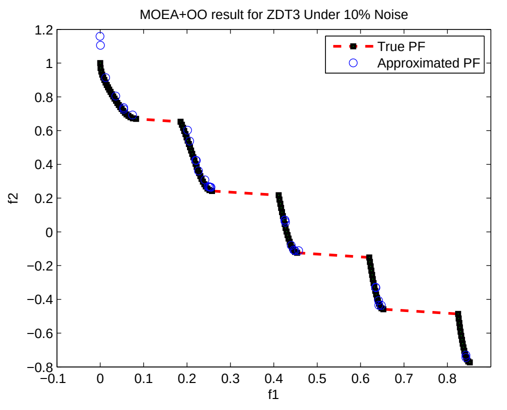
(a)



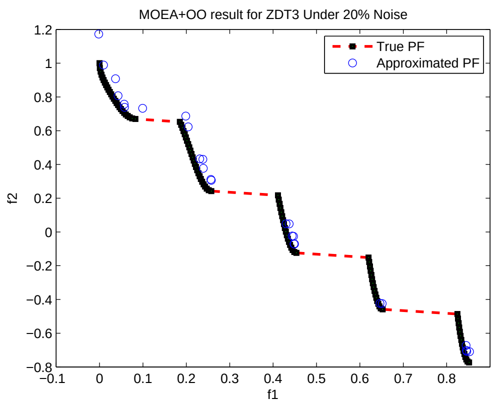
(b)



(c)



(d)



(e)

Figure 4.10: Pareto Front of noisy ZDT3 under the influence of noise level at (a)1%, (b)2%, (c)5%, (d)10%, (e)20% by MOEA/D+OO.

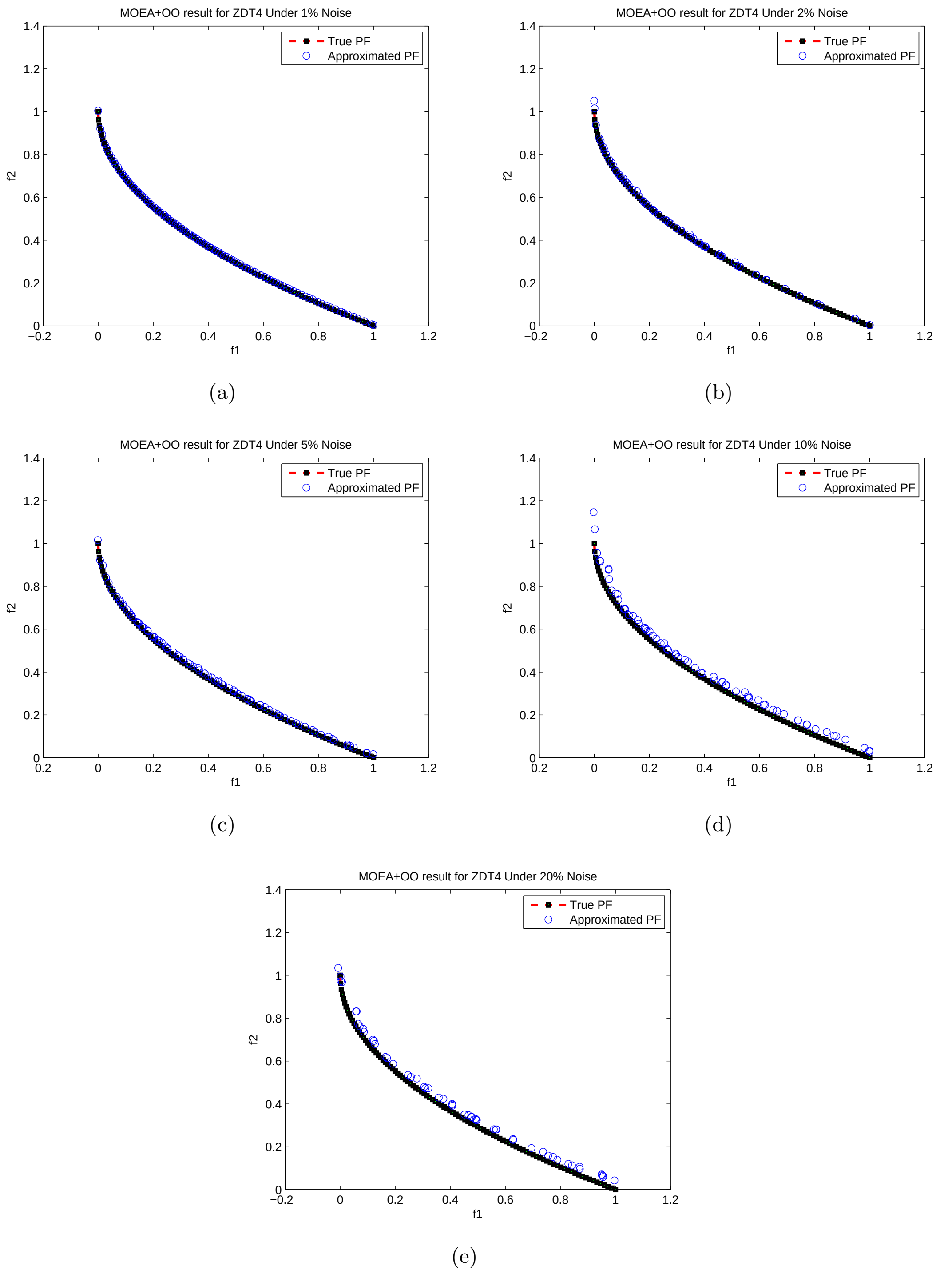
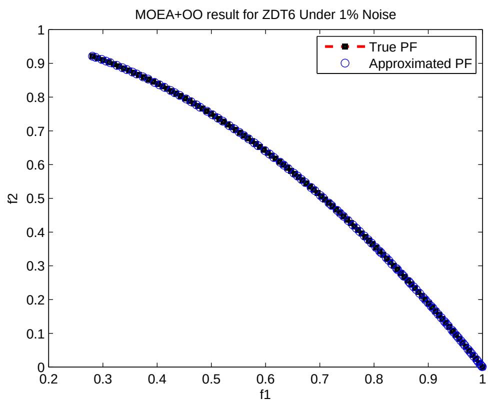
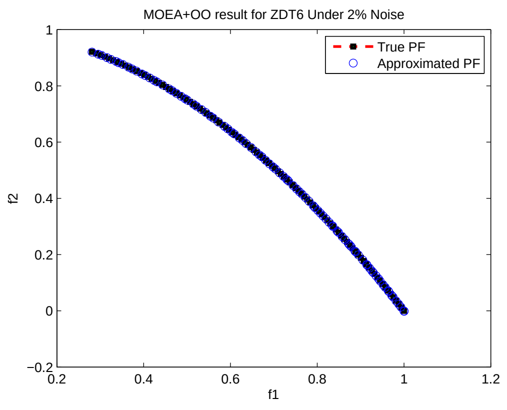


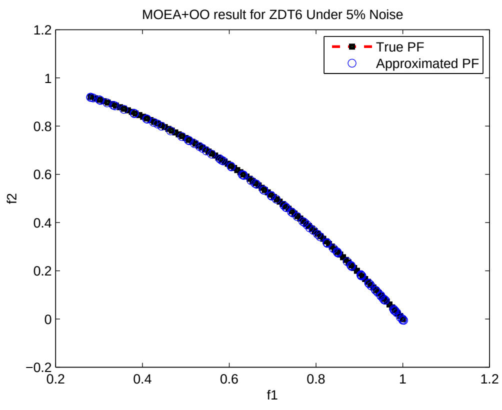
Figure 4.11: Pareto Front of noisy ZDT4 under the influence of noise level at (a)1%, (b)2%, (c)5%, (d)10%, (e)20% by MOEA/D+OO.



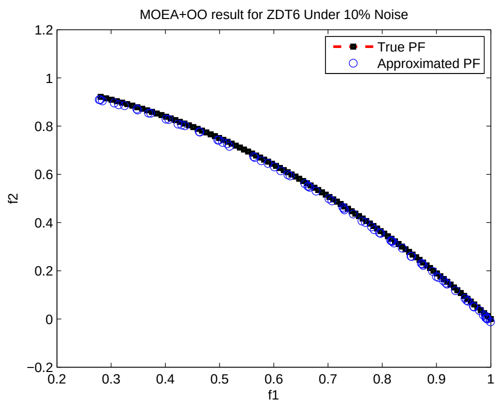
(a)



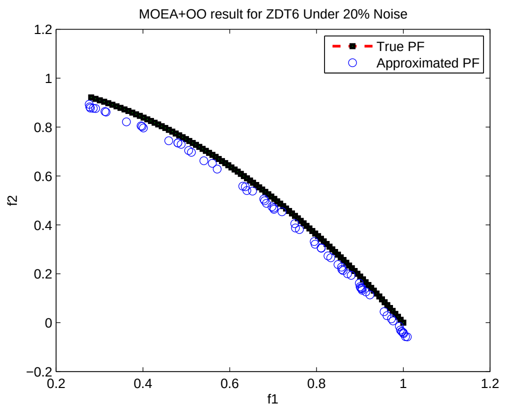
(b)



(c)

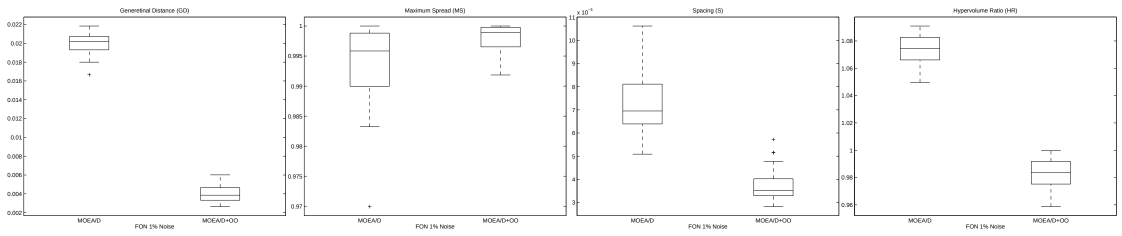


(d)



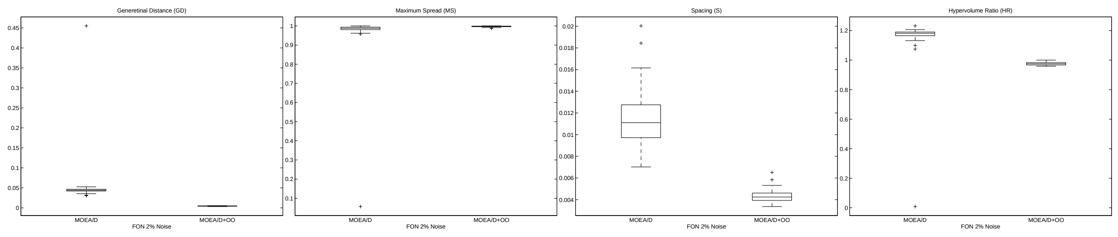
(e)

Figure 4.12: Pareto Front of noisy ZDT6 under the influence of noise level at (a)1%, (b)2%, (c)5%, (d)10%, (e)20% by MOEA/D+OO.



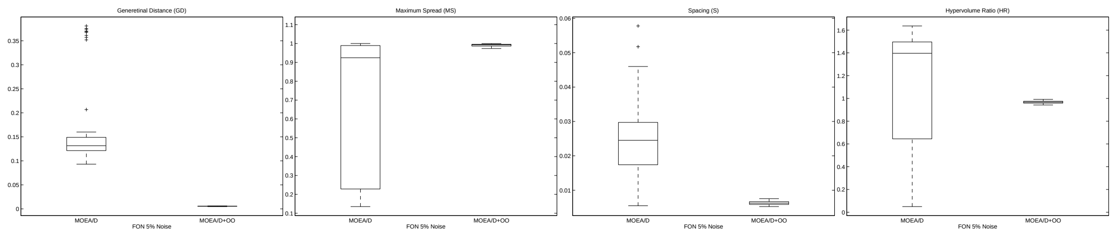
(a) (b) (c) (d)

Figure 4.13: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for FON with 1% noise



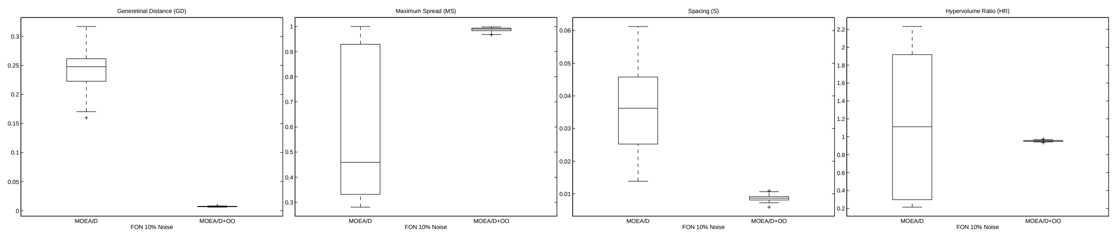
(a) (b) (c) (d)

Figure 4.14: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for FON with 2% noise



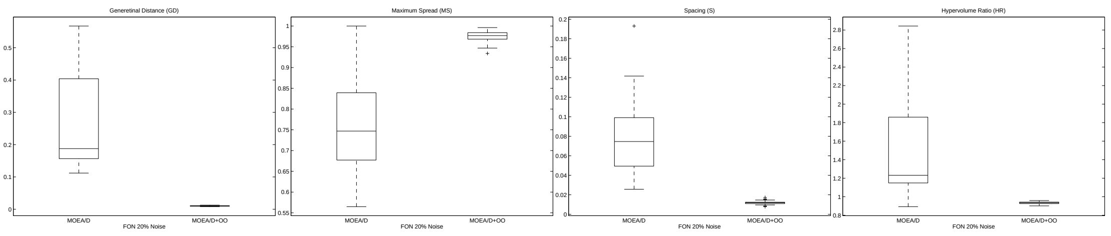
(a) (b) (c) (d)

Figure 4.15: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for FON with 5% noise



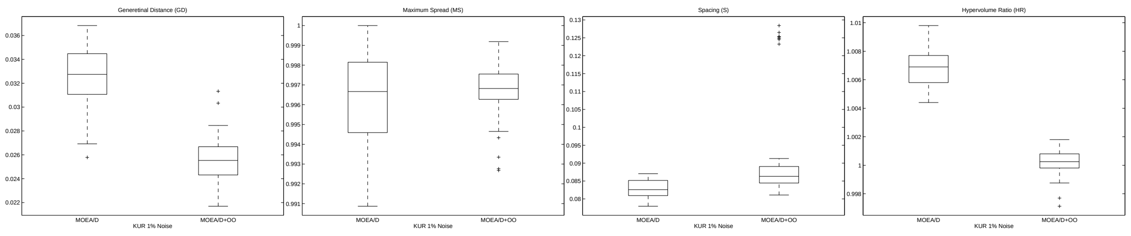
(a) (b) (c) (d)

Figure 4.16: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for FON with10% noise



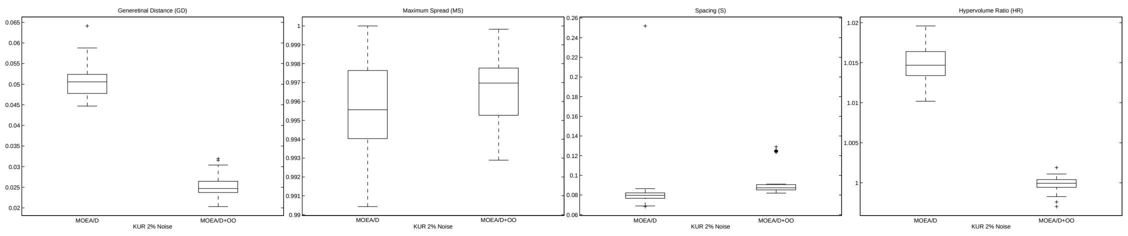
(a) (b) (c) (d)

Figure 4.17: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for FON with 20% noise



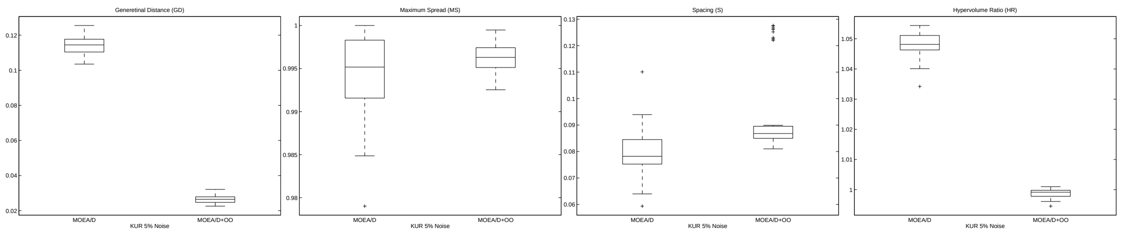
(a) (b) (c) (d)

Figure 4.18: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for KUR with 1% noise



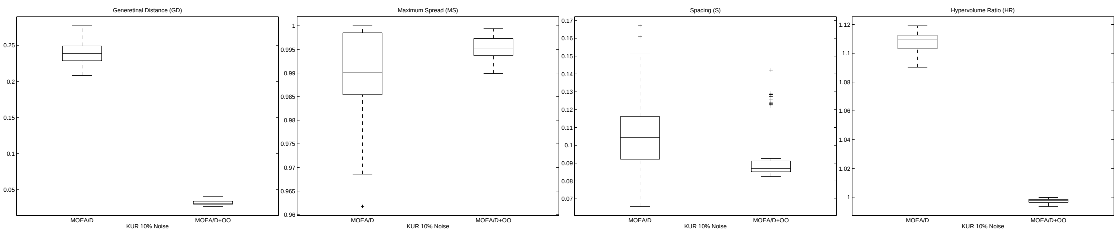
(a) (b) (c) (d)

Figure 4.19: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for KUR with 2% noise



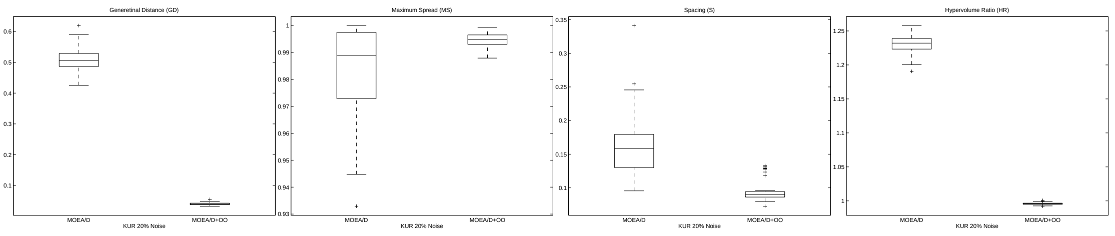
(a) (b) (c) (d)

Figure 4.20: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for KUR with 5% noise



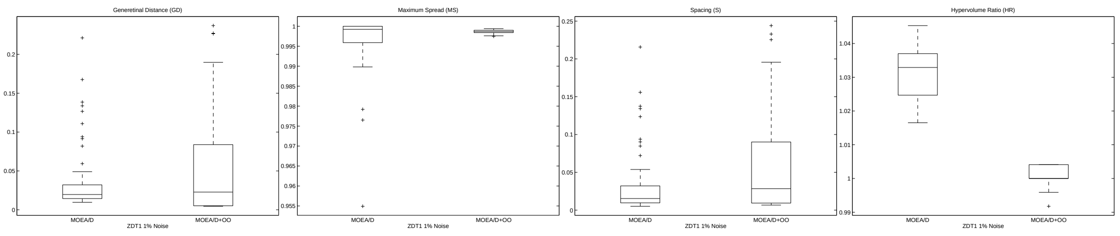
(a) (b) (c) (d)

Figure 4.21: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for KUR with10% noise



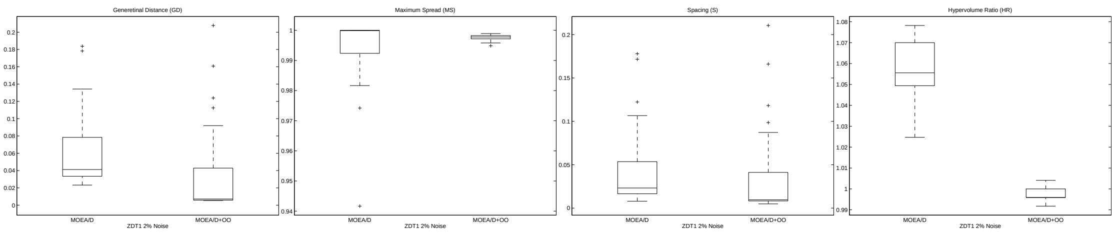
(a) (b) (c) (d)

Figure 4.22: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for KUR with 20% noise



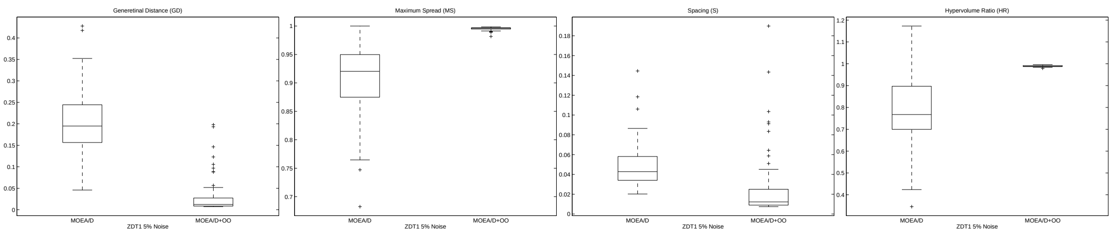
(a) (b) (c) (d)

Figure 4.23: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT1 with 1% noise



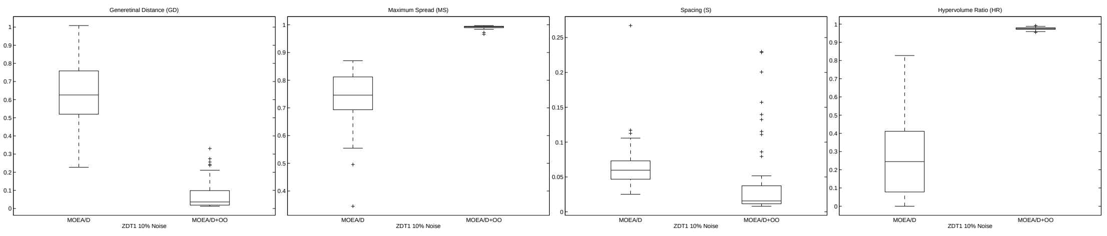
(a) (b) (c) (d)

Figure 4.24: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT1 with 2% noise



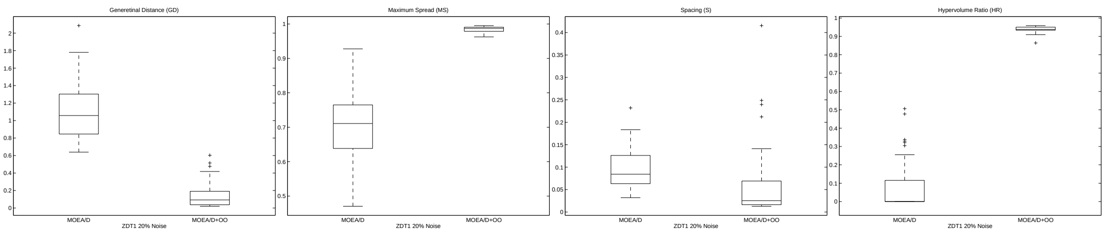
(a) (b) (c) (d)

Figure 4.25: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT1 with 5% noise



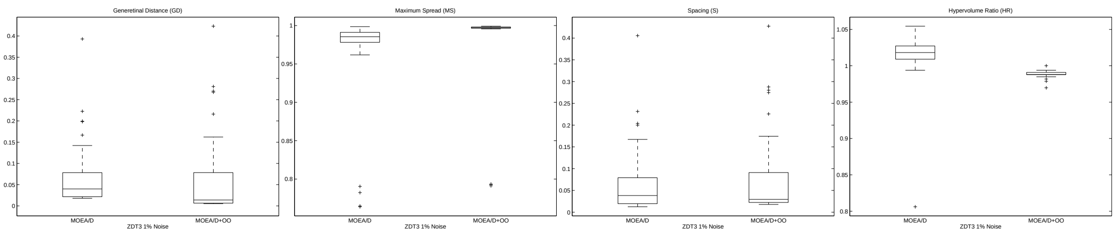
(a) (b) (c) (d)

Figure 4.26: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT1 with10% noise



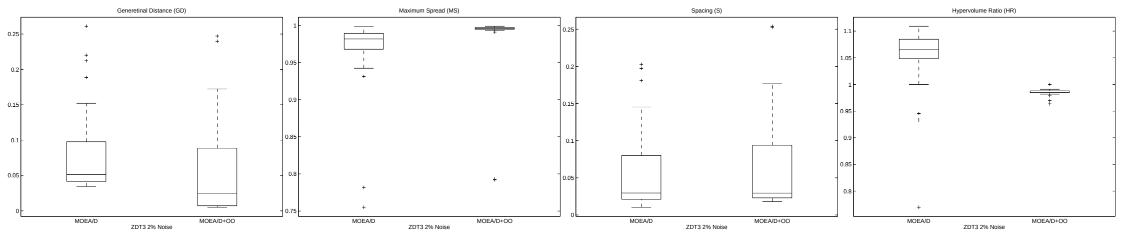
(a) (b) (c) (d)

Figure 4.27: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT1 with 20% noise



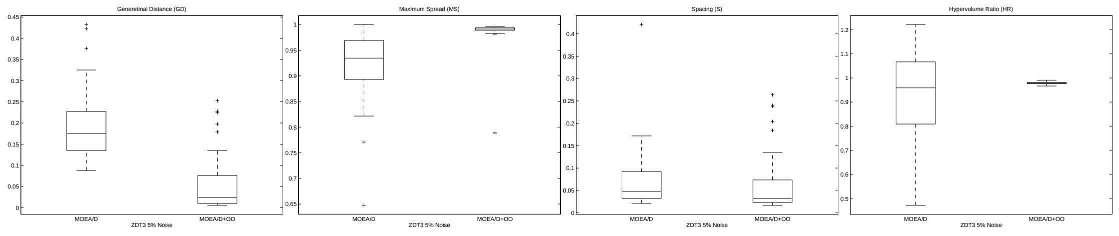
(a) (b) (c) (d)

Figure 4.28: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT3 with 1% noise



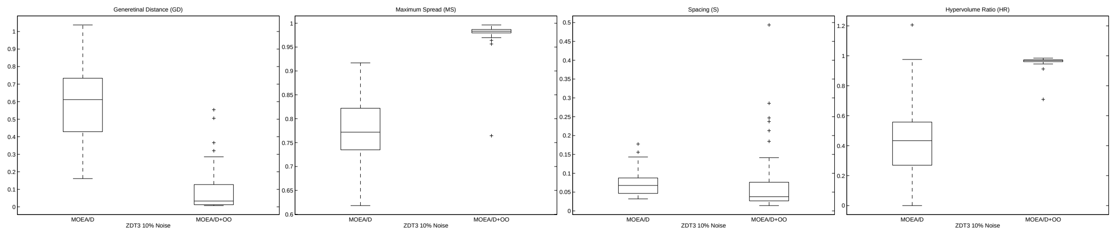
(a) (b) (c) (d)

Figure 4.29: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT3 with 2% noise



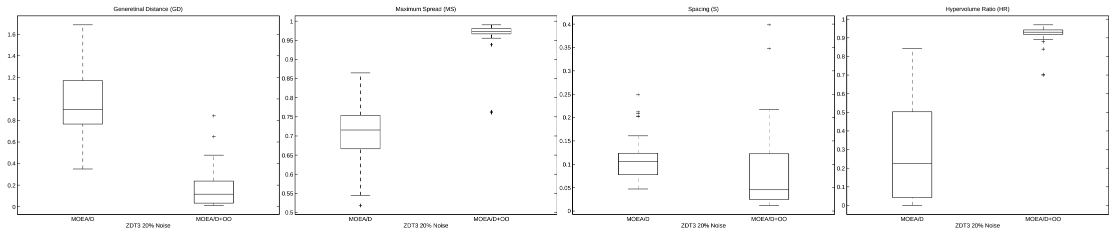
(a) (b) (c) (d)

Figure 4.30: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT3 with 5% noise



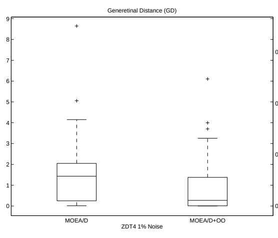
(a) (b) (c) (d)

Figure 4.31: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT3 with10% noise

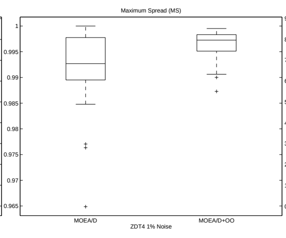


(a) (b) (c) (d)

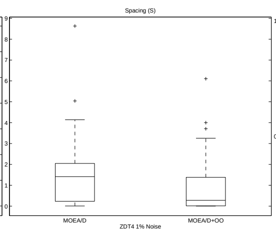
Figure 4.32: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT3 with 20% noise



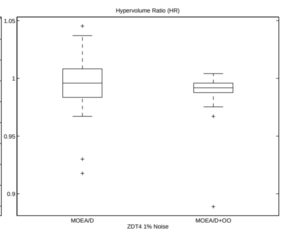
(a)



(b)

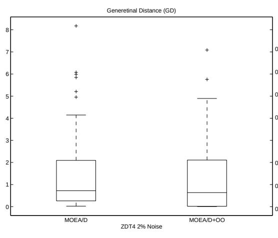


(c)

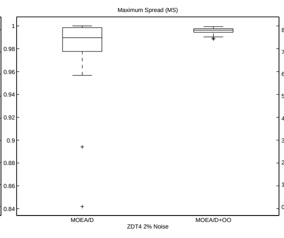


(d)

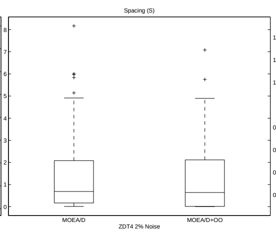
Figure 4.33: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT4 with 1% noise



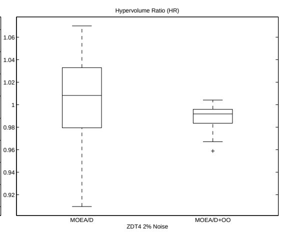
(a)



(b)

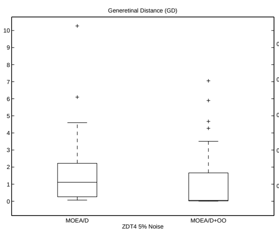


(c)

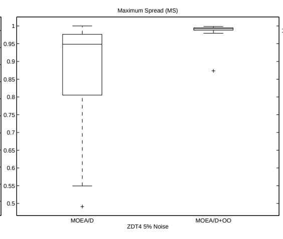


(d)

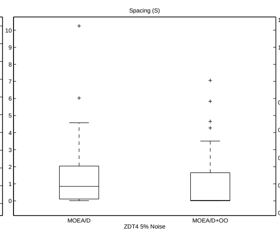
Figure 4.34: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT4 with 2% noise



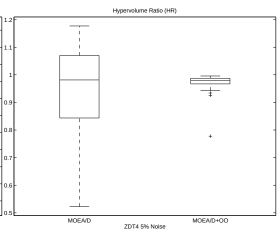
(a)



(b)

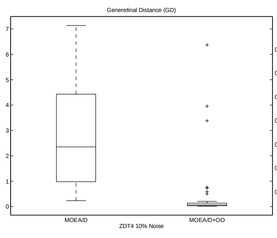


(c)

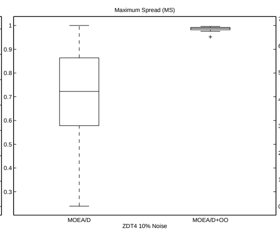


(d)

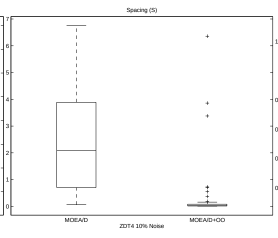
Figure 4.35: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT4 with 5% noise



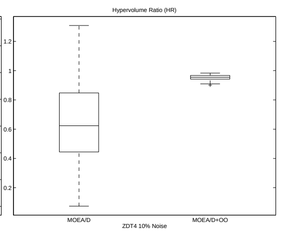
(a)



(b)

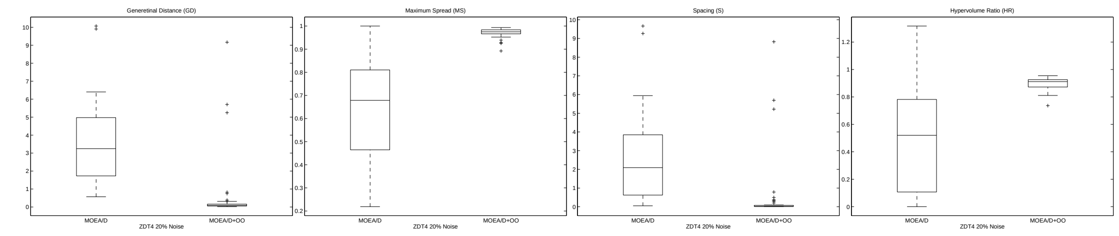


(c)



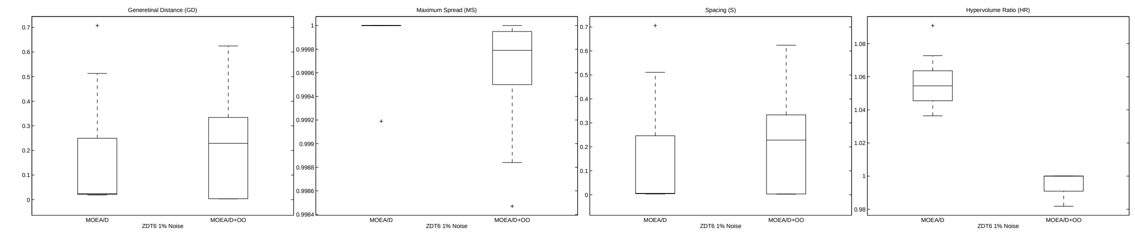
(d)

Figure 4.36: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT4 with10% noise



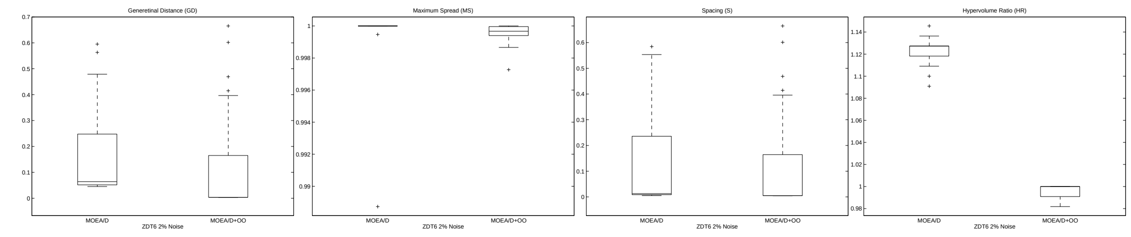
(a) (b) (c) (d)

Figure 4.37: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT4 with 20% noise



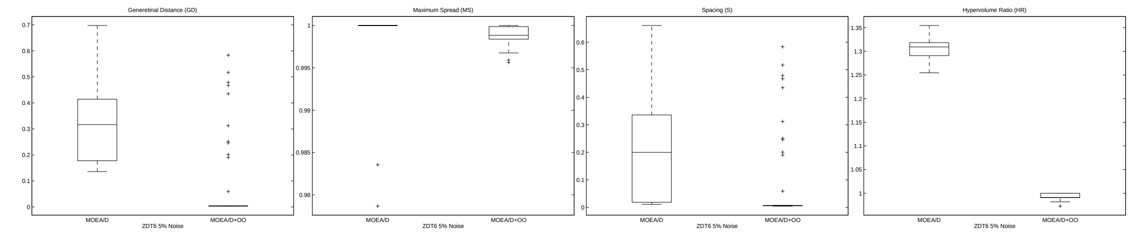
(a) (b) (c) (d)

Figure 4.38: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT6 with 1% noise



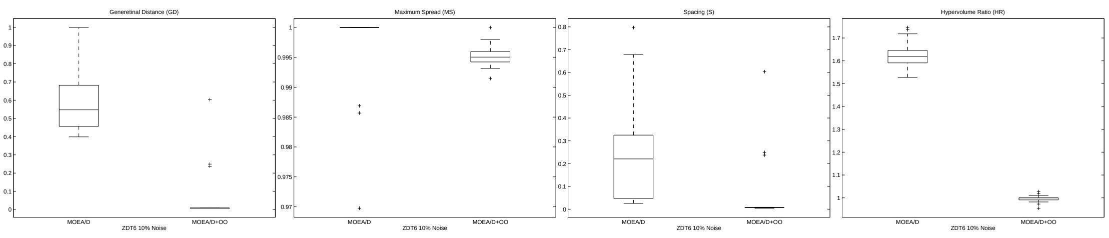
(a) (b) (c) (d)

Figure 4.39: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT6 with 2% noise



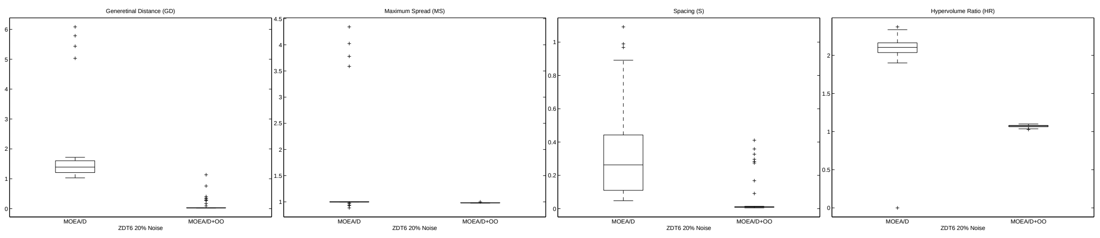
(a) (b) (c) (d)

Figure 4.40: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT6 with 5% noise



(a) (b) (c) (d)

Figure 4.41: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT6 with10% noise



(a) (b) (c) (d)

Figure 4.42: Performance metric of (a) GD, (b) MS, (c) S and (d) HVR for ZDT6 with 20% noise

### 4.3.3 Discussion

As we argued in Section 3.5.3 the true Pareto front is the best estimation of each benchmark problem. It can be seen from the results of our experiment that MOEA/D+OO is capable of evolving a near optimal, diverse and uniformly distributed Pareto front for the different benchmark problems discussed in Section 3.5.2. By comparing the results of MOEA/D in Section 3.5.3 and MOEA/D+OO in Section 4.3.2 it can be clearly observed that MOEA/D+OO significantly outperforms MOEA/D in noisy environments.

The evidence of MOEA/D+OO's superiority over its basic version in relation to the different performance metrics can be established as follows:

#### 4.3.3.1 GD

Generational distance measures the gap between the evaluated Pareto front and true Pareto front, as outlined in Section 3.4. Looking at the GD values for MOEA/D+OO in Tables 4.1 to 4.6, they are very small and almost zero in most cases, indicating that the new algorithm is able to approximate solutions very close to the true Pareto front. With the noisy, multimodal featured ZDT4 problem, the algorithm was certainly faced with a bigger challenge, even in low levels of noises, but in general it performs quite satisfactorily (See Figure 4.11).

By considering the values in parentheses, we can see how the performance of MOEA/D deteriorates when noise intensifies because the values increase with noise. As GD is a proximity indicator its measure shows that MOEA/D+OO converges to solutions much better than MOEA/D in noisy environments. Thus, the GD metric demonstrates that MOEA/D+OO outperforms MOEA/D for noisy MOP.

#### 4.3.3.2 MS

$MS$ (Maximum Spread) is a diversity indicator. When it converges to one it means that the true Pareto front has been properly covered by the approximated Pareto front. Tables 4.1 to 4.6 in Section 4.3.2 include calculated values for the MS metric for MOEA/D+OO. Given that the MS values derived by the new algorithm are almost equal to one, it is clear that the approximated Pareto front covers the true Pareto front very well.

By contrast, the values reported in parentheses that show the differences between the two algorithms prove that the MOEA/D cannot maintain diversity of solution in a noisy environment. Thus, according to the MS indicator, the modified algorithm is superior to its basic version in the presence of noise.

#### 4.3.3.3 S

The spacing metric indicates the distribution of solutions along the Pareto front. This metric must return a small value close to zero (ideally zero) for a fine and evenly distributed Pareto front. By studying the value of the $S$ metrics in Tables 4.1 to 4.6, the success of MOEA/D+OO at obtaining evenly distributed solutions along the Pareto front can clearly be seen. According to this indicator, MOEA/D+OO can maintain a better distribution of solutions than basic MOEA/D.

#### 4.3.3.4 HVR

As described in Section 3.4, this indicator of general quality measures the diversity and proximity of the approximated Pareto front. It returns one for the ideal estimated solution. By tracking the HVR values in Tables 4.1 to 4.6, it is evident that MOEA/D+OO is significantly better than MOEA/D at approximating the near optimal in noisy environments. For MOEA/D, the HVR values increase on all benchmark problems when noise intensifies (as shown by the values in parentheses)

#### 4.3.3.5 Shape of Pareto front

Apart from performance metrics, the shape of the Pareto front itself can reveal testimony as to the performance of an algorithm. For this reason, Figures 4.7 to 4.12 represent the relevant Pareto fronts estimated by MOEA/D+OO in noisy problems. These Pareto fronts show some missing points from the estimated Pareto front in comparison with the true Pareto front. This reveals the diversity of solutions influenced by noise and shows how severe the effects can be. For example, Figures 4.9 and 3.4 show that MOEA/D+OO maintain better diversity than MOEA/D on the same problem in presence of different noise levels. These figures, together with the help of the true Pareto front, offer visual evidence to support the conclusions drawn from the analysis of the performance metrics.

### 4.3.4 Analytical Comparison

For this analysis, we draw box plots relating to the performance metrics of both MOEA/D+OO and MOEA/D in the presence of different levels of noise for each benchmark problem discussed in Section 3.5.2. MOEA/D+OO's superiority over MOEA/D on the different benchmark problems can be established as follows:

#### 4.3.4.1 FON

The two algorithms' performance metrics for the noisy benchmark problem FON are compared by box plots in Figures 4.13(a)-(d) to 4.17(a)-(d), from 1% noise to 20%

noise. It can be observed from Figure 4.13 that MOEA/D+OO outperforms MOEA/D in all performance metrics. This is because the box plot related to MOEA/D+OO shows the median of the data set closer to a sensible value for each performance metric. As previously indicated, the sensible value for GD and S is 0 whilst for MS and HVR it is desirable they converge to 1. Accordingly, Figures 4.14 to 4.17 testify to MOEA/D+OO's superiority over its basic version.

#### 4.3.4.2 KUR

The two algorithms' performance metrics for the noisy benchmark problem KUR are compared by box plots in Figures 4.18(a)-(d) to 4.22(a)-(d), from 1% noise to 20% noise. In Figures 4.18, 4.19 and 4.20 only the S metric would indicate that MOEA/D performed better than MOEA/D+OO. According to all other metrics MOEA/D+OO outperforms its basic version in all levels of noise.

#### 4.3.4.3 ZDT1

The two algorithms' performance metrics for the noisy benchmark problem ZDT1 are compared by box plots in Figures 4.23(a)-(d) to 4.27(a)-(d), from 1% noise to 20% noise. As can be seen from the box plots related to this problem, MOEA/D+OO outperforms its basic version in all performance metrics in the presence of different noise levels. This is because the median of each data set obtained by MOEA/D+OO is closer to the desirable value of each metric.

#### 4.3.4.4 ZDT3

The two algorithms' performance metrics for noisy benchmark problem ZDT3 by box plots in Figures 4.28(a)-(d) to 4.32(a)-(d), from 1% noise to 20% noise. Similar to its performance in problem ZDT1, the revised algorithm proposed in this thesis (MOEA/D+OO) handles noise better than MOEA/D with respect to all metrics and in the presence of different levels of noise on the noisy benchmark problem ZDT3 as well.

#### 4.3.4.5 ZDT4

Figures 4.33(a)-(d) to 4.37(a)-(d) compare the two algorithms in box plots, subject to the results of the different performance metrics on the noisy benchmark problem ZDT4 in 1% to 20% noise. In part (d) of Figure 4.33 MOEA/D has a closer median to 1 in the presence of 1% noise, but MOEA/D+OO obtains a shorter range for HVR, which is more sensible. For all other scenarios the results reveal that MOEA/D+OO definitively outperforms MOEA/D.

#### 4.3.4.6 ZDT6

Figures 4.38(a)-(d) to 4.42(a)-(d) compare the two algorithms in box plots, subject to the results of the different performance metrics on the noisy benchmark problem ZDT6 in 1% to 20% noise. From these figures, it can be seen that in the presence of low noise levels, such as 1% or 2%, MOEA/D's performance is comparative to that of MOEA/D+OO. This is due to the fact that in parts (a) and (c) of Figure 4.38 and in part (c) of Figure 4.39 the estimated value of both algorithms show almost the same quality. But in other instances MOEA/D+OO outperforms its basic version.

## 4.4 Conclusions

This chapter introduced a new algorithm called 'MOEA/D+OO'. This is a modified version of MOEA/D and is significantly better suited to handling noise. Ordinal optimization (OO) is a technique that softens the goal by compromising on a set of good enough solutions rather than a best solution. According to this technique the order of solutions is more robust than their value in noisy environments [3, 74].

The major contribution of this chapter is the proof that MOEA/D+OO significantly outperforms MOEA/D in the noisy multi-objective optimization problems detailed in Section 3.5.2, according to the following performance measures,

- GD for proximity of solutions. See Section 4.3.3.1 and 4.3.4.
- MS for diversity of solutions. See Section 4.3.3.2 and 4.3.4.

- S for distribution of solutions. See Section 4.3.3.3 and 4.3.4.
- HVR, a general quality metric, for diversity and proximity of solutions. See Section 4.3.3.4 and 4.3.4.
- Estimated Pareto front discussed in Sections 4.3.3.5 and 4.3.2.

# 5 Noisy Portfolio Optimization Problem

Economic noise, or simply noise, describes a theory of pricing developed by Fischer Black [120]. Black describes noise as the opposite of information. His theory states that noise is everywhere in the economy and we can rarely tell the difference between it and information [120].

This chapter studies the impact of noise on the noisy portfolio optimization problem. This is a classic problem in finance and economics where the goal is to maximize returns on investment whilst minimizing risk and thereby increasing wealth. Thus, in this chapter, we are going to solve the problem using both our modified MOEA/D+OO and basic MOEA/D, followed by a comparison of the performance results for both algorithms.

## 5.1 Introduction

Portfolio management is an important research topic in the finance world. A portfolio includes a number of assets, the returns of which vary in the market according to the patterns of various stochastic processes.

The objective is to maximize the total wealth of the portfolio by finding an optimal allocation of capital to a set of assets. Accordingly, an optimal asset weight must be selected.

The following factors are the key complicators in the portfolio optimization problem.

- Asset interrelationship.
- Decision maker's preferences.
- Resource allocation.
- Total budget limitation.

There exist several other factors, over and above these elements, that are also involved in making the portfolio optimization problem a complicated one.

Markowitz was awarded an economic sciences Nobel prize in 1990 for his modelling of the portfolio optimization problem. He proposed a fundamental answer to this problem based on the mean-variance model. Markowitz formulated this as an optimization problem with two criteria: maximize the reward (measured by the mean) and minimize the risk to the portfolio (measured by the variance of return). The trade-off between risk and return leads to a set of optimal portfolios that is called an efficient portfolio. From Markowitz until now enormous amounts of research studies have been published that either extend or modify the basic model in three major aspects [121,122] as follows:

1. Simplification of the type and amount of input data. When the number of portfolios for selection is large, estimating the covariance will become computationally impractical. [122–124]

2. Alternative measure of risk. Value-at-risk (VaR) has become a popular risk measure since its first recommendation. Unlike most widely used risk measures, which are based on historical returns, VaR is a forward-looking measure of risk for estimating future portfolio losses. When financial crises cause significant loss to many investors, Conditional value at Risk (CVaR) can be implemented which brings with it a higher confidence level [125, 126].

3. Additional criteria and/or constraints. The portfolio optimization problem can also be modelled as a tree objectives optimization problem [122,127] with liquidity or a number of securities in the portfolio along with risk and return.

In our research we use a two objective portfolio optimization based on the Markowitz mean-variance model.

## 5.2 Problem Definition

A mean-variance, two objectives portfolio optimization with $Q$ asset problems could be formulated as follows:

$$\begin{aligned} \max \quad & R(x) = \sum_{i=1}^{Q} x_i r_i \\ \min \quad & V(x) = \sum_{i=1}^{Q} \sum_{j=1}^{Q} x_i x_j \sigma_{ij} \\ s.t. \quad & x \geq 0 \in X \\ & \sum_{i=1}^{Q} x_i = 1 \end{aligned} \tag{5.1}$$

where $r_i$ is the expected return for asset $i$ and $\sigma_{ij}$ is the covariance between asset $i$ and $j$. Finally $x_i$ is the decision variable with a value of $[0, 1]$ for $i = 1, 2, \cdots Q$, denoting the composition of asset $i$ in the portfolio as a proportion of the total available capital. Non negativity constraint $x \geq 0$ indicates that no short sales are allowed [128].

As we have established, the portfolio optimization problem is multi-objective with its two objectives being minimised risk and maximized return. The trade-off between these represents a Pareto front, which is the set of all non-dominated solutions in the optimal portfolio.

Despite the existence of many traditional methods for solving multi-objective optimization problems, during the last two decades a number of evolutionary algorithms have been proposed that work with a population of candidate solutions that lead to a Pareto optimal solution after a specific number of generations in a single run.

Amongst all of these multi-objective optimization evolutionary algorithms (MOEA), the decomposition based MOEA (MOEA/D) is emerging as a very promising optimization framework. This decomposes the MOP into a number of scalar subproblems, the optimal solutions of which are Pareto optimal to the MOP.

In this chapter we will use the noisy portfolio optimization problem based on Markowitz's mean-variance model as an application to test MOEA/D and MOEA/D+OO. MOEA/D was proposed in [11] and MOEA/D+OO was introduced in Chapter 4 of this thesis. This algorithm combines the ordinal optimization technique with MOEA/D to handle noise.

## 5.3 Uncertainty in Portfolio Optimization Problem

Equation 5.1 outlines a mean-variance portfolio optimization problem. Based on this problem, we will define the noisy portfolio optimization as follows.

### 5.3.1 Definition of Uncertainty

The terms 'Risk' and 'Noise' that represent uncertainty in this real-life problem could be considered identical, but this would be a mistake. Whilst it is a fact that noise in our system can lead to a high level of risk for investors, this does not mean that risk and noise are the same. For this reason we study the impact of noise on the portfolio optimization problem. For clarity, we emphasise below the definitions of noise and risk.

Risk is the chance that the actual return on investment will be different from the expected return. Risk includes the possibility of losing some or all of the original investment. Different versions of risk are usually measured by calculating the standard deviation of the historical returns or average returns of a specific investment. High standard deviation indicates a high degree of risk.

Noise is the opposite of information and, due to the complex nature of the world's markets, not all market data is 'information'. Moreover, many of the price fluctuations we see on a day-to-day basis are due to random change rather than meaningful trends. As a result, portfolio optimization is noisy by its nature. Also noise can be caused by political changes, market policy changes, natural disasters, war and so on.

In this study we add a zero mean normal distributed noise with different standard divination to the return of the portfolio optimization problem in Equation 5.1. Different standard divinations are conducted for a diverse level of noises.

$$\begin{aligned} \bar{R}(x) &= R(x) + \delta \\ \delta &\sim N(0, \sigma^2) \end{aligned} \tag{5.2}$$

Equation 5.2 models noise in the market. In this model we only add noise to the return in the objective space. This thesis makes a first attempt at a portfolio optimization with noise. In this thesis, we focus on noises in the return objective function. Note, that adding noise to risk is a worthy extension of Markowitz mean-variance model. It will be left for future research.

This study shows how noise could be handled in our financial system by reducing the effects of noise on the decision making progress. For this purpose, we employ the ordinal optimization technique to handle uncertainty. A review of ordinal optimization is provided in Section 2.5

## 5.4 Experiment

The purpose of this experiment is to assess the performance of the MOEA/D+OO and MOEA/D algorithms in handling the portfolio optimization problem in the presence of different levels of noise. We compare the performance of MOEA/D+OO and the

Table 5.1: Result of portfolio optimization returns with 30 assets by MOEA/D

| 30 Assets | Min | Max | Range | St.D. | Mean |
|---|---|---|---|---|---|
| No Noise | 0.00952 | 0.02153 | 0.01200 | 0.00848 | 0.01553 |
| 1% Noise | 0.00773 | 0.02905 | 0.02132 | 0.01507 | 0.01839 |
| 2% Noise | 0.01120 | 0.05778 | 0.04658 | 0.03293 | 0.03449 |
| 5% Noise | 0.02315 | 0.12758 | 0.10443 | 0.07384 | 0.07536 |
| 10% Noise | 0.11287 | 0.32321 | 0.21034 | 0.14873 | 0.21804 |
| 20% Noise | 0.31974 | 0.56067 | 0.24094 | 0.17037 | 0.44020 |

Table 5.2: Result of portfolio optimization returns with 60 assets by MOEA/D

| 60 Assets | Min | Max | Range | St.D. | Mean |
|---|---|---|---|---|---|
| No Noise | 0.01330 | 0.03684 | 0.02354 | 0.01664 | 0.02507 |
| 1% Noise | 0.00816 | 0.02784 | 0.01967 | 0.01391 | 0.01800 |
| 2% Noise | 0.01703 | 0.06242 | 0.04539 | 0.03209 | 0.03972 |
| 5% Noise | 0.03539 | 0.13542 | 0.10003 | 0.07073 | 0.08540 |
| 10% Noise | 0.09729 | 0.29031 | 0.19301 | 0.13648 | 0.19380 |
| 20% Noise | 0.35623 | 0.58276 | 0.22654 | 0.16019 | 0.46950 |

original MOEA/D to see whether the modification has improved its ability to handle noise.

The data used in this thesis are the daily returns of 64 different industries over a period of more than five years. These data are collected from Yahoo Finance.

### 5.4.1 How to evaluate results?

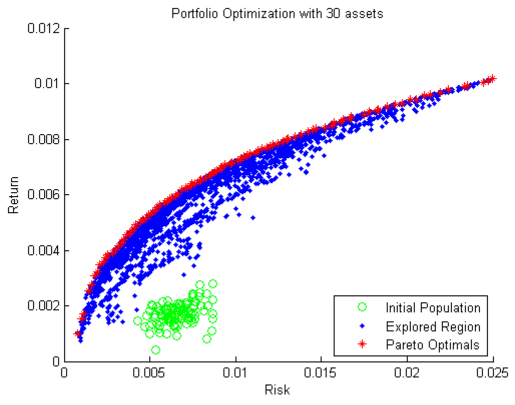
First it is necessary to clarify the meaning of certain terms such as 'better performance' and 'good solutions' whilst the true Pareto front is still unknown to us.

We consider Algorithm A better than Algorithm B if, in the presence of noise, A generates solutions closer to noiseless solutions than B. This is shown in Figure 5.5. The solutions obtained by the superior algorithm are considered good solutions.

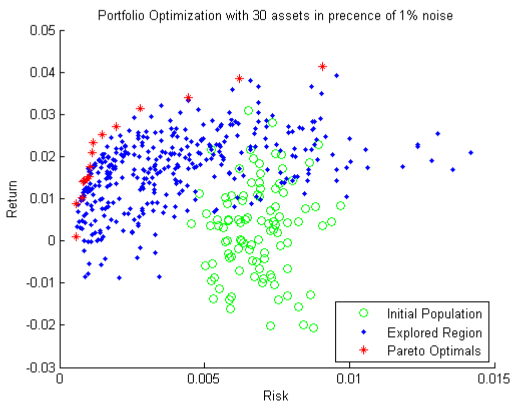
By noiseless solutions, we mean solutions generated in the absence of noise. We use the noiseless solutions produced by MOEA/D as our basis of comparison.

### 5.4.2 Discussion

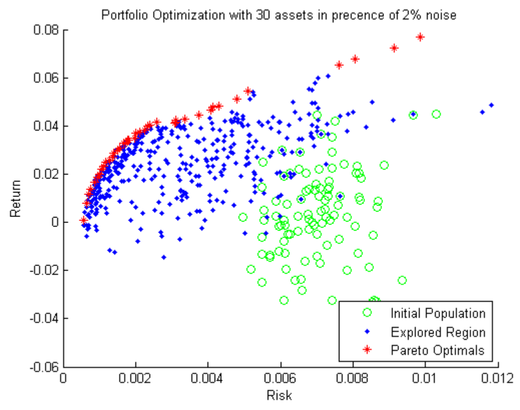
This experiment challenged the algorithms to find optimal portfolios for problems of 30 and 60 assets in turn. They were faced with low noise levels (1% and 2%), medium



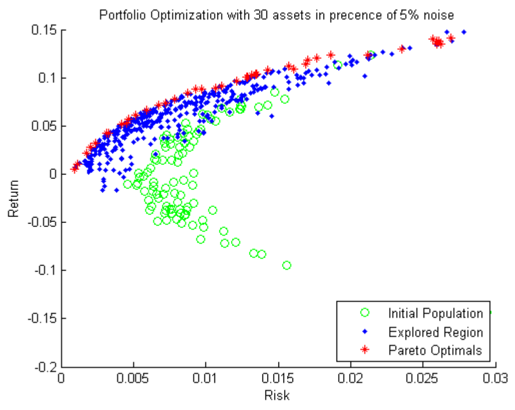
(a)



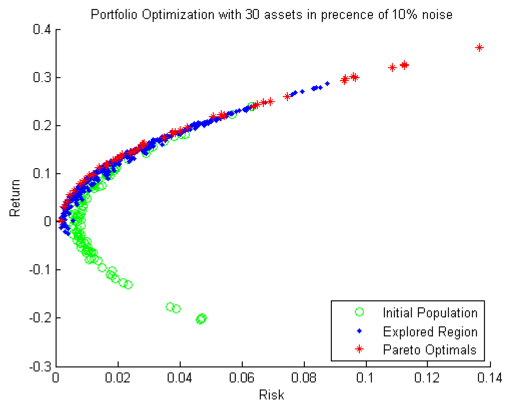
(b)



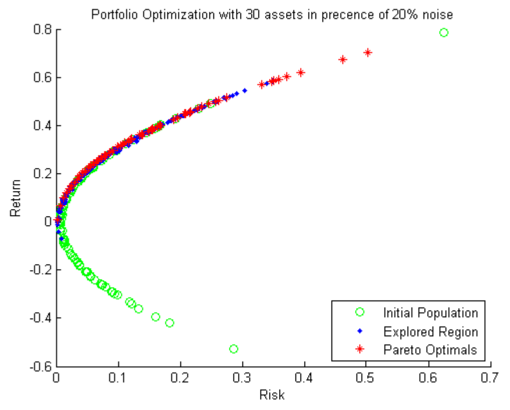
(c)



(d)



(e)



(f)

Figure 5.1: MOEA/D results for portfolio optimization problem with 60 assets (a) without noise, (b) 1%, (c) 2%, (d) 5%, (e) 10%, (f) 20%.

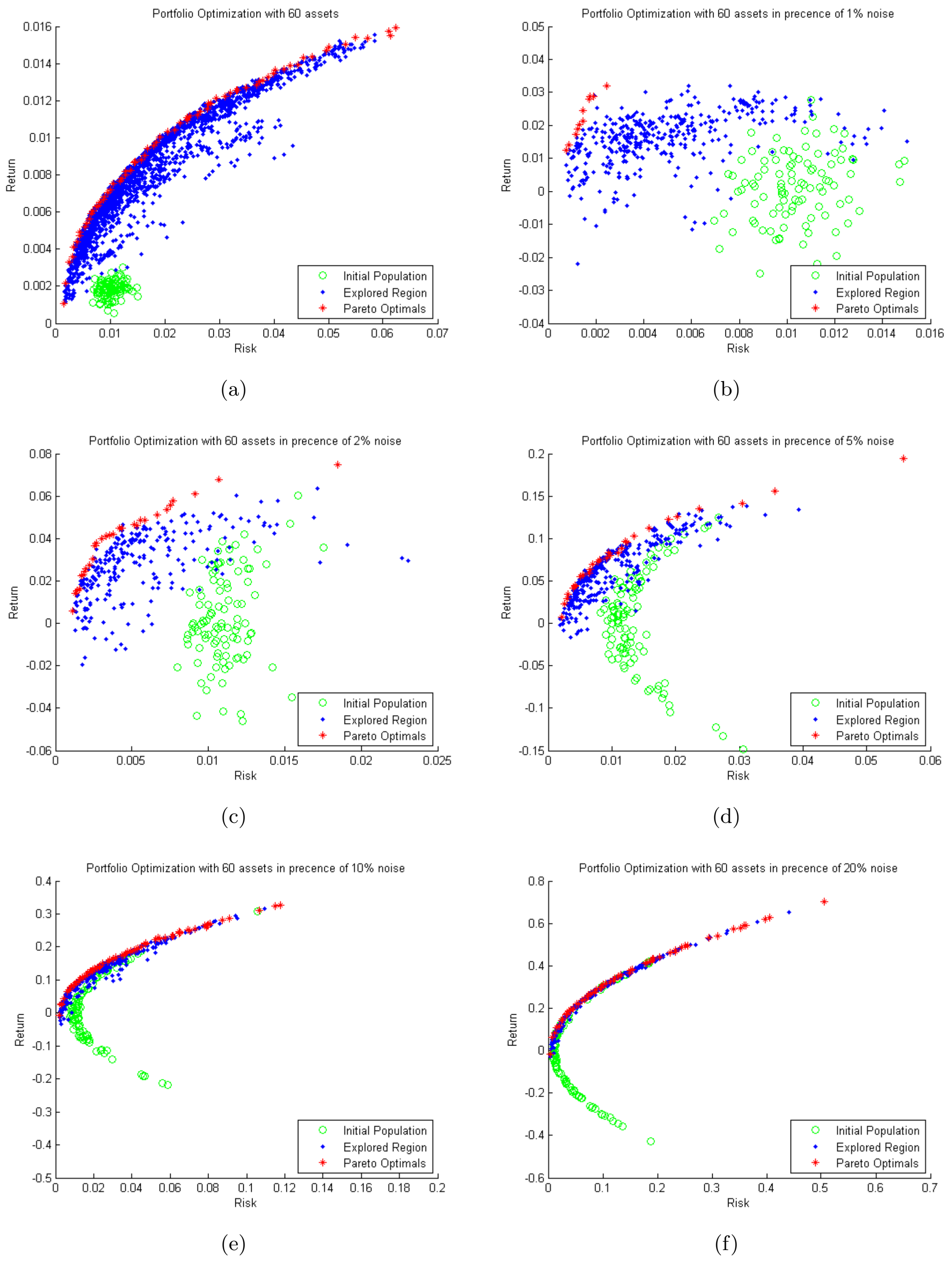
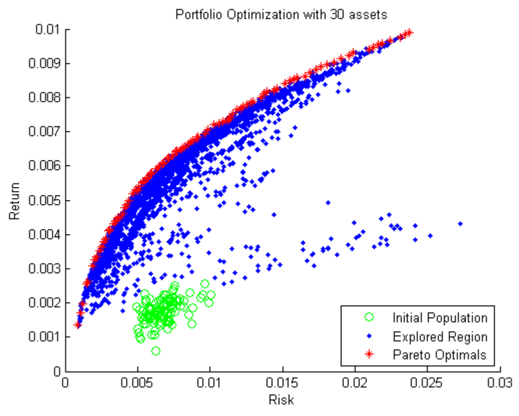
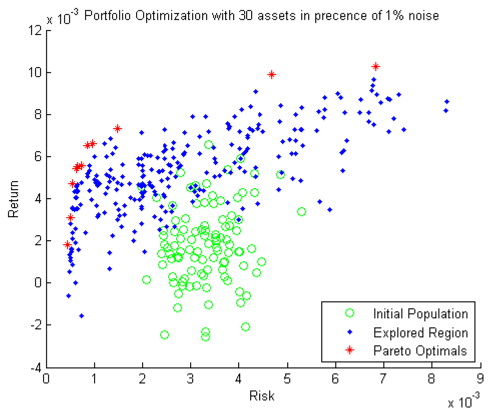


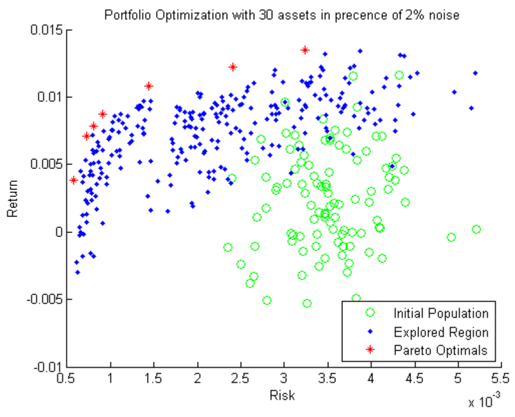
Figure 5.2: MOEA/D results for portfolio optimization problem with 60 assets (a) without noise, (b) 1%, (c) 2%, (d) 5%, (e) 10%, (f) 20%.



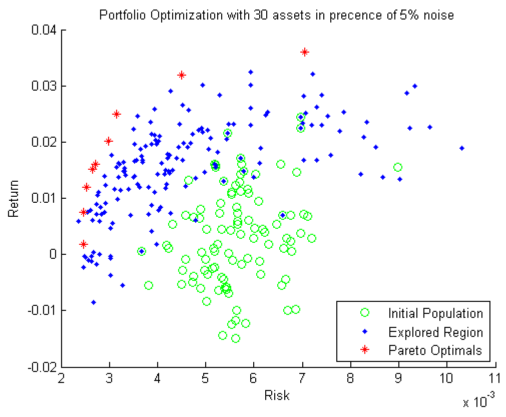
(a)



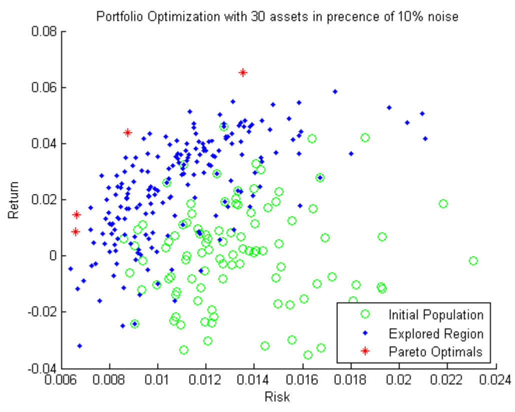
(b)



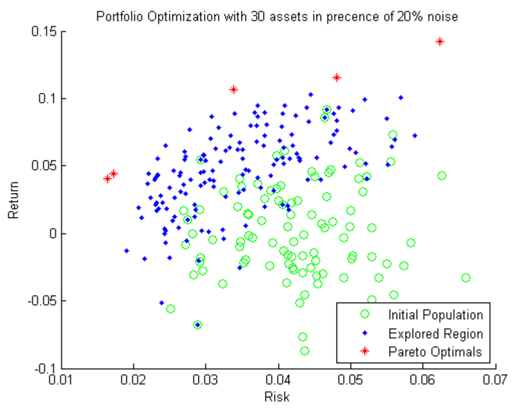
(c)



(d)

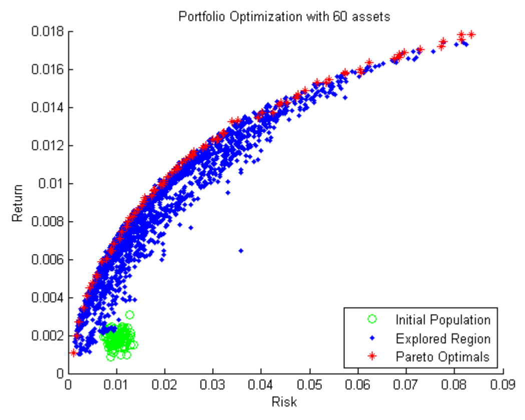


(e)

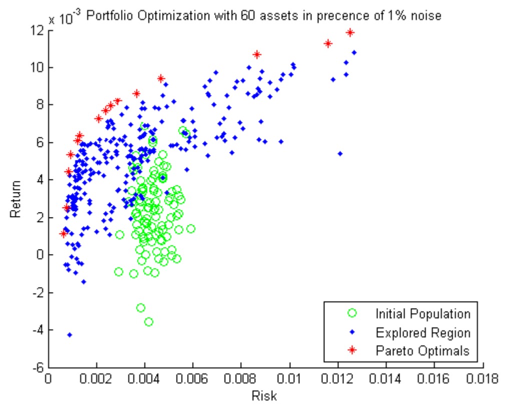


(f)

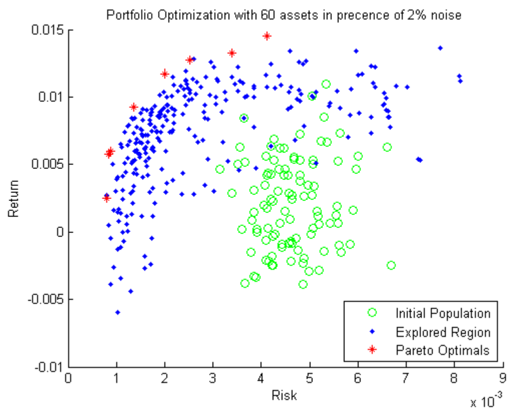
Figure 5.3: MOEA/D+OO results for portfolio optimization problem with 30 assets (a) without noise, (b) 1%, (c) 2%, (d) 5%, (e) 10%, (f) 20%.



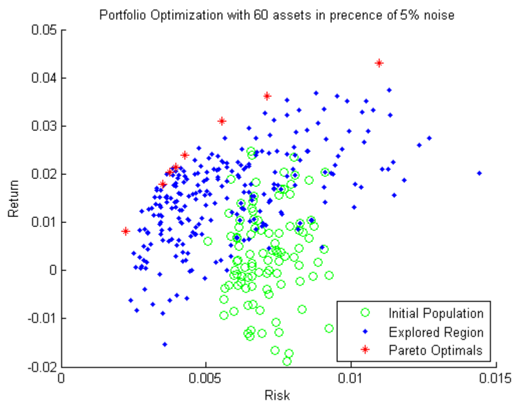
(a)



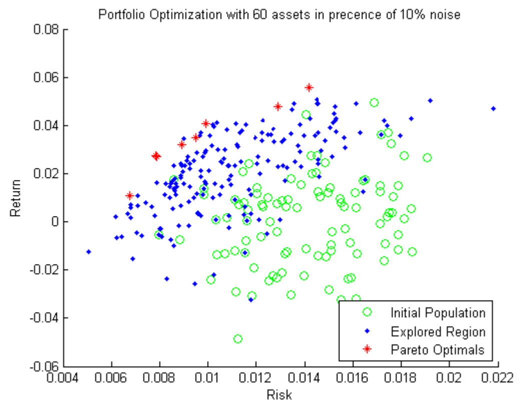
(b)



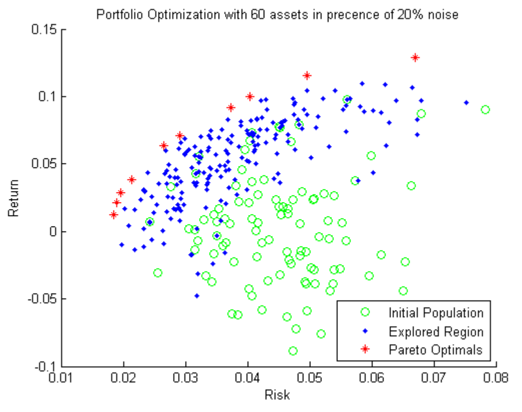
(c)



(d)



(e)



(f)

Figure 5.4: MOEA/D+OO results for portfolio optimization problem with 30 assets (a) without noise, (b) 1%, (c) 2%, (d) 5%, (e) 10%, (f) 20%.

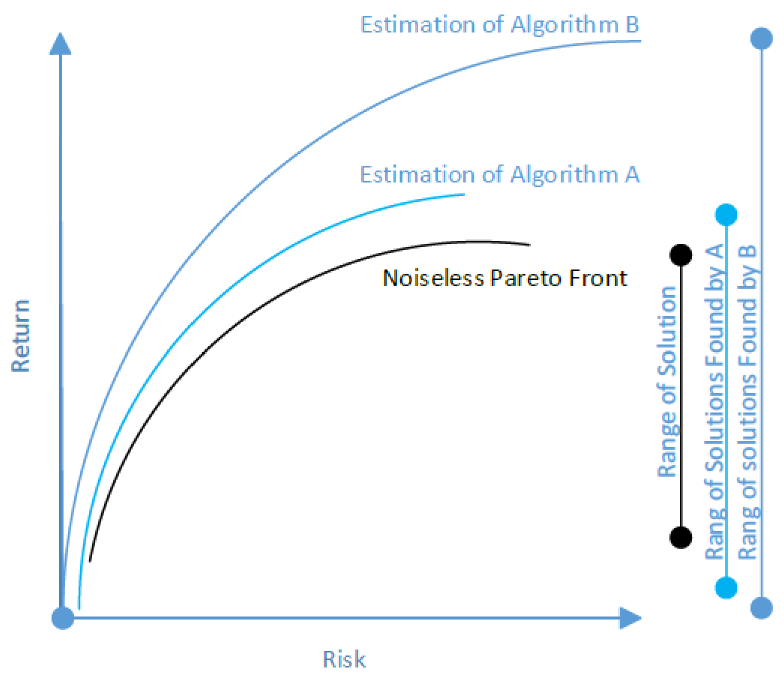

Estimation of Algorithm B
Estimation of Algorithm A
Noiseless Pareto Front
Return
Risk
Range of Solution
Rang of Solutions Found by A
Rang of solutions Found by B

Figure 5.5: Algorithm A is better than B

Table 5.3: Result of portfolio optimization returns with 30 assets by MOEA/D+OO

| 30 Assets | Min | Max | Range | St.D. | Mean |
|---|---|---|---|---|---|
| No Noise | 0.00952 | 0.02153 | 0.01200 | 0.008480 | 0.01553 |
| 1% Noise | 0.00243 | 0.01365 | 0.01121 | 0.002013 | 0.01034 |
| 2% Noise | 0.00141 | 0.02565 | 0.02424 | 0.004022 | 0.01940 |
| 5% Noise | 0.00156 | 0.05987 | 0.05831 | 0.015169 | 0.03520 |
| 10% Noise | 0.00219 | 0.11584 | 0.11364 | 0.029165 | 0.06544 |
| 20% Noise | 0.00940 | 0.21456 | 0.20516 | 0.050858 | 0.13313 |

Table 5.4: Result of portfolio optimization returns with 60 assets by MOEA/D+OO

| 60 Assets | Min | Max | Range | St.D. | Mean |
|---|---|---|---|---|---|
| No Noise | 0.01330 | 0.03684 | 0.02354 | 0.01664 | 0.02507 |
| 1% Noise | 0.00417 | 0.01428 | 0.01010 | 0.00182 | 0.01025 |
| 2% Noise | 0.00440 | 0.02553 | 0.02112 | 0.00461 | 0.01837 |
| 5% Noise | 0.00225 | 0.05075 | 0.04850 | 0.01263 | 0.03661 |
| 10% Noise | 0.00157 | 0.12257 | 0.12100 | 0.02990 | 0.07577 |
| 20% Noise | 0.00659 | 0.22424 | 0.21765 | 0.05679 | 0.14036 |

noise (5%) and high noise levels (10%, 20%). We compared the performance of both algorithms to see whether the modifications to MOEA/D made the modified algorithm better at handling noise.

As can be seen from Figure 5.1, without noise, the MOEA/D found solutions with returns from 0.1% to 1% for the portfolio problem with 30 assets but this range could not be maintained when noise intensified. Thus, the solutions found by MOEA/D in the presence of 20% noise are between 0% and 80%, which could easily mislead investors. On the other hand, the MOEA/D+OO found solutions with returns from 1% to 14% for the same problem in the presence of 20% noise, as shown in Figure 5.3. This represents a real solution range. MOEA/D+OO maintains this tendency more or less over the other noise levels, proving it better than MOEA/D.

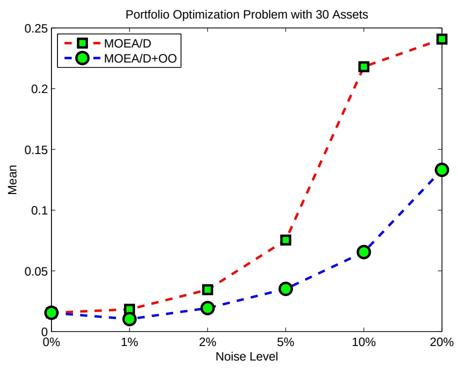
By tracking the results in Figures 5.2 and 5.4 for the portfolio optimization problem with 60 assets, we can see that MOEA/D+OOT maintains its superiority over MOEA/D.

Figure 5.1 shows MOEA/D faced with an exploration challenge for the portfolio optimization problem. As can be seen, there is a meaningful reduction in the portfolio regions explored portfolios. Blue dots represent explored portfolios in Figure 5.1.

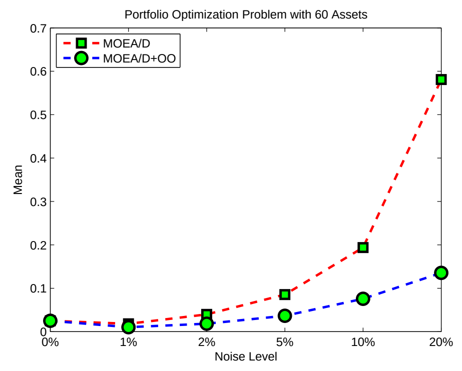
The modified algorithm (MOEA/D+OO) might not be the best algorithm for solving the noisy portfolio optimization problem, but it is the first attempt at handling noise in this problem. To adequately handle noise in this type of problem, not only does the algorithm need to be equipped for handling noise but also provided with a modelling technique developed enough for the problem.

#### 5.4.2.1 Analytical Comparison

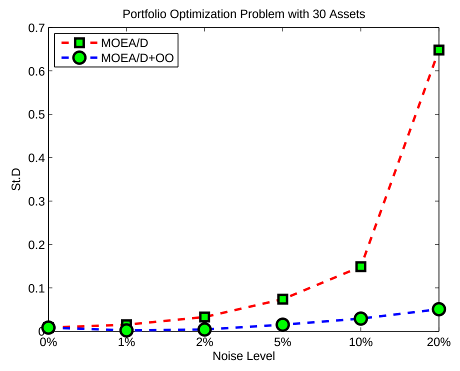
In this study, noise is only added to the return in objective space. For this reason, we have collected statistics purely from the objective function of the returns portion of the portfolio optimization problem. Tables 5.1 and 5.2 report the minimum, maximum, range of estimated solutions, standard divination and mean of solutions which have been found by MOEA/D for portfolios with 30 and 60 assets respectively. We provide



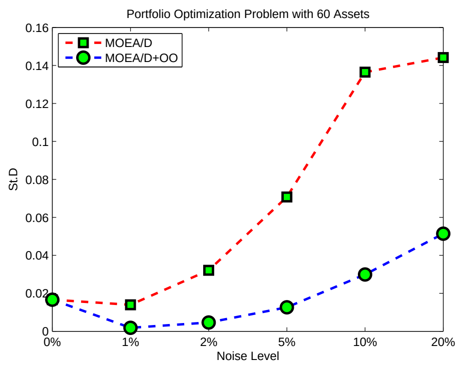
(a)



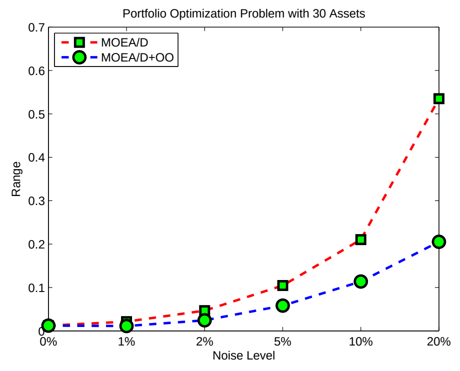
(b)



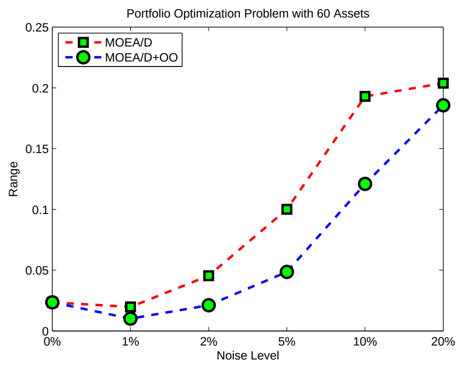
(c)



(d)



(e)



(f)

Figure 5.6: MOEA/D+OO vs. MOEA/D with respect to mean, standard divination and range of return for noisy portfolio optimization problem

the same information for MOEA/D+OO in Tables 5.3 and 5.4.

Tables 5.1 to 5.4 show which values of each column increase from top to bottom. This indicates that the performance of the algorithms deteriorate when noise intensities increase. In fact, both algorithms show the same trend but MOEA/D+OO obtains values closer to noiseless evaluation values than MOEA/D. To further illustrate this point, the respective values are plotted in Figure 5.6 for a clear comparison of both algorithms.

MOEA/D+OO is compared with its basic version in Figure 5.6 with respect to 1) the means in Parts (a) and (b) for 30 and 60 asset portfolios respectively, 2) its standard divination in Parts (c) and (d) for 30 and 60 asset portfolios respectively and 3) its range of solutions in Parts (e) and (f) for 30 and 60 asset portfolios respectively, against levels of 0%, 1%, 2%, 5%, 10% and 20% noise. As is clearly demonstrated, MOEA/D+OO out performs MOEA/D by estimating solutions closer to a noiseless solution (0%) with respect to mean, standard divination and range of solutions.

## 5.5 Conclusions

The portfolio optimization problem makes use of historical stock market data to assist investors in planning future investments. Based on how much risk an investor is willing to take, for a certain return a proportional investing strategy can be accomplished. Naturally, there is no guarantee on results as this optimization problem produces answers by looking at the market's past behaviour. In other words, the predicted returns found through portfolio optimization are noisy by nature. As a result of this noise, and general noise in the market, making a profit depends entirely on the robustness of the portfolio in a noisy environment.

As can be seen by the results of this study, portfolio optimization is very sensitive to noise - even gentle turbulence has an impact. This fact could render investors very vulnerable to sudden changes in the market, making a noise handling strategy extremely important. Their success or failure could depend on the strategy chosen.

In this thesis we have solved the noisy portfolio optimization problem with our noise-robust algorithm (MOEA/D+OO). We compared MOEA/D+OO with MOEA/D to demonstrate the significance of our modification. Thus, the major contributions of this chapter are listed below:

1. Noisy portfolio optimization has never been done before.

2. Portfolio optimization is very sensitive to noise.

3. MOEA/D+OO is better than MOEA/D at handling noise in the portfolio optimization problem. See Section 5.4.2 and Figure 5.6.

# 6 Conclusions

This study set out to investigate the performance of decomposition evolutionary algorithms in noisy multi-objective optimization problems. It is a fact that many real world problems are both multi-objective and noisy. Noise can be produced from a range of different sources, as discussed in Section 1.1. As a result, it is necessary to educate ourselves on this natural phenomenon and to find robust optimization techniques with which to handle noisy problems.

## 6.1 Summary

Our intended aim for this research was to find a modification for MOEA/D that allowed it to handle noisy problems. This work, therefore, contributes towards understanding the impact of noise on MOEA/D. As described in Section 3.1, we modelled the noise that disrupts the objective function on a Gaussian distribution. This is the most common way to introduce noise, although there are some studies that do consider other distributions [35].

Following the modelling of noisy problems, we studied the basic MOEA/D for a deeper understanding of the impact of noise on its performance. In the presence of noise, the performance of MOEA/D can deteriorate rapidly, as seen in Section **??**.The detrimental effects of noise on selection, elitism and diversity preservation present a significant challenge to the algorithm in its efforts to achieve a good estimation of the Pareto front. For this reason, benchmark problems with different characteristics, as described in Table 3.1, and different performance metrics, as outlined in Section 3.4, were comprehensively applied.

MOEA/D was modified for handling noisy problems through combination with an optimization technique known as *Ordinal Optimization* [3]. Due to the fact that order of solutions is more robust than their values (see Section 2.5) this technique proves itself to be highly effective at handling noise. Its performance is discussed in Section 4.3.3. The new algorithm is called MOEA/D+OO.

For the first time, we investigated the noisy portfolio optimization problem on real world problems and solved it using both algorithms. In spite of being noisy in nature, this problem shows high sensitivity to the slightest noise in our experiments, the results of which can be seen in Section 5.4.2.

## 6.2 Contributions

The main empirical finding is that the modified algorithm MOEA/D+OO significantly outperforms the basic MOEA/D.

The major contributions of this work can be summarized as follows:

- MOEA/D was studied in the presence of different noise levels to gain a deeper understanding of the impact of noise deterioration.
- We introduced a new algorithm called 'MOEA/D+OO' to handle noisy problems. It significantly outperformed basic MOEA/D on our diverse selection of benchmark problems.
- We employed the proposed algorithm to solve the noisy portfolio optimization problem with different numbers of assets. This is a classic finance optimization problem in which the goal is to find an optimal policy to use on all types of available assets in the market to make the total return maximal by taking a minimum risk. The efficient portfolios estimated by 'MOEA/D+OO' are closer to noiseless than those found by MOEA/D.

## 6.3 Future Work

Several interesting issues remain to be addressed and the most interesting topics for further study are listed below.

- Using a surrogate model, constructed as a crude model, to evaluate the objective function. There are various methods to introduce a surrogate model, including Support Vector Regression (SVR), Artificial Neural Network (ANN) and many more [78].
- Using a budget allocation technique to reduce the total simulation cost. In this method, critical designs receive a larger portion of the computing budget to reduce the estimator variance. Ordinal optimization can significantly reduce the computational cost, therefore using an intelligent budget controlling process could be a future improvement [129].

## 6.4 Conclusion

Despite the many ground breaking studies that have been undertaken on optimization in noisy environments in recent decades, there still exist many areas for continued development. This thesis is just the beginning of a long journey and this area demands greater attention in future studies.

# References

[1] K. Miettinen, *Nonlinear Multiobjective Optimization*, ser. International Series in Operations Research & Management Science. Springer US, 1999.

[2] C. Coello, G. Lamont, and D. van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*, ser. Genetic and Evolutionary Computation. Springer, 2007.

[3] Y. Ho, Q. Zhao, and Q. Jia, *Ordinal Optimization: Soft Optimization for Hard Problems*, ser. The International Series on Discrete Event Dynamic Systems. Springer, 2007.

[4] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *Evolutionary Computation, IEEE Transactions on*, vol. 9, no. 3, pp. 303–317, 2005.

[5] E. J. Hughes, "Evolutionary multi-objective ranking with uncertainty and noise," in *Evolutionary multi-criterion optimization.* Springer, 2001, pp. 329–343.

[6] H.-G. Beyer, "Evolutionary algorithms in noisy environments: theoretical issues and guidelines for practice," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2, pp. 239–267, 2000.

[7] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *IEEE Transactions on Evolutionary Computation*, pp. 257–271, 1999.

[8] D. Büche, P. Stoll, R. Dornberger, and P. Koumoutsakos, "Multiobjective evolutionary algorithm for the optimization of noisy combustion processes," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 32, no. 4, pp. 460–473, 2002.

[9] C. K. Goh and K. C. Tan, "An investigation on noisy environments in evolutionary multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 11, no. 3, pp. 354–381, 2007.

[10] M. Babbar, A. Lakshmikantha, and D. E. Goldberg, "A modified nsga-ii to solve noisy multiobjective problems," in *Genetic and Evolutionary Computation Conference (GECCO 2003)*, vol. 2723, 2003, pp. 21–27.

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

[12] A. Engelbrecht, *Computational Intelligence: An Introduction.* Wiley, 2007.

[13] K. Deb, *Multi-Objective Optimization Using Evolutionary Algorithms*, ser. Wiley Interscience Series in Systems and Optimization. Wiley, 2001.

[14] J. L. Cohon, "Multicriteria programming: Brief review and application," *Design optimization*, pp. 163–191, 1985.

[15] C.-L. Hwang and A. S. M. Masud, *Multiple objective decision making-methods and applications: a state-of-the-art survey.* Springer Science & Business Media, 2012, vol. 164.

[16] K. Deb and R. B. Agrawal, "Simulated binary crossover for continuous search space," *Complex Systems*, vol. 9, no. 3, pp. 1–15, 1994.

[17] H. Kita, I. Ono, and S. Kobayashi, "Theoretical analysis of the unimodal normal distribution crossover for real-coded genetic algorithms," in *Evolutionary Compu-*

*tation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on.* IEEE, 1998, pp. 529–534.

[18] 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.

[19] T. Higuchi, S. Tsutsui, and M. Yamamura, "Theoretical analysis of simplex crossover for real-coded genetic algorithms," in *Parallel Problem Solving from Nature PPSN VI.* Springer, 2000, pp. 365–374.

[20] R. Hinterding, "Gaussian mutation and self-adaption for numeric genetic algorithms," in *Evolutionary Computation, 1995., IEEE International Conference on*, vol. 1. IEEE, 1995, p. 384.

[21] T. Robič and B. Filipič, "Demo: Differential evolution for multiobjective optimization," in *Evolutionary multi-criterion optimization.* Springer, 2005, pp. 520–533.

[22] T. Bäck and H.-P. Schwefel, "An overview of evolutionary algorithms for parameter optimization," *Evolutionary computation*, vol. 1, no. 1, pp. 1–23, 1993.

[23] A. C. Koenig, "A study of mutation methods for evolutionary algorithms," *Advanced Topics in Artificial Intelligence, CS*, vol. 447, 2002.

[24] A. V. Veldhuizen and B. Lamont, "Evolutionary Computation and Convergence to a Pareto Front," in *Stanford University, California.* Morgan Kaufmann, 1998, pp. 221–228.

[25] T. Hanne, "On the convergence of multiobjective evolutionary algorithms," *European Journal of Operational Research*, vol. 117, no. 3, pp. 553–564, 1999.

[26] J. Horn, N. Nafpliotis, and D. E. Goldberg, "A niched pareto genetic algorithm for multiobjective optimization," in *Evolutionary Computation, 1994. IEEE World*

*Congress on Computational Intelligence., Proceedings of the First IEEE Conference on.* Ieee, 1994, pp. 82–87.

[27] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *evolutionary computation, IEEE transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

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

[29] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," 2001.

[30] C. M. Fonseca and P. J. Fleming, "Genetic algorithms for multiobjective optimization: Formulation, discussion and generalization," 1993.

[31] J. Knowles and D. Corne, "The pareto archived evolution strategy: a new baseline algorithm for pareto multiobjective optimisation," in *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, vol. 1, 1999, pp. –105 Vol. 1.

[32] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: Nsga-ii," 2000.

[33] E. Zitzler, M. Laumanns, and L. Thiele, "Spea2: Improving the strength pareto evolutionary algorithm," Tech. Rep., 2001.

[34] C. Fonseca and P. Fleming, "Multiobjective genetic algorithms made easy: selection sharing and mating restriction," in *Genetic Algorithms in Engineering Systems: Innovations and Applications, 1995. GALESIA. First International Conference on (Conf. Publ. No. 414)*, Sep 1995.

[35] C. Goh and K. Tan, *Evolutionary Multi-objective Optimization in Uncertain Environments: Issues and Algorithms*, ser. Studies in Computational Intelligence. Springer, 2009.

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

[37] E. Zitzler and S. Knzli, "Indicator-based selection in multiobjective search," in *in Proc. 8th International Conference on Parallel Problem Solving from Nature (PPSN VIII.* Springer, 2004, pp. 832–842.

[38] J. Fieldsend, R. Everson, and S. Singh, "Using unconstrained elite archives for multiobjective optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 3, pp. 305–323, June 2003.

[39] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 577–601, 2014.

[40] H. Jain and K. Deb, "An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: handling constraints and extending to an adaptive approach," *Evolutionary Computation, IEEE Transactions on*, vol. 18, no. 4, pp. 602–622, 2014.

[41] S. Boyd, L. Xiao, A. Mutapcic, and J. Mattingley, "Notes on decomposition methods," *Notes for EE364B, Stanford University*, pp. 1–36, 2007.

[42] G. B. Dantzig and P. Wolfe, "Decomposition principle for linear programs," *Operations research*, vol. 8, no. 1, pp. 101–111, 1960.

[43] M. Liu, X. Zou, Y. Chen, and Z. Wu, "Performance assessment of dmoea-dd with cec 2009 moea competition test instances." in *IEEE Congress on Evolutionary Computation*, vol. 1, 2009, pp. 2913–2918.

[44] H. Ishibuchi and T. Murata, "A multi-objective genetic local search algorithm and its application to flowshop scheduling," *Systems, Man, and Cybernetics, Part*

*C: Applications and Reviews, IEEE Transactions on*, vol. 28, no. 3, pp. 392–403, 1998.

[45] V. J. Bowman Jr, "On the relationship of the tchebycheff norm and the efficient frontier of multiple-criteria objectives," in *Multiple criteria decision making.* Springer, 1976, pp. 76–86.

[46] Y. Qi, X. Ma, F. Liu, L. Jiao, J. Sun, and J. Wu, "Moea/d with adaptive weight adjustment," *Evolutionary computation*, vol. 22, no. 2, pp. 231–264, 2014.

[47] H. Li and D. Landa-Silva, "An adaptive evolutionary multi-objective approach based on simulated annealing," *Evolutionary Computation*, vol. 19, no. 4, pp. 561–595, 2011.

[48] L. Ke, Q. Zhang, and R. Battiti, "Multiobjective combinatorial optimization by using decomposition and ant colony," *Working Report*, vol. 132, no. 9, pp. 125–140, 2010.

[49] N. Al Moubayed, A. Petrovski, and J. McCall, "A novel smart multi-objective particle swarm optimisation using decomposition," in *Parallel Problem Solving from Nature, PPSN XI.* Springer, 2010, pp. 1–10.

[50] S. Zapotecas Martínez and C. A. Coello Coello, "A multi-objective particle swarm optimizer based on decomposition," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation.* ACM, 2011, pp. 69–76.

[51] A. Alhindi and Q. Zhang, "Moea/d with tabu search for multiobjective permutation flow shop scheduling problems," in *Evolutionary Computation (CEC), 2014 IEEE Congress on.* IEEE, 2014, pp. 1155–1164.

[52] ——, "Moea/d with guided local search: Some preliminary experimental results," in *Computer Science and Electronic Engineering Conference (CEEC), 2013 5th.* IEEE, 2013, pp. 109–114.

[53] W. Huang and H. Li, “On the differential evolution schemes in moea/d,” in *Natural Computation (ICNC), 2010 Sixth International Conference on*, vol. 6. IEEE, 2010, pp. 2788–2792.

[54] C.-M. Chen, Y.-p. Chen, and Q. Zhang, “Enhancing moea/d with guided mutation and priority update for multi-objective optimization,” in *Evolutionary Computation, 2009. CEC’09. IEEE Congress on.* IEEE, 2009, pp. 209–216.

[55] K. Sindhya, S. Ruuska, T. Haanpää, and K. Miettinen, “A new hybrid mutation operator for multiobjective optimization with differential evolution,” *Soft Computing*, vol. 15, no. 10, pp. 2041–2055, 2011.

[56] Y. Lai, “Multiobjective optimization using moea/d with a new mating selection mechanism,” *Master’s thesis, Taiwan Normal University*, 2009.

[57] Q. Zhang, H. Li, D. Maringer, and E. Tsang, “Moea/d with nbi-style tchebycheff approach for portfolio management,” in *Evolutionary Computation (CEC), 2010 IEEE Congress on.* IEEE, 2010, pp. 1–8.

[58] H. Ishibuchi, Y. Sakane, N. Tsukamoto, and Y. Nojima, “Simultaneous use of different scalarizing functions in moea/d,” in *Proceedings of the 12th annual conference on Genetic and evolutionary computation.* ACM, 2010, pp. 519–526.

[59] ——, “Adaptation of scalarizing functions in moea/d: An adaptive scalarizing function-based multiobjective evolutionary algorithm,” in *Evolutionary Multi-Criterion Optimization.* Springer, 2009, pp. 438–452.

[60] F.-Q. Gu and H.-L. Liu, “A novel weight design in multi-objective evolutionary algorithm,” in *Computational Intelligence and Security (CIS), 2010 International Conference on.* IEEE, 2010, pp. 137–141.

[61] S. Jiang, Z. Cai, J. Zhang, and Y.-S. Ong, “Multiobjective optimization by decomposition with pareto-adaptive weight vectors,” in *Natural Computation (ICNC), 2011 Seventh International Conference on*, vol. 3. IEEE, 2011, pp. 1260–1264.

[62] P. C. Chang, S. H. Chen, Q. Zhang, and J. L. Lin, "Moea/d for flowshop scheduling problems," in *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on.* IEEE, 2008, pp. 1433–1438.

[63] Y. Mei, K. Tang, and X. Yao, "Decomposition-based memetic algorithm for multiobjective capacitated arc routing problem," *Evolutionary Computation, IEEE Transactions on*, vol. 15, no. 2, pp. 151–165, 2011.

[64] S. Pal, B. Qu, S. Das, and P. Suganthan, "Linear antenna array synthesis with constrained multi-objective differential evolution," *Progress In Electromagnetics Research B*, vol. 21, pp. 87–111, 2010.

[65] R. d. Carvalho, R. R. Saldanha, B. Gomes, A. C. Lisboa, and A. Martins, "A multi-objective evolutionary algorithm based on decomposition for optimal design of yagi-uda antennas," *Magnetics, IEEE Transactions on*, vol. 48, no. 2, pp. 803–806, 2012.

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

[67] A. Waldock and D. Corne, "Multiple objective optimisation applied to route planning," in *Proceedings of the 13th annual conference on Genetic and evolutionary computation.* ACM, 2011, pp. 1827–1834.

[68] J. Zhang, Q. Tang, Q. Zhang, and Z. Feng, "The research on multiple-impulse correction submunition multi-objective optimization based on moea/d," *Journal of Projectiles, Rockets, Missiles and Guidance*, vol. 30, no. 2, pp. 195–197, 2010.

[69] Q.-b. ZHANG, Z.-w. FENG, Z.-m. LIU, and T. YANG, "Fuel-time multiobjective optimal control of flexible structures based on moea/d [j]," *Journal of National University of Defense Technology*, vol. 6, p. 015, 2009.

[70] Y.-H. Chan, T.-C. Chiang, and L.-C. Fu, "A two-phase evolutionary algorithm for multiobjective mining of classification rules," in *Evolutionary Computation (CEC), 2010 IEEE Congress on.* IEEE, 2010, pp. 1–7.

[71] M. C. Fu, F. W. Glover, and J. April, "Simulation optimization: a review, new developments, and applications," in *Proceedings of the 37th conference on Winter simulation.* Winter Simulation Conference, 2005, pp. 83–95.

[72] Y.-C. Ho, C. G. Cassandras, C.-H. Chen, and L. Dai, "Ordinal optimisation and simulation," *Journal of the Operational Research Society*, pp. 490–500, 2000.

[73] C. Chen, Q. Jia, and L. Lee, *Stochastic Simulation Optimization for Discrete Event Systems: Perturbation Analysis, Ordinal Optimization and Beyond.* World Scientific Publishing Company Pte Limited, 2013.

[74] Q. Zhao, Y.-C. Ho, and Q.-S. Jia, "Vector ordinal optimization," *journal of optimization theory and applications*, vol. 125, no. 2, pp. 259–274, 2005.

[75] S. Teng, L. H. Lee, and E. P. Chew, "Multi-objective ordinal optimization for simulation optimization problems," *Automatica*, vol. 43, no. 11, pp. 1884–1895, 2007.

[76] L. Wang, L. Zhang, and D.-Z. Zheng, "A class of order-based genetic algorithm for flow shop scheduling," *The International Journal of Advanced Manufacturing Technology*, vol. 22, no. 11-12, pp. 828–835, 2003.

[77] L. Zhang and L. Wang, "Genetic ordinal optimization for stochastic traveling salesman problem," in *Intelligent Control and Automation, 2004. WCICA 2004. Fifth World Congress on*, vol. 3. IEEE, 2004, pp. 2086–2090.

[78] S.-C. Horng and S.-Y. Lin, "Combining evolution strategy with ordinal optimization," *International Journal of Information Technology & Decision Making*, vol. 12, no. 02, pp. 233–260, 2013.

[79] S.-Y. Lin and S.-C. Horng, "Application of an ordinal optimization algorithm to the wafer testing process," *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 36, no. 6, pp. 1229–1234, 2006.

[80] S.-C. Horng and F.-Y. Yang, "Ordinal optimization based algorithm for hotel booking limits problem," in *Computer, Consumer and Control (IS3C), 2012 International Symposium on.* IEEE, 2012, pp. 759–762.

[81] C. W. Reynolds, "Evolution of corridor following behavior in a noisy world," *From animals to animats*, vol. 3, pp. 402–410, 1994.

[82] D. Büche, P. Stoll, R. Dornberger, and P. Koumoutsakos, "Multiobjective evolutionary algorithm for the optimization of noisy combustion processes," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 32, no. 4, pp. 460–473, 2002.

[83] J. Branke, P. Funes, and F. Thiele, "Evolving en-route caching strategies for the internet," in *Genetic and Evolutionary Computation–GECCO 2004.* Springer, 2004, pp. 434–446.

[84] A. N. Aizawa and B. W. Wah, "Dynamic control of genetic algorithms in a noisy environment," *vol*, vol. 2, p. 1, 1993.

[85] ——, "Scheduling of genetic algorithms in a noisy environment," *Evolutionary Computation*, vol. 2, no. 2, pp. 97–122, 1994.

[86] P. J. Darwen, "Computationally intensive and noisy tasks: Co-evolutionary learning and temporal difference learning on backgammon," in *Evolutionary Computation, 2000. Proceedings of the 2000 Congress on*, vol. 2. IEEE, 2000, pp. 872–879.

[87] B. L. Miller and D. E. Goldberg, "Genetic algorithms, tournament selection, and the effects of noise," *Complex Systems*, vol. 9, no. 3, pp. 193–212, 1995.

[88] J. Branke and C. Schmidt, "Selection in the presence of noise," in *Genetic and Evolutionary ComputationGECCO 2003.* Springer, 2003, pp. 766–777.

[89] J. Teich, "Pareto-front exploration with uncertain objectives," in *Evolutionary multi-criterion optimization.* Springer, 2001, pp. 314–328.

[90] E. J. Hughes, "Evolutionary multi-objective ranking with uncertainty and noise," in *Evolutionary multi-criterion optimization.* Springer, 2001, pp. 329–343.

[91] ——, "Constraint handling with uncertain and noisy multi-objective evolution," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 2. IEEE, 2001, pp. 963–970.

[92] D. E. Goldberg, *The design of innovation: Lessons from and for competent genetic algorithms.* Springer Science & Business Media, 2013, vol. 7.

[93] M. Rattray and J. Shapiro, "Noisy fitness evaluation in genetic algorithms and the dynamics of learning," 1998.

[94] Y. Sano and H. Kita, "Optimization of noisy fitness functions by means of genetic algorithms using history of search with test of estimation," in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*, vol. 1. IEEE, 2002, pp. 360–365.

[95] ——, "Optimization of noisy fitness functions by means of genetic algorithms using history of search," in *Parallel Problem Solving from Nature PPSN VI.* Springer, 2000, pp. 571–580.

[96] J. Branke, C. Schmidt, and H. Schmec, "Efficient fitness estimation in noisy environments," in *Proceedings of genetic and evolutionary computation.* Citeseer, 2001.

[97] T. Beielstein and S. Markon, "Threshold selection, hypothesis tests, and doe methods," in *wcci.* IEEE, 2002, pp. 777–782.

[98] G. Rudolph, "A partial order approach to noisy fitness functions," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1. IEEE, 2001, pp. 318–325.

[99] S. Markon, D. V. Arnold, T. Back, T. Beielstein, and H.-G. Beyer, "Thresholding-a selection operator for noisy es," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1. IEEE, 2001, pp. 465–472.

[100] P. D. Stroud, "Kalman-extended genetic algorithm for search in nonstationary environments with noisy fitness evaluations," *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 1, pp. 66–77, 2001.

[101] A. Singh, "Uncertainty based multi-objective optimization of groundwater remediation design," *Master's Thesis, University of Illinois at Urbana-Champaign*, 2003.

[102] C. K. Goh and K. C. Tan, "Noise handling in evolutionary multi-objective optimization," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on.* IEEE, 2006, pp. 1354–1361.

[103] E. J. Hughes, "Multi-objective probabilistic selection evolutionary algorithm (mopsea)," Technical Report No. DAPS/EJH/56/2000, Department of Aerospace, POwer & Sensors, Cranfield University, Tech. Rep., 2000.

[104] S. Rana, L. D. Whitley, and R. Cogswell, "Searching in the presence of noise," in *Parallel Problem Solving from NaturePPSN IV.* Springer, 1996, pp. 198–207.

[105] L. T. Bui, H. A. Abbass, and D. Essam, "Fitness inheritance for noisy evolutionary multi-objective optimization," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation.* ACM, 2005, pp. 779–785.

[106] T. Bäck and U. Hammel, "Evolution strategies applied to perturbed objective functions," in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on.* IEEE, 1994, pp. 40–45.

[107] D. V. Arnold and H.-G. Beyer, "A general noise model and its effects on evolution strategy performance," *Evolutionary Computation, IEEE Transactions on*, vol. 10, no. 4, pp. 380–391, 2006.

[108] H. Li, “Combination of Evolutionary Algorithms with Decomposition Techniques for Multiobjective Optimization,” Ph.D. dissertation, University of Essex, UK, 2007.

[109] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, “Performance assessment of multiobjective optimizers: an analysis and review,” *Evolutionary Computation, IEEE Transactions on*, vol. 7, no. 2, pp. 117–132, 2003.

[110] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff, “Generalizing surrogate-assisted evolutionary computation,” *Evolutionary Computation, IEEE Transactions on*, vol. 14, no. 3, pp. 329–355, 2010.

[111] D. W. Corne, J. D. Knowles, and M. J. Oates, “The pareto envelope-based selection algorithm for multiobjective optimization,” in *Parallel problem solving from nature PPSN VI.* Springer, 2000, pp. 839–848.

[112] K. C. Tan, T. H. Lee, and E. F. Khor, “Evolutionary algorithms with dynamic population size and local exploration for multiobjective optimization,” *Evolutionary Computation, IEEE Transactions on*, vol. 5, no. 6, pp. 565–588, 2001.

[113] K. C. Tan, C. K. Goh, Y. Yang, and T. H. Lee, “Evolving better population distribution and exploration in evolutionary multi-objective optimization,” *European Journal of Operational Research*, vol. 171, no. 2, pp. 463–495, 2006.

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

[115] C. M. Fonseca and P. J. Fleming, “Multiobjective optimization and multiple constraint handling with evolutionary algorithms. i. a unified formulation,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, no. 1, pp. 26–37, 1998.

[116] ——, “Multiobjective optimization and multiple constraint handling with evolutionary algorithms. ii. application example,” *Systems, Man and Cybernetics, Part A: Systems and Humans, IEEE Transactions on*, vol. 28, no. 1, pp. 38–47, 1998.

[117] F. Kursawe, “A variant of evolution strategies for vector optimization,” in *Parallel Problem Solving from Nature.* Springer, 1990, pp. 193–197.

[118] J. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, ser. Wiley Series in Discrete Mathematics and Optimization. Wiley, 2005.

[119] S.-C. Horng and F.-Y. Yang, “Combining particle swarm with ordinal optimization for stochastic simulation optimization problems,” in *Control Conference (ASCC), 2011 8th Asian.* IEEE, 2011, pp. 982–987.

[120] F. Black, “Noise,” *The journal of finance*, vol. 41, no. 3, pp. 529–543, 1986.

[121] A. Ponsich, A. L. Jaimes, and C. A. C. Coello, “A survey on multiobjective evolutionary algorithms for the solution of the portfolio optimization problem and other finance and economics applications,” *Evolutionary Computation, IEEE Transactions on*, vol. 17, no. 3, pp. 321–344, 2013.

[122] K. Anagnostopoulos and G. Mamanis, “A portfolio optimization model with three objectives and discrete variables,” *Computers & Operations Research*, vol. 37, no. 7, pp. 1285–1297, 2010.

[123] Y. Ali and S. Mehrotra, “Simplifying the portfolio optimization process via single index model,” *Northwestern University*, 2008.

[124] E. J. Elton, M. J. Gruber, and J. C. Rentzler, “Professionally managed, publicly traded commodity funds,” *Journal of Business*, pp. 175–199, 1987.

[125] C. Fulga and S. Dedu, “A new approach in multi-objective portfolio optimization using value-at-risk based risk measure,” in *Information and Financial Engineering (ICIFE), 2010 2nd IEEE International Conference on.* IEEE, 2010, pp. 765–769.

[126] G. J. Alexander and A. M. Baptista, "A comparison of var and cvar constraints on portfolio selection with the mean-variance model," *Management Science*, vol. 50, no. 9, pp. 1261–1273, 2004.

[127] F. He, R. Qu, and R. John, "A compromise based fuzzy goal programming approach with satisfaction function for multi-objective portfolio optimisation," 2015.

[128] H. Markowitz, "Portfolio selection," *The journal of finance*, vol. 7, no. 1, pp. 77–91, 1952.

[129] C.-H. Chen, J. Lin, E. Yücesan, and S. E. Chick, "Simulation budget allocation for further enhancing the efficiency of ordinal optimization," *Discrete Event Dynamic Systems*, vol. 10, no. 3, pp. 251–270, 2000.

# A

# True Pareto fronts

The approximated Pareto front as proposed by the System Optimization group[1] in Zurich is considered as the true Pareto front for each of the test problems used in this thesis. The head of this group is Professor Eckart Zitzler who, in conjunction with Professor K. Deb and Professor L. Thiele [114], devised the ZDT problems . This information along with more complementary information is available online[2].

[1]http://www.tik.ee.ethz.ch/sop/
[2]http://www.tik.ee.ethz.ch/sop/download/supplementary/testproblems/

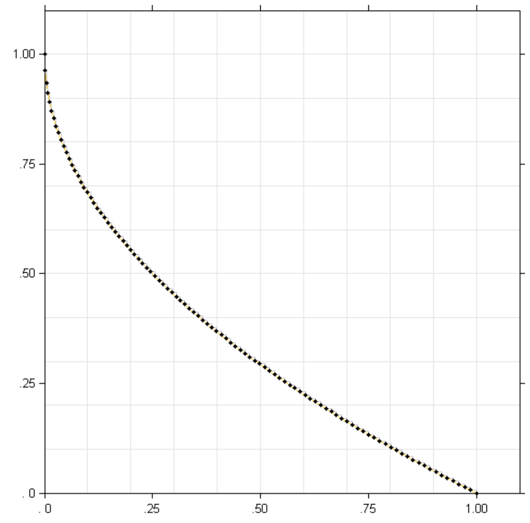


Figure A.1: True Pareto front of ZDT1.

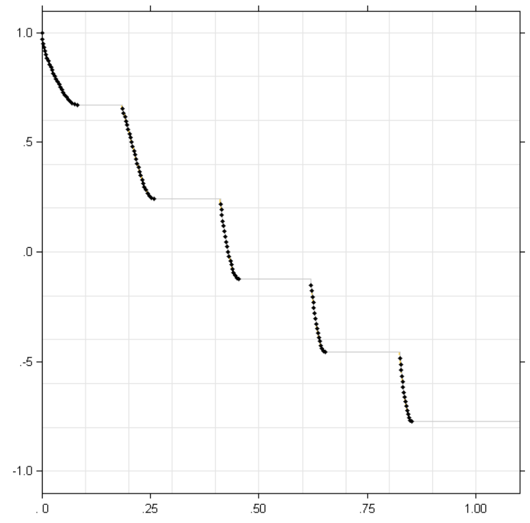


Figure A.2: True Pareto front of ZDT3.

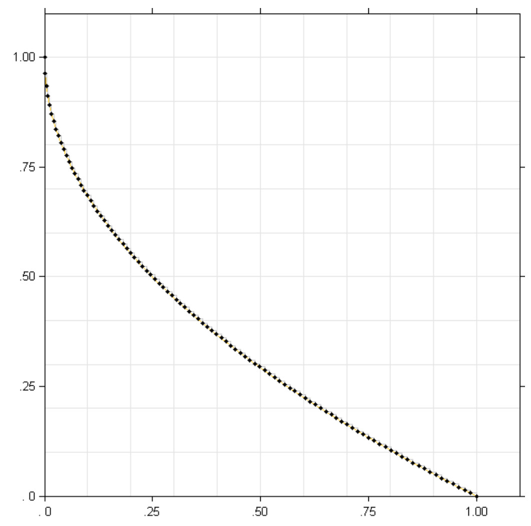


Figure A.3: True Pareto front of ZDT4.

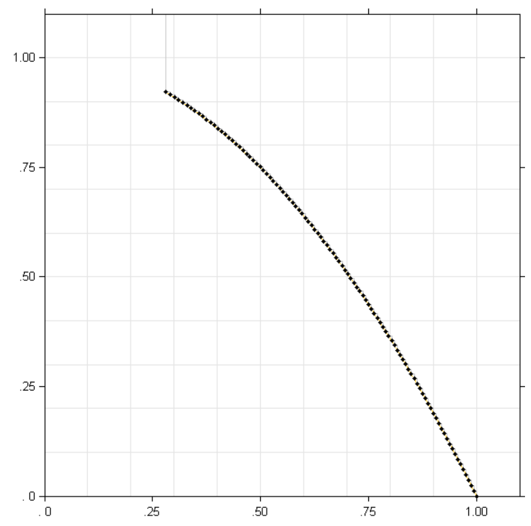


Figure A.4: True Pareto front of ZDT6.

# B

# Ordinal Optimization Demonstration

The following demonstration has been designed by Yu-Chi Ho from the University of Harvard[1] who proposed the ordinal optimization technique.

This demonstration is designed in Microsoft Excel.

- 1. Open a new work sheet.
- 2. Enter the value of "1" in cell A1.
- 3. Enter the formula "=A1+1" in cell A2.
- 4. Copy cell A2.
- 5. Paste into cells A3:A200.

This will yield for column $A1$ through $A200$ the values $1, 2, ..., 200$ which represents an Ordered Performance Curve (OPC) of a complex system. Note the OPC must be

[1]http://people.seas.harvard.edu/ ho/DEDS/OO/Demo/Simple.html

monotonic and one dimensional regardless of the complexity of the system. The best performance is "1", the second best "2", ..., and so on. You can change the values in the cell later on if you want.

- 6. Enter the formula "=rand()*100" into cell B1.
- 7. Copy B1 and paste into cells B2 through B200.

This enters a column of random numbers uniformly distributed between 0 and 100. In column B, note the range of the noise is half as large as the range of the system performance. You can of course change the value defining the range which is "100" currently.

- 8. Enter the formula "=A1+B1" in cell C1.
- 9. Copy cell C1 and paste into C2 through C200.

This operations enters the estimated (or noise corrupted) system performance of the 200 design in column C

- 10. Now copy the three-column region from A1 to C200.
- 11. Define another three column region from D1 to F200.
- 12. Use "paste special" command to "value-paste" the contents (not the formula) of A1 - C200 to D1 - F200.

Because of the design of Excel we cannot just paste A1 - C200 to D1 - F200. You will note that at the end of step 12, the content of D1-F200 is what was in A1-C200 before the execution of the paste special command. The content of A1-C200 is a NEW set of noises and estimated system performance.

- 13. Define the region F1-D200, and choose the sort command from the "data" menu. Now sort the F column in ascending order by rows.
- 14. Now look at column D, and see how many numbers from 1 through n appear in the top n rows. This correspond to one sample realization of the alignment between the actual top-n design with the estimated top-n design.

If you did the above steps correctly, you should see on the average 4 of the true top-12 in the first twelve rows of column D for the given value of the parameters, i.e.,

- N = total of designs considered and estimated = 200
- W = range of estimation noise = 100
- n = range of good enough subset, the top-n designs = 12

Now you are ready to repeat steps 12-14 again for another sample realization.